Ю. И. Рыжиков

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

теория и технологии

Санкт-Петербург, 2004

Оглавление

Предисловие **12**

1 Имитационное моделирование и анализ сложных систем **21**
1.1 Математическое моделирование 21
1.2 Общая характеристика имитационного моделирования СМО 35
1.3 Организация моделирования 39
1.4 Достоверность модели 45
1.4.1 Валидация 45
1.4.2 Верификация 47
1.4.3 Аккредитация 47

2 Принципы построения имитационных моделей **48**
2.1 Элементы имитационной модели 48
2.2 Базовый алгоритм моделирования 52
2.3 Лабораторная работа с моделью 61
2.4 Варианты моделей систем обслуживания 64
2.4.1 Неоднородный поток заявок 64
2.4.2 Кольцевая система очередей 65
2.4.3 Статический относительный приоритет 65
2.4.4 Динамический приоритет 65
2.4.5 Многоуровневое квантованное обслуживание . . . 66
2.4.6 Определение длины периода непрерывной занятости 66
2.4.7 Абсолютный приоритет, одноканальная система . 66
2.4.8 Многоканальные приоритетные системы 67
2.5 Моделирование сетей обслуживания 79
2.6 Моделирование вычислительных систем 92
2.7 О надежности программ моделирования 93

3 Элементы теории и аппроксимации распределений 96
3.1 Постановка задачи аппроксимации 96
3.2 Принципы аппроксимации распределений 98
3.3 Показательное распределение 100
3.4 Распределения фазового типа 101
3.4.1 Распределение Эрланга 101
3.4.2 Гиперэкспоненциальное распределение 102
3.4.3 Распределение Кокса 103
3.4.4 Гиперэрлангово распределение 104
3.4.5 Распределение Ньютса 105
3.5 Преобразования распределений 106

4 Генерация случайных чисел 108
4.1 Схема получения случайных чисел 108
4.2 Генерация равномерно распределенных чисел 110
4.2.1 Физические и программные датчики 110
4.2.2 Классические конгруентные генераторы 113
4.2.3 Генераторы Смита 119
4.2.4 Раздельные датчики 123
4.3 Метод обратной функции 125
4.3.1 Идея метода и точные обращения 125
4.3.2 Численное обращение 129
4.3.3 Аппроксимация обратной ФР 129
4.3.4 Дискретные распределения 132
4.4 Аппроксимация плотностей 133
4.5 Метод режекции 136
4.5.1 Идея метода 136
4.5.2 Логарифмически вогнутые плотности 137
4.6 Метод композиции 138
4.7 Генерация по заданной функции интенсивности 138
4.8 Генерация по моментам 139
4.9 Частные методы 140
4.9.1 Непрерывные распределения 141
4.9.2 Дискретные распределения 147
4.10 Генерация пары коррелированных величин 149
4.11 Случайные выборки 149
4.12 Случайные объекты 150
4.13 Дополнительная литература 150

5 Обработка результатов моделирования 156
5.1 Точечные оценки 156
5.2 Дисперсия и корреляция 162
5.3 Классический подход к интервальным оценкам 162

6 Уточненная обработка статистики 166
6.1 Регенерирующий процесс и его обработка 166
6.2 Приближенно регенерирующие модели 169
6.3 Метод «складного ножа» 171

7 Эксперименты над моделями 173
7.1 Введение 173
7.1.1 Постановка задачи 173
7.1.2 Целевая функция 174
7.1.3 Базовые понятия теории экспериментов 174
7.2 Дисперсионный анализ 175
7.3 Скрининг-эксперименты 179
7.4 Оптимизирующие эксперименты 183
7.4.1 Общий подход 183
7.4.2 Новый метод вычисления градиента 184

8 Понижение дисперсии 188
8.1 Проблема и методы 188
8.2 Выбор оценок с наименьшей дисперсией 189
8.3 Расслоенные выборки 194
8.4 Значимые выборки 196
8.5 Контрольные переменные 198
8.5.1 Расчетная схема для одномерного случая 198
8.5.2 Выбор контрольных переменных 200
8.5.3 Расчетная схема для многомерного случая 201
8.5.4 Доверительный интервал из теории регрессий . . 203
8.5.5 Доверительный интервал для «складного ножа» . 204
8.5.6 Потери в уменьшении дисперсии 205
8.5.7 Контрольные переменные и замкнутые сети 206
8.6 «Противоположные» выборки 210
8.7 Параллельные выборки 218
8.8 Условная имитация 219

9 Языки моделирования 224
9.1 Универсальные и специализированные языки 224
9.1.1 Универсальные языки программирования 224
9.1.2 Системы моделирования 225
9.2 СЛЭНГ 231
9.3 Универсальная система Симула 67 233
9.3.1 Базисные средства 233
9.3.2 Наборы 235
9.3.3 Средства моделирования 236
9.3.4 Оценка языка 240
9.4 Новые зарубежные пакеты 241
9.5 Моделирование и математические пакеты 242

10 Введение в GPSS 244
10.1 Общая характеристика 244
10.1.1 Рождение системы 244
10.1.2 Базовые понятия 245
10.1.3 GPSS World 248
10.2 Пример программы 250
10.3 Инсталляция 255
10.4 Методические указания 255

11 Объекты GPSS 257
11.1 Классы сущностей 257
11.2 Стандартные числовые атрибуты 258
11.3 Транзакты 259
11.3.1 Сегменты модели 259
11.3.2 Параметры транзактов 260
11.3.3 Цепи транзактов 262
11.3.4 Планировщик транзактов 265
11.4 Ансамбли и группы 266
11.5 Устройства и памяти 269
11.6 Приоритет транзактов и прерывания 272
11.6.1 Механизм приоритета 272
11.6.2 Прерывание обслуживания 273
11.6.3 Приоритет и временн´ые узлы 273
11.7 Логические ключи 274
11.8 Переменные и сохраняемые значения 274

11.8.1 Переменные 274
11.8.2 Сохраняемые значения 275
11.9 Накопление статистики 275
11.9.1 Очереди 275
11.9.2 Таблицы общего назначения 276
11.10 Трассировка 277
11.11 Генераторы случайных чисел 277
11.12 Потоки данных 278

12 Язык GPSS 279
12.1 Виды операторов GPSS 279
12.1.1 Команды 279
12.1.2 Блоки GPSS 281
12.2 Структура операторов 283
12.2.1 Формат 283
12.2.2 Поле метки 284
12.2.3 Поле операций 284
12.2.4 Поле операндов 285
12.2.5 Комментарии 285
12.3 Алфавит и элементарные конструкции 285
12.4 Стандартные числовые атрибуты 287
12.5 Переменные и выражения 288
12.6 PLUS и процедуры 290
12.6.1 Структура и вызов процедуры 290
12.6.2 Подоператоры процедуры 292
12.7 Библиотеки процедур 293
12.7.1 Запросные процедуры 294
12.7.2 Строковые процедуры 294
12.8 Распределения вероятностей 296
12.8.1 Непрерывные распределения 297
12.8.2 Дискретные распределения 302

13 Функциональные комплексы 304
13.1 Траекторный комплекс 304
13.2 Атрибутный комплекс 312
13.3 Ансамблевый комплекс 321
13.4 Ресурсный комплекс 329
13.5 «Цепной» комплекс 335

13.6 Комплекс управления 341
13.7 Комплекс ввода-вывода 343

14 Технология моделирования 346
14.1 Среда 346
14.1.1 Меню 346
14.1.2 Боксы диалогов и Help 347
14.1.3 Настройки 348
14.1.4 Окна 349
14.1.5 Этапы моделирования 349
14.2 Останов моделирования 350
14.3 Сбор статистики 351
14.4 ANOVA 352
14.5 Синтаксис Эксперимента 353
14.6 Стандартный отчет 354
14.7 «Горячие клавиши» 355
14.8 Проблемы эффективности 357
14.8.1 Программирование 357
14.8.2 Окна 358
14.8.3 Память 358
14.8.4 Работа с файлами 359
14.9 Начальные условия
и стационарный режим 359

15 Практическая работа в GPSS World 361
15.1 Первая программа 361
15.2 Технологический инструментарий 369
15.3 Текстовый редактор 371
15.4 Транзакты, GENERATE и TERMINATE 375
15.5 Конец сеанса 379
15.6 Блок ADVANCE 379
15.7 Устройства. Блоки SEIZE и RELEASE 381
15.8 Очереди и таблицы. Блоки QUEUE, DEPART 384
15.9 Команды GPSS World 389
15.10 Ручное моделирование 389
15.11 Графические окна GPSS World 391
15.12 Работа с внешними файлами 403
15.13 ANOVA 412

15.14 Отчет 415
15.15 Журнал сессии 416
15.16 Непрерывные модели 417
15.17 Отладка моделей 419
15.17.1 Организация отладки 419
15.17.2 Пример отладки 421
15.17.3 Оперативная семантическая отладка 423
15.17.4 Работа с командными файлами 426
15.17.5 Отладка PLUS-процедур 426
15.18 Эксперименты в GPSS World 428
15.19 Генерация скрининг-эксперимента 435
15.20 Оптимизирующий Эксперимент 440

16 Типовые модели теории очередей 441
16.1 Модель $M/M/1$ и датчики случайных чисел 441
16.1.1 Смитовский равномерный датчик 441
16.1.2 Датчики с произвольными распределениями . . . 444
16.2 Многоканальная система 447
16.3 Приоритетные системы обслуживания 453
16.4 Квантование времени 458
16.5 Круговое обслуживание 459
16.6 Замкнутые системы 462
16.7 Нетерпеливые заявки 463
16.8 Групповые заявки 464
16.9 Работа с памятями 464
16.10 Обслуживание с разогревом 465
16.11 Обслуживание с порогом включения 465
16.12 Учет недельного и суточного циклов 466
16.13 Моделирование цепями правил планирования 467
16.13.1 Вводный пример 468
16.13.2 Варианты дисциплин обслуживания 469
16.13.3 Ограничение времен ожидания в очередях 470
16.14 Моделирование процессов сборки 470
16.15 Прикладные задачи 472
16.16 Оценка GPSS World 475

17 NetCracker **478**
17.1 Общая характеристика 478
17.2 Инсталляция 480
17.3 Панели инструментов 481
17.4 База данных 484
17.5 Создание проекта 484
17.6 Синтез простейшей сети 485
17.6.1 Приборы 485
17.6.2 Функции 486
17.6.3 Связи 487
17.7 Многоуровневый проект 488
17.8 Анимация и имитация 489
17.8.1 Основные понятия 489
17.8.2 Анимация 489
17.8.3 Имитация 490
17.8.4 Оценка работы сети 491
17.9 Режим «Discovery» 491

18 Конференция ИММОД-2003 **494**
18.1 История имитационного моделирования 494
18.2 Современное состояние имитационного моделирования . 495
18.3 Имитационный ренессанс 496
18.4 Общая оценка и результаты конференции 496
18.5 Методология моделирования 497
18.5.1 Общие принципы 497
18.5.2 Визуализация моделирования 499
18.5.3 Распределенное моделирование 500
18.5.4 Методы моделирования 501
18.5.5 Достоверность модели 502
18.6 Языки и системы моделирования 504
18.6.1 Обзор новинок 504
18.6.2 GPSS 505
18.6.3 C, C++ 505
18.6.4 Паскаль и Delphi 506
18.6.5 Фортран 507
18.7 Моделирование в образовании 507
18.8 Применение моделирования 508
18.8.1 Промышленность 508

18.8.2 Логистика 510
18.8.3 Транспорт 512
18.8.4 Авиация 513
18.8.5 Горное дело 513
18.8.6 Медицина 514
18.8.7 Финансы 514
18.8.8 Информатика 515
18.8.9 Коммунальное хозяйство 516
18.9 Недостатки конференции 516

Заключение **519**

Литература **521**

Предисловие

В разных областях техники, в организации производства, в социальной сфере и в военном деле постоянно возникает необходимость решения вероятностных задач, связанных с работой систем *массового обслуживания* разного вида требований. Термин «массовое обслуживание» предполагает многократную повторяемость ситуаций в том или ином смысле (много прибывших в систему и обслуженных заявок, большое число находящихся в эксплуатации аналогичных систем) и статистическую устойчивость картины. Выводы и рекомендации, получаемые методами теории массового обслуживания (ТМО), применимы лишь при наличии одного или обоих перечисленных факторов повторяемости.

По справедливому замечанию виднейшего советского специалиста в этой области Б.В.Гнеденко, легче указать ситуации, где не может быть использована ТМО (она же — теория очередей), чем перечислить все сферы ее потенциального применения. К таким сферам, в частности, относятся:

- в технике связи — проектирование телефонных станций и сетей связи;

- на транспорте — анализ процессов дорожного движения, прохождения туннелей, очередей у светофоров, технического осмотра автомашин, формирования железнодорожных составов на сортировочных станциях, регистрации пассажиров, взлета и посадки самолетов, диспетчерской службы в аэропортах, разгрузки и погрузки судов в речных и морских портах, резервирования билетов;

- в промышленности — при планировании сборочных операций (расчет числа линий и емкости бункеров), гибких автоматизированных производств, организации ремонта оборудования, расчете объема

восстанавливаемого ЗИП, работе пунктов обслуживания (например, инструментальных складов), расчете зон обслуживания (в прядильном и ткацком производстве), определении числа резервных забоев в шахтах;

- в автоматизированных системах — для оценки своевременности обслуживания заявок на вычислительные работы и подготовку данных, обработки результатов экспериментов, управления технологическими процессами, при исследовании производительности вычислительных систем с асинхронно работающими устройствами;

- в торговле — для определения количества магазинов, продавцов, фасовщиц, узлов расчета, торговых и кассовых автоматов;

- в здравоохранении — при определении необходимого количества аптек, больничных коек, станций и бригад скорой помощи, врачей и младшего медицинского персонала, потребностей в диагностической и лечебной аппаратуре, пропускной способности специализированных санаториев;

- в службе быта — при расчете сети парикмахерских, мастерских разного назначения, количества техники и работников в них, транспортных средств обеспечения услуг населению, пунктов приема коммунальных платежей, количества ремонтников в жилищно-эксплуатационных конторах и трестах, сотрудников бюро по трудоустройству, по обмену жилой площади и т. д.;

- в органах юстиции и внутренних дел — при расчете количества судебных органов и их штатной численности, емкости исправительно-трудовых учреждений, численности сотрудников следственного аппарата и оперативников, планировании работы контрольно-пропускных пунктов и пунктов таможенного досмотра;

- в сфере науки и образования — при иссследовании некоторых видов природных процессов (регистрация элементарных частиц, фильтрация, диффузия, действие катализаторов химических реакций), запуске ИСЗ и обработке спутниковой информации, расчете количества лабораторных установок, экзаменаторов (в том числе автоматических), возможностей учебно-опытных производств, бытово-

го и медицинского обслуживания студентов; при проектировании и анализе работы крупных библиотек;

- в издательском деле — при расчете численности сотрудников, участвующих в подготовке рукописей к изданию;
- в военном деле — при проектировании систем ПВО (каждая цель может рассматриваться как «заявка» на обслуживание, то есть обстрел), организации охраны границ, расчете патрульных нарядов и во многих других случаях типа рассмотренных выше, но применительно к боевой и повседневной деятельности войск.

Сразу же подчеркнем, что задачи указанных типов приходится решать не только при проектировании вновь создаваемых систем и сетей обслуживания (СМО, СеМО), но и в процессе эксплуатации имеющихся:

- при увеличении нагрузки;
- изменении трудоемкости обработки заявок;
- выходе из строя, деградации или модернизации техники;
- снижении квалификации персонала;
- пересмотре требований к оперативности обработки заявок и т. п.

Характерной чертой массового обслуживания является случайный поток заявок и случайная продолжительность времени обслуживания. Поэтому прогноз относительно единичного события может быть только вероятностным.

Применение теории очередей особенно эффективно при большой роли случайных факторов, что подтверждается несколько экстремистским заявлением Г.Поттгофа: « ...В этих опытах теория и наблюдения хорошо совпадали, так как благодаря большой нерегулярности или отсутствию должного порядка, что имелось в годы войны, были созданы особенно благоприятные предпосылки для применения теории вероятностей» [51, с.103]. Добавим, что столь же «благоприятно» в указанном смысле и разрегулированное состояние экономики. Впрочем, и в налаженной экономике проблема очередей актуальности не теряет. Дело в том, что *возникновение очередей является естественной*

и неизбежной платой за совместное использование ограниченных ресурсов. «С развитием цивилизации люди все больше и больше времени проводят в очередях, и необходимо найти пути преодоления этих неприятностей» [29, т.1, с.17].

Многообразие приложений теории массового обслуживания определяет постоянно растущий интерес к ней, а сложность возникающих задач не позволяет получить исчерпывающие решения на базе аналитических методов [55, 61] даже при численной реализации последних. В таких ситуациях приходится прибегать к *имитационному моделированию*, представляющему собой незаменимый инструмент анализа эксплуатационных и многих других проблем.

Настоящая книга написана на базе раздела лекций по моделированию систем, читанных автором в Военно-космической академии им. А. Ф. Можайского (первоначально в объеме главы из [16]), радикально переработанного в 1990 г. [?] и постоянно дополнявшегося с учетом современного состояния вопроса и потребностей прикладных исследований в течение последних десяти лет.

В первой главе излагаются базовые концепции математического моделирования вообще и имитационного в особенности и дается общая характеристика последнего.

Содержание глав 2–4 можно рассматривать как введение в построение моделей и технику моделирования. Здесь к традиционному перечню вопросов добавлены некоторые новые идеи в области генерации случайных чисел (схема перемешивания, формирование пар нормально распределенных чисел), обработки результатов моделирования (нейтрализация смещения средних вследствие принятого на многих ЭВМ правила округления, рекуррентный расчет среднего, дисперсии и корреляционных моментов), построения моделей сетей обслуживания.

Главы 5–8 образуют продвинутую часть курса. Здесь прежде всего обоснованы и описаны регенеративный подход к анализу моделей и метод «складного ножа» уменьшения смещения оценок. Обсуждаются основные понятия теории экспериментов и их применения к дисперсионному и регрессионному анализу и оптимизации имитационных моделей. Далее рассмотрены методы понижения дисперсии результатов моделирования, применение которых заставляет пересмотреть традиционную оценку имитационного моделирования как тупого и непомерно трудоемкого инструмента исследования. Приведены новые данные о корреляции

взаимно дополняющих переменных.

В главе 9 обсуждаются проблемы программирования имитационных моделей на универсальных языках, обсуждаются целесообразность и основные особенности языков моделирования. Приведен обзор последних и представление о языках *СЛЭНГ* и *Симула* 67. Дан обзор средств имитационного моделирования, включаемых в общецелевые математические пакеты.

Главы 11–15 содержат подробное описание общецелевой системы имитационного моделирования *GPSS World* (далее *GPSS/W*) и программирования в ней, при составлении которого учитывались три обстоятельства:

- *GPSS/W* обсуждается после достаточно подробного изложения фундаментальных идей имитационного моделирования;
- отличия *GPSS/W* от *GPSS/PC* и более ранних версий *GPSS* четко выделены в электронной документации, интересны для очень узкого круга лиц и потому здесь не оговариваются;
- изучение этой книги предполагается в ходе практической работы с *GPSS/W*, так что технологические детали (форматы операндов блоков и частные схемы их использования) удобнее искать в системном `Help`'е.

Общеизвестна трудность написания руководств по программированию, порождаемая тесной связью обсуждаемых понятий, многообразием контекстов их использования и областей применения. Любая логика их изложения, исключая гипертекст, неизбежно будет уязвимой для критики. Мы выбрали стратегию «сверху вниз», группируя материал по родству *абстрактных* применений. Глава 15 для удобства практической работы с системой отчасти дублирует ранее сообщенные сведения.

Неисчерпаемость проблемы вынудила провести жесткий отбор материала и свести к минимуму число примеров законченных программ: вновь вводимые понятия иллюстрируются преимущественно связанными с ними *фрагментами* моделирующих программ. Относительно достаточности их числа и объема комментариев возможны различные мнения. Однако следует иметь в виду, что учебник — не дамский роман: его надо не читать, но *прорабатывать за компьютером*. В утешение читателю отметим, что для работы на начальном уровне необходимо небольшое подмножество средств. При накоплении достаточного опыта переход

к более сложному инструментарию окажется безболезненным и вполне естественным. Таким образом, систему *GPSS/W* заинтересованный читатель должен рассматривать как объект *перманентного* изучения.

Изучение *GPSS* следует начинать с освоения ее среды и языка системы — здесь серьезных затруднений не предвидится. Далее следует ознакомиться с главой 11 относительно объектов *GPSS*, которая содержит упорядоченное по родству функций или областей применения описание основных ее сущностей. Разделы «Цепи событий», «Ансамбли транзактов», «Приоритет транзактов и прерывания устройств» целесообразно для начала пролистать, а более обстоятельную проработку отложить до возникновения практической потребности в них. Эту главу следует использовать как справочник при освоении главы 15 — «технологического практикума», а также при разработке собственных моделей. Для оперативных справок при работе за компьютером (например, об операндах некоторого блока) удобнее пользоваться системным справочником с алфавитным или контекстным доступом — см. разд. 14.1.2.

Практикум является продуктом глубокой переработки системного `Tutorial` — главным образом в смысле устранения повторов, уточнения формулировок, расстановки логических акцентов и обогащения личным опытом автора, приобретенным в процессе освоения системы. Были также исправлены обнаруженные ошибки — в частности, вкравшиеся в текст пережитки *DOS*-версий *GPSS* (например, закрепление команд за *буквенными* клавишами). Поскольку глава 15 базируется на сопровождающем систему пакете типовых файлов (папка `Sample Models`), цитируемые из них тексты программ оставлены без изменения — исключая перевод на русский язык комментариев. Сразу же заметим, что кириллические комментарии не диагностируются компилятором как ошибки, но, как ни странно, исключают правильную работу программ [1].

В главе 16 приведены сводные рекомендации по программированию на *GPSS*, сгруппированные по абстрактным проблемам. Там же даны «целеуказания» для интересующихся конкретными приложениями — в частности, перечень типовых программ, включенных в электронную документацию к системе (папка `Sample Models`).

Уникальной особенностью глав, посвященных *GPSS*, является критический анализ обсуждаемой системы — в частности, конкретные

[1] Во всех известных автору системах программирования компиляторы игнорируют текст примечаний.

данные по быстродействию программ в сопоставлении с написанными на *Фортране*.

Глава 17 содержит краткое описание интегрированной системы проектирования сетей связи *NetCracker*, использующей имитационное моделирование как средство анализа их характеристик.

Раздел 18 содержит аналитический обзор материалов проведенной в ЦНИИ технологии судостроения (Санкт-Петербург) 23–24 октября 2003 г. 1-й Всероссийской научно-практической конференции «Опыт практического применения языков и программных систем имитационного моделирования в промышленности и прикладных разработках». Обзор демонстрирует размах и многообразие исследований по имитационному моделированию в России и их практическую реализацию. Доклады сопоставляются по методам моделирования, инструментальным средствам и прикладным задачам.

Список литературы содержит практически все доступные в России и достойные внимания источники (прежде всего нужно отметить [28, 80, 84]).

Предполагается, что читатель знаком с основными понятиями теории массового обслуживания и фундаментальными соотношениями в ней типа формулы Литтла хотя бы в объеме третьей главы [?].

Работа в области прикладной математики без уверенного владения техникой программирования не только бессмысленна, но и опасна (и для общества, и для самого горе-специалиста). К сожалению, в последние десятилетия увлечение «информационными технологиями» и демагогия о «непрограммирующих пользователях» заметно ухудшили программистскую подготовку будущих математиков и инженеров в их основной массе. В связи с этим в книгу включены типичные процедуры имитационного моделирования — полностью или в виде фрагментов. Автор имеет сорокалетний личный опыт программирования и почти такой же — его преподавания и уверен, что тщательный разбор упомянутых процедур многому научит читателей. Синтаксис включаемых в текст фрагментов программ по понятным причинам подчинен синтаксису языков программирования.

Несколько слов о выборе языка «вычислительного» программирования. Для расчетных приложений сейчас самыми массовыми являются *ПЛ/1* (на больших машинах), *Паскаль*, *C* и *Фортран*. По поводу их

сравнения см., например, [60], где делается вывод о явной предпочтительности *Фортрана* 90. По изложенным причинам процедуры приводятся на *Фортране* 90 с *минимальным* набором средств, отсутствующих в *Фортране* 77 (для возможности быстро адаптировать их к последнему). При адаптации необходимо учитывать следующие различия (в режиме свободного формата):

Таблица 1. Различия версий Фортрана

Свойство	*Фортран* 90	*Фортран* 77
Операторов в строке	произвольно (через ;)	один
Признак продолжения	& в конце строки	минус в конце строки
Строка-комментарий	" или ! в первой колонке	C или *
Описание allocatable	<тип>, allocatable :: <список вида <имя>(:)>	отдельными dimension <> [allocatable](:)

Автор стремился использовать традиционные и по возможности простые обозначения. В частности, большими буквами — например, (X, U) — обозначаются случайные величины, а малыми (x, u) — их возможные значения. Малыми буквами с соответствующими нижними индексами обозначены моменты распределений. Если индекс опущен, то подразумевается первый момент. Малые буквы m и d зарезервированы для обозначения математического ожидания и дисперсии указанных индексом случайных величин, большие M и D — для обозначения соответствующих операторов. Знаком «^» отмечены *оценки* числовых характеристик (в отличие от их истинных значений). Поскольку в контексте данной книги все оценки случайны, применение «^» однозначно характеризует объект как случайный и делает необязательным шрифтовое различие.

На протяжении всей книги используются следующие сокращения:

ТМО — теория массового обслуживания,
СМО — система массового обслуживания,
СеМО — сеть массового обслуживания.

Локальные обозначения и сокращения вводятся непосредственно в тексте. Специальные термины снабжаются переводом только при первом их упоминании.

Перечислим технические соглашения, принятые при написании текста. Предполагается, что буквенные клавиши нажимаются на нижнем регистре, то есть без [Shift]. Если нужно нажать несколько клавиш одновременно, их имена заключены в одной паре скобок и соединяются плюсами. Под обозначением вида [Ctrl] подразумеваются (в зависимости от контекста) как сама кнопка, так и ее нажатие. Термины «кнопка», «клавиша» и «иконка» используются как синонимы. «Щелчок мышью» подразумевает нажатие левой ее кнопки. Необходимость щелчка по кнопке [OK], завершающего выбор режимов или набор команды, подсказывается наличием этой кнопки в окне текущего диалога и поэтому оговаривается явно только в первых «уроках» работы с $GPSS/W$.

Последовательный выбор команд из меню дается через слэш. Стандартные последовательности команд (наподобие выбора файла с типовой моделью) подробно поясняются при первом упоминании и — для лучшего запоминания — при нескольких последующих.

Набор и верстку книги автор выполнял в издательской системе LaTeX. Соответственно пришлось мириться с заложенными в этой системе не вполне обычными решениями — в частности, с нумерацией сносок в пределах *главы*.

Глава 1

Имитационное моделирование как метод анализа сложных систем

1.1 Математическое моделирование

Модель есть материально или теоретически сконструированный объект, который заменяет (представляет) объект исследования в процессе познания, находится в отношении сходства с последним (изоморфизм, аналогия, физическое сходство и т. п.) и более удобен для исследования. Как следствие, изучение модели и выполненные над ней операции позволяют получить информацию о реальном объекте исследования. О моделировании можно говорить лишь при использовании модели для познания оригинала: в игре ребенка с моделью паровоза новое знание относительно паровоза не рождается.

Наиболее естественная и важная область применения моделирования — анализ *систем*, позволяющий вести исследование в достаточно широком контексте. Система есть совокупность объектов, функционирующих и взаимодействующих друг с другом для достижения определенной *цели*. Понятие системы зависит от задач конкретного исследования. Ракеты, к примеру, суть компоненты *системы вооружения*, включающей не только саму ракету, но и стартовую установку, средства выявления

цели и определения ее координат, логистическую поддержку, оперативную обстановку, сведения о тактике цели и т. д. [89, с. 2]. *Состояние системы* определяется как совокупность переменных, необходимая для ее описания и прогнозирования поведения в соответствии с задачами исследования.

При анализе системы должны учитываться:

- общесистемные цели;
- окружение системы (ограничения);
- ресурсы системы;
- компоненты системы (активные элементы, цели и показатели эффективности);
- управление системой.

Все это в совокупности определяет структуру системы. Цель может быть сформулирована только с позиций вышележащего уровня.

Математическая модель *концентрирует* в себе написанную на определенном языке (естественном, математическом, алгоритмическом) совокупность наших знаний, представлений и гипотез о соответствующем объекте или явлении. Поскольку эти знания никогда не бывают абсолютными, а гипотезы *вынужденно* или *намеренно* не учитывают некоторые эффекты, модель лишь приближенно описывает поведение реальной системы и является ее абстракцией. Например, для механических конструкций (исключая пружины) обычно можно пренебречь явлениями растяжения и сжатия. Для резисторов в качестве свойства не рассматривается емкость. Для полей оперативной памяти ЭВМ имеет смысл «следующий адрес», но это свойство теряет смысл для регистров процессора.

«Моделирование охватывает широкий диапазон актов человеческого общения — от наскальной живописи и сооружения идолов до составления систем сложных математических уравнений, описывающих полет ракеты в космическом пространстве... По существу, прогресс и история науки и техники нашли свое наиболее точное выражение в развитии способности человека создавать модели естественных явлений, понятий и объектов» [73]. По авторитетному мнению акад. А.А. Самарского [63], «методология математического моделирования может и должна быть ядром информационных технологий, всего процесса информатизации

общества». Любые научно-технические расчеты (на прочность, устойчивость, надежность, безопасность, точность и т. п.) — важнейшие элементы профессиональной деятельности инженера — в сущности являются специализированными видами моделирования.

Развитие языков программирования и распространение компьютеров привели к распространению моделирования не только в науке и технике, но и в бизнесе, экономике, социологии, экологии, политике. О многообразии применений математического моделирования можно судить хотя бы по перечню моделей, рассмотренных в [63]:

- динамика боевых действий, включая партизанскую войну;
- эффект распределения власти в иерархии;
- безопасность ядерного реактора;
- загрязнение грунтовых вод;
- экологические последствия сжигания углеводородных топлив;
- управляемый термоядерный синтез;
- тунгусский метеорит;
- климатические последствия «100-мегатонного» ядерного конфликта;
- магнитогидродинамическая теория циклов солнечной активности;
- процессы в переходной экономике.

Целями моделирования могут являться:

- осмысление действительности;
- постановка над моделью экспериментов с последующей интерпретацией
  их результатов применительно к моделируемой системе;
- прогнозирование будущего поведения системы;
- общение с другими лицами, общественными группами, устройствами обработки информации;

- обучение и тренаж специалистов.

Как иллюстрацию к первому пункту перечня отметим необходимость построения модели для описания каких-то реальных процессов в медицине, экологии, социологии, экономике. Здесь, в отличие от техники, невозможно независимое изучение компонент, и необходимо прежде всего выяснить структуру взаимодействий. К примеру, модель управления запасами в сложной системе [61] не сможет предсказать точный уровень запаса, но обеспечит понимание влияния задержек на уровень обеспечиваемости и общую устойчивость системы.

Модели играют важную и всевозрастающую роль в развитии общества. «Часто новые идеи (например, в управлении производством и экономикой) длительное время не находят применения на практике по той причине, что система, в которую они должны быть внедрены, обладает большой внутренней сложностью и последствия предлагаемых преобразований трудно предсказать. С помощью имитации можно организовать проверку и демонстрацию новых идей и обосновать, таким образом, их принятие или отклонение» [75, с. 15]. В частности, в ВЦ АН СССР под руководством акад. А.А. Петрова [45] в свое время были смоделированы последствия «шоковой терапии» российской экономики. Действительность полностью подтвердила мрачные прогнозы, которые были проигнорированы власть имущими.

Важнейшая особенность модели состоит в возможности неограниченного накопления специализированных знаний без потери целостного взгляда на объект исследования. Моделирование процессов в природе, обществе и технических системах — это основная компонента системного подхода к познанию этих процессов и управлению ими. Математическое моделирование является важнейшим элементом прикладного исследования. От исследований в области теоретической математики прикладную математику отличают:

1. Социальный заказ на каждую конкретную разработку.

2. Жесткие сроки, требования экономичности и эффективности.

3. Сложность и комплексный характер как поставленных проблем, так и методов их решения.

4. Принципиальная необходимость в многообразном использовании ЭЦВМ.

5. Возможность некоторого смягчения традиционно принятых в математических исследованиях стандартов строгости:

   - широкое использование аналогий, правдоподобных рассуждений и эвристических решений;
   - поиск решений, единственность и даже существование которых не доказаны;
   - использование итерационных процедур, сходимость которых теоретически не доказана, и т. д.

Разумеется, перечисленные в п.5 «послабления» вынуждены. Прикладная математика «делает то, что нужно, — так, как может» в отличие от чистой, которая «делает, как нужно, — то, что может».

Решение прикладной проблемы с применением математики неизбежно требует математического моделирования. Общая схема этапов и компонент такого моделирования показана на рис. 1.1.

Адекватность модели объекту всегда ограничена и зависит от цели моделирования. Всякая модель не учитывает некоторые свойства оригинала и потому является его *абстракцией*. Смысл абстрагирования состоит в мысленном отвлечении от несущественных в данном контексте свойств предмета и одновременном выделении свойств существенных. Примерами абстракций из разных областей знания являются точка, линия, материальная точка, математический маятник, абсолютно упругое соударение, идеальный газ, несжимаемая жидкость, абсолютно твердое тело, потребитель, доход, правонарушение и т. п. Абстрагирование может быть многоступенчатым. По мере повышения уровня абстракции объем понятия увеличивается, а содержание уменьшается.

Чем более высок уровень обобщения, тем грубее модель и шире область интерпретации результатов моделирования. Проиллюстрируем эту мысль на математическом моделировании математических же объектов. Члены низкого порядка разложения функции в ряд Фурье несут обобщенную информацию о ней, высокого — более частную. При распознавании плоских фигур сопоставляют с эталонными моменты относительно координатных осей, начиная с моментов нулевого порядка (площади). Первые моменты дают данные о положении центра масс, вторые — о моментах инерции и т. д. Сходство двух образов оценивается по совпадению нескольких начальных моментов. В частности, этот подход можно применить к сравнению распределений случайных величин.

Рис. 1.1. Организация математического эксперимента

Порой удается построить достаточно простую модель, которая передает основные особенности явления. Например, аэродинамика первоначально строилась в предположении, что воздух является идеальной несжимаемой жидкостью. Такая модель среды действовала безотказно, пока скорость самолета не превышала 200 км/час. При ее увеличении потребовались поправки на сжимаемость воздуха. Принципиальные изменения пришлось внести для сверхзвуковых скоростей. Наконец, актуальная модель гиперзвуковой аэродинамики учитывает сложные физико-химические процессы в ближайшей окрестности летательного аппарата и на его поверхности.

Другой пример исключительно простой и эффективной модели — ньютоновская теория тяготения, созданная для построения картины солнечной системы (небесные тела рассматривались как материальные точки). Она позволила определить массы планет и их расстояния от Солнца и, более того, предсказать существование и открыть две планеты — Нептун и Плутон. Однако она потребовала уточнений для движения Меркурия (учет релятивистских эффектов) и для низкоорбитальных спутников и ракет (нецентральность поля тяготения Земли и его локальные искажения, моделируемые системой дополнительных точечных масс).

Рассмотрим, наконец, математические модели оптических явлений. Геометрическая оптика позволила вывести законы отражения и преломления света, волновая помогла объяснить и рассчитывать эффекты интерференции и дифракции, квантовая стала основой расчета лазеров.

При разработке новой системы для начала строится и оптимизируется стандартными методами простейшая модель, учитывающая важнейшие факторы. При этом упрощается структура изолиний уровней значений целевой функции, оценивающей качество исследуемого процесса, а вычисление каждого значения функции становится менее трудоемким. Затем модель постепенно усложняется за счет учета все новых и новых факторов и ставятся задачи оптимизации по все большему числу параметров. Решение каждой из них обычно оказывается хорошим начальным приближением для следующей.

Точная модель как правило имеет бóльшую непосредственную ценность, но в узкой области применения. С другой стороны, исключение из обобщенной модели второстепенных факторов часто заставляет ставить вопрос о границах применимости полученных результатов. Сравниваемые модели должны быть сопоставимы — например, должны работать

при одних и тех же внешних условиях. Модель выбирают минимальной сложности при заданной точности либо максимальной точности — при заданной сложности. Понятию сложности при этом придается весьма широкий смысл:

- количество внутрисистемных связей;
- длина программы моделирования;
- ее сложность по Холстеду [72], и т. п.

Принцип *баланса точности* требует соизмеримости погрешностей, вызываемых различными причинами: неполным соответствием модели объекту, неточностью задания исходных параметров модели, случайным характером результатов моделирования.

При исследовании сложных систем может потребоваться разработка набора моделей, соответствующих различным иерархическим уровням рассмотрения и функциональным разрезам деятельности системы. Такое стратифицированное описание на каждом уровне использует свой набор концепций, понятий и терминов. Разработка иерархии моделей для предотвращения пропусков и бросовых затрат должна вестись «сверху вниз».

В сравнении с натурным экспериментом математическое моделирование имеет следующие преимущества:

1. Экономичность (в частности, сбережение ресурса реальной системы).
2. Возможность моделирования гипотетических (то есть не реализованных в натуре) объектов и процессов.
3. Возможность реализации режимов, опасных или трудновоспроизводимых в натуре (критический режим ядерного реактора, работа системы ПРО).
4. Возможность изменения масштаба времени.
5. Легкость многоаспектного анализа.
6. Сжатие информации и ее единообразное представление, способствующие усмотрению основных закономерностей.

7. Отсутствие погрешностей измерения, которые в реальных условиях могут замаскировать искомый эффект.

8. Наличие строго сформулированных правил, позволяющее:

   - вскрыть ложность некоторых предубеждений;
   - заменить содержательные рассуждения формальным преобразованием выражений — в частности, выполнить оптимизацию;
   - выявить глубинные (сущностные) свойства и отношения и как следствие получить бóльшую прогностическую силу и построить разнообразные аналогии.

9. Универсальность технического обеспечения проводимой работы (ЭЦВМ, системы программирования и пакеты прикладных программ широкого назначения).

Средством перевода физических моделей в математические и обратно служит *теория подобия*. Функциональная зависимость между характеризующими процесс величинами отражается зависимостью между составленными из них критериями подобия. Рассматриваемая величина может быть принята в качестве переменной при исследовании системы, если с ее использованием удается составить безразмерные комплексы. В противном случае она исключается из рассмотрения или дополняется другими величинами.

Математическое моделирование на определенных этапах исследования может сочетаться с физическим. Классическим примером такого подхода является исследование динамики летательного аппарата (самолета, ракеты, спутника) на комплексе из математической модели самого аппарата, воспроизводимой на аналоговой или цифровой ЭВМ, и макета реальной аппаратуры управления. Весьма поучительный пример комбинированного моделирования с использованием критериев подобия приводится в воспоминаниях А.Н. Крылова о его службе в опытовом бассейне [37, с. 133]: «Я приказал вычисление сопротивления (воды — движению судна. — Ю.Р.) делать следующим образом. Из наблюденного при работе с моделью сопротивления вычесть рассчитанное сопротивление от трения: получится остаточное сопротивление, которое умножается на куб масштаба и дает остаточное сопротивление корабля. К этому остаточному сопротивлению прибавляется рассчитанное сопротивление

от трения на корабль — получится полное сопротивление корабля при скорости его, соответствующей скорости модели».

Практическое использование модели возможно лишь после тщательного ее исследования и настройки, в процессе которых необходимо решить задачи проверки адекватности, идентификации параметров, оценки значимости параметров и структурного преобразования.

Адекватность модели устанавливается проверкой для нее основных законов предметной области типа законов сохранения и сопоставлением результатов моделирования частных вариантов с известными для этих вариантов аналитическими решениями.

Задачей *идентификации* является определение значений рабочих параметров модели по наблюдениям реальной системы. При этом тип модели предполагается известным. Эта задача обычно ставится в форме минимизации функционала отклонения траектории модели от траекторий исследуемой системы. На практике для ее решения традиционно применяются методы наименьших квадратов и наибольшего правдоподобия.

Выявление *значимых параметров* (и пренебрежение остальными) позволяет уменьшить размерность пространства параметров. Оно базируется на определении коэффициентов чувствительности выходных показателей по отношению к вариациям входных параметров и представляет значительную ценность при решении задач оптимизации. Техника планирования и обработки результатов таких исследований составляет предмет теории экспериментов.

Задачи *преобразования модели* обычно решаются после накопления некоторого опыта работы с моделью. К ним относятся: упрощение, усложнение, структурное изменение. Такие преобразования могут быть как эквивалентными (проводимыми с целью сокращения числа состояний системы, количества связей и т. п.), так и неэквивалентными (существенное огрубление закономерностей, исключение или введение дополнительных факторов). Важнейшим способом упрощения моделей является использование методов теории подобия — в частности, переход от исходных параметров к меньшему числу обобщенных. Изменение модели отражается как на потребных вычислительных ресурсах, так и на возможности получения тех или иных характеристик.

Математическому моделированию в любой его форме предшествует создание *содержательной* (концептуальной) модели, определяющей

объект, цель и условия моделирования.

Возможность исследования того или иного явления реального мира некоторым математическим методом определяется формальными свойствами построенной модели. Познание разделяется на два этапа: формирование математической модели и получение логических следствий из нее. Оба этапа имеют свои специфические сложности. Составление модели требует проникновения в природу вещей. Второй этап требует умения найти и использовать нужный аппарат. На последний часто наталкивает рассмотрение частных случаев, в том числе результатов численных расчетов. Это прежде всего отбор влияющих факторов, а также предельное поведение параметров (например, в задачах теории надежности — быстрое восстановление, пуассоновский поток отказов).

Разработка формальных моделей начинается с простейших подходов, при которых учитываются лишь наиболее существенные характеристики исследуемых процессов, важнейшие законы сохранения и базовые вариационные принципы типа принципа Гамильтона, определяющие направление развития процессов. Необходимая степень уточнения достигается последовательным усложнением модели с целью ее приближения к реальности. Каждое такое усложнение должно быть достаточно значимым. В [47] рекомендуется минимум удвоение сложности модели, измеряемой, например, по Холстеду [72] — по аналогии с принципом Рунге при вычислении определенного интеграла с автоматическим удвоением числа шагов. Быстрая и полная проверка выдвигаемых гипотез (например, правомерности облегчающих аналитический расчет упрощающих предположений) должна быть проведена в ходе специально поставленного эксперимента.

Завершающим этапом математического моделирования обычно является оптимизация, результаты которой затем переносятся на моделируемую систему.

Математическое моделирование можно разделить на аналитическое, имитационное и комбинированное.

При *аналитическом* моделировании процессы функционирования элементов системы записываются в виде алгебраических, интегральных, дифференциальных, конечно-разностных и иных соотношений и логических условий. Результаты исследования аналитической модели могут быть:

- качественными (установлены такие свойства решения, как существование, единственность, устойчивость в большом и в малом, характер зависимости выходных параметров от входных и т. д.);
- аналитическими — получены явные зависимости для искомых характеристик;
- численными (получены числовые значения ответных величин для заданных входных данных).

Два первых подхода предполагают сведение задачи к стандартной математической постановке — решению системы линейных алгебраических, интегральных или дифференциальных уравнений и т. п. Применению аналитических методов мешают физические ограничения, нелинейности, запаздывания. Многие аналитические результаты выражаются через серии сложных интегралов, специальные функции. Поэтому модель обычно должна быть аппроксимирована или абстрагирована.

Качественные результаты полезны и необходимы, но не дают конкретного решения практической задачи. Примером качественных методов анализа являются общая теория дифференциальных уравнений и аппарат сетей Петри [46], см. также [17]. Теоремы существования дают уверенность в правильности постановки, подсказывают качественные свойства решения, что крайне важно при выборе алгоритма. Наличие известных частных решений позволяет производить проверку точности предлагаемых методов и облегчить численное решение более сложных задач. Вместо численного решения уравнений, описывающих движение самолета, построения графика и выявления по картинке колебательности движения квалифицированный математик может вывести общее соотношение между параметрами, которое порождает колебательность, не перебирая частные случаи.

Примером *аналитического* решения в интересующей нас области может служить элементарно выводимая (см. [61]) формула Полячека—Хинчина для среднего времени ожидания начала обслуживания в одноканальной системе:

$$w = \frac{\lambda b_2}{2(1 - \lambda b_1)}.$$

Здесь

λ — интенсивность простейшего потока заявок,

b_1 — среднее время обслуживания,

$b_2 = b_1^2 + D$ — второй момент распределения длительности обслуживания (D — дисперсия).

Из этой формулы мгновенно следует, что:

- при приближении коэффициента загрузки системы $\rho = \lambda b_1$ к единице среднее время ожидания стремится к бесконечности;

- при постоянной длительности обслуживания ($D = 0$) и прочих равных условиях среднее время ожидания минимально и ровно вдвое меньше, чем при показательно распределенной длительности обслуживания ($D = b_1^2$).

Для получения этих следствий из численных расчетов потребовались бы немалые усилия.

Аналитическое решение предпочтительно всегда, но его обычно удается получить лишь после ряда упрощающих предположений. Возможность его достижения очень критична к изменениям модели, для нахождения решения требуются высокая квалификация и значительные творческие усилия разработчика.

Гораздо чаще удается применить *численные* методы. Здесь творческий элемент при наличии хорошей библиотеки подпрограмм может быть сведен к разумному минимуму. Однако промежуточные данные, полученные в процессе длительных выкладок или расчетов, обычно не имеют наглядной физической интерпретации. Это затрудняет как выявление первопричин специфического поведения системы, так и смысловой контроль вычислений. Основные черты исследуемого варианта системы, легко обнаруживаемые непродолжительным, а иногда единичным экспериментом, при аналитико-числовом подходе выявляются лишь по завершении всех вычислений.

Численные методы дают частные результаты, по которым трудно делать обобщающие выводы. При оптимизации модели необходим многовариантный счет. Аналитические и численные методы расчета СМО описаны в [15, 44, 55, 61].

При *имитационном* моделировании реализующий модель алгоритм воспроизводит процесс функционирования системы во времени

и пространстве, причем имитируются составляющие процесс элементарные явления с сохранением его логической и временной структуры. Результаты каждого шага моделирования могут интерпретироваться как состояние системы в определенный момент времени, а метод может быть определен как наблюдение во времени за характеристиками динамической модели системы. Это роднит имитационное моделирование с физическим экспериментом. Другим примером являются кросс-системы автоматизации программирования — прогон программ для несуществующей ЭВМ на имитаторе ее команд.

Имитационное моделирование, линейное программирование и регрессионный анализ по диапазону и частоте использования давно и устойчиво занимают три первых места среди всех методов исследования операций [74, с. 7]. Теория, техника и системы автоматизации имитационного моделирования разрабатывались прежде всего для моделирования систем с очередями. Об этом свидетельствуют как содержание первой отечественной монографии по моделированию[1], так и первая глава книги признанного классика *GPSS* Т.Дж. Шрайбера [74] — «Вопросы моделирования систем массового обслуживания».

Одной из первых областей применения имитационного моделирования явилось *управление запасами* [61], что вполне объясняется сложностью вероятностных задач этого вида и их практической важностью. Здесь следует упомянуть работы Робинсона (1957) об иерархической системе складов нефтепродуктов, Бермана (1961) о перераспределении запасов, Джислера (1964) о снабжении авиационных баз. Практически все прикладные аспекты недавно переведенной книги Кельтона и Лоу [98] связаны с задачами управления запасами и теории очередей.

Чрезвычайно полезно и перспективно *комбинированное* использование аналитических и имитационных методов, позволяющее сочетать достоинства обоих подходов.

Целью данной книги являются изложение, развитие и пропаганда новых идей и технологий в имитационном моделировании применительно к системам и сетям массового обслуживания.

---

[1] *Н.П. Бусленко, Ю.А. Шрейдер*. Метод статистических испытаний и его реализация на электронных цифровых машинах. — М.: Наука, 1962.

1.2 Общая характеристика имитационного моделирования СМО

Аналитическими методами может быть исследован сравнительно узкий круг задач массового обслуживания. С другой стороны, уникальность и дороговизна многих реальных систем, а также ответственность решаемых ими задач мешают проведению натурных экспериментов — особенно в критических режимах.

Имитационное моделирование в принципе позволяет воспроизводить любой *марковский* процесс в целях анализа, оптимизации или обучения персонала системы. Естественными требованиями к модели являются:

- полнота (получаются все необходимые характеристики, причем с требуемой точностью и достоверностью);
- гибкость (допустимо варьирование структуры и параметров);
- минимальная длительность разработки;
- блочная структура;
- эффективность воспроизведения на ЭЦВМ.

Считается [98] что имитационное моделирование особенно эффективно в целях:

- проектирования и анализа производственных систем;
- оценки различных видов вооружения и требований к их материально-техническому обеспечению;
- определения требований к оборудованию, программному обеспечению, протоколам сетей связи различных компьютерных систем;
- проектирования и анализа работы транспортных систем (аэропортов, морских портов, автомагистралей, метрополитена);
- оценки проектов создания различных организаций массового обслуживания (центров обработки заказов, заведений быстрого питания, больниц, отделений связи);

- модернизации различных проектов в деловой сфере;
- определения стратегий управления запасами;
- анализа финансовых и экономических систем.

При этом ожидаются:

- увеличение производительности;
- сокращение производственного цикла;
- уменьшение производственных запасов;
- сокращение оборотного капитала и производственных расходов;
- гарантия желаемого функционирования;
- достижение лучшего понимания — уже в процессе сбора информации для моделирования;
- выявление потенциальных проблем.

Растущий масштаб применения имитационного моделирования определяется как остротой потребностей в нем (ужесточение конкуренции; рост стоимости объектов исследования — завод по производству микроэлектроники может стоить миллиард долл.), так и облегчением разработки и применения моделей (рост быстродействия и удешевление компьютеров и создание программного инструментария, в том числе для визуализации моделей).

Система моделируется путем прослеживания характерных событий в модельном времени, для каждого из которых создается ряд программ обработки событий. Эти программы подробно описывают изменения состояний, происходящие при наступлении каждого события. Развитие моделируемой системы во времени осуществляется путем выполнения программ обработки событий в порядке возрастания времени их возникновения, причем внутри подпрограмм модельное время не продвигается.

Процесс — это упорядоченная во времени последовательность взаимосвязанных и разделенных промежутками времени событий, отражающая полный «жизненный путь» объекта в системе.

Реализации процесса моделируются на ЭЦВМ с помощью серий случайных или псевдослучайных величин. *Управление* работой модели

строится в зависимости от цели исследования и может быть организовано по текущему значению счетчика, связанного с этой целью. В более сложных моделях останов может определяться некоторой комбинацией условий, а также достижением требуемой точности (подробнее см. разд. 2.1). *Усреднение* результатов моделирования по времени функционирования модели или числу реализаций процесса позволяет методами математической статистики получить искомые характеристики. Эту обработку также следует проводить на ЭЦВМ.

Отметим, что существуют задачи, при решении которых также используются случайные числа, но к *имитационному* моделированию не относящиеся. Характерным их образцом и первым применением явилось исследование процесса размножения и поглощения нейтронов в массе обогащенного урана, выполненное под руководством Дж. фон Неймана в рамках Манхэттенского проекта. Этот метод (*Монте-Карло*) не вытекает из метода имитационного моделирования, не включает в себя этот метод и не включается в него, хотя и имеет с ним много общего.

Благодаря возможности достаточно полного отражения реальности (например, многокаскадного обслуживания, неоднородных потоков и каналов, блокировок из-за ограниченной емкости буферов, сложной системы приоритетов и дисциплин обслуживания и т. п.) имитационное моделирование удобно для исследования практических задач: определения показателей эффективности, сравнения вариантов построения и алгоритмов функционирования системы, проверки устойчивости режимов системы при малых отклонениях входных переменных от расчетных значений и др. Полнота имитации может быть проверена построением серии последовательно уточняемых моделей. Если дальнейшая детализация практически не влияет на интересующие нас показатели, расчеты можно прекратить и принять в качестве выходных последние полученные результаты.

В любом случае моделируются все те и только те стороны процесса, которые влияют на выбранный показатель эффективности или критичны к наложенным ограничениям. В частности, заявка может характеризоваться не только временем ее обслуживания, но и своим объемом (весом, температурой), видом обслуживания, необходимыми затратами энергии и материалов и т. п.

Между длительностью моделируемого процесса и длительностью имитации нет прямой связи. Длительность имитации определяется уровнем детальности и соотношением характерных времен. К примеру,

моделирование работы пользователя вычислительной системы коллективного доступа, выдающего запросы с 15-минутными интервалами, трудно сочетается с моделированием регистров процессора ЭВМ, работающего в наносекундном диапазоне. Эта ситуация аналогична решению «жестких» систем дифференциальных уравнений. Проблематика жестких систем стимулировалась космонавтикой (например, полет космического аппарата к Юпитеру с учетом вибраций конструкции и притяжения планетами солнечной системы). Второй пример — процесс накопления плутония в костях работника радиохимической промышленности (константа на шесть порядков меньше, чем у обычных обменных процессов в организме) с учетом суточного и годового ритмов.

Соотношение характерных времен практически должно быть не более 100:1, а в упомянутом примере системы коллективного доступа превышает 10 порядков. Здесь модель завязнет в деталях. В подобных случаях детальное моделирование нужно выполнять отдельно и полученные средние значения подставлять в модели более высокого уровня как константы или переменные, зависящие от «медленных» параметров.

Промежуточные результаты имитационного моделирования, в отличие от результатов аналитического счета, имеют четкий физический смысл. Это облегчает обнаружение ошибок в программе — в особенности при работе в интерактивном режиме.

Метод имитационного моделирования свободен от каких-либо ограничений на класс решаемых задач, нагляден, прост в освоении, обладает повышенной устойчивостью к сбоям ЭЦВМ; все промежуточные результаты легко интерпретируются и контролируются.

Недостатками имитационного моделирования являются:

- большой расход машинного времени;
- малая точность вероятностных характеристик редких событий;
- трудность получения обобщающих выводов и рекомендаций;
- сложность оптимизации системы (поиск оптимума требует многовариантных расчетов и ведется при наличии вероятностных помех — случайных ошибок в результатах);
- вероятностная оценка погрешностей (впрочем, для вероятностных задач такая оценка вполне естественна).

Перечисленные недостатки имитационного моделирования несколько смягчаются постоянным ростом технических характеристик (быстродействия и объема памяти) современных ЭВМ и использованием методов, описанных в главе 8.

По указанным причинам применение имитационного моделирования (особенно в *квалификационных* работах) нужно сводить к разумному минимуму. Такое применение представляется целесообразным:

- для накопления первичных данных об изучаемом явлении, если эти данные нельзя получить в натурном эксперименте;
- для проверки правомерности допущений, сделанных разработчиком в целях перехода к аналитическим методам;
- для демонстрации конечных результатов исследования на достаточно полной модели реальной ситуации;
- при «безысходности», когда сложность ситуации намного превосходит возможности аналитических методов, известных разработчику.

1.3 Организация моделирования

В типичном случае процесс моделирования проходит следующие фазы [73]:

1. Определение системы.
2. Формализация описания.
3. Подготовка данных.
4. Трансляция модели.
5. Оценка адекватности.
6. Планирование эксперимента.
7. Планирование прогонов.
8. Машинный эксперимент.
9. Анализ результатов.

10. Интерпретация.

11. Реализация.

12. Документирование.

Перечисленные этапы в значительной степени перекрываются по времени (например, документирование должно вестись с первых дней работы над проектом) и в зависимости от результатов этапов охвачены многочисленными обратными связями.

Организация является важным элементом работы вследствие значительной трудоемкости последней. Согласно [73], трудоемкость разработки даже простых моделей в США оценивается в 5–6 человекомесяцев (30 тыс. долларов), а сложных — на два порядка больше. В организацию входят:

- установление связей с заказчиком;
- определение задач;
- определение ресурсов (включая состав исполнителей);
- установление взаимодействия, отчетности и контроля;
- планирование работы.

Следует подчеркнуть особую важность конструктивного взаимодействия с заказчиком: это ключ к достаточной реалистичности модели и успешному внедрению результатов ее исследования. В частности, весьма желательна работа над моделью в привычных для заказчика категориях.

Определение системы включает уточнение ее границ с внешней средой; характеристики среды и внешних воздействий; изучение состава, назначения, внешних и внутренних связей; выявление ограничений и выбор показателей эффективности; постановку задачи на исследование. Описание представляется в виде схем, текстов, формул, таблиц экспериментальных данных.

На этапе *формализации* строится математическая модель системы: устанавливаются ее структура и существенные зависимости между элементами. Здесь особенно важно выбрать модель минимально необходимой сложности. При большой сложности системы создается совокупность моделей по функциональному и/или иерархическому признаку.

Тогда показатели работы подсистем должны выводиться из целей системы в целом (см. [47, с. 219–222]).

Подготовка исходных данных состоит в сборе и обработке результатов наблюдений за моделируемой системой. Обработка в типичном случае заключается в построении функций распределения соответствующих случайных величин или вычислении числовых характеристик распределений. Данные могут быть использованы и в необработанном виде (например, результаты трассировки вычислительного процесса на реальной установке — при прогоне ее имитационной модели); однако в этом случае модель будет иметь ограниченную прогностическую силу.

К подготовке исходных данных следует отнести и сбор информации о предполагаемых изменениях в нагрузке системы (о прогнозируемой нагрузке — для проектируемой системы).

Трансляция модели — это запись ее на одном из языков программирования (общецелевом или специализированном). Следует стремиться к блочному (модульному) построению программы, позволяющему независимо вносить изменения в отдельные модули и повысить производительность программирования путем повторного использования ранее созданных модулей.

Нужно иметь в виду, что при простом объединении для анализа системы в целом полных моделей подсистем нижних уровней возникает диспропорция между требуемой точностью и фактической сложностью модели. Эта диспропорция может быть устранена загрублением моделей низшего уровня (после детального автономного исследования их). Возможными вариантами такого загрубления являются [13]:

- укрупнение состояний и фаз процессов;
- аппроксимация выявленных зависимостей (например, кусочно-линейное представление функций распределения);
- усреднение характеристик процессов по их аргументам;
- снижение требований к точности итераций;
- замораживание медленно меняющихся параметров;
- пренебрежение взаимной зависимостью переменных;
- замена нескольких обслуживающих устройств одним с суммарной производительностью;

- сведение детальных описаний многокомпонентного процесса к главной составляющей с поправочными коэффициентами.

Оценка адекватности модели заключается в проверке:

- полноты учета основных факторов и ограничений, влияющих на работу системы;

- соответствия исходных данных модели реальным (в частности, согласия постулируемых законов распределения с первичными данными);

- наличия в модели всех данных, необходимых для накопления ответных величин;

- синтаксической корректности программы моделирования;

- правильности преобразования исходных данных в конечные результаты (например, при условиях, приводящих к известному аналитическому решению);

- осмысленности результатов при нормальных условиях (поступательный ход модельного времени, отсутствие переполнения буферов и счетчиков, допустимый процент отказов в обслуживании) и в предельных случаях (пробные прогоны в условиях перегрузки системы обслуживания или работа с распределением Коши, не имеющим моментов).

Планирование экспериментов определяет совокупность исследуемых вариантов (в частности, наборов исходных данных) и/или стратегию их перебора. Здесь учитываются прежде всего:

- цель проекта (анализ или оптимизация);

- степень достоверности исходных данных (при малой достоверности необходимы дополнительные исследования чувствительности модели к вариациям параметров);

- ресурсы календарного и машинного времени.

При планировании работы с имитационной моделью полезно применение методов общей теории планирования экспериментов (см. главу 7).

Планирование прогонов имеет целью получить для фиксированной точки пространства варьируемых параметров возможно лучшие статистические оценки показателей эффективности — несмещенные (в крайнем случае, асимптотически несмещенные) и с минимальной дисперсией — при определенном объеме вычислительной работы. Может быть поставлена и обратная задача — получить оценки с заданной дисперсией при минимальном объеме работы. Отдельным прогоном (репликой) считается часть процесса имитации, в котором системное время монотонно возрастает.

Решение этих задач связано с проблемами:

- начальных условий,
- уменьшения или исключения корреляции результатов,
- уменьшения дисперсии результатов (см. главу 8),
- критерия останова прогона.

Данный этап имеет обратную связь к этапу трансляции (способы решения перечисленных проблем должны быть непосредственно отражены в программе модели) и к этапу планирования экспериментов (число возможных экспериментов при ограниченном машинном времени обратно пропорционально трудоемкости одного прогона).

Моделирование как правило ориентировано на получение *стационарных* характеристик. В связи с этим первостепенное значение приобретают вопросы о длительности разгонного участка и стационарного режима. Определение времени вхождения в стационарный режим выполняется экспериментально. После этого накопленная статистика сбрасывается, и моделирование продолжается с достигнутого к концу разгонного участка состояния системы. Такая технология предпочтительнее, чем длительное моделирование без сброса — см. [74, с. 125].

Даже при функционировании в стационарном режиме происходят статистические флуктуации измеряемых показателей. Если время работы в стационарном режиме мало, они могут оказаться недостоверными. После сброса статистики разгонного участка для оценки стабильности результатов следует периодически выдавать накопленную с начала стационарного режима стандартную статистику. Установив необходимую и

достаточную его длительность, можно для рабочего прогона запросить дополнительную информацию.

Особая проблема — учет при моделировании редких событий (отказов) и временны́х ритмов. Вряд ли на этот вопрос можно дать общий ответ. Скорее всего,

- «разгонный» участок должен быть длиннее самого короткого периода (суток);
- требовать на нем наступления редких событий не стоит;
- на стационарном участке событие каждого из запланированных типов должно произойти минимум по одному разу — хотя бы для того, чтобы убедиться в правильности программирования его обработки и в том, что система справляется с его последствиями (часто требуется, чтобы при одиночном отказе с длительным восстановлением ни один узел системы не оказался перегруженным).

Подробнее о редких событиях см. разд. 8.8.

Этапы *машинного эксперимента* и *анализа результатов* в дополнительных пояснениях не нуждаются.

Интерпретация результатов состоит в переносе их с модели на исследуемую (проектируемую) систему.

Реализация заключается в практическом осуществлении полученных рекомендаций и оценке их эффективности в процессе опытной эксплуатации.

На этих двух этапах тесное взаимодействие с заказчиком играет особую роль.

Тщательное и полное *документирование* процесса разработки модели и хода экспериментов дисциплинирует участников проекта; повышает вероятность его успешной реализации; обеспечивает накопление научно-технического опыта и обучение специалистов по моделированию; позволяет модифицировать модель в целях ее дальнейшего использования.

1.4 Достоверность модели

Рост масштабов и ответственности применений имитационных моделей породили повышенные требования к их корректности. Последняя не может быть добавлена к модели *после* ее разработки, но должна обеспечиваться на всем протяжении проекта. Здесь особую роль играют начальные этапы, допущенные на которых ошибки исправляются с наибольшим трудом.

1.4.1 Валидация

Концептуальная модель включает в себя [98]:

- обзорный отдел (общие цели проекта, конкретные проблемы, целевые показатели);
- предварительное описание подсистем и их взаимодействия;
- сделанные упрощения и их обоснование;
- располагаемые числовые данные и распределения;
- источники информации; оценка информации, в том числе по важности и непротиворечивости.

В процессе разработки концепции проводятся:

- регулярное взаимодействие с заказчиком (уточнение постановки задачи, апробация допущений, поддержание интереса к проекту и чувства сопричастности, содействие внедрению);
- консультации с различными специалистами (проектировщики, пользователи, наладчики, системные администраторы, менеджеры);
- наблюдение за системой (указать: что надо, единицы измерения, формат представления информации, требуемая точность, гарантия объективности);
- изучение опыта и инструментов решения подобных задач.

Особенно важно тщательно определить проблемы и оцениваемые показатели. В США военный аналитик в течение 6 мес. работал над моделью, не взаимодействуя с генералом, который подал на нее заявку. Генерал покинул брифинг по итогам исследования через 5 мин. после его начала, заявив, что это не та проблема, которая его интересует [98, с. 320].

Отчет о концептуальной модели должен быть понятен менеджерам. После рассылки его следует обсудить в кругу исполнителей, заказчиков и консультантов. Принятие положительного решения (с учетом поправок) рассматривается как *валидация* (признание ценности) концепции модели. Одним из оснований валидации может служить репутация разработчиков модели.

Вторым этапом валидации является обсуждение количественных аспектов представления исходных данных. Здесь могут приниматься решения по таким вопросам, как:

- стартовая детализация (детали, сборки, изделия, коробки);
- типы распределений случайных величин — по критериям согласия или по соотношениям между моментами распределений;
- объединение выборок — по критериям однородности;
- характер потоков событий (стационарность, рекуррентность, ординарность — из общетеоретических соображений, качественного или количественного анализа ситуации);
- аргумент потока отказов (календарное время, наработка, число обслуженных заявок);
- отбор учитываемых факторов — по критериям чувствительности (с применением методов планирования эксперимента);
- методика расчета «штрафов» в случае отказов моделируемой системы, нехватки запасных частей и т. п.;
- основания для расчета стоимости хранения материальных запасов и организации их восполнения.

По последнему пункту сошлемся на пример валидации модели для оценки влияния организации снабжения на боеготовность бомбардировщиков [98, с. 333].

Как валидацию *завершенного проекта в целом* можно рассматривать натурные испытания рекомендаций модели (в особо ответственных случаях) и проверку (со временем, для будущих исследований) ее прогностических возможностей

1.4.2 Верификация

Под верификацией модели понимается проверка соответствия программной реализации модели ее концепции. Эта проблема в основном решается обычными средствами повышения надежности программирования [56], что гарантируется наличием у разработчика сертификата соответствующего технологического уровня. Как дополнительное средство отметим визуализацию выходного процесса имитации, часто позволяющую вскрыть дефекты в постановке задачи (например, при имитации транспортных потоков — необходимость учета задержек на ликвидацию последствий дорожно-транспортных происшествий).

1.4.3 Аккредитация

Аккредитация модели заключается в официальном признании компетентным органом ее пригодности или полезности для некоторого класса ситуаций и соответственно — в рекомендации модели к широкому использованию.

При акккредитации учитываются: валидация и верификация модели; история ее разработки и использования; качество доступных данных; качество документации; возникавшие проблемы, известные и обнаруженные ограничения.

Глава 2

Принципы построения имитационных моделей

2.1 Элементы имитационной модели

Имитационная модель образуется взаимодействием следующих элементов:

- состояний,
- событий,
- датчиков случайных чисел,
- таймера,
- цепей событий,
- цели моделирования,
- счетчиков,
- блока инициализации,
- критерия остановки,
- методов обработки результатов.

Состояние системы должно быть определено со степенью детальности, необходимой и достаточной для вероятностного продолжения процесса моделирования (процесс должен быть сведен к марковскому). В частности, состояние системы массового обслуживания задается текущим числом заявок в ней, фазами текущего обслуживания (прибытия) и моментами наступления ближайших событий каждого вида.

Под *событием* модели понимается скачкообразное изменение ее состояния. События могут быть первичными (прибытие заявки, завершение обслуживания) и вторичными (по отношению к прибытию — прием заявки на обслуживание, продвижение очереди и т. п.), которые наступают как следствие первичных.

С помощью *датчиков случайных чисел* (ДСЧ) в модели формируются ее очередные состояния (моменты наступления следующих первичных событий каждого вида, объемы спроса на запасные части и т. д.). Случайные величины генерируются в соответствии с заданными распределениями.

Имитируемый процесс развивается в модельном (системном) времени. Счетчик модельного времени называется *таймером*. Наиболее сложные процессы моделируются упрощенно: с постоянным шагом по оси времени (имеется в виду «системное» время). При этом отдельные стороны процесса (прибытие заявок, обслуживание) моделируются «суммарно» — через распределение числа событий данного рода за выбранный интервал. Подобное построение моделирующего алгоритма используется также при моделировании так называемых непрерывных систем, то есть систем, описываемых дифференциальными уравнениями (обыкновенными или в частных производных), когда невозможно предварительное выделение узловых событий, определяющих динамику системы, и приходится отслеживать всю траекторию.

Следующим уровнем является схема «однофакторного слежения». В этом варианте состояние системы фиксируется только в моменты событий определенного рода (например, окончания обслуживания), а остальные параметры воспроизводятся суммарно — за случайные интервалы между событиями первого рода. Легко усмотреть аналогию данной схемы с аналитическим методом вложенных цепей Маркова — см. [15, 55, 61].

Для наших целей наибольший интерес представляют чисто дискретные модели, состояния которых меняются скачкообразно при наступлении *событий*. Функционирование любого процесса разбивается на этапы (активные фазы), каждый из которых соответствует некото-

рому событию и реализуется в один момент системного времени. Между смежными активными фазами находится пассивная, в которой с данным процессом ничего не происходит, но может произойти любое число событий других процессов. Событие может изменить значение атрибута, создать или уничтожить сущность, начать или прекратить активность. Моделирование требует программы, которая выстраивает последовательность событий в их взаимной зависимости.

Логика модели реализуется в процессе обработки *цепей событий*. В цепи *текущих* событий ЦТС находятся события, которые наступают в один момент модельного времени (уход из системы обслуженной заявки, продвижение очереди, выборка на обслуживание головной заявки очереди, формирование для нее момента завершения обслуживания). Ясно, что последовательность их обработки должна быть строго определенной. В цепи *будущих* событий ЦБС находятся события, запланированные посредством ДСЧ на последующие моменты системного времени (завершение обслуживания в других каналах, прибытие очередных заявок различных типов, поломка обслуживающего устройства, восстановление ранее отказавшего, уход из канала либо очереди нетерпеливой заявки и т. п.). В цепи *задержанных* событий находятся события, развитие которых заблокировано сложившимися в системе на данный момент модельного времени условиями (например, занятостью необходимых ресурсов). Могут использоваться и другие цепи событий.

Списки событий для облегчения вставок и удалений рекомендуются двунаправленные. Цепь может быть неупорядоченной или упорядоченной по моментам наступления соответствующих событий. К неупорядоченной цепи проще добавляется новое событие, но после каждого добавления или выборки требуется ее просмотр для поиска наиболее раннего события. Такой поиск имеет смысл начинать с конца цепи. При вставке очередного события в список, упорядоченный по моментам наступления событий, его целесообразно просматривать с конца или применять половинное деление. Наиболее эффективен поиск по методу двоичного дерева, реализуемый для упорядоченных таблиц.

Модели этого типа имеют наибольшую логическую сложность; зато они исключают из рассмотрения «пустые» интервалы времени, не ознаменованные никакими событиями. Это уменьшает затраты машинного времени на прогон модели.

Специализированные программные системы моделирования дискретных событий обычно содержат встроенные средства ведения цепей.

Поэтому они позволяют программировать модели в *потоковом* (процессном) стиле, описывая траектории прохождения заявок через устройства системы и указывая узловые точки для сбора статистики. При работе на языках общего назначения организация и обработка цепей событий обеспечивается программистом. Цепи будущих событий могут быть представлены табличными или списковыми структурами. Первый способ проще в реализации, но связан с избыточным расходом распределяемой памяти и необходимостью сдвигов при выборке или вставке элемента цепи. Второй способ свободен от указанных недостатков и особенно удобен в языках программирования типа *Паскаль* или *Фортран* 90, имеющих встроенные средства работы со списковыми структурами. Примеры подобных списков приводятся в [80]. Организация календаря событий подробно рассматривается в [38].

Под *инициализацией* понимается приведение модели до начала прогона в исходное состояние. Простейший аспект инициализации — обнуление всех накапливающих счетчиков. Для обеспечения воспроизводимости результатов может потребоваться установка в исходное состояние генераторов псевдослучайных чисел.

Цель моделирования при построении модели трактуется в узком смысле — как определение показателей качества функционирования системы. Выбор цели существенно влияет на структуру модели через счетчики, необходимые для накопления результатов моделирования (например, для подсчета среднего времени ожидания начала обслуживания нужно иметь счетчики суммарного времени ожидания и количества выбранных на обслуживание заявок).

Упомянутые показатели обычно связываются с некоторыми *стационарными* (установившимися при устремлении к бесконечности системного времени) характеристиками. Данные, накопленные за время переходного к такому режиму процесса, будут вносить погрешность в окончательные результаты. Инициализация модели, претендующей на повышенную точность, должна либо сразу задать начальные условия, близкие к стационарному режиму, либо обеспечить отбрасывание статистики переходного процесса. Обычно начальный участок отбрасывают с неконтролируемым «запасом», что ведет к непроизводительным потерям машинного времени. При необходимости нескольких туров (а такая необходимость возникает почти всегда) это отбрасывание становится особенно накладным. В таких случаях для *эргодических* процессов, стационарный режим которых не зависит от начальных условий, в качестве туров

рассматриваются отрезки одного длинного прогона.

Критерий останова определяет момент прекращения прогона модели. В простейшем случае прогон прекращается по достижению заданного значения таймера, счетчика числа обслуженных заявок и т. п. Однако правильнее управлять прогоном по достижению заданной точности одного из определяемых показателей. Обоснованный выбор правила остановки моделирования нетривиален, так как на этапе планирования эксперимента ни оценка определяемой величины, ни тем более ее дисперсия как правило не известны. Здесь возможны следующие варианты:

1. двухэтапный прогон, когда на первом этапе грубо определяются величины, необходимые для формирования критерия остановки на втором этапе;

2. рекуррентная обработка результатов моделирования;

3. метод последовательного анализа (в смысле Вальда [14]).

Обработка результатов моделирования состоит в сжатии получаемой информации, вычислении статистических (точечных и интервальных) оценок типа математических ожиданий и высших моментов для искомых показателей, оценке статистической значимости различия средних, построении гистограмм и статистических функций распределения. Другим аспектом обработки результатов является выдача их на печать в компактной и удобной для дальнейшего анализа форме. Соответственно в *систему моделирования* должны включаться необходимые подпрограммы — как правило широкого применения, слабо связанные со спецификой модели.

2.2 Базовый алгоритм моделирования

На рис. 2.1 дается детальная структурная схема имитационной модели СМО типа $GI/G/n/R$, с помощью которой определяются стационарное распределение $p[0:R]$ числа заявок в системе и начальные моменты распределения времени ожидания. В этой модели используются два класса событий: прибытие заявки и завершение обслуживания. Поток заявок моделируется как рекуррентный (момент прибытия очередной заявки получаем добавлением случайного интервала к предыдущему).

Рис. 2.1. Структурная схема имитационной модели

Моменты освобождения каналов получаем добавлением к текущему моменту случайной длительности обслуживания. Упомянутые интервалы формируются посредством датчиков псевдослучайных чисел, настроенных на требуемые законы распределения.

Управление прогоном производится по числу обслуженных заявок. Схема рассчитана на выполнение 10 прогонов с выдачей результатов в конце каждого из них.

Приведем словарь этой модели:

`tt`	—	таймер;
`told`	—	момент предыдущего события;
`tlib[1:n]`	—	моменты завершения текущего обслуживания
`ttlib`	—	наиболее ранний из них;
$\tau[n+1:R]$	—	моменты прибытия заявок очереди
`tz`	—	момент прибытия очередной заявки;
`p[0:R]`	—	счетчики стационарных вероятностей;
`r`	—	текущее число заявок в системе;
`nz`	—	общее число выбранных на обслуживание заявок (ограничено 5000);
`n1`	—	число заявок в данной серии (ограничено 500)
`nref`	—	количество отказов в обслуживании
`s[1:4]`	—	счетчики для моментов распределения времени ожидания начала обслуживания;
`w[1:4]`	—	искомые моменты.

Работа модели начинается с установки в нуль таймера (счетчика модельного времени) `tt`, момента предыдущего события `told` и всех остальных накапливающих счетчиков. Моменты {`tlib[i]`} выхода из системы обслуженных заявок полагаем равными бесконечности (точнее — числу, заведомо превышающему предельное значение таймера в процессе прогона). Далее формируется случайный момент `tz` прибытия заявки и выбираются `ttlib` (минимальный из моментов освобождения каналов {`tlib[i]`}) и индекс минимума `j`.

Вариант обработки текущего события определяется соотношением между `tz` и `ttlib`. Если `tz` $<$ `ttlib`, то ближайшим событием в системе является прибытие заявки. Соответственно продвигается таймер `tt`, к счетчику `p[r]` времени пребывания системы в состоянии с `r` заявками добавляется разность `tt-told` и обновляется значение `told`. Если новая заявка застает в системе предельное число `R` ранее прибывших, то к счетчику отказов `nref` добавляется единица. Затем формируется момент `tz`

прибытия новой заявки. При наличии в системе свободного места r увеличивается на единицу и проверяется, есть ли свободные каналы. Если r $>$ n, то их нет, для прибывшей заявки запоминается момент $\tau[r]$ прибытия и формируется очередная заявка. В обоих рассмотренных случаях мы выходим на метку B — сравнение ttlib с обновленным tz.

При r $\leq$ n в системе имеется хотя бы один свободный канал. Тогда определяется его номер m; в элементе массива tlib[m] формируется момент освобождения m-го канала; корректируются счетчики числа случаев ожидания n1 и nz (длительность ожидания в данном случае нулевая, и соответствующие счетчики времени ожидания не изменяются); опять же формируется момент прибытия новой заявки; при n1 $<$ 500 происходит возврат к выбору $\min_i$ {tlib[i]}, поскольку один из элементов этого массива обновляется.

В случае tz $>$ ttlib очередным событием оказывается завершение начатого обслуживания. Здесь также продвигается таймер, пересчитывается счетчик p[r] времени пребывания в состоянии r, обновляется момент последнего изменения состояния told. Затем число r заявок в системе *уменьшается* на единицу. Если r $<$ n, то очереди в системе нет, в массив tlib для освободившегося j-го канала записывается «бесконечность» и выполняется переход на выбор минимальной компоненты массива tlib.

При r $\geq$ n в системе существует очередь, так что в освободившийся канал будет выбрана на обслуживание новая заявка. Соответственно увеличиваются на единицу счетчики n1 и nz случаев выбора на обслуживание, вычисляется время x ожидания начала обслуживания головной заявкой очереди. Далее этот x и его последовательные степени добавляются к счетчикам {s[i]}, накапливающим данные для расчета моментов распределения времени ожидания. Затем продвигается очередь (точнее, записанные в ней моменты $\{\tau[i]\}$ прибытия в систему ожидающих заявок) и формируется время tlib[j] освобождения вновь занимаемого j-го канала.

При n1 $<$ 500 модель возвращается на выбор $\min_i$ {tlib[i]}. При n1=500 прогон серии заканчивается, счетчик n1 обнуляется, рассчитываются и выводятся на печать оценки {w[i]} моментов распределения времени ожидания вместе с текущим значением nz. Указанные промежуточные выдачи нужны для контроля темпа прохождения модели (чтобы рассчитать потребное машинное время в случае снятия задачи до полно-

го завершения моделирования), а также для оценки влияния на результаты начальных условий и выполненного числа испытаний. После достижения nz=5000 на печать выдаются стационарные вероятности {p[r]}, определяемые как доля общего времени tt, в течение которого в системе находилось ровно r заявок, а также количество nref отказов в приеме на обслуживание. На этом моделирование завершается.

Ниже приведена программа, реализующая эту модель на языке *Фортран* 90. Внешние процедуры со стандартными именами FLIB и FTZ служат для настройки модели на распределения длительности обслуживания и интервалов между смежными заявками соответственно (подробнее см. главы 3–4). В данном случае это вырожденное распределение (регулярное время обслуживания) и экспоненциальное.

```
program testmod
   interface
      subroutine model(n,rr,p,pi,kmax,theta,zmax)
         integer n,rr,kmax,zmax
         real p(0:rr),pi(0:rr),theta(kmax)
      end subroutine model
   end interface
   integer n,rr,kmax,zmax
   real p(0:19),pi(0:19),theta(3),s
   data n,rr,kmax/2,19,3/,zmax/200000/
   open(7,file='d:\fps4\projects\resmod.doc')
   call model(n,rr,p,pi,kmax,theta,zmax)
   write(7,*) 'p'
   write(7, *)  p
   write(7,*) 'pi'
   write(7,*)  pi
   write(7,*) 'theta'
   write(7,*)  theta
   s=0
   do i=n+1,rr
      s=s+(i-n)*p(i)
   end do
   write(7,*) 'Средняя длина очереди = ',s
end program testmod
```

```
subroutine model(n,rr,p,pi,kmax,theta,zmax)
!*****************************************
!*  Имитационная модель системы GI/G/n/R    *
!*****************************************

   integer    n     ! Число каналов обслуживания
   integer    rr    ! Предельное число заявок в системе
   real       p     ! Стационарные вероятности состояний
   real       pi    ! Вероятности перед прибытием заявки
   integer    kmax  ! Предельное к-во вычисляемых моментов
   real       theta ! Моменты распределения времени ожидания
   integer    zmax  ! Число обслуженных заявок
   dimension  p(0:rr),pi(0:rr),theta(kmax)

   interface
      function ftz(tt)
         real tt
      end function ftz
      function flib(tt)
         real tt
      end function flib
   end interface

   integer           i,j,m,n1,n2,nref,ns,nz,r
   real              b,flib,ftz,told,tt,ttlib,tz
   real, allocatable, dimension (:) :: s,tau,tlib

   allocate (s(kmax),tau(n+1:rr),tlib(n))
   ttlib=1e15
   data n1,ns,nz,nref /4*0/
   n2=zmax/10; n1=0
   p=0; pi=0; s=0     ! Обнуление счетчиков
   r=0                ! Начальное число заявок в системе
   tlib=1e15          ! Моменты освобождения каналов
   ttlib=1e15         ! Самый ранний из них
   tt=0               ! Модельное время
   told=0             ! Момент предыдущей смены состояния
   tz=ftz(tt)         ! Момент прибытия первой заявки
```

```
do while(ns<zmax)  ! Пока заявок было мало
   do while(n1<n2) ! Пока начальный участок
      if (tz<ttlib) then
         !** Обработка прибытия заявки ***
         tt=tz
         nz=nz+1           ! Число прибывших заявок
         pi(r)=pi(r)+1     ! Для сост. перед прибытием
         p(r)=p(r)+tt-told ! Для стационарных
         told=tt
         if (r==rr) then
            nref=nref+1    ! Подсчет числа отказов
         else
            !* Заявка принимается
            r=r+1
            if (r<=n) then
               ! непосредственно в канал
               do m=1,n
                  if (tlib(m)==1e15) exit
               end do
               tlib(m)=flib(tt) ! Момент его освобождения
               ns=ns+1; n1=n1+1 ! Счетчики выбр. заявок
               call first
            else
               tau(r)=tt    ! Запомин. момента прибытия
            end if
         end if
         tz=ftz(tt)         ! Прибытие новой заявки
      else
         !** Обработка завершения обслуживания ***
         tt=ttlib
         p(r)=p(r)+tt-told !Счетчик для стац. вероятностей
         told=tt
         r=r-1
         if (r<n) then
            !* Канал освобождается *
            tlib(j)=1e15    ! (в "бесконечность")
         else
```

```
               !* Берем заявку из очереди
               n1=n1+1; ns=ns+1 ! Накопление числа выбранных
               x=tt-tau(n+1)    ! Ожидание головной заявки
               b=x
               do i=1,kmax
                  s(i)=s(i)+b; b=b*x
               end do
               do i=n+1,r
                  tau(i)=tau(i+1)! Продвижение очереди
               end do
               tlib(j)=flib(tt)  ! Освобождение j-канала
            end if
            call first()
         end if
      end do
      theta=s/ns
      print *, 'ns=',ns
      print *, 'Моменты распределения времени ожидания'
      print *, theta
      n1=0
   end do
   p=p/tt; pi=pi/zmax
   print *, 'Число отказов = ',nref
   deallocate (s,tau,tlib)

contains
   subroutine first
      !* Выбор момента ближайшего обслуживания
      integer i
      ttlib=tlib(1); j=1
      do i=2,n
         if (tlib(i)<ttlib) then
            ttlib=tlib(i); j=i
         end if
      end do
   end subroutine first
end subroutine model
```

```
function ftz(tt)
   !* Очередной момент прибытия заявки *
   !* Предыдущая прибыла в момент tt   *
   !* Входящий поток - простейший      *
   real     c,lam,x,tt
   lam=7e-1
   call random_number(x)
   c=log(x)/lam
   ftz=tt-c
end function ftz

function flib(tt)
   !*  Момент завершения обслуживания, *
   !*     начинаемого в момент tt      *
   !*  Длительность обслуживания = b   *
   real b,tt
   b=2e0
   flib=tt+b
end function flib
```

Теперь приведем результаты счета, опуская внутренние выдачи (они встроены в программу для контроля за сходимостью результатов к установившимся значениям):

```
  p
1.661490E-01    2.681018E-01    2.368847E-01    1.535796E-01
8.443638E-02    4.416471E-02    2.271876E-02    1.169181E-02
6.011551E-03    3.197770E-03    1.611588E-03    6.780776E-04
3.825238E-04    1.965745E-04    7.669851E-05    2.532698E-05
4.166035E-05    2.602405E-05    2.022878E-05    5.193876E-06
  pi
1.656300E-01    2.700350E-01    2.365450E-01    1.526850E-01
8.418000E-02    4.435000E-02    2.258000E-02    1.155000E-02
6.270000E-03    3.160000E-03    1.570000E-03    7.100000E-04
3.450000E-04    1.900000E-04    5.000000E-05    4.000000E-05
6.000000E-05    2.500000E-05    2.500000E-05    1.000000E-05
  theta
9.864905E-01        3.025315       13.534090
 Средняя длина очереди =   6.903524E-01
```

Средняя длина очереди практически совпала со значением 0.692, полученным при численной реализации метода Кроммелина [55] для модели $M/D/2$ с теми же исходными данными. Отметим также хорошее согласие между собой двух выводимых массивов распределений вероятностей (теоретически в случае простейшего входящего потока и неограниченной очереди они должны совпадать).

В процессе моделирования было зафиксировано два отказа в приеме заявок на обслуживание. Поскольку вероятность отказа равна вероятности иметь предельную длину очереди в момент прихода очередной заявки, упомянутый результат также соответствует ожиданиям.

2.3 Лабораторная работа с моделью

Описанная выше модель используется в читаемом автором курсе «Моделирование вычислительных систем» как объект изучения и средство экспериментальной проверки законов сохранения ТМО [15, 61]. Во всех вариантах работы для системы $M/G/n$ (с простейшим входящим потоком, распределением длительности обслуживания общего типа, n-канальной, с неограниченной очередью) проверяются:

- закон сохранения заявок

$$\sum_{j=0}^{n-1}(n-j)p_j = n - \lambda b;$$

- закон сохранения стационарной очереди для высших моментов $q_{[k]} = \lambda^k w_k\,, \quad k = \overline{1,3}$, в частности формулы Литтла для средних значений

$$q = \lambda w, \quad L = \lambda V$$

  числа заявок в очереди и в системе соответственно.

В этих формулах n — число каналов; λ — интенсивность входящего потока; b — средняя длительность обслуживания; $\{p_j\}$ — стационарное распределение числа заявок в системе; L — среднее число заявок в ней; V — среднее время пребывания заявки в системе; w — средняя длительность ожидания начала обслуживания; w_k — k-й момент распределения длительности ожидания; $q_{[k]} = \sum_{j=k}^{\infty} j(j-1)\ldots(j-k+1)p_j$ есть k-й

факториальный момент распределения длины очереди; $\rho = \lambda b/n = 0.8$ — расчетный коэффициент загрузки системы. Часть этих величин задавалась студентами как входные данные, часть определялась из имитационной модели, остальные (например, факториальные моменты распределения длины очереди) — применением к результатам моделирования программ из описанного в [58] пакета. Предлагаемые варианты различались числом каналов и типом распределения длительности обслуживания. Приведем «Методические указания» к данной работе из [58]:

1. Необходимые статистические данные доставляются процедурой `MODEL`. Пользователь должен самостоятельно запрограммировать и автономно оттранслировать внешние процедуры `FTZ(TT)` и `FLIB(TT)`, задающие случайные моменты прибытия очередной заявки и завершения начинаемого обслуживания по текущему значению `TT` таймера модели.

2. Расчет числовых исходных данных следует начинать с выбора свободных параметров распределения времени обслуживания. Затем вычисляются моменты $\{b_k\}$ этого распределения и по коэффициенту загрузки $\rho = \lambda b/n = 0.8$ — интенсивность входящего потока λ. Вычисление моментов и формирование случайных величин производятся с учетом характеристических диаграмм соответствующих распределений — см. главу 3 (задержки в каждой фазе распределены показательно). Числовые значения должны быть присвоены параметрам распределений *внутри* процедур `FLIB` и `FTZ` соответственно.

3. Моменты распределения времени пребывания заявки в системе вычисляются сверткой (процедура `CONV`) получаемых из модели моментов ожидания и теоретических значений моментов распределения чистой длительности обслуживания.

4. Выполнение законов сохранения оценивается относительной невязкой левой и правой частей соответствующих равенств в процентах.

В качестве *контрольных вопросов* при защите отчетов предлагались следующие:

1. Нарисуйте базовую схему имитационной модели.

2. Как формируются случайные числа с необходимыми для реализации модели законами распределения?

3. Сформулируйте закон сохранения стационарной очереди.

4. Выведите общую формулу, связывающую плотность распределения времени ожидания начала обслуживания с распределением числа заявок в очереди при рекуррентном входящем потоке.

5. Получите вариант этой формулы при простейшем входящем потоке как частный случай и непосредственным выводом.

6. Выведите соотношение между моментами распределения стационарной очереди и распределения времени ожидания в системе $M/G/n$. Что можно дополнительно сказать о системе $M/D/n$? На каком основании?

7. Опишите схему расчета моментов распределений E_2, H_2, C_2.

8. Что нужно сделать для получения интервальных оценок определяемых в данной работе статистических величин?

9. Какой метод уменьшения дисперсии результатов моделирования применен в данной работе?

10. Дайте краткую характеристику известных вам методов понижения дисперсии.

Для примера ответим на вопрос об особенностях расчета $M/D/n$. Закон сохранения стационарной очереди, следствием которого являются обобщенные формулы Литтла, выводится в предположении «первый пришел — первый выбран». Поскольку при регулярном обслуживании ($D = \texttt{deterministic}$) обгон в процессе обслуживания исключен, в данном случае упомянутые формулы будут верны и для системы в целом, то есть применительно к моментам полной длительности пребывания в системе и распределения числа заявок в ней. На том же основании этот тезис справедлив и для одноканальных систем с произвольным распределением длительности обслуживания.

Последние три вопроса этого перечня предполагают, что к моменту выполнения работы начитан соответствующий теоретический материал — см. [15] и последующие главы этой книги.

Методической целью работы являлось осознание принципов работы имитационных моделей, генерации случайных чисел и базовых понятий ТМО (распределения, моменты, коэффициент загрузки, законы сохранения). Эта цель достигалась вполне, хотя и не сразу. Особенно поучительными были ошибки, связанные с неверным определением параметров входных распределений и приводящие к перегрузке системы (о перегрузке свидетельствуют монотонный рост вероятностей $\{p_j\}$ по j и большое число отказов в приеме заявок). Заметим, что работа (на языке *ПЛ/1*) с готовыми программами из пакета МОСТ [44, 58] избавляет обучаемых от трудоемкого программирования самой модели, сводит к минимуму проблему отладки и позволяет сосредоточиться на существе предмета.

2.4 Варианты моделей систем обслуживания

В этом разделе для каждого варианта обсуждаются только *впервые* возникающие в нем особенности.

2.4.1 Неоднородный поток заявок

Здесь возможны два варианта. В первом из них необходимо иметь независимые указатели времени поступления заявок каждого типа и для сравнения с моментом окончания обслуживания отдельно хранить наиболее ранний из них. После приема в систему очередной заявки некоторого типа вырабатывается (при рекуррентных потоках) время прибытия новой заявки того же типа, выбираются новый минимум и индекс соответствующего источника. В упрощенном варианте моделируется поступление заявок *суммарного* простейшего потока интенсивности $\Lambda = \sum_i \lambda_i$ с дополнительным разыгрыванием типа заявки на основе вероятностей $\{\lambda_i/\Lambda\}$. Этот вариант дает приемлемые результаты лишь при составляющих потоках, близких к простейшим.

При неоднородном потоке необходимо либо помнить индекс типа заявки до ухода ее из системы, если заявка находится в общей очереди, либо отдельно строить очереди разнотипных заявок. Тип заявки учитывается при постановке ее в очередь, при формировании случайной

продолжительности обслуживания, при определении допустимого времени ожидания (пребывания в системе), а также при дифференцированной по классам заявок обработке результатов (подсчет числа отказов в обслуживании, моментов распределения времени пребывания в СМО и т. п.).

2.4.2 Кольцевая система очередей

Здесь очереди с независимыми источниками заявок обслуживаются «по кругу» согласно одному из вариантов дисциплины:

- по одной заявке из каждой непустой очереди;
- до исчерпания заявок, скопившихся в текущей очереди к моменту начала ее обслуживания;
- до исчерпания текущей очереди.

Дополнительных рекомендаций требует только второй вариант: следует запоминать момент начала обслуживания очереди и сравнивать его с моментами прибытия заявок. Для выбора следующей очереди после $i = n$ принимается $i = 1$.

2.4.3 Статический относительный приоритет

Предпочтительна организация отдельной очереди по каждому классу приоритета. При завершении обслуживания очередная заявка выбирается из головы непустой очереди с наивысшим приоритетом.

2.4.4 Динамический приоритет

В общей очереди размещаются паспорта заявок с указанием их типов и моментов прибытия. В момент завершения обслуживания начинается просмотр очереди с вычислением текущего диспетчерского приоритета каждой заявки и его сравнением с текущим максимумом. При необходимости значение максимума и его координата обновляются. По окончании просмотра «максимальная» заявка выбирается в канал, а следующая за ней часть очереди уплотняется.

2.4.5 Многоуровневая система очередей с квантованным обслуживанием

Здесь появляется новый тип события — исчерпание кванта текущего обслуживания. Соответственно прерванная заявка перемещается в конец следующей очереди, где она будет дожидаться предоставления большего кванта.

2.4.6 Определение длины периода непрерывной занятости

При прибытии в n-канальную систему заявки, увеличивающей их текущее количество до n, фиксируется начало ПНЗ — значение таймера. При завершении обслуживания и уменьшении текущего числа заявок до $n-1$ отмечается окончание ПНЗ и к счетчикам добавляются последовательные степени его длительности (для вычисления моментов). К счетчику числа ПНЗ добавляется единица.

2.4.7 Абсолютный приоритет, одноканальная система

Случайное время обслуживания для новой заявки генерируется в момент ее прибытия. Тип вновь прибывшей заявки сравнивается с типом обслуживаемой и определяется необходимость прерывания. Прерванная заявка помещается в голову очереди своего приоритета с потребным временем обслуживания, равным разности между расчетным моментом завершения его и текущим значением таймера (режим дообслуживания). Для обсуждаемых ниже фазовых распределений можно запоминать не остаток времени, а номер фазы, на которой произошло прерывание. Возможны варианты генерации времени обслуживания в момент приема в канал (независимо от кратности захода) или сохранения первоначально сформированной длительности (все три варианта рассматриваются в теории очередей).

Просмотр очередей для выбора следующей заявки начинается с номера, соответствующего типу обслуженной (при наличии более приоритетных заявок она была бы прервана).

В паспорт заявки может быть внесен раздел кратности прерываний,

куда при каждом прерывании добавляется единица. Статистика накапливается при нормальном завершении обслуживания.

На рис. 2.2 показана блок-схема модели системы $\overrightarrow{GI}/\vec{G}/1/\vec{R}$ с абсолютным приоритетом и сохранением выполненной до прерывания работы для расчета числа отказов и среднего времени пребывания заявки каждого типа в системе. Предполагается, что прерванная заявка ставится в голову своей очереди. При недостатке места последняя заявка данного класса получает отказ. Приведем словарь дополнительных (по отношению к рис. 2.1) обозначений на схеме:

`tz`	—	массив моментов прибытия заявок каждого типа,
`ttz`	—	его минимальная компонента,
`j`	—	ее индекс,
`tin`	—	момент прихода в систему обслуживаемой заявки,
`z`	—	массив количества заявок каждого типа,
`rr`	—	общее число заявок в системе,
`p`	—	класс приоритета текущего обслуживания,
`s`	—	счетчики суммарного времени пребывания заявок каждого типа,
`n`	—	счетчики обслуженных заявок,
`tau`	—	оставшиеся длительности обслуживания,
`t`	—	моменты прибытия заявок в систему,
`ref`	—	счетчики отказов.

2.4.8 Многоканальные приоритетные системы

При моделировании *многоканальных* систем, как это показано в разд. 2.2, необходимо учитывать число и номера занятых каналов, а также возможную их дифференциацию по типам принимаемых заявок и характеристикам распределения времени обслуживания. Учет моментов освобождения каналов ведется аналогично обработке моментов поступления заявок неоднородного рекуррентного потока общего вида. При однородных каналах простейшей дисциплиной выбора занимаемого канала является «первый свободный», выявляемый по «запредельному» моменту освобождения. При нескольких свободных каналах может потребоваться равновероятный выбор из них.

Рис. 2.2. Модель одноканальной СМО с абсолютным приоритетом

Если таких каналов m, то для определения выбираемого канала следует вычислить $N = [m * U] + 1$, где U — случайное число, равномерно распределенное на полуинтервале $[0, 1)$ (см. главу 4), а квадратные скобки означают взятие целой части. Далее начинается просмотр моментов освобождения каналов, и для каждого свободного в счетчик добавляется единица. Заявка принимается в тот канал, для которого содержимое счетчика сравняется с N.

Особенно сложную логику имеет моделирование *многоканальных* систем с абсолютным приоритетом (эта задача весьма актуальна ввиду отсутствия аналитических методов ее решения в достаточно общей постановке). Хранение паспортов заявок, находящихся в системе, здесь целесообразно организовать в массиве `R` $\times$ 4, где `R` — предельное число заявок. Паспорт включает в себя тип заявки `z` (он же индекс приоритета), время прибытия в систему τ, требуемое время обслуживания θ. Для прерванной заявки в качестве θ записывается остаток времени обслуживания, вычисляемый как разность моментов планового завершения обслуживания и прерывания его. Четвертой компонентой паспорта для заявок, находящихся на обслуживании, является номер с занимаемого канала. Необходимость быстрого поиска прерываемых и выбираемых на обслуживание после освобождения канала заявок диктует упорядоченность списка паспортов по убыванию приоритетов, а при равных приоритетах — по возрастанию времени прибытия в систему. Первые n позиций списка заняты обслуживаемыми заявками. В этом случае приоритет вновь прибывшей заявки достаточно сопоставить с обслуживаемой в последнем канале. Такая технология требует переупорядочения списка при прерывании и при выборе заявки из очереди после завершении обслуживания.

Индекс π здесь представляет младший приоритет текущего обслуживания и сравнивается с типом `i` новой заявки. При `i` $< \pi$ происходит прерывание заявки, находящейся в `n`-й позиции списка; по номеру канала определяется остаток обслуживания; прерванная заявка помещается в очередь. При `i` $\geq \pi$ в очередь помещается новая заявка. Для быстрого поиска места вставки целесообразно иметь массив указателей начал «частных очередей» каждого приоритета. Место вставки для прерванной заявки нужно искать в начале частной очереди, а для новой — в конце. Если общая очередь уже имеет максимальную длину, то (в зависимости от соотношения приоритетов) получает отказ либо новая заявка, либо последняя заявка очереди. В паспорт прерывающей заявки заносится

номер освобожденного канала, после чего для нее определяется позиция вставки. Затем выполняются вставки, сдвигаются нижележащие паспорта и корректируются указатели частных очередей. Заявка, выбираемая из очереди после завершения обслуживания, всегда помещается в n-ю строку списка паспортов.

Приведем соответствующую программу:

```
program tnprtypr
   integer n,k,jmax,nmax
   real*8  w(3,3),z
   n=1; k=3; jmax=20; nmax=20000
   do i=1,5000; call random_number(z); end do
   call modprn(n,k,jmax,w,nmax)
end

      subroutine modprn(n,k,jmax,w,nmax)
!*************************************************/
!*  Имитационная модель системы GI/G/n/jmax      */
!* с абсолютными приоритетами и дообслуживанием  */
!*         Разработчик Ю.И.Рыжиков               */
!*                    (c)                        */
!*            Версия 12.05.2000                  */
!*************************************************/
      integer    n    !! число каналов
      integer    jmax !! предельная длина очередей в системе
      integer    k    !! число типов заявок
      real*8     w    !! моменты длительности пребывания
      integer    nmax !! задано завершений обслуж.
                      !! по заявкам первого типа
      dimension  w(k,3)

      integer                   i,j,jlib,jz,m,prc,rr
      real*8                    b,flib,ftz,tt,ttlib,ttz,x
      real*8,  allocatable ::   q(:,:,:),s(:,:),chn(:,:),tz(:)
      integer, allocatable ::   r(:),nref(:),ns(:)

      allocate (q(k,jmax,2),s(k,3),chn(n,3),tz(k),r(k), &
                nref(k),ns(k))
```

```
   print *, ' Процедура Model - Абсолютный PRTY'

   rr=0; tt=0          ! Число заявок в системе и таймер
   do i=1,k            ! По типам
      nref(i)=0        ! Число отказов
      ns(i)=0          ! Счетчик выборок
      r(i)=0           ! Длина очереди
      tz(i)=ftz(tt,i) ! Момент прибытия заявки
      do j=1,3
         s(i,j)=0      ! Счетчики моментов пребывания
      end do
   end do
   do i=1,n; chn(i,3)=1d20; end do !! Каналы свободны
do while(ns(1)<nmax)
 !! продолжаем моделирование
   ttz=1d20
   do i=1,k
      !* Выбираем тип ближайшей заявки и момент прибытия *
      if (tz(i)<ttz) then
         jz=i; ttz=tz(i)
      end if
   end do
   ttlib=1d20
   do j=1,n
      !*  Выбираем канал и момент ближайшего обслуживания *
      if (chn(j,3)<ttlib) then
         jlib=j; ttlib=chn(j,3)
      end if
   end do

   if (ttz<ttlib) then
      !*** Прибытие заявки типа jz ***
      tt=ttz                  !! Сдвиг модельного времени
      tz(jz)=ftz(tt,jz)       !! момент прибытия новой
      if (r(jz)==jmax) then
         nref(jz)=nref(jz)+1 !! накопление числа отказов
      else
         !* прием в систему *
```

```
      rr=rr+1   !! Общее число заявок
      if (rr<=n) then
         !* немедленный прием на обслуживание *
         do j=1,n
            if (chn(j,3)==1d20) then
               m=j; exit        !! первый свободный
            end if
         end do
         !* Занятие канала m *
         chn(m,1)=jz            !! Тип заявки
         chn(m,2)=tt            !! Время прибытия
         chn(m,3)=tt+flib(jz)   !! Время освобождения
      else
   !* Определяем max индекс prc приоритета в каналах *
         prc=0
         do i=1,n
            if (chn(i,1)>prc) then
               prc=chn(i,1); m=i
            end if
         end do
         if (jz<prc) then
            !* Прерывание prc-заявки в канале m *
            if (r(prc)==jmax) then
               ! Прерванная заявка теряется
               r(prc)=r(prc)-1; nref(prc)=nref(prc)+1
               rr=rr-1
            end if
            do i=r(prc),1,-1
               !* Сдвиг вниз очереди по прерванному типу *
               q(prc,i+1,1)=q(prc,i,1) !! Моменты прибытия
               q(prc,i+1,2)=q(prc,i,2) !! Длительности обсл.
            end do
            !* Прерванная -> в голову очереди своего типа *
            q(prc,1,1)=chn(m,2)     !! Прибытие прерванной
            q(prc,1,2)=chn(m,3)-tt  !! Остаток ее обслуж.
            r(prc)=r(prc)+1
            !* Прием прерывающей в канал m *
            chn(m,1)=jz             !! Тип заявки
```

```
                chn(m,2)=tt             !! Ее прибытие в систему
                chn(m,3)=tt+flib(jz)    !! Освобождение канала
             else
                !*  постановка jz-заявки в конец своей очереди *
                r(jz)=r(jz)+1
                q(jz,r(jz),1)=tt; q(jz,r(jz),2)=flib(jz)
             end if      !! Есть/нет прерывания
          end if         !! канал/очередь
       end if            !! прием/отказ
    else
 !*** Завершение обслуживания в канале jlib ***
       tt=ttlib; rr=rr-1
       m= chn(jlib,1)     !! Тип уходящей заявки
       x=tt-chn(jlib,2)   !! Время ее пребывания в системе
       b=x
       do i=1,3
          !** Накопление моментов пребывания **
          s(m,i)=s(m,i)+b; b=b*x
       end do
       ns(m)=ns(m)+1       !! Только при завершении обслуж.
       if (rr<n) then
          chn(jlib,3)=1d20 !! Канал jlib остается свободным
       else
          !** Выбор из очереди и ее продвижение **
          do i=1,k
             if (r(i)>0) then
                jq=i; exit        !! Первая непустая очередь
             end if
        end do
        r(jq)=r(jq)-1             !! Уменьшение длины очереди
        chn(jlib,1)=jq            !! Тип выбранной заявки
        chn(jlib,2)=q(jq,1,1)     !! Время прибытия в систему
                                  !! головной заявки очереди
        chn(jlib,3)=tt+q(jq,1,2)  !! Освоб. занимаемого канала
        do i=1,r(jq)
           q(jq,i,:)=q(jq,i+1,:)  !! Продвижение очереди
        end do
     end if !! Наличие очереди
```

```
   end if     !! Заявка/обслуживание
end do        !! Главного цикла

   do i=1,k
      if (ns(i)>0) then
          do j=1,3; w(i,j)=s(i,j)/ns(i); end do
      else
          w(i,:)=0
      end if
   end do
   print *, ' Количество обслуженных заявок по типам'
   write (*,'(3i10)') ns
   print *, ' Количество отказов в приеме на обслуживание'
   write (*,'(3i10)') nref
   print *, ' Длины очередей к концу моделирования'
   write (*,'(3i10)') r
   print *, ' Моменты распределения времени пребывания'
   write (*,'(3e14.5)') ((w(i,j),j=1,3),i=1,k)

   deallocate (q,s,chn,tz,r,nref,ns)
end !! modprn
```

Отметим дополнительные особенности моделирования приоритетного обслуживания с прерыванием и возобновлением обслуживания со старой или новой реализациями его случайной длительности (схемы RW и RS соответственно [32]). В первом варианте требуемое время обслуживания формируется один раз (при прибытии заявки) и в случае прерывания пересчету не подлежит. Во втором варианте целесообразно формировать требуемое время обслуживания в момент занятия канала (независимо от кратности захода) и соответствующую компоненту из паспорта заявки исключить.

В случае динамических (переменных во времени) приоритетов обслуживание заявок обычно ведется без прерываний, а диспетчерские приоритеты заявок растут по времени ожидания — как правило, линейно с коэффициентами $\{\beta_j\}$, убывающими по мере понижения базового приоритета. Здесь следует хранить неупорядоченную общую очередь, а в паспорте каждой заявки фиксировать ее тип и момент прибытия. При

освобождении канала для всех заявок очереди поочередно вычисляется диспетчерский приоритет и при необходимости обновляются значение и позиция текущего максимума. По окончании просмотра заявка с максимальным приоритетом выбирается на обслуживание, а очередь уплотняется.

До сих пор мы рассматривали выбор заявок из очередей в порядке их прибытия, то есть по принципу FIFO (First IN — First Out). При работе по правилу LIFO (Last IN — First Out) ожидающие обслуживания заявки образуют *магазин*, указателем которого служит текущая длина очереди. Но как обычную очередь, так и магазин можно организовать в виде закольцованного массива длины (закольцованность достигается относительной адресацией всех обращений к этому массиву по модулю M). В этом случае, как видно из рис. 2.3, перемещение информационных элементов заменяется пересчетом соответствующих указателей (**Г** и **К** — голова и конец очереди, **У**— указатель магазина), что реализуется значительно экономнее.

Обозначим: In — входная переменная (паспорт заявки); Out — выходная переменная; H — голова очереди; E — конец очереди; P — указатель магазина; C(0:M-1)— кольцевой массив. Тогда работа с очередью программируется следующим образом:

```
* Запись                    * Выборка
C(E)=In                     Out=C(H)
E=mod(E+1,M)                H=mod(H+1,M)
```

Аналогично для магазина

```
* Запись                    * Выборка
C(P)=In                     P=mod(P+M-1,M)
P=mod(P+1,M)                Out=C(P)
```

Для моделирования случайного выбора заявки из очереди текущая длина очереди Q умножается на случайное число $U \in [0,1)$, результат (с добавлением единицы) определяет номер выбираемой заявки, очередь уплотняется.

Рис. 2.3. Кольцевая организация очереди и магазина

При моделировании системы с *временны́ми ограничениями* на пребывание в ней заявок каждую из них в очереди (магазине) должен сопровождать допустимый момент начала (завершения) обслуживания. Кроме того, отдельно должен фиксироваться минимум из этих моментов с привязкой к «наиболее нетерпеливой»заявке. Потеря «нетерпеливой» заявки должна сопровождаться уплотнением очереди (магазина) и выбором следующей по «степени нетерпения». Такой выбор производится и в тех случаях, когда «нетерпеливая»заявка ушла из очереди в канал (при ограничении на время ожидания), при завершении обслуживания (ограничение на время пребывания), а также при прибытии новой заяв-

ки. Иметь упорядоченную очередь не обязательно — достаточно каждый раз при освобождении канала выбирать наиболее нетерпеливую заявку.

В системах с временны́ми ограничениями учет заявок, ушедших необслуженными, ведется раздельно по каждому виду заявок.

При составлении наиболее сложных моделей незаменимы *списки* — динамические наборы данных, каждый элемент которых состоит из информационных полей, хранящих целевые данные, и адресных, содержащих ссылки на смежные элементы. В отличие от очередей и таблиц, требующих резервирования постоянных объемов памяти, работа со списками (рис. 2.4) позволяет добавлять и исключать элементы — следовательно, использовать память по реальной потребности в ней. Применение списков не требует физического перемещения записей, что уменьшает трудоемкость моделирования при интенсивной динамике (обновление цепей событий — текущих, будущих, задержанных по условию, прерванных процессов, а также групп отобранных объектов).

Рис. 2.4. Структура списка

Логика работы модели может меняться в зависимости от *целей исследования*. При расчете распределения стационарных вероятностей состояний в модели должны иметься счетчики по числу возможных состояний, и к их содержимому при каждом изменении состояний должно прибавляться время, проведенное в предыдущем состоянии.

Для расчета распределения числа заявок перед прибытием очередной заявки при прибытии новой заявки добавляется единица к счетчику, соответствующему текущему (старому) числу заявок к системе. В конце моделирования содержимое всех этих счетчиков делится на полное число прибывших заявок. Аналогично могут быть определены вероятности состояний на моменты завершения обслуживания.

Разработка аналитических методов расчета многоканальных приоритетных систем потребовала найти распределение занятых каналов по типам заявок. Здесь можно в момент прибытия заявки в полностью занятую систему опросить все каналы и для каждого добавить единицу в счетчик, соответствующий типу обслуживаемой этим каналом заявки. Оказалось, что для систем с неограниченным ожиданием искомые вероятности пропорциональны частным коэффициентам загрузки системы по типам заявок, т. е. произведениям $\{\lambda_i b_{i,1}\}$, — независимо от назначенных приоритетов.

Для определения средних интенсивностей потоков некоторых событий следует иметь отдельные счетчики по каждому классу таких событий и счетчик времени функционирования модели (системного времени). Для вычисления временны́х характеристик (занятости канала, пребывания заявки в системе) должны быть предусмотрены фиксация начала и конца соответствующих интервалов, а также накапливающие счетчики. Если требуется вычислить несколько статистических моментов исследуемой случайной величины, счетчики необходимы по числу моментов.

Рассмотрим построение распределения длительности периода полной непрерывной занятости системы (ПНЗ). Здесь в момент прибытия заявки, увеличивающей их количество до n, фиксируется начало ПНЗ. Окончание его наступает, когда число заявок в системе при завершении обслуживания уменьшается с n до $n-1$. В этот момент к счетчику ПНЗ добавляется единица, а в счетчиках моментов искомого распределения накапливаются соответствующие степени длительности ПНЗ.

При построении статистического распределения некоторой непрерывной случайной величины следует разбить возможный диапазон ее изменения на отрезки (обычно 15-20) и завести на каждый отрезок отдельный счетчик. Обработка содержимого этих счетчиков дает гистограмму распределения, а сопоставление ее (например, по критерию χ^2) с постулируемым теоретическим распределением позволит оценить приемлемость последнего. Статистические моменты можно непосредственно применить для апроксимации непрерывной плотности распределения ее

разложением по обобщенным многочленам Лагерра (см. [55]) или смесью показательных распределений — глава 3.

2.5 Моделирование сетей обслуживания

Поскольку сеть обслуживания состоит из $M > 1$ реальных узлов, ее модель в общем случае имитирует рассмотренными выше способами каждый из узлов и дополнительно — их взаимодействие. Здесь возникают следующие особенности:

1. Вводятся дополнительные фиктивные узлы «0» — источник, « $M+1$ » — сток.

2. Для разомкнутой сети моменты прибытия внешних заявок формируются в узле «0». По элементам нулевой строки матрицы $\{r_{i,j}\}$ вероятностей переходов между узлами посредством ДСЧ разыгрывается номер узла–преемника, для которого затем моделируется процесс приема заявки.

3. При завершении обслуживания в узле i аналогичным образом определяется узел–преемник j, который продолжает ее обслуживание. Если $j = M+1$, заявка считается покинувшей сеть.

4. При моделировании замкнутой сети работа узла «0» не имитируется. Первоначально сеть случайным образом заполняется заданным числом R заявок, а затем при каждом переходе в узел $M+1$ аналогично п.2, но по $(M+1)$-й строке матрицы переходов выбирается узел–преемник.

5. Для управления процессом моделирования в каждом узле выбирается наиболее ранний момент освобождения канала, а по сети в целом — минимальный из них. Эти данные должны обновляться при каждом завершении обслуживания или прибытии внешней заявки.

Основными показателями работы сетей обслуживания являются моменты распределения времени пребывания заявки в сети и маргинальные (частные) распределения числа заявок в узлах. Время пребывания в сети каждой конкретной заявки определяется следующим образом:

1. Для входящей в сеть заявки формируется отметка времени ее прибытия.

2. На каждый j-й узел сети заводится отдельная очередь, в которой первые n_j (по числу каналов в узле) позиций хранят моменты прибытия в систему проходящих обслуживание заявок, а остальные — аналогичные моменты для заявок ожидающих.

3. Отметка времени при переходе заявки из узла в узел ее сопровождает.

4. При переходе заявки в сток определяется время ее пребывания в сети как разность между текущим значением таймера и отметкой времени прибытия. Далее вычисляются и накапливаются степени времени пребывания.

В целях подсчета маргинальных (частных) распределений числа заявок для каждого узла необходимо иметь указатель момента последнего изменения состояния узла и отдельный счетчик на каждое возможное состояние узла.

Сети обслуживания могут иметь весьма разнообразные особенности поведения, из которых мы относительно подробно рассмотрим два: блокировки и расщепление/слияние заявок. Под *блокировкой* понимается ситуация, когда заявка, завершившая обслуживание в одном из узлов, не может быть принята очередным из-за отсутствия места в очереди. Соответствующий канал помечается как блокированный и в выборе момента следующего ближайшего завершения обслуживания не участвует. С другой стороны, у преемника блокированной заявки формируется очередь «внешних» запросов, которые по мере завершения обслуживаний принимаются в конец очереди. В обработке блокировок для j-го узла сети должны принимать участие следующие элементы вспомогательных массивов:

`ext(j)` — количество накопленных внешних запросов;

`blin(j,k,1)` — откуда сделан k-й запрос;

`blin(j,k,2)` — в какой момент модельного времени;

`blout(j,l)` — признак блокировки l-го канала (равен номеру блокирующего узла или нулю при отсутствии блокировки).

Максимальное количество внешних запросов к узлу не превышает суммарного числа каналов в остальных узлах сети.

Ниже приводится программа на *Фортране* 90 моделирования сети с блокировками. Из распечатки исключены тексты процедур `ftz` формирования моментов прибытия очередной внешней заявки (простейший поток), `flib` моментов завершения обслуживания принятой в канал заявки (обслуживание предполагалось регулярным) и `gauss` решения системы линейных уравнений методом Гаусса с расширенной матрицей коэффициентов, расположенной по строкам, и поиском главных элементов по столбцу. Последняя используется для нахождения из системы уравнений баланса

$$\lambda_i = \Lambda r_{0,i} + \sum_{j=1}^{M} \lambda_j r_{j,i}, \qquad i = \overline{1, M}$$

интенсивностей потоков через узлы (Λ — интенсивность внешнего потока). По ним определяются коэффициенты загрузки узлов $\rho_i = \lambda_i b_i / n_i$, без знания которых моделирование сети бессмысленно. При указанных в тексте исходных данных коэффициенты загрузки равны 0.3605, 0.7868, 0.7748, 0.4672 соответственно. Остальные необходимые пояснения включены в программу как комментарии к составляющим ее частным процедурам, фрагментам и отдельным операторам.

```
program testnw
!****************************
!* Моделирование разомкнутой  *
!*    сети обслуживания       *
!****************************
   integer m        !! Число узлов
   real    r(0:4,5) !! Матрица передач
   integer rr       !! Предельное число заявок в узле
   real    lext     !! Интенсивность внешнего потока
   integer n(4)     !! Количество каналов (по узлам)
   real    b(4)     !! Средние времена обслуж. (по узлам)
   integer k        !! Кол-во вычисляемых моментов вр. пребыв.
   integer zmax     !! Расчетное число прибывших заявок
   integer nref     !! Кол-во отказов внешним заявкам
   real    vnw(3)   !! Моменты распред. времени пребыв. в сети

   interface
```

```
      subroutine modnw(m,r,rr,lext,n,b,k,zmax,nref,vnw)
         integer m,rr,n(:),k,zmax,nref
         real    r(0:m,m+1),lext,b(:),vnw(:)
      end subroutine modnw
   end interface

   data r/0.7, 0,   0.1, 0,   0.1,  &
          0.2, 0.5, 0.2, 0.2, 0.2,  &
          0.1, 0.3, 0.3, 0.3, 0.2,  &
          0,   0.2, 0.2, 0.2, 0,    &
          0,   0,   0.2, 0.3, 0.5/
   data m/4/,rr /30/,k/3/, lext/0.4/, zmax/1000/, &
        n/2,2,2,3/, b/2,3,3,5/
    open(7,file='newnet2.res')
   call modnw(m,r,rr,lext,n,b,k,zmax,vnw)
   write(7,*) 'vnw=', vnw
   write(7,*) 'nref=',nref
   close(7)
end program testnw

subroutine modnw(m,r,rr,lext,n,b,k,zmax,nref,vnw)
!*********************************
!*   Моделирование разомкнутой    *
!*       сети обслуживания        *
!*********************************

!MS$DEBUG
   integer m          !! Число узлов
   real    r(0:m,m+1) !! Матрица передач
   integer rr         !! Предельное число заявок в сети
   real    lext       !! Интенсивность внешнего потока
   integer n(:)       !! Количество каналов (по узлам)
   real    b(:)       !! Среднее время обслуж. (по узлам)
   integer k          !! Кол-во подсчитываемых моментов
   integer zmax       !! Заданное к-во обслуженных заявок
   integer nref       !! Количество отказов внешним заявкам
   real    vnw(:)     !! Моменты распред. времени пребывания
```

```
interface
   subroutine gauss(n,a,x)
      integer n
      real a(:,:),x(:)
   end subroutine gauss
   real function flib(tt,b)
   ! Время завершения обслуживания
      real tt,b
   end function flib
   real function ftz(tt,lam)
   ! Время прибытия заявки
      real tt,lam
   end function ftz
end interface

real      te,tlib,tz,ts,tt,x
integer   i,j,key,ll,nmax,nn,nref,z10,zw
real, allocatable ::  a(:,:),lam(:),rho(:),cump(:,:), &
          p(:,:),pp(:),q(:,:),s(:),t(:),tau(:,:),theta(:)
integer,  allocatable, dimension(:) :: l,z
! Для работы с блокировками
real, allocatable :: ext(:),blin(:,:,:),blout(:,:)

! Проверка коэффициентов загрузки узлов
allocate (a(m,m+1),lam(m),rho(m),z(m+1))
do i=1,m
   do j=1,m; a(i,j)=r(j,i); end do
   a(i,i)=a(i,i)-1e0; a(i,m+1)=-r(0,i)*lext
end do
call gauss(m,a,lam)
do i=1,m; rho(i)=lam(i)*b(i)/n(i); end do
if (any(rho>=1)) then
   print * ,'Имеются перегруженные узлы'
   write(7,*) rho; stop
end if
nmax=1; nn=0
do i=1,m
   if (n(i)>nmax) nmax=n(i)
```

```
   nn=nn+n(i)
end do

allocate (cump(0:m,m+1),p(m,0:rr),pp(0:rr),s(k),  &
          ext(m),blin(m,nn,2),blout(m,nmax),      &
          l(m),t(m),tau(m,nmax),theta(m),q(m,rr))

!**** Инициализация модели ***
tt=0       ! Модельное время
z=0        ! Счетчики заявок в узлах
p=0        ! Счетчики маргинальных распределений
t=0        ! Моменты предыдущ. измен. состояния узлов
tau=1e15   ! Моменты освобождения каналов
theta=1e15 ! Самые ранние из них по узлам
q=1e15
! В массиве q для каждого j первые n(j) позиций
! хранят моменты прибытия в сеть заявок из каналов,
! остальные - из очереди
s=0     ! Счетчики моментов распред. времени пребывания
nblk=0  ! Счетчик количества блокировок
ndebl=0 ! Счетчик разблокировок

! Формирование кумулянтных вероятностей
do i=0,m
   cump (i,1)=r(i,1)
   do j=2,m; cump(i,j)=cump(i,j-1)+r(i,j); end do
   cump(i,m+1)=1e0
end do
! * Раскрутка ДСЧ *
do i=1,2000; call random_number(x); end do
z10=zmax/10 ! Начальный отрезок
nref=0      ! Счетчик отказов
ext=0       ! Обнуление счетчиков блокировок
blout=0     ! Обнуление блокировок выхода каналов
key=1       ! Установка ключа начального состояния
theta=1e15
tlib=1e15
tz=ftz(tt,lext) ! Момент прибытия внешней заявки
```

```
!*** ОСНОВНАЯ ДИНАМИКА ***
zw=0
do while(zin<zmax)   ! По числу входящих заявок
   if (tlib<tz) then
      ! Завершено обслуживание в узле ll, канал l(ll)
      tt=tlib        ! Модельное время
      te=q(ll,l(ll)) ! Вр. прибытия в сеть
      !* Выбор узла-преемника
      call random_number(x)
      j=select(cump(ll,:),x)
      if ((z(j)==rr).and.(j<=m)) then
         ! ***  Блокировка  ***
         ! Обслуженной заявке дальше хода нет
         ! Состояние узла не меняется
         nblk=nblk+1         ! Накопл. числа блокировок
         blout(ll,l(ll))=j   ! Осв. канал блокируется
         ext(j)=ext(j)+1     ! Копим запросы к узлу j
         blin(j,ext(j),1)=ll ! Откуда запрос (из ll)
         blin(j,ext(j),2)=tt ! и когда сделан
      else
         ! Будут изменения
         call done(ll,l(ll)) ! Продвижение в узле ll
         if (j==m+1) then
            !** Обработка ухода заявки из сети
            zw=zw+1       ! Подсчет количества уходов
            d=tt-te; x=d  ! Время пребыв. заявки в сети
            ! Накопление его моментов
            do i=1,k
               s(i)=s(i)+x; x=x*d
            end do
         else
            call reception(j,te)! Передача заявки в узел j
            ! Проверка наличия внешних запросов
            if (ext(ll)>0) then
            ! Разблокировка первой поз. внешней очереди
               j=blin(ll,1,1)   ! Откуда запрос
               tbl=blin(ll,1,2) ! Когда сделан
```

```
            ext(ll)=ext(ll)-1
            do i=1,ext(ll)
               blin(ll,i,:)=blin(ll,i+1,:)
            end do
            call unblock(ll,j,tbl)
         ! Эта заявка попадает в конец внутр. очереди
         ! Число заявок в узле ll не меняется
         end if
      end if
   end if
else
   !* Прибытие внешней заявки
   if (zin==z10) then
      if (key==1) then
         !* Исключение начального отрезка моделирования
         p=0; s=0; zw=0; zin=0; nblk=0; ndebl=0
         ts=tt; key=0
      end if
   end if
   tt=tz; zin=zin+1
   !* Выбор узла-преемника
   call random_number(x)
   j=select(cump(0,:),x)
   if (z(j)==rr) then
      nref=nref+1
      do i=1,m; call newstate(i,0); end do
   else
      call reception(j,tt)
   end if
   tz=ftz(tt,lext)
end if    ! Главных вариантов

! Выбор ближайшего обслуживания
tlib=1e15
do i=1,m
   call min(i,n(i),tau(i,:),theta(i),l(i))
   if (theta(i)<tlib) then
      tlib=theta(i)
```

```
            ll=i ! Узел с ближайшим завершением
         end if
      end do
   end do ! * Динамики

   ! *** Финальная обработка ***
   do i=1,m
      pp=p(i,:)/(tt-ts)
      write(7,*) 'i=',i
      write(7,*) pp
   end do
   do i=1,k; vnw(i)=s(i)/zw; end do
   write(7,*) 'nref=',nref
   write(7,*) 'nblk=',nblk
   write(7,*) 'ndebl=',ndebl
   deallocate   &
   (ext,blin,blout,cump,p,pp,s,t,tau,theta,q,a,l,lam,rho,z)
contains
   subroutine min(j,l,x,s,k)
      ! Выбор самого раннего момента s
      ! освобождения канала k в узле j
      ! (с исключением блокированных)
      integer j        ! Номер узла
      integer l        ! Число каналов в узле
      real    x(l)     ! Моменты их освобождения
      real    s        ! Минимальный среди неблокированных
      integer k        ! Его номер
      integer i
      s=1e15; k=0
      do i=1,l
         if ((x(i)<s).and.(blout(j,i)==0)) then
            s=x(i); k=i
         end if
      end do
   end subroutine min

   integer function select(cp,x)
      real  cp(m+1)    ! Кумулянтные вероятности
```

```
      real  x          ! Случайная ордината
      integer i
      do i=1,m+1
         if(x<cp(i)) exit
      end do
      select=i
   end function select

   subroutine newstate(j,k)
      integer j   ! Узел
      integer k   ! Приращение числа заявок
      real    d
      d=tt-t(j)
      p(j,z(j))=p(j,z(j))+d
      z(j)=z(j)+k ! Изменение  числа заявок
      t(j)=tt     ! Обновление момента измен.
   end subroutine newstate

   subroutine done(j,lj)
      integer j  ! Номер узла
      integer lj ! Освободившийся канал
      integer i
      call newstate(j,-1)
      if (z(j)<n(j)) then
         ! * Освобождение канала
         tau (j,lj)=1e15; q(j,lj)=1e15
      else
         ! Занятие канала
         q(j,lj)=q(j,n(j)+1)
         tau(j,lj)=flib(tt,b(j)) ! Сформирован момент освоб.
         ! * Продвижение очереди
         do i=n(j)+1,z(j); q(j,i)=q(j,i+1); end do
         q(j,z(j)+1)=1e15
      end if
  end subroutine done

  subroutine reception(j,te)
     integer j   ! Номер принимающего узла
```

```
   real    te  ! Время прибытия заявки в сеть
   ! Обработка прибытия заявки в узел j
   call newstate(j,1)
   if (z(j)<=n(j)) then
      ! Заявка принимается в j-й канал
      ! Ищем первый свободный
      do i=1,n(j)
         if (tau(j,i)==1e15) exit
      end do
      q(j,i)=te   ! * Перенос момента прибытия
      tau(j,i)=flib(tt,b(j))
   else
      ! * Заявка ставится в очередь
      q(j,z(j))=te ! * Запись момента прибытия
   end if
end subroutine reception

recursive subroutine unblock(ll,j,tbl)
   integer ll  ! Разблокировавший узел
   integer j   ! Деблокируемый узел
   real    tbl ! Момент снимаемой блокировки
   integer ib
   real    tentr,tnext
   ! Выбор деблокируемого канала ib
   do ib=1,n(j)
      if (tau(j,ib)==tbl) exit
   end do
   blout(j,ib)=0  ! Снимаем с него признак блокировки
   tau(j,ib)=1e15 ! Помечаем как свободный
   ! Передача заявки в разблокировавший узел
   tentr=q(j,ib)  ! Момент прибытия блокированной
   !                заявки в сеть
   call reception(ll,tentr) ! Прием ее в ll-м узле
   ! Продвижения в деблокируемом узле
   call done(j,ib)
   if (ext(j)>0) then
      ! Снятие первой блокировки
      jb=blin(j,1,1)    ! Откуда
```

```
        tnext=blin(j,1,2) ! Когда сделана
        ext(j)=ext(j)-1
        ! Продвижение списка блокировок
        do i=1,ext(j)
           blin(j,i,:)=blin(j,i+1,:)
        end do
        call unblock(j,jb,tnext)
     end if
  end subroutine unblock
end subroutine modnw
```

К этим пояснениям следует добавить принципиальную необходимость в *рекурсивности* процедуры `unblock`: разблокированный узел может вызвать цепь разблокировок других узлов и соответственно повторные обращения к той же процедуре.

Для отладки блокировок была использована копия блока основной динамики со специально подготовленными исходными данными, выводящими программу сначала на обычное завершение обслуживания, а затем на однократное и двойное деблокирование. Эта копия завершалась программным остановом.

Достойны обсуждения и результаты счета. При предельной емкости узлов `rr`=30 встретились лишь два случая отказа в приеме внешней заявки и четыре акта блокирования. Среднее время пребывания заявки в сети составило 22.04. При уменьшении `rr` до 20 через сеть быстро (в смысле астрономического времени) проходило около 1900 заявок, после чего программа, окончание работы которой управлялось числом прошедших полное обслуживание заявок, зависала. Завершений обслуживания не происходило, новые внешние заявки получали отказ. Эта ситуация иллюстрирует хорошо известный из теории (и практики) операционных систем эффект «смертельных объятий», когда два или более процессов взаимно блокируют друг друга. Причина названного эффекта — не алгоритм моделирования, а алгоритм *работы сети*; мы не входим в его дальнейшее рассмотрение.

По указанным причинам мы перешли к управлению циклом по числу внешних запросов. Ниже воспроизведена соответствующая распечатка для `rr`=15.

```
 zmax=1000, rr=15
  i =          1
  sp=       1.000463
2.666609E-01    2.177942E-01    8.116386E-02    2.828570E-02
1.009264E-02    2.241195E-03    7.466776E-04    1.288333E-04
3.876469E-03    1.405546E-04    2.967208E-04    1.670791E-04
1.389278E-03    5.735862E-04    3.116251E-03    3.837888E-01
  i =          2
  sp=       1.001664
6.791828E-02    1.182255E-01    1.012686E-01    7.134412E-02
6.120515E-02    5.808616E-02    3.016390E-02    1.827593E-02
1.424350E-02    1.124614E-02    1.263812E-02    1.261988E-02
7.345975E-03    5.432472E-03    4.651287E-03    4.069993E-01
  i =          3
  sp=       1.000143
9.597181E-02    2.360979E-01    1.220058E-01    9.353096E-02
6.032599E-02    6.848758E-02    4.079330E-02    4.264247E-02
1.356767E-02    9.198396E-03    9.666163E-03    1.871408E-02
5.440454E-03    3.692727E-02    2.420753E-03    1.443529E-01
  i =          4
  sp=   9.999999E-01
1.692818E-01    2.790169E-01    4.397313E-01    6.441776E-02
3.590020E-02    1.030060E-02    1.351335E-03    0.000000E+00
0.000000E+00    0.000000E+00    0.000000E+00    0.000000E+00
0.000000E+00    0.000000E+00    0.000000E+00    0.000000E+00
 nblk =         10
 ndebl=          2
 vnw  =      20.974900      745.385100    39594.000000
 nref =        358
```

Переменная sp представляет контрольные суммы выведенных маргинальных вероятностей состояний узлов. Большое значение p(15) для первых трех узлов свидетельствует о том, что именно они явились причиной блокировок.

В сетях обслуживания часто встречаются процессы типа «Split and Join» — расщепления и слияния заявок. Примером может служить поступление в систему сложного изделия, блоки которого после разборки проходят по индивидуальным траекториям ремонта и затем поступают

на сборку. Каждый блок при разборке получает двойной номер: исходной заявки и дополнительный индекс; для него определяется номер следующего узла сети. Затем продолжается работа стандартного алгоритма моделирования сети. В точке сборки накапливаются комплекты блоков. При достижении полноты одного из комплектов его компоненты аннулируются и запускается процесс сборки первоначальной заявки. Описанный процесс аналогичен передаче сообщений по сетям связи с коммутацией пакетов.

Дополнительными усложняющими моментами при моделировании сетей связи являются обработка «квитанций» — подтверждений о правильном приеме сообщения или пакета, а также множественность копий пакетов.

2.6 Моделирование вычислительных систем

Современные вычислительные системы являются не только незаменимым средством, но и важнейшим объектом имитационного моделирования, поскольку сложность протекающих в них процессов существенно ограничивает возможности применения к их расчету аналитических методов. Эта сложность порождается:

- иерархичностью построения вычислительных систем: узлы, блоки, устройства, машины, комплексы, локальные и глобальные сети — с возникновением на каждом уровне специфических проблем (здесь уместно вспомнить о многоуровневых протоколах взаимодействия открытых вычислительных сетей);

- сложностью алгоритмов их функционирования (достаточно сослаться на такие режимы, как конвейерная обработка, параллельные вычисления, квантованное обслуживание, режим реального времени, асинхронная обработка потоков данных);

- возможной потребностью заявки в одновременном использовании разнотипных аппаратных и программных ресурсов;

- сложностью и гибкостью операционных систем, развитие которых протекает значительно динамичнее совершенствования аппа-

ратных средств и постоянно ставит все новые проблемы оценки временны́х характеристик, дополнительного расхода памяти, корректности взаимодействия процессов, пропускной способности, надежности, защиты информации;

- необозримым разнообразием применений, со спецификой которых нельзя не считаться, и низкой достоверностью исходных данных для новых машин, алгоритмов и применений;

- включением в автоматизированные системы человека — в сочетании с плохой изученностью работы оператора АСУ;

- особой важностью задач, возлагаемых на ЭВМ в ряде случаев;

- трудностью прогнозирования социальных аспектов как штатных режимов вычислительных центров коллективного доступа, так и в особенности нештатных ситуаций (сошлемся на проблемы разграничения доступа, «виртуальной реальности» и компьютерных вирусов).

Ясно, что давать конкретные рекомендации по столь широкому кругу проблем было бы крайней самонадеянностью.

2.7 О надежности программ моделирования

По поводу техники программирования на *Фортране* 90 мы отсылаем читателя к [8, 9, 59]. Однако о методике разработки и отладки программ *имитационного моделирования* полезно поговорить именно здесь.

Прежде всего отметим сложность логической структуры программ моделирования, выявление которой требует структурированной записи программ, — следовательно, работы в *свободном формате*.

Необходимо по возможности избегать проверок на строгое равенство двух величин, даже если они целочисленные. При автоматическом выборе из списка альтернатив нужно предусматривать контроль невыхода за границы списка.

Укажем далее полезность встраивания в каждую подпрограмму контроля допустимости ее входных параметров с выдачей сигнальной

информации о месте и характере ошибки. Такие вставки рекомендуются современной технологией программирования для всех ответственных программ (в особенности управляющих). Мы изъяли часть таких вставок из окончательного текста приводимых программ для экономии места и концентрации внимания читателя на принципиальных моментах.

Надежность программ повышается при раздельном программировании многофазных операций. Примером может служить завершение обслуживания заявки при наличии в системе очереди. Здесь имеет смысл сначала «освободить» канал, заслав в него «бесконечность», и уже затем отрабатывать его занятие. Аналогично можно поступать и при прибытии заявки, переполняющей очередь: сначала принять ее и увеличить число заявок в системе на единицу, а затем отработать отказ.

Не следует увлекаться чрезмерной рационализацией программы. В частности, в схеме нашей «базовой» модели в зависимости от вида отработанного события предусмотрен возврат в разные точки алгоритма. Более «безопасна» схема, в которой условный оператор обработки различных событий размещен внутри основного цикла, управляемого условием завершения моделирования. В зависимости от типа очередного события можно устанавливать значения управляющей переменной, которая в начале следующего цикла задавала бы обновление только одного подсписка будущих событий. Две последних рекомендации несколько снижают эффективность программ моделирования, но благодаря большей типизации фрагментов и ускорению отладки повышают производительность программирования.

При отладке обязательно использование в основной процедуре метакоманды компилятора *Фортрана* `!MS$DEBUG`, позволяющей выявить случаи неописанных переменных и переменных с не установленными до их использования значениями, а также факты выхода индексов за границы массивов из-за неправильной обработки блокировок. Те же возможности, а также указание номера строки с оператором, вызвавшим «математическую» ошибку при счете, предоставляет указание опций компилятора `/4Yb /Zi`.

В заключение этого «методического» раздела посоветуем исследователю:

- приобрести первоначальный опыт в написании и отладке простых моделей;
- при переходе к более сложным задачам максимально прояснять для

себя их специфику, пользуясь консультациями специалистов;

- искать в литературе, в том числе и в данной книге, не готовые ответы, но полезные аналогии.

Глава 3

Элементы теории и аппроксимации распределений

3.1 Постановка задачи аппроксимации

Уже первое знакомство с основными идеями имитационного моделирования показывает огромную роль в нем реализаций случайных величин, задаваемых своими функциями распределения. Естественно потребовать, чтобы функция распределения:

- находилась в удовлетворительном соответствии с опытными данными;
- была по возможности обоснована, то есть опиралась на вероятностные схемы, отражающие существо изучаемого явления;
- задавалась возможно более простым уравнением;
- была удобной для генерации в соответствии с ней случайных чисел.

Проблема распределений имеет две стороны: описание вероятностных аспектов исходной (моделируемой) ситуации и воспроизведение их статистических аналогов в процессе моделирования (генерация случайных чисел). В этой главе речь пойдет о первой из них.

В общем случае обработка (выравнивание) статистического распределения состоит из следующих этапов:

- построение статистического ряда;
- расчет моментов статистического распределения;
- выбор типа теоретического распределения;
- определение его параметров;
- проверка согласия полученного распределения с опытными данными.

Перечисленные этапы достаточно подробно описаны в любом руководстве по математической статистике. Отметим лишь, что все критерии согласия при корректной обработке статистики имеют малую вероятность отвергнуть неподходящее распределение.

С другой стороны, исходные данные могут быть представлены в уже обработанном виде. При этом также могут возникать проблемы:

- распределение построено по малой выборке;
- имеются только суммарная статистика (моменты, медиана, минимальное и максимальное значения);
- информация представлена качественно (экспертные оценки);
- имеются только агрегированные оценки (например, спрос за месяц вместо данных о ежедневной потребности);
- данные устарели или получены при других условиях;
- выборка цензурирована (отброшена часть данных);
- разрешение выборки недостаточно (погрешность данных соизмерима с дискретностью значений).

При отсутствии опытных данных *вид* распределения обычно постулируется из теоретических соображений — см. известные предельные теоремы о суммировании потоков и случайных величин. Например, при суммировании случайных эффектов следует ожидать нормального распределения результирующей величины, при перемножении — логарифмически нормального, при выборе экстремальных значений — распределения Вейбулла. Случайное прореживание и суммирование рекуррентных потоков приближают результирующий поток к простейшему, регулярное — к регулярному [61].

При переносе имеющихся данных на новые условия чаще всего предполагается сохранение типа распределения и параметров формы. *Масштабные параметры* и параметры *положения распределения* (сдвига) назначаются на базе аналогий с реальными системами с учетом предполагаемых изменений во входной нагрузке, скорости обслуживания, данных об их расчетной надежности и восстанавливаемости и т. п.

3.2 Принципы аппроксимации распределений

Выравнивание статистических распределений, то есть подбор теоретических зависимостей, описывающих фактически наблюдавшиеся данные, обычно проводится по принципу наибольшего правдоподобия или сохранения значений интегральных числовых характеристик распределений, аккумулирующих его основные свойства. В роли таких последних чаще всего выступают начальные моменты

$$f_i \stackrel{\text{def}}{=} \int\limits_0^\infty t^i f(t)\,dt, \qquad i = 1, 2, \ldots$$

Существует обширная литература (см., например, [6]) по проблеме моментов — определению условий, когда заданный набор чисел может служить моментами некоторого распределения (требуется гарантировать нормировку к единице и неотрицательность плотности $f(t)$ для всех t). «Можно ожидать, что если два распределения имеют некоторое число одинаковых моментов, то они в какой-то степени схожи и аппроксимируют друг друга. Обоснованием этого суждения является совпадение их аппроксимаций по методу наименьших квадратов при конечном размахе и разложений по ортогональным многочленам — при бесконечном... Практически аппроксимация такого рода оказывается очень хорошей, даже если совпадают только первые три или четыре момента» [26, с.127]. В той же книге приводятся необходимые и достаточные условия возможности построения характеристической функции, однозначно определяющей распределение во всех точках его непрерывности. Если определитель

$$\begin{vmatrix} 1 & \mu_1 & \dots & \mu_n \\ \mu_1 & \mu_2 & \dots & \mu_{n+1} \\ \dots & \dots & \dots & \dots \\ \mu_n & \mu_{n+1} & \dots & \mu_{2n} \end{vmatrix} \leq 0$$

для всех целых n, то существует распределение с моментами $\{\mu_i\}$ (в случае строгого неравенства — без точек концентрации).

Для единственности распределения на положительной полуоси установлено *достаточное* условие (Карлемана)

$$\sum_{i=0}^{\infty} \mu_i^{-1/2i} = \infty.$$

Если распределение имеет плотность, то необходимо и достаточно условие Крейна

$$\int_{-\infty}^{\infty} \frac{\ln(f(x))}{1+x^2} = -\infty.$$

Эти условия как правило выполняются, хотя известны и отрицательные примеры (логарифмически нормальное распределение). Для него $\mu_r = \sigma^2 r^2/2$. В [84, с. 693–694] приводятся семейство плотностей Хейде (Heyde) и плотность Кендалла-Стьюарта $f(x) = \frac{1}{4}e^{-\sqrt{|x|}}(1 + a\cos(\sqrt{|x|})$ — с параметром $a \in [-1, 1]$, а также еще две *различных* плотности с совпадающими моментами. Наращивание числа учитываемых моментов не всегда сопровождается монотонным улучшением качества аппроксимации.

При всех своих недостатках (чаще потенциальных, чем реальных) метод моментов остается наиболее удобным в работе и применяется чаще других. Количество сохраняемых моментов рассматривается как порядок аппроксимации и равно числу свободных параметров теоретической кривой. Существуют методы автоматического выбора типа выравнивающего распределения в заданном семействе (например, одной из кривых К. Пирсона — по четырем моментам [53, 84]) [1].

Критерии согласия (χ^2 Р. Фишера, $n\omega^2$ Н.В. Смирнова, А.Н. Колмогорова) статистических и теоретических распределений с несколько-

[1] В [84] приводятся еще несколько семейств распределений, автоматически выбираемых по соотношениям между моментами.

ми общими моментами мало чувствительны к виду последних. Это позволяет выбирать теоретические зависимости, дающие те или иные вычислительные преимущества — например, допускающие эффективное вычисление функции распределения, облегчающие расчет функционалов специального вида или генерацию случайных чисел. Именно такие распределения и будут обсуждаться в дальнейшем.

3.3 Показательное распределение

Показательным (экспоненциальным) называется распределение с плотностью

$$f(t) = \mu e^{-\mu t}, \tag{3.3.1}$$

дополнительной функцией распределения

$$\bar{F}(t) = e^{-\mu t} \tag{3.3.2}$$

и начальными моментами

$$f_i = i\,!/\mu^i, \qquad i = 1, 2, \ldots \tag{3.3.3}$$

Примем для определенности, что речь идет о процессе обслуживания. Условная плотность распределения длительности остатка обслуживания

$$\tilde{f}(t|\tau) = \frac{f(t+\tau)}{\bar{F}(\tau)} = \frac{\mu e^{-\mu(t+\tau)}}{e^{-\mu\tau}} = \mu e^{-\mu t}$$

независимо от уже истекшей длительности обслуживания. Говорят, что показательное распределение обладает *марковским* свойством — *отсутствием последействия* (памяти). Параметр потока обслуживаний с функцией распределения $B(t)$

$$\mu(\tau) \stackrel{\text{def}}{=} \lim_{\Delta t \to 0} \frac{B(\tau + \Delta t) - B(\tau)}{\bar{B}(\tau)\Delta t} = -\frac{\bar{B}'(\tau)}{\bar{B}(\tau)} \tag{3.3.4}$$

в случае показательно распределенной длительности равен μ и не зависит от уже истекшего времени τ. Соответственно вероятность завершения обслуживания на малом интервале длины Δt

$$P(t, t + \Delta t) = \mu\Delta t + o(\Delta t)$$

не зависит от положения этого интервала на оси времени.

Отмеченные уникальные свойства показательного распределения делают его исключительно удобным в аналитических выкладках, связанных с описанием процессов обслуживания. Реально такому распределению подчиняется только длительность телефонных разговоров[2]. С другой стороны, процессы *поступления* заявок часто имеют близкое к показательному распределение интервалов между смежными заявками. В особенности это относится к потокам редких событий, число которых на интервале времени описывается распределением Пуассона.

3.4 Распределения фазового типа

Эти распределения порождаются системой подлежащих прохождению фаз обслуживания с показательно распределенной длительностью пребывания в каждой.

3.4.1 Распределение Эрланга

Распределение Эрланга r-го порядка с плотностью

$$f(t) = \frac{\mu(\mu t)^{r-1}}{(r-1)\,!} e^{-\mu t}, \tag{3.4.1}$$

дополнительной функцией распределения

$$\bar{F}(t) = \sum_{i=0}^{r-1} \frac{(\mu t)^i}{i\,!} e^{-\mu t} \tag{3.4.2}$$

и моментами

$$f_k = r(r+1)\cdots(r+k-1)/\mu^k, \qquad k = 1, 2, \ldots \tag{3.4.3}$$

предполагает последовательное прохождение r фаз (см. рис. 3.1) и соответственно является r-кратной сверткой показательного закона.

[2] Классик теории вероятностей В. Феллер ехидно объясняет это тем, что львиную долю загрузки телефонных каналов создают женщины, оставшаяся длительность разговора которых по местной связи не зависит от истекшей длительности.

Рис. 3.1. Эрлангово распределение

Дисперсия эрлангова распределения $D = r/\mu^2$.

Распределение Эрланга — двухпараметрическое, причем параметр r должен быть целым. Это обстоятельство определяет следующую схему подбора его параметров:

1. найти $\tilde{r} = [f_1^2/D + 0.5]$ (ближайшее целое к указанному отношению);

2. вычислить вещественный параметр $\mu = \tilde{r}/f_1$.

При этом строго выравнивается первый момент и приближенно — второй. *Обобщенное* распределение Эрланга имеет различные параметры задержки в фазах.

3.4.2 Гиперэкспоненциальное распределение

Представление ДФР в виде

$$\bar{F}(t) = \sum_{i=1}^{n} y_i e^{-\mu_i t} \tag{3.4.4}$$

позволяет считать исходный процесс проходящим одну из n альтернативных фаз. Здесь $\{y_i\}, \quad 0 \le y_i \le 1, \quad \sum_{i=1}^{n} y_i = 1$, интерпретируются как вероятности выбора i-й фазы, а $\{\mu_i\}$ — как параметры показательного распределения времени пребывания в ней. Диаграмма гиперэкспоненциального распределения приведена на рис. 3.2.

Среди названных величин могут быть комплексные (попарно сопряженные), что подчеркивает фиктивный характер расщепления процесса на фазы. Допустимость таких параметров была впервые отмечена Д. Коксом в 1955 г. [83].

Рис. 3.2. Гиперэкспоненциальное распределение

Применение формулы для моментов показательного распределения приводит к системе уравнений относительно параметров аппроксимации

$$\begin{array}{lllll} y_1 & + & y_2 & +\cdots+ & y_n & = 1, \\ y_1/\mu_1 & + & y_2/\mu_2 & +\cdots+ & y_n/\mu_n & = f_1, \\ y_1/\mu_1^2 & + & y_2/\mu_2^2 & +\cdots+ & y_n/\mu_n^2 & = f_2/2, \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_1/\mu_1^i & + & y_2/\mu_2^i & +\cdots+ & y_n/\mu_n^i & = f_i/i\,!, \qquad i=\overline{0,2n-1}. \end{array} \tag{3.4.5}$$

Для случая $n=2$ задача сводится к квадратному уравнению, а в общем случае решается методом, описанным в книге Р.В. Хемминга [71].

3.4.3 Распределение Кокса

Каждая i-я фаза обслуживания, подчиненного распределению Кокса, требует продолжения обслуживания с вероятностью y_i и завершает его с дополнительной вероятностью $\bar{y}_i = 1-y_i$. Это распределение обобщает классическое эрланговское в том смысле, что допускает случайное число фаз и смену параметра показательного закона от фазы к фазе. На рис. 3.3 показана диаграмма распределения Кокса.

Рис. 3.3. Распределение Кокса

Моменты двухфазного распределения Кокса C_2 могут быть получены из символического разложения

$$f^k = (1-y)a^k + y(a+b)^k,$$

где a и b представляют собой моменты распределения длительности первой и второй фаз обслуживания соответственно. Следовательно, для нахождения параметров y, μ_1, μ_2 распределения C_2 необходимо решить систему уравнений

$$\begin{aligned}
&\frac{1}{\mu_1} && +\frac{y}{\mu_2} && = f_1, \\
&\frac{1}{\mu_1^2} && +y\left(\frac{1}{\mu_1\mu_2} + \frac{1}{\mu_2^2}\right) && = f_2/2, \\
&\frac{1}{\mu_1^3} && +y\left(\frac{1}{\mu_1^2\mu_2} + \frac{1}{\mu_1\mu_2^2} + \frac{1}{\mu_2^3}\right) && = f_3/6.
\end{aligned} \tag{3.4.6}$$

3.4.4 Гиперэрлангово распределение

По аналогии с гиперэкспоненциальным можно построить гиперэрлангово распределение. Наиболее удобна для расчетов смесь двух эрланговских распределений со смежными значениями порядка и общим непрерывным параметром — см. рис. 3.4.

Рис. 3.4. Вариант гиперэрлангова распределения

При данном распределении состояние процесса полностью определяется одной дискретной переменной — номером текущей фазы. Его ДФР

$$\bar{F}(t) = \sum_{i=0}^{r-1} \frac{(\mu t)^i}{i\,!} e^{-\mu t} + y \frac{(\mu t)^r}{r\,!} e^{-\mu t}. \qquad (3.4.7)$$

Параметр r определяется как целая часть отношения f_1^2/D, а два оставшихся параметра y, μ — из системы уравнений

$$\begin{array}{lll} (1-y)\frac{r}{\mu} & +y\frac{r+1}{\mu} & = f_1, \\ (1-y)\dfrac{r(r+1)}{\mu^2} & +y\dfrac{(r+1)(r+2)}{\mu^2} & = f_2. \end{array}$$

Выразив из первого уравнения $\mu = (r+y)/f_1$ и подставив результат во второе, получаем квадратное уравнение относительно y с решением вида $y = d + \sqrt{d^2 + rd}$, где

$$\begin{array}{ll} x & = f_2/f_1^2, \\ r & = [1/(x-1)] \quad \text{(целая часть)}, \\ d & = (r+1)/x - r. \end{array}$$

Эта аппроксимация обеспечивает строгое выравнивание двух первых моментов и приближенное — третьего. В отладочном примере выравнивания гамма-распределения порядка α =2.3 с моментами f_1 =2, f_2 =5.739, f_3 =21.302 третий момент гиперэрлангова распределения составил 21.459. Качество выравнивания высших моментов возрастает с уменьшением коэффициента вариации; однако одновременно растет и число фаз, что увеличивает трудоемкость расчетов.

3.4.5 Распределение Ньютса

Наиболее общей формой представления распределений фазового типа (Ph-распределений) является схема М. Ньютса. Длительность каждой реализации процесса здесь соответствует случайному времени блуждания заявки по изображенной на рис. 3.5 сети с показательно распределенной (параметры $\{\nu_i\}$, $i = \overline{1, M}$) задержкой в узлах и одним поглощающим состоянием.

Рис. 3.5. Распределение Ньютса

Класс Ph-распределений замкнут по отношению к конечным смесям, сверткам, операциям взятия максимума и минимума, схемам соединения рассматриваемых в теории надежности видов. Для любой из этих операций можно по Ph-представлению операндов получить Ph-представление результата.

Подбор параметров сети общего вида из условия выравнивания заданного числа моментов оказывается очень сложным, что затрудняет практическое использование распределений Ньютса.

Специализируя структуру сети, из рис. 3.5 можно получить как частные случаи ранее рассмотренные варианты фазовых распределений.

3.5 Преобразования распределений

Смеси распределений могут быть дискретными:

$$f(x) = \sum_{i=1}^{\infty} \Pr\{Y = i\} f_i(x)$$

(обычно сумма конечна) и непрерывными:

$$f(x) = \int f_y(x) g(y)\, dy.$$

Часто используются *отношения* стандартно распределенных величин. Пусть, например, даны две показательных плотности

$$f_1(x) = \lambda e^{-\lambda x}, \quad f_2(y) = \mu e^{-\mu y}.$$

Полагая $Z = X/Y$, имеем $X = YZ$. Функция распределения

$$F(z) \stackrel{\text{def}}{=} \Pr\{Z < z\} = \int\limits_0^\infty (1 - e^{-\lambda yz})\mu e^{-\mu y}\, dy.$$

Соответственно плотность распределения

$$f(z) = \int\limits_0^\infty \lambda y \mu e^{-\mu y}\, dy = \lambda\mu \int\limits_0^\infty y e^{-y(\lambda+\mu)}\, dy = \frac{\lambda\mu}{(\mu + \lambda z)^2}.$$

Отношение E_1/E_2 стандартных показательно распределенных величин с единичным средним имеет плотность распределения $1/(z+1)^2$.

Свертки распределений приводят к суммированию средних и дисперсий. При свертке Г-распределений с одинаковым масштабным параметром суммируются параметры формы.

Если $X_1, \ldots X_n$ имеют одинаковые ФР $F(x)$, то функция распределения *максимума* из них есть $F^n(x)$, а распределение *минимума* $T_n(x) = 1 - [1 - F(x)]^n$. Можно показать [84], что положение i-го элемента упорядоченного ряда имеет плотность

$$\frac{n!}{(i-1)!(n-i)!} F^{i-1}(x)(1 - F(x))^{n-i} f(x).$$

Глава 4

Генерация случайных чисел

4.1 Схема получения случайных чисел

Практическое моделирование систем массового обслуживания на ЭЦВМ требует большого количества случайных чисел (интервалы между заявками, длительности обслуживания, продолжительность ремонта отказавшего устройства, допустимые времена ожидания, запрашиваемые очередной задачей объемы оперативной памяти). Статистикам нужны генераторы для испытания оценок прежде, чем использовать оценки в реальной жизни. В исследовании операций случайные числа есть ключевая компонента масштабного моделирования. В компьютерных науках случайность нужна при испытании программ, в играх, для сравнения алгоритмов.

Задачи на составные распределения, генерацию случайных структур и т. п. более характерны для процессов с постоянным шагом по оси времени и в схеме однофакторного слежения, чем в схеме «от события к событию».

Первичные данные должны быть получены наблюдением за работой реальной системы (например, для вычислительной установки типа ЕС — посредством универсального средства трассировки УСТ). В модели они могут использоваться либо непосредственно (в «сыром» виде), либо через датчики случайных чисел (ДСЧ), воспроизводящие их статистические аналоги. Применение «сырых» данных обеспечивает наилучшее приближение к фактически наблюдавшемуся процессу, однако при этом:

- не гарантируется типичность данных;
- длительность моделируемого процесса ограничивается длительностью реального;
- модель лишается прогностической силы, поскольку для проектируемых систем или измененных вариантов построения и использования реальных систем такие данные отсутствуют;
- исключается применение методов уменьшения дисперсии результатов, основанных на использовании специально подобранных серий случайных чисел.

По указанным причинам непосредственное использование «сырых» данных в процессе моделирования производится только для верификации модели. В практике моделирования почти исключительно применяются ДСЧ, формирующие случайные числа с нужным законом распределения.

Получение случайных чисел с требуемым законом распределения обычно выполняется в два этапа:

1. Формирование физическим или программным методом случайного числа U_i, равномерно распределенного на $[0,1)$, $i = 1, 2 \ldots$.
2. Программный переход от U_i к случайному числу X_i, имеющему требуемое распределение $F_X()$.

В программах некоторых современных ДСЧ предусматривается генерация за одно обращение *массива* случайных чисел. Тогда эффективное быстродействие датчиков увеличивается из-за уменьшения числа обращений к подпрограмме.

Доверие к выработанной серии случайных чисел возрастает, если они прошли специализированный тест — упрощенный вариант модели исходной системы, для которой известно аналитическое решение. Таким тестом при решении задач ТМО может служить, например, хорошо изученная схема аналитического расчета системы типа $M/M/1$.

Генераторы оценивают по качеству формируемой последовательности, быстродействию, трудоемкости инициализации, машинной независимости, диапазону применений, простоте и понятности для пользователей. Обстоятельное сопоставление ряда генераторов для различных распределений проведено, например, в [108]. Простой и понятный алгоритм

при прочих равных условиях будет жить дольше и применяться чаще. Обычно простые алгоритмы медленнее более хитрых. Любой алгоритм, будучи переписан на языке ассемблера, даст значительный выигрыш в быстродействии (по данным [62] для генератора равномерных чисел — вдвое).

Скорость генерации можно увеличить, применяя приближенные методы вычисления обратной функции распределения. Напомним также о целесообразности генерации *массивов* чисел с требуемым распределением.

Вес различных показателей работы ДСЧ определяется ситуацией. При их разработке нежелательны приемы, специфические для конкретных систем программирования (как common-блоки *Фортрана*), однако их применение может оказаться вполне оправданным (те же common-блоки — для *настройки* автономных ДСЧ).

Алгоритмы генерации ниже описываются в программоподобном стиле, позволяющем сочетать типовые управляющие конструкции *Паскаля* и *Фортрана* с привычной математической записью выражений и определением смысла происходящего на естественном языке. Для большей наглядности начало их отмечается знаком $\bigtriangledown$, а конец посредством $\bigtriangleup$.

Алгоритмы обеспечивают точное или приближенное следование требуемому распределению. Термин «точное» является условным и предполагает пренебрежение ограниченностью машинного слова, округлениями при арифметических операциях и погрешностями вычисления стандартных функций ($\exp, \log, \mathrm{sqrt}, \sin, \cos, \arctan$). Приближенные алгоритмы помимо перечисленных источников погрешности имеют дополнительные, порождаемые применяемыми аппроксимациями.

4.2 Генерация равномерно распределенных чисел

4.2.1 Физические и программные датчики

Равномерно распределенное на $[0, 1)$ случайное число представляется в ЭЦВМ в двоичной форме в виде n-разрядной последовательности нулей и единиц, причем в каждом разряде нуль или единица должны наблюдаться с вероятностью 0.5.

Физические датчики равномерно распределенных на [0,1) чисел состоят из n идентичных по своим параметрам триггеров со счетным входом, каждый из которых регистрирует независимый поток импульсов от счетчика радиоактивных частиц или шумовые выбросы электронной лампы. Такой поток можно считать простейшим. При интенсивности потока λ и интервале между снятиями отсчета Δt вероятности появления в одном разряде нуля или единицы различаются между собой на $e^{-2\lambda\Delta t}$, и при достаточно большом произведении $\lambda\Delta t$ (и идеальной работе счетчика) разбаланс может быть сделан сколь угодно малым. Многочисленные примеры схем физических ДСЧ приводятся, например, в [107] и в [47, с. 70–74].

Достоинствами физических ДСЧ являются:

- истинная случайность получаемых чисел (если считать это свойство достоинством);
- исключение затрат процессорного времени на генерацию случайных чисел.

Схемная нестабильность физических датчиков, в частности неустранимая нестабильность элементов, реализующих механизм квантования, вызывает необходимость введения аппаратурного и математического контроля датчика, существенно усложняя его математическую эксплуатацию. Работа каждого экземпляра физического датчика нуждается в *периодической* проверке.

Программные ДСЧ фактически генерируют псевдослучайные числа. Все ДСЧ этого класса обеспечивают близкое к равномерному перемешивание разрядов исходных чисел некоторым закономерным способом. Согласно Лемеру (Lehmer, [24, с. 452]), мы можем считать последовательность псевдослучайных чисел случайной, если «каждый ее член непредсказуем для непосвященного и она удовлетворяет ряду традиционных статистических тестов, в некоторой степени зависящих от цели выработки последовательности». Специалисты по теории автоматического управления рассматривают программные ДСЧ как неустойчивые автоколебательные системы ([47, с. 76–77]), причем переходный режим, нежелательный при обычном использовании автогенератора, является здесь единственно приемлемым.

Программные ДСЧ имеют следующие преимущества:

- отсутствие дополнительного оборудования;
- возможность повторения прогона с той же последовательностью случайных чисел в целях контроля вычислений, уменьшения дисперсии или сравнительного анализа вариантов;
- необходимость лишь однократной проверки ДСЧ после его разработки.

Заметим, что иногда равномерность может быть важнее случайности.

Работа как физического, так и программного равномерного датчика оценивается по согласию статистического распределения $\{U_i\}$ с теоретическим — прежде всего по равномерности заполнения r-мерного единичного гиперкуба точками $\eta_1 = (U_1, U_2, \ldots, U_r)$, $\eta_2 = (U_{r+1}, U_{r+2}, \ldots, U_{2r})$, ... (при r=1 — отрезка $[0,1)$)[1].

Согласие фактических распределений с ожидаемыми устанавливается, например, по критерию χ^2. Дополнительно проверяется отсутствие корреляции между последовательными числами и отдельными разрядами чисел.

В [47], с. 88 и далее, обсуждаются статистические тесты равномерности: по заполнению *всех* подобластей гиперкуба, по заполнению выбранных частей (покер-тест), «шаровой» тест (вероятность попадания в гиперсферу единичного диаметра), тест серий (из нулей или единиц), проверка корреляционной связи между последовательными числами. Наглядным средством визуального контроля для двумерного случая является равномерность засветки на экране единичного квадрата.

Отсутствие корреляции не означает отсутствия статистической зависимости вообще, однако при формировании нормально распределенных величин эти понятия эквивалентны.

Полезен тест *крайних значений*: здесь нужно убедиться, что распределение максимума серии из n чисел $F_n(x) = x^n$.

В книге [62] описан ряд *программ* тестирования случайных последовательностей для ЕС ЭВМ.

Поскольку в n-разрядной сетке количество различных двоичных чисел равно 2^n (для машин с 32-разрядным словом $\approx 4 \cdot 10^9$),

[1] При необходимости генерации многомерных случайных величин с повышенными требованиями к равномерности используются специальные *последовательности Соболя* [62].

отрезки последовательности различных $\{U_i\}$ рано или поздно начнут *повторяться*. Тестирование программного ДСЧ должно быть произведено на сериях различной длины вплоть до длины периода (может получиться так, что тесты для периода в целом дают хороший результат, а для отдельных его частей — неудовлетворительный). Подобную проверку должна пройти каждая новая программа, предлагаемая к использованию на ЭЦВМ данного типа. Использовать в одной задаче количество чисел, превышающее длину отрезка апериодичности L, не рекомендуется. Поэтому программист должен знать длину периода для применяемой им программы ДСЧ. При необходимости в большем количестве чисел применяются методы «возмущения» последовательности $\{U_i\}$ — например, с помощью другой программы ДСЧ. Перетасовка нарушает любую корреляцию и значительно увеличивает период генератора. При случайной перетасовке нельзя получить число по номеру, не сгенерировав все предшествующие.

Потенциальным недостатком программных ДСЧ является опасность *вырождения* — получения на некотором шаге чисто нулевого кода, последующие преобразования которого дадут опять же нули.

В настоящее время используются почти исключительно программные ДСЧ и программные же преобразователи их к требуемым распределениям. В последующих разделах этой главы обсуждаются основные типы таких преобразователей.

4.2.2 Классические конгруентные генераторы

Исторически первым программным ДСЧ явился метод «середины произведения», в котором в качестве очередного числа берется середина произведения двух n-разрядных чисел, а вторым сомножителем служат оставшиеся разряды. Этот метод дает невысокое качество случайных последовательностей и склонен к вырождению, однако долгое время применялся, например, в системе моделирования $GPSS$ [74]. Как этот, так и другие используемые методы выработки псевдослучайных чисел состоят в выборе некоторой функции f, отображающей множество целых чисел в себя:

$$U_{k+1} = f(U_k), \qquad k = 0, 1, \ldots$$

Если правило вычисления x_{k+1} зависит только от x_k, то длина периода не превосходит количества M различных чисел. На основе теории чисел

можно выбрать такую функцию f, для которой период близок к максимальному, а вырождение исключено. Чаще всего применяют функцию вида

$$U_{k+1} = (\lambda U_k + c) \pmod M, \qquad k = 0, 1, \ldots$$

(«линейный конгруэнтный генератор» Лемера), где для n-разрядных целых чисел $M = 2^n$. В книге Д. Кнута [30] следующим образом подытожены рекомендации по выбору параметров таких генераторов:

1. Начальное нечетное число U_0 может быть произвольно.
2. Выбор λ должен удовлетворять трем требованиям:
   - $\lambda \pmod 8 = 5$,
   - $M/100 < \lambda < M - \sqrt{M}$,
   - двоичные знаки λ не должны иметь очевидного шаблона.
3. В качестве λ следует выбирать нечетное число, такое, что

$$\lambda/M \approx 1/2 - \sqrt{3}/6 = 0.21132.$$

Согласно [47, с. 109]), минимум коэффициента корреляции последовательных псевдослучайных чисел обеспечивает выбор мультипликативной константы $\lambda \approx \sqrt{M}$.

Обычно поставляемый пакет SSP программ для научных расчетов содержит подпрограмму RANDU с λ=65539=$2^{16}+3$ и =0, дающую крайне высокую корреляцию между тремя подряд идущими целыми числами. От этого недостатка свободна подпрограмма URAND (см. приложение к [57]). Перед первым обращением к URAND целой переменной IY следует присвоить произвольное начальное значение.

Рекомендуемая в [108] программа также имеет стартовую установку 65539. Предыдущее целое число умножается на 69069 и добавляется 1. U-число из интервала [0,1] получаем извлечением 24 левых битов и умножением на 2^{-24}.

Во всех равномерных датчиках, обсуждаемых в [62], множитель $\lambda = 5^{13} = 1220703125$.

Еще один вариант линейного конгруэнтного генератора — выбор в качестве λ иррационального числа, большего 1, с оставлением в результате дробных разрядов. В частности, в [78] рекомендуется $\lambda = (\sqrt{5}+1)/2$.

В одном из генераторов [108] множитель λ получается умножением 2^k (k — разрядность) на «золотое сечение» $(\sqrt{5}-1)/2$ и выбором в качестве λ ближайшего целого числа, удовлетворяющего условию $\lambda \pmod 8 = 5$. Таковым оказалось 1073741821. Приведем *Фортран*-программу этого генератора:

```
      real function randw
*************************************
*  Датчик псевдослучайных чисел,      *
*  равномерно распределенных на [0,1) *
*       <<Золотой>> множитель         *
*************************************
      integer*4        ix,iy
      real*8
      data ix /65539/
      iy=ix*1073741821
      if (iy.le.0) iy=iy+2147483647+1
      ix=iy
      rn=dreal(iy)
      rn=rn*4.656613e-10
      randw=rn
      end !! randw
```

Переполнение разрядной сетки приводит к появлению единицы в знаковом разряде, и число интерпретируется как отрицательное. Этот эффект устраняется в условном операторе добавлением (за два шага) $2^{31}-1$ и единицы. Результат используется как исходный множитель при следующем обращении, а для получения $U \in [0,1]$ переводится в вещественную форму и нормируется 2^{31} (для повышения быстродействия деление заменено умножением на обратную величину).

Согласно [108], для мультипликативного генератора $X_{i+1} = (\lambda X_n) \pmod M$ и $M = 2^m$ максимальный период есть 2^{m-2} и достигается для λ, равных 3 или 5 по модулю 8 и нечетных стартовых значений. Генерируемые последовательности суть перестановки следующих значений:

Таблица 4.1. Мультипликативные генераторы

Тип	$\lambda \pmod 8$	X_0	Члены последовательности	
I	3	1,3,9,11 $\pmod{16}$	$8\nu+1, 8\nu+3$	$\nu = 0, 1, \ldots, 2^{m-3}-1$
II	3	5,7,13,15 $\pmod{16}$	$8\nu+5, 8\nu+7$	$\nu = 0, 1, \ldots, 2^{m-3}-1$
III	5	1 $\pmod 4$	$4\nu+1$	$\nu = 0, 1, \ldots, 2^{m-2}-1$
IV	5	3 $\pmod 4$	$4\nu+3$	$\nu = 0, 1, \ldots, 2^{m-2}-1$

Метод со сложением работает по формуле $X_{i+1} = (\lambda X_n + \mu) \pmod M$. Для $M = 2^m$ максимальный период есть 2^m и достигается для λ, равных 1 по модулю 4 и $\mu = 1$. Генерируются перестановки значений $0, 1, \ldots, 2^m - 1$.

Если U_{k+1} зависит только от U_k, то длина периода не превосходит количества M различных чисел. Если же $U_{k+1} = f(U_{k-1}, U_k)$, то максимальный период равен M^2. Поэтому лучшие результаты дают генераторы высших порядков вида

$$U_{i+n} = (a_1 U_{i+n-1} + a_2 U_{i+n-2} + \ldots a_n U_i) \pmod M.$$

Их частным видом являются *генераторы Фибоначчи*, обсуждаемые и пропагандируемые, например, в [24]. В программе `UNI`

$$x_k = x_{k-17} - x_{k-5}.$$

Работа с датчиками подобного типа предполагает их инициализацию: задание базовых целых значений и заполнение закольцованного стартового массива.

Ниже приводится текст написанного на *Фортране* 90 датчика, разработанного Дж. Марсалья и А. Заманом в университете штата Флорида (1987 г.), процедура его инициализации и тестовая программа. Авторы считают датчик «наилучшим из известных». Он имеет период $2^{144} \approx 10^{43}$ и дает одинаковые результаты на всех машинах, имеющих 24-битовую мантиссу при операциях с плавающей точкой. Базовые целые `ij, kl` должны принадлежать к диапазону $[0, 30000]$ и выбираются независимо, что позволяет иметь 900 миллионов различных псевдослучайных последовательностей.

```
program tstran
   real     temp(100)
   integer  ij,kl,len
```

```
  !  Инициализация
     ij=1802; kl=9373
     call rmarin(ij,kl)
     len=100
  !  Генерация 20000 случайных чисел
     do i=1,200
        call ranmar(temp,len)
     end do
     len=18
     call ranmar(temp, len)
     write(6,20) (temp(i),i=1,18)
20   format (3e12.5)
  end

  subroutine rmarin(ij,kl)
   ! Это процедура инициализации генератора ranmar()
     real       u(97),c,cd,cm
     integer    i97, j97
     logical    test
     data test  /.false./
     common /raset1/ u,c,cd,cm,i97,j97,test

     if (ij<0 .or. ij>31328 .or. kl<0 .or. kl>30081) then
        print 'Начальные установки должны быть &
             & между нулем и 30000'
        stop
     end if

     i=mod(ij/177,177)+2; j=mod(ij,177)+2
     k=mod(kl/169,178)+1; l=mod(kl,169)

     do ii=1,97
        s=0.0
        t=0.5
        do jj=1,24
           m=mod(mod(i*j,179)*k,179)
           i=j; j=k; k=m
           l=mod(53*l+1,169)
```

```
            if (mod(l*m,64) .ge. 32) s=s+t
            t=0.5*t
         end do
         u(ii)=s
      end do

      c =362436.0  /16777216.0
      cd=7654321.0 /16777216.0
      cm=16777213.0/16777216.0

      i97=97; j97=33
      test=.true.
      return
   end

   subroutine ranmar(rvec,len)
      real    rvec(*)
      real    u(97),c,cd,cm
      integer i97,j97
      integer ivec
      logical test
      common /raset1/ u,c,cd,cm,i97,j97,test

      if (.not. test) then
         print 'Сначала вызовите процедуру &
              & инициализации rmarin'
         stop
      end if

      do ivec=1,len
         uni=u(i97)-u(j97)
         if (uni<0.0) uni=uni+1.0
         u(i97)=uni;  i97=i97-1
         if (i97 .eq. 0) i97=97
         j97=j97-1
         if (j97 .eq. 0) j97=97
         c=c-cd
         if (c<0.0) c=c+cm
```

```
        uni=uni-c
        if( uni<0.0) uni=uni+1.0
        rvec(ivec)=uni
      end do
      return
   end
```

Получены следующие контрольные результаты:

```
.38945E+00  .84759E+00  .43363E+00
.36789E+00  .49797E+00  .63379E+00
.33967E+00  .61337E+00  .89652E+00
.28297E+00  .66277E-01  .94529E+00
.56366E+00  .79408E+00  .88443E+00
.57736E+00  .44619E+00  .50157E+00
```

Наконец, возможно введение дополнительного перемешивания, при котором сначала заполняется таблица из M чисел. Далее генерируется целое число $i \in [0, M-1]$, выдается i-й элемент таблицы, а на его место записывается новое случайное число. Если выбрать, например, M =128, то можно на каждом шаге формировать только одно случайное число, а в качестве входа в таблицу использовать семь младших разрядов этого числа.

4.2.3 Генераторы Смита

В книге М. В. Антипова [5, с. 18] идеи Д. Кнута подвергнуты резкой критике: «Кнут обратил свой замысел в пародию, получив вырожденную короткопериодическую последовательность». Предложенные в [5] линейные конгруентные генераторы используют принципиально иной подход к выбору модуля и множителей, основанный на числах Смита. Модулем конгруэнтного генератора служит простое число P, а *множителем* — его первообразный корень. Первообразным корнем простого числа P называется целое число N, $1 < n < N$, если наименьшая степень k, при которой верно сравнение $N^k \pmod P = 1$, равна $P-1$. Количество первообразных корней равно количеству чисел, меньших значения $P-1$ и взаимно простых с ним. При указанном выборе множителя обеспечивается максимальная длина периода, равная $P-1$.

Поскольку для $P \geq 3$ число $P-1$ четное, наибольшее относительное количество первообразных корней будет достигнуто при $P-1 = 2P_1$,

где P_1 — также простое число. *Числом Смита* P_s называется большее из двух простых чисел, связанных соотношением $P_s = 2P + 1$.

Разработка генератора с заданными свойствами статистической многомерной равномерности требует:

- поиска числа (чисел) Смита;
- поиска первообразных корней — претендентов на роль множителей генератора;
- многомерного корреляционного анализа для отбора множителя, дающего последовательность наивысшего статистического качества.

Из элементарных соображений следует, что числа Смита, исключая начальные значения 3, 5 и 7, удовлетворяют соотношению $P_s \equiv 11 \pmod{12}$, и количество их падает относительно количества простых чисел. В [5] показано, что для ДСЧ на 31 разряд число Смита не должно превосходить величины $(2^{31})^{2/3} \approx 1.7 \cdot 10^6$. Тогда переполнения целых не произойдет. Необходимость отбора первообразных корней и множителей из них несколько уменьшает величину возможного модуля P_s. После определения и задания числа Смита ведется поиск первообразных корней, начиная с $[2^{31}/P_s]$, в сторону уменьшения. Из них отбираются множители, отвечающие наиболее качественному генератору.

В [5] практически проверялась автокоррелированность последовательности в диапазоне размерностей k единичного гиперкуба от единицы до 100. Множители отбирались по четырем характеристикам: минимаксная, гильбертова, квадратичная и показательная. Ниже (табл. 4.2) приведено извлечение из итоговых данных:

Таблица 4.2. Рекомендуемые параметры генераторов Смита

Число Смита	Наилучшие множители
100067	1580, 1894, 2118
300023	5682, 6525, 6754, 6790, 7000
1000919	1373, 2076, 2084
1001003	1430, 1856, 2000, 2063

К отобранной паре P_s, λ, компоненты которой обозначены целыми (MP,L), следует добавить начальное целое число $M1$, $1 \le N1 \le P_s - 1$, и нормирующий множитель $A = P_s^{-1}$.

```
      real function rands
************************************
*  Датчик Смита случайных чисел,      *
*  равномерно распределенных на [0,1) *
************************************
      integer*4        ix,iy,jx
      real             rn
      data ix /12345/
      jx=ix*1580
      iy=mod(jx,100067)
      rn=dreal(iy)
      rn=rn*9.993304e-6  !! =/100067d0
      ix=iy
      rands=rn
      end !! rands
```

Этот датчик соответствует первому числу Смита и первому множителю для него из табл. 4.2. Переход к другой последовательности возможен заменой начального числа, множителя при том же модуле или числа Смита.

Произвольный объем последовательности псевдослучайных чисел может обеспечить только *система* генераторов с различными модулями При комбинировании нескольких ДСЧ период равен произведению периодов составляющих. Улучшаются и свойства равномерности. Опишем предлагамую М. В. Антиповым [5] и исправленную автором схему комбинированного использования нескольких смитовских генераторов. Пусть выбраны числа Смита $\{P_i\}$ и соответствующие множители $\{\lambda_i\}$, $i = \overline{0, n}$. Комбинированный рекуррентный генератор задается системой

$$
\begin{aligned}
M_0 &= (M_n\lambda_0) \pmod{P_0},\\
M_1 &= (M_0\lambda_1) \pmod{P_1},\\
&\dots\dots\dots\dots\dots\dots\\
M_i &= (M_{i-1} * \lambda_i) \pmod{P_i},\\
&\dots\dots\dots\dots\dots\dots\\
M_n &= (M_{n-1} * \lambda_n) \pmod{P_n}.
\end{aligned}
$$

Начальное значение M_n полагается равным единице. Выходом генератора является отношение M_n/P_n.

Ниже приводится программа мультигенератора, использующая все числа Смита и первые множители из табл. 4.2.

```
 real function rndsmith
************************************
*      Мультигенератор Смита       *
*      псевдослучайных чисел,      *
* равномерно распределенных на (0,1) *
************************************
      integer*4      M(0:3),L(0:3),P(0:3),NN
      real           rn
      data           P /100067,300023,1000919,1001003/,  &
                     L /1580,5682,1373,1430/, M(3)/1/, NN/3/
         M(0)=mod(M(NN)*L(0),P(0))
      do i=1,NN
         M(i)=mod(M(i-1)*L(i),P(i))
      end do
      rn=dreal(M(NN))
      rn=rn/P(NN)
      randsm=rn
      end !! rndsmith
```

В описанной форме датчик требует $NN+1$ операций умножения и вычисления вычета на каждое результирующее число. Этого недостатка можно избежать, применив common-блок с циклическими перестановками в нем.

Согласно [5], конкретная реализация системы на ПЭВМ с модулями P_s порядка 10^6 и числе компонент порядка 100 обеспечивает период более 10^{600}. Это шестая степень так называемого «гугола», превосходящего число элементов наблюдаемой вселенной.

В заключение этого раздела приведем сводную табл. 4.3 тестирования результатов тестирования четырех описанных датчиков по среднему и (для наглядности) двенадцатикратной дисперсии в сериях из 20 тыс. испытаний. Среднее и дисперсия вычислялись по рекуррентным формулам (5.1.5) и (5.1.6).

Таблица 4.3. Сравнение генераторов равномерных чисел

Показатель	Базовый	«Золотой»	Марсальи	Смита	Мульти
Среднее	0.497237	0.498356	0.498654	0.500293	0.485587
$12 * D$	0.999839	1.003325	1.002483	0.996183	1.023732

Таким образом, все обсуждаемые датчики эти необходимые, но не достаточные проверки выдержали.

4.2.4 Раздельные датчики

Принципиальным условием применения методов понижения дисперсии результатов моделирования является независимость случайных величин каждого типа (для модели СМО — интервалов между заявками, длительностей обслуживания, типов заявок, узлов-преемников в сетях обслуживания и т. п.). Оно достигается использованием для их генерации раздельных ДСЧ. Для удобства реализации датчики должны быть однотипны. Как правило это генераторы мультипликативного вида с общим множителем, но различными начальными значениями. Требуется гарантировать непересекаемость генерируемых серий, но конструктивные рекомендации на этот счет в литературе отсутствуют.

Упомянутую проблему можно решить, если разделить период генератора (для хороших ДСЧ это 2^{m-2}, где m — разрядность датчика) на требуемое число серий и взять в качестве начальных значений числа с соответствующими номерами. Для получения этих чисел можно применить алгоритм ускоренного получения случайного числа, основанный на двоичном разложении его номера k:

```
subroutine fastrand(k,p)
   integer k,p
   integer x0,z
   integer j,j1,j2
   data x0 /57539/

   j=k
   p=x0
   z=1220703125
   do while(j.gt.0)
```

```
            j1=j/2
            j2=j-2*j1
            if (j2.eq.1) then
               p=p*z
               if (p.le.0) p=(p+2147483647)+1
            end if
            z=z*z
            if    (z.le.0) z=(z +2147483647)+1
            j=j1
         end do
      end
```

Здесь как очередной «полуфабрикат» p, так и степень множителя z копятся по модулю датчика 2^{31}. Возведение в степень выполняется на каждом шаге, а домножение p — в среднем на половине шагов, что дает для трудоемкости получения k-го числа оценку $\frac{3}{2}\log_2 k$ умножений. Заметим, что двоичный логарифм *миллиарда* составляет всего 27. Разумеется, алгоритм применим к любым начальным значениям и множителям и (после очевидной модификации) — к любому модулю датчика.

Идея этого алгоритма некогда была предложена А. П. Ершовым для реализации возведения в степень в Альфа-трансляторе; применительно к имитации она использовалась Ю. Г. Полляком [47], но была изложена неотчетливо и не связывалась с разнесением серий чисел. Правильность приведенной программы подтверждается совпадением результатов прямой и ускоренной генерации для заданного номера. Приведем первые 11 чисел, полученных с шагом номера в 5 млн.

Таблица 4.4. Начальные установки ДСЧ

k	Число	k	Число
0	57539	6	236093123
1	1664722371	7	1325220803
2	921004739	8	2119895235
3	268709827	9	824954307
4	60159171	10	2087686851
5	647674307	11	1965447107

Эта таблица иллюстрирует только возможность легкого получения «весьма далеких» чисел. Нужно иметь в виду, что многие датчики тестировались во времена машин с быстродействием на несколько порядков мень-

ше, чем у современных — следовательно, в «ближней» зоне, и должное качество их работы в дальней не гарантируется.

При практическом использовании датчиков целые числа для получения $\{U_i\} \in [0, 1)$ делятся на модуль датчика, а лучше — умножаются на обратную ему величину. На большой выигрыш в быстродействии от этой замены рассчитывать не стоит. Согласно [42], трудоемкость умножения вещественных чисел по отношению к присваиванию целых оценивается в 5 единиц, деление — в 9, а вычисление стандартных функций — в 150. По данным [98, с. 527], при генерации показательно распределенных чисел 72% времени занимает вычисление логарифма.

Из полученного набора начальных установок по степени согласия результатов моделирования с эталонными можно выбрать наилучшую комбинацию датчиков. В частности, было проверено 50 комбинаций, для которых средние времена ожидания изменялись от 6.02 до 12.20. Наилучшей оказалась комбинация четвертой и шестой установок, давшая результат 8.9788. При ее поиске была использована построенная по 30 точкам грубая «карта» поверхности отклика.

4.3 Метод обратной функции

4.3.1 Идея метода и точные обращения

Универсальным способом перехода к требуемому распределению $F(x)$ случайной величины является метод обратной функции.

Рис. 4.1. Метод обратной функции

На рис. 4.1 показана его графическая реализация. Здесь U — случайное число, равномерно распределенное на интервале [0,1). Таким образом, из уравнения $F() = U$ определяется

$$X = F^{-1}(U). \tag{4.3.1}$$

Показательное распределение имеет ФР $F(x) = 1 - e^{-\lambda x}$, и (4.3.1) превращается в $1 - e^{-\lambda X} = U$, откуда $X = -\ln(1-U)/\lambda$. Поскольку $(1-U)$ имеет то же распределение, что и U, удобнее применять формулу

$$X = -\frac{1}{\lambda}\ln U. \tag{4.3.2}$$

Натуральный логарифм правильной дроби отрицателен, так что расчет дает положительную величину. Сразу же предупредим авторов программ имитационного моделирования о часто встречающейся при работе с этой формулой ошибке — потере минуса. Ее следствием является обратный ход системного времени, катастрофически искажающий логику модели.

Обычные стандартные программы предусматривают вычисление $\ln U$ с высокой точностью, как правило излишней при имитации случайных воздействий. Поэтому целесообразно использовать более простые и быстрые вычислительные схемы. Представляя случайное число U в нормализованном виде как $U = m \cdot 2^p$, можно аппроксимировать $\ln U = -\ln 2(p - 2.6797 + 4.0391m - 1.3594m^2)$ с погрешностью, не превышающей по абсолютной величине 0.001 [47, с. 167]. Вообще чем бóльшую часть времени моделирования составляет генерация случайных чисел и чем менее точны данные о параметрах распределений, тем более оправданы наиболее простые и быстродействующие генераторы.

Все реальные равномерные датчики генерируют псевдослучайные числа, равномерно распределенные на полуинтервале [0,1), причем единица исключается, но существует *конечная* (для каждой программы своя) вероятность получить *нулевое* значение. Последнее в ряде алгоритмов преобразования распределений может привести к аварийному завершению счета (проверено на личном опыте). Поэтому упомянутое упрощение следует применять с осторожностью.

Замкнутые СМО работают с переменной интенсивностью потока. В этом случае при любом изменении числа заявок в системе (в том числе завершении обслуживания) заново генерируется время прибытия очередной заявки. Такая ситуация характерна для задач о восстанавливаемом ЗИП и о вычислительных системах коллективного пользования.

Распределение Релея с функцией распределения

$$F(x) = 1 - e^{-x^2/2\sigma^2} \tag{4.3.3}$$

приводит к генератору вида

$$X = \sigma\sqrt{-2\ln U}. \tag{4.3.4}$$

Распределение Вейбулла имеет функцию распределения

$$F(x) = 1 - \exp(-x^k/T) \tag{4.3.5}$$

и моменты

$$f_m = T^{m/k}\Gamma(1 + m/k), \quad k = 1, 2, \ldots \tag{4.3.6}$$

Соответственно оказывается, что

$$X = \sqrt[k]{-T\ln(1 - U)}. \tag{4.3.7}$$

Распределение Коши с медианой μ и масштабом σ имеет плотность

$$f(x) = \sigma/[\pi(\sigma^2 + (x - \mu)^2)]$$

и функцию распределения

$$F(x) = \frac{1}{2} - \frac{1}{\pi}\arctan\frac{x - \mu}{\sigma}.$$

Получаем генератор

$$X = \sigma\tan(\pi(U - 1/2)) + \mu.$$

Закону Коши подчинено, в частности, распределение отношения нормально распределенных величин. Это распределение *не имеет моментов* (!!!) и потому полезно при «стрессовом» тестировании модели.

Логистическое распределение общего вида имеет ФР

$$F(x) = \frac{1}{1 + e^{-(x-a)/b}}.$$

Соответственно имеем уравнение $U[1 + e^{-(X-a)/b}]$ с решением

$$X = a - b\ln(1/U - 1).$$

У стандартного логистического распределения $a = 0$ и $b = 1$, так что
$F(x) = (1 + e^{-x})^{-1}$. Его плотность

$$f(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = F(x)(1 - F(x)),$$

среднее есть $2 \ln 2$, а второй момент — $\pi^2/3$. Можно показать, что $X = \ln[U/(1 - U)]$.

Метод обратной функции непосредственно применим к нахождению максимума и минимума из n одинаково распределенных величин. Для максимума основное уравнение имеет вид $F_n(x) = F^n(x) = U$, откуда

$$X = F^{-1}(U^{1/n}).$$

Аналогичным образом может решаться задача о распределении минимума: $T_n(x) = 1 - [1 - F(x)]^n = U$. Имеем $1 - F(x) = (1 - U)^{1/n}$. Поскольку U и $1 - U$ распределены одинаково, последнее равенство сводится к $F(x) = 1 - U^{1/n}$, так что

$$X = F^{-1}(1 - U^{1/n}).$$

Приведем сводную таблицу 4.5 генераторов этого класса для наиболее популярных распределений:

Таблица 4.5. Метод обратной функции

Тип	Плотность	$F(x)$	$X = F^{-1}(U)$	Упрощенно
Экспонента	$\lambda e^{-\lambda x}$	$1 - e^{-\lambda x}$	$-\frac{1}{\lambda}\ln(1 - U)$	$-\frac{1}{\lambda}\ln U$
Коши	$\sigma/\pi(x^2 + \sigma^2)$	$\frac{1}{2} + \frac{1}{\pi}\arctan(x/\sigma)$	$\sigma\tan[\pi(U - 1/2)]$	$\sigma\tan[\pi U]$
Релея	$\frac{x}{\sigma}e^{-x^2/2\sigma^2}$	$1 - e^{-x^2/2\sigma^2}$	$\sigma\sqrt{-\ln(1 - U)}$	$\sigma\sqrt{-\ln U}$
Треугольное на $[0, a]$	$\frac{2}{a}(1 - x/a)$	$\frac{2}{a}(x - x^2/2a)$	$a(1 - \sqrt{1 - U})$	$a(1 - \sqrt{U})$
Парето, $x \geq b$	$ab^\alpha/x^{\alpha+1}$	$1 - (b/x)^\alpha$	$b/(1 - U)^{1/\alpha}$	$b/U^{1/\alpha}$

Более обширная таблица способов генерации некоторых распределений: Релея, Райса, χ^2, Стъюдента, Фишера, бета, гамма (фактически — Эрланга) приведена в [47, с. 170–172]. Из более современных источников укажем [12, 98, 84].

Метод обратной функции — единственный истинно универсальный метод генерации случайных чисел с требуемым распределением.

4.3.2 Численное обращение

Для *численного* обращения применяются методы половинного деления, секущих, Ньютона — Рафсона. Первый и второй методы сходятся всегда, метод Ньютона — только при постоянстве знака второй производной $F(x)$. Универсальных правил остановки алгоритма нет. Выбор алгоритма зависит от:

- гарантий сходимости,
- скорости сходимости,
- априорной дополнительной информации,
- знания плотности (она нужна в методе Ньютона).

4.3.3 Аппроксимация обратной ФР

Многие современные пакеты для решения статистических и математических задач предлагают как готовые аппроксимации функций распределения и им обратных (обычно в виде дробно-линейных функций), так и средства для их построения. Кроме того, в них имеются и готовые подпрограммы генерации случайных чисел (до двух десятков типов распределений). К сожалению, заставить совместно работать такой пакет и серьезную программу на универсальном языке высокого уровня — дело не простое.

Решение уравнений вида (4.3.1) для получения каждого нового числа X (а их нужны десятки и сотни тысяч) требует больших затрат машинного времени, в особенности если аналитическую формулу типа (4.3.2) получить не удается. Поэтому широко применяются различные приближенные табличные методы. Наиболее употребительный из них состоит в кусочно-линейной аппроксимации обратной функции распределения. В системе моделирования GPSS до недавних пор функция $\ln(1-U)$ воспроизводилась этим способом по 24 точкам. При проверке ее на 5000 случайных точек максимальный модуль относительного отклонения составил 0.09, а среднеквадратическое отклонение — 0.007. Трудоемкость этой аппроксимации оказывается довольно значительной — на 34% больше, чем при обращении к логарифмической функции.

Основные затраты на реализацию линейных аппроксимаций связаны с поиском интервала опорных значений, в который попадает очередной аргумент U. В процедуре XPDIS [74] в этих целях применен простейший метод — последовательный просмотр, трудоемкость которого для таблицы длины n в среднем составляет $n/2$. При двоичном поиске среднее число попыток уменьшается до $[\log_2 n]+1$. Если (при заданном числе узлов) пойти на некоторое ухудшение качества аппроксимации, то можно получить таблицу с вычисляемым входом. В этом случае отрезок $[0,1]$ разбивается на n равных и, следовательно, равновероятных частей, номер интервала определяется как $[nU]+1$, а относительное расстояние точки от левой границы интервала — как дробная часть произведения nU. Трудоемкость обращения к такой таблице существенно меньше и к тому же не зависит от выбранного числа узлов.

Часто для уменьшения суммарной трудоемкости операций обращения предварительно строится *аппроксимация* $g(U) = F^{-1}(U)$. Для *нормального* распределения известна аппроксимация Оде и Ивэнса

$$g(u) = t + A(t)/B(t),$$

где $t = \sqrt{-2\ln(U)}$, а коэффициенты многочленов A и B при соответствующих степенях t для $10^{-20} \le U \le 1/2$ приведены в табл. 4.6.

Таблица 4.6. Аппроксимации нормального закона

i	a_i	b_i
0	-0.322232431088	0.0993484626060
1	-1.0	0.588581570495
2	-0.342242088547	0.531103462366
3	-0.0204231210245	0.103537752850
4	-0.0000453642210148	0.0038560700634

Для диапазона $1-10^{-20} \ge u \ge 1/2$ берется $-g(1-u)$.

Приведем еще две аппроксимации для стандартного нормального распределения: согласно [47], можно пользоваться формулами

$$X = \frac{2.30753 + 0.27061t}{1 + 0.99229t + 0.04481t^2} \tag{4.3.8}$$

с погрешностью не более $3 \cdot 10^{-3}$ и

$$X = -\frac{2.515517 + 0.802853t + 0.0103288t^2}{1 + 1.432788t + 0.189269t^2 + 0.001308t^3} \tag{4.3.9}$$

с погрешностью не более $4.5 \cdot 10^{-4}$. Здесь переменная $t = \sqrt{-2 \ln U'}$, а $U' = \max\{U, 1-U\}$.

Простую аппроксимацию, работающую во всем диапазоне значений аргумента, обычно подобрать не удается. В таких случаях применяется кусочно-линейная аппроксимация.

На «хвосте» распределения обычные полиномиальные аппроксимации часто работают плохо. Можно оставить «естественный хвост». Пусть для экспоненциального распределения получено равномерно распределенное число $U > 1 - \Delta$. Тогда можно записать

$$U = 1 - \Delta + \Delta[1 - e^{-\lambda(X - x^*)}] = 1 - \Delta e^{-\lambda(X - x^*)}, \qquad (4.3.10)$$

где $x^* = F^{-1}(1 - \Delta)$. Решая уравнение (4.3.10) относительно X, получаем

$$X = x^* + \frac{1}{\lambda} \ln \frac{\Delta}{1 - U}.$$

При этом подходе трудоемкое логарифмирование придется выполнять в $1/\Delta$ раз реже, чем при использовании основной формулы. За это приходится платить предварительной работой по подбору аппроксимаций для диапазона $[0, 1 - \Delta]$. Ту же идею можно применить и для генерации переменных, распределенных по закону Релея.

Недостатки применения аппроксимаций:

- метод неприменим, если F часто меняется;
- если ФР заменяется отношением полиномов или применяется кусочно-полиномиальная аппроксимация, надо хранить много коэффициентов;
- точность аппроксимации фиксирована, и при необходимости ее повышения надо менять всю функцию;
- возникают трудности при аппроксимации многопараметрических семейств.

Неудобством табличных методов является необходимость построения новых таблиц для всех комбинаций параметров моделируемого распределения даже при сохранении его типа. Исключениями здесь являются показательное и нормальное распределения, для которых переход

от стандартного к произвольному распределению производится линейным преобразованием — см. формулу (4.3.2). Последнее обстоятельство оправдывает применение для этих распределений в их стандартной форме более сложных аппроксимаций.

4.3.4 Дискретные распределения

В табл. 4.7 сведены наиболее употребительные дискретные распределения.

Таблица 4.7. Дискретные распределения

Название	$P(X=i)$	Диапазон
Пуассоново ($\lambda > 0$)	$e^{-\lambda}\lambda^i/i!$	$i \geq 0$
Биномиальное (n,p)	$\binom{n}{i}p^i(1-p)^{n-i}$	$i=\overline{0,n}$
Отрицат. биномиальное, $n \geq 1, p)$	$\binom{n+1-i}{i}p^i(1+p)^{n+i}$	$i \geq 0$
Логарифм. ряд, $0<\theta<1)$	$-\theta^i/i\ln(1-\theta)$	$i \geq 1$
Геометрическое, $0<p<1$	$p(1-p)^{i-1}$	$i \geq 1$

Для формирования дискретных случайных величин непрерывная функция распределения заменяется ступенчатой кумулянтой. Метод *последовательных сравнений* является дискретным аналогом метода обратной функции. Он заключается в переборе значений X, пока не окажется

$$F(X-1) = \sum_{i<X} p_i < U \leq \sum_{i \leq X} p_i. \tag{4.3.11}$$

При этом $\Pr(X=i) = F(i) - F(i-1) = p_i$. Метод неудобен при длинных хвостах. Возможная модификация его — половинное деление при работе с таблицей функции распределения. Тот же метод можно применить для случайного розыгрыша возможных событий, если эти события предварительно перенумеровать и упорядочить по убыванию соответствующих вероятностей.

Пуассоново распределение формируется по следующему алгоритму:

$\triangledown$ $X=0$; $b=\exp(-\lambda)$; $s=b$.

Сформировать равномерно распределенное U.

Пока $U > s$, выполнять

$$X = X + 1\,;\ b = b * \lambda/X\,;\ s = s + b\,.$$

Конец цикла.

Вернуть X. $\triangle$

Аналогично можно работать с прочими распределениями (меняются начальные установки и правила пересчета):

биномиальное:

$$b = b * \frac{n - X}{X + 1}\frac{p}{1 - p};$$

отрицательное биномиальное:

$$b = b * \frac{n + i}{i + 1}p(1 + p);$$

логарифмическое: в данном случае

$$E[i] = ap/(1 - p), \quad D[i] = ap/(1 - p)^2;$$

пересчет идет по формуле $c = c * \theta$ (член ряда представляется в виде $b = c/i$);

геометрическое: $b = b * (1 - p)$.

Если расчет вероятностей частных значений оказывается слишком трудоемким, можно использовать базовую идею в табличной форме. Поскольку эти значения упорядочены, для поиска удовлетворяющего условию (4.3.11) X можно применить метод половинного деления. При работе с вероятностями целесообразно переупорядочить таблицу по убыванию вероятностей $\{p_i\}$. Еще один вариант — использование функции расстановки для обратной ФР.

4.4 Аппроксимация плотностей

Рассмотрим для примера график показательной плотности — рис. 4.2.

Разобьем диапазон значений аргумента на n равных отрезков длины a и «хвост». Каждую из положенных на бок криволинейных трапеций разобьем на прямоугольник и «клин» и подсчитаем вероятности:

Рис. 4.2. Аппроксимация показательной плотности

для i-й трапеции $\pi_i = \int\limits_{(i-1)a}^{ia} e^{-x}\,dx = e^{-(i-1)a} - e^{-ia}$,

для i-го прямоугольника $\tilde{\pi}_i = ae^{-ia}$,

для i-го клина $\pi_i = \pi_i - \tilde{\pi}_i$.

Наконец, вычислим вероятность хвоста $\pi_t = e^{-na}$. Разобьем интервал $[0,1]$ на соответствующие отрезки и перенумеруем из заново, отведя индексы $i = \overline{1,n}$ для прямоугольников, $i = \overline{n+1,2n}$ для клиньев и $i = 2n+1$ — для хвоста.

Первый шаг генерации EXP случайного числа состоит в отображении равномерно распределенного U_1 на этот интервал, что обеспечивает выбор i-го фрагмента аппроксимации.

На *втором* шаге выбирается новое равномерно распределенное U_2. Дальнейшая обработка зависит от вида фрагмента. При $i \le n$ (прямоугольник) выходная переменная $X = a(i-1+U_2)$.

При $n < i \le 2n$ (клин) условная плотность распределения $\varphi(x) = \frac{2}{a}(1-x/a)$, а функция распределения $\Phi(x) = \frac{2}{a}(x - x^2/2a)$. Из условия $\Phi(X_2) = U_2$ находим $X_2 = a(1-\sqrt{U_2})$. Эту величину нужно добавить к левой границе интервала, то есть к $(i-n-1)a$.

Наконец, при $i = 2n+1$ (хвост) считаем, что U_2 равномерно распределена на суммарной вероятности хвоста e^{-na}, откуда отсчитываемый от границы na $X_2 = -\ln U_2$. Итак, $X = na - \ln U_2$. Логарифмирование необходимо только при попадании на хвост и при достаточ-

но большом произведении na будет требоваться редко. Плата за это — большой объем хранимой информации.

Возможен вариант алгоритма без логарифмирования: в этом случае при попадании на хвост к генерируемым в дальнейшем значениям X добавляется na, умноженное на количество таких попаданий, и U_2 генерируется заново.

На *третьем* этапе алгоритма полученная величина X масштабируется: делится на параметр экспоненциального распределения.

Плотность распределения может быть также представлена как дискретная смесь элементарных плотностей:

$$f(x) = \sum_{i=1}^{\infty} p_i f_i(x),$$

где $\{p_i\}$ — вероятностный вектор, $\{f_i(x)\}$ — легко генерируемые через U плотности (треугольные и равномерные). Типичен случай

$$f(x) = \sum_{i=1}^{k-2} p_i f_i + p_{k-1} f_{k-1} + p_k f_k,$$

где сумма представлена прямоугольниками, соответствует центральной части распределения, обращается в 0 вне $[a, b]$ и близка к 1. Левая часть аппроксимируется f_{k-1}, в случае J-образной плотности сильно заострена и генерируется обсуждаемым ниже методом режекции. Последнее слагаемое представляет собой хвост распределения.

Расссмотрим пример для нормальной плотности с $k = 3$. Ясно, что вероятность p_2 должна быть максимально велика. Так как $p_2 f_2(x) \le 1$, наибольшее p_2 должно удовлетворять условию

$$p_2 \le \inf_x \frac{f(x)}{f_1(x)}.$$

Поскольку $\int_{-\theta}^{\theta} f_2(x)\,dx = 2k\theta$, где θ — полуширина центрированного прямоугольника и k — его высота, имеем $k = 1/(2\theta)$. Теперь мы можем положить

$$p_2 = \inf_{|x| \le \theta} \frac{2\theta e^{-x^2/2}}{\sqrt{2\pi}} = 2\frac{\theta}{\sqrt{2\pi}} e^{-\theta^2/2}.$$

Эта вероятность максимальна при $\theta = 1$, так что $p_2 = \sqrt{2/\pi e}$. Она довольно далека от 1, и этот расчет просто иллюстрирует метод.

4.5 Метод режекции

4.5.1 Идея метода

Если плотность распределения не интегрируется или обращение интеграла требует сложных вычислений, можно применить предложенный фон Нейманом метод режекции (отбрасывания). Здесь для плотности $f(x)$ строится возможно более близкая к ней мажоранта $\varphi(x)$. В процессе моделирования для случайного числа U вычисляются $X = \Phi^{-1}(U)$ и затем Y, равномерно распределенный на интервале $[0, \varphi(X)]$. Если $Y < f(X)$, то X принимается как реализация случайной величины с плотностью $f(x)$; в противном случае попытка повторяется. Процент успешных попыток определяется близостью мажоранты к исходной плотности. Иллюстрирующая эту идею схема представлена на рис. 4.3.

Рис. 4.3. Схема метода режекции

Исходная плотность распределения $f(x)$ заменяется более удобной плотностью $g(x)$, в соответствии с которой легко генерируются числа $\{X_i\}$. В класс типовых g включаются равномерное, треугольное и другие. Плотность g должна иметь более «толстый» хвост и (в случае J-образной плотности) более острый бесконечный пик, чем f. Далее вычисляется константа

$$c = \sup_x f(x)/g(x)$$

или устанавливаются возможно более точные неравенства.

Алгоритм генерации числа, распределенного в соответствии с $f(x)$:

▽ Повторять:

сформировать X с плотностью g и равномерно распределенное U;

положить $T = cg(x)/f(x)$

до выполнения $UT \le 1$.

Выдать X. $\triangle$

Количество попыток имеет геометрическое распределение, параметр которого равен вероятности успеха $p = 1/c$. Среднее число попыток $N = 1/p = c$.

По утверждению [84, с. 43], метод режекции (в терминологии других источников — метод огибающей) «имеет почти безграничный потенциал».

4.5.2 Логарифмически вогнутые плотности

В книге [84] специально рассматриваются *логарифмически вогнутые* плотности, логарифмы плотностей которых имеют отрицательную вторую производную. Для них рекомендуется метод режекции с мажорантой специфического вида (равномерная в районе моды и экспоненциальные хвосты). Специальные алгоритмы применяются для плотностей, удовлетворяющих условию Липшица порядка α, то есть

$$\sup_{x \ne y} \frac{|f(x) - f(y)|}{|x - y|^\alpha} \le C,$$

$0 < \alpha < 1$ (при других α функция $f(x)$ не может быть плотностью).

Если $f(x)$ логарифмически вогнута и максимальна в нуле, то, опираясь на неравенство $1 - F(x) \le e^{-xf(0)}$, имеем [84, с. 325]

$$f(x) \le \frac{\sqrt{8C}}{f(0)} \left[\frac{f(0)}{2} e^{-xf(0)/2} \right].$$

Соответственно метод режекции для них:

$\triangledown$ Повторять:

Сформировать E_1, E_2.

Положить $X = 2E_1/f(0)$

до выполнения $-E_2 - E_1 \le \ln[f(X)/\sqrt{2C}]$.

Выдать X. $\triangle$

Условие можно преобразовать к виду $-E_2 - E_1 \le -\ln[\sqrt{2C/f(X)}]$, или $E_1 + E_2 \ge \ln[\sqrt{2C/f(X)}]$. Вообще показательно распределенные числа весьма полезны при генерации по другим законам.

4.6 Метод композиции

Пусть плотность $f(x)$ можно представить в форме

$$f(x) = \int\limits_{-\infty}^{\infty} g_z(x)\, dH(z),$$

где мы знаем, как генерировать числа для $g(x)$ и $H(z)$. Тогда необходимо:

1. Сформировать пару (U_1, U_2).
2. Определить $Z = H^{-1}(U_1)$.
3. Вычислить $X = G^{-1}(Z, U_2)$.

Возможны смешанные варианты генераторов случайных чисел.

4.7 Генерация по заданной функции интенсивности

Интенсивность некоторого распределения

$$h(x) \stackrel{\text{def}}{=} f(x)/\bar{F}(x).$$

Соответственно

$$H(x) = \int\limits_{0}^{x} h(y)\, dy = -\ln \bar{F}(x).$$

Применив метод обратной функции, убеждаемся, что

$$X = H^{-1}(E),$$

где E — показательно распределенное число. В самом деле,

$$\Pr\{H^{-1}(E) \le x\} = \Pr\{E \le H(x)\} = 1 - e^{-H(x)} = F(x).$$

Примеры этого подхода сведены в табл. 4.8:

Таблица 4.8. Генераторы по функции интенсивности

Название	Плотность	$h(x)$	$H(x)$	$H^{-1}(E)$
Вейбулла	$ax^{a-1}e^{-x^a}$	e^{-x^a}	ax^{a-1}	$(E/a)^{1/(a-1)}$
Парето	$a/(1+x)^{a+1}$	$a/(1+x)$	$a\ln(1+x)$	$e^{E/a}-1$
Степенно́е	$ax^{a-1},\ x \le 1$	$ax^{a-1}/(1-x^a)$	$-\ln(1-x^a)$	$(1-e^{-E})^{1/a}$

4.8 Генерация по моментам

Приведем алгоритм генератора случайных величин, сохраняющего четыре момента [84]. Для настройки генератора необходимо вычислить среднеквадратическое отклонение $\sigma = \sqrt{\mu_2 - \mu_1^2}$ и перейти к нормированным высшим моментам:

$$\begin{aligned} \mu_3' &= (\mu_3 - 3\mu_2\mu_1 + 2\mu_1^3)/\sigma^3, \\ \mu_4' &= (\mu_4 - 4\mu_3\mu_1 + 6\mu_2\mu_1^2 - 3\mu_1^4)/\sigma^4. \end{aligned}$$

Далее рассчитываются вспомогательные параметры

$$\begin{aligned} q &= \frac{1}{2}\left(1 + \sqrt{(\mu_4' - 1)/(\mu' 4 + 3)}\right), \\ p &= \frac{1}{2}\left(1 + \frac{\mu_3'}{\mu_4' - 1}\right). \end{aligned}$$

Собственно генерация требует следующих действий:

1) Получить U.

2) Если $(U \le p)$, то

сформировать $X = I(U \le pq)$ (распределение Бернулли по указанному условию);

рассчитать $X = \mu_1 + \sigma(X - q)/\sqrt{q(1-q)}$,

иначе

получить $X = I(U \le p + (1-p)q)$ (распределение Бернулли с другим условием);

вычислить $X = \mu_1 - \sigma(X - q)/\sqrt{q(1-q)}$.

конец условия.

3) Выдать X.

Если известен r-й абсолютный момент распределения μ_r, то на основе неравенства Чебышева для всех r

$$1 - F(x) \le \frac{\mu_r}{|x|^r}.$$

Отсюда

$$f(x) \le \sqrt{2C} \min\left\{1, \frac{\sqrt{\mu_r}}{|x|^{r/2}}\right\}.$$

Оно полезно только для $r > 2$. Интеграл от мажоранты есть $\sqrt{8C}\frac{r}{r-2}\mu_r^{1/r}$. Какое из r лучше, зависит от распределения: $r/(r-2)$ монотонно убывает по r, тогда как $\mu_r^{1/r}$ есть неубывающая функция (это следует из неравенства Иенсена).

4.9 Частные методы

Здесь мы рассмотрим ситуации, когда для моделирования случайной величины непосредственно воспроизводится вероятностная схема, порождающая интересующее нас распределение.

4.9.1 Непрерывные распределения

Многие сложные СМО могут быть рассчитаны численно после аппроксимации исходных распределений фазовыми. Поэтому моделирование с применением фазовых распределений часто необходимо для верификации аналитических результатов. Рассмотрим соответствующие способы генерации.

Эрланговское распределение с параметрами λ и r есть распределение суммы r чисел, получаемых согласно (4.3.2). Следовательно,

$$X = \sum_{k=1}^{r} \left(-\frac{1}{\lambda} \ln U_k \right) = -\frac{1}{\lambda} \sum_{k=1}^{r} \ln U_k = -\frac{1}{\lambda} \ln \prod_{k=1}^{r} U_k. \qquad (4.9.1)$$

Последний вариант уменьшает число сравнительно трудоемких операций логарифмирования.

Гамма-генераторы. Так как сумма гамма-распределенных величин с одинаковым масштабным параметром вновь имеет Г-распределение с тем же масштабным параметром и суммарным параметром формы, то целая часть последнего обеспечивается сверткой показательных законов (то есть генерируется случайная величина, распределенная по закону Эрланга соответствующего порядка), и остается прибавить случайную величину с параметром $0 < \alpha < 1$. В справочнике [12] предлагается (к сожалению, без обоснования) следующий алгоритм для этой последней задачи:

$\triangledown$ Повторять:

сформировать равномерно распределенные U_1, U_2;

вычислить $S_1 = U_1^{1/\alpha}$, $S_2 = U_2^{1/(1-\alpha)}$

до выполнения $S_1 + S_2 \le 1$;

сформировать равномерно распределенное U_3;

вычислить

$$X = -\frac{S_1 \ln U_3}{\lambda(S_1 + S_2)}.$$

Выдать X. $\triangle$

Гиперэрлангово распределение с возможным обходом первой фазы может быть получено в двухэтапном процессе. На первом этапе при $U > y$, где y — вероятность полнофазного обслуживания, число фаз процесса Эрланга уменьшается на единицу. Второй этап аналогичен предыдущему случаю.

Гиперэкспоненциальное распределение независимо от его порядка (числа составляющих) требует двух обращений к равномерному ДСЧ. С помощью числа U_1 выбирается номер экспоненты i (см. рекомендации разд. 4.3 по выбору дискретных случайных величин), а посредством U_2 формируется согласно (4.3.2) величина, имеющая показательное распределение с параметром λ_i.

Распределение Кокса C_r генерируется в соответствии со схемой рис. 3.3. На каждой из r его фаз из ранее накопленной суммы вычитается $-\ln(U_i)/\mu_i$ и проверяется $U_i' > y_i$ (кроме последней фазы). При выполнении этого условия формирование результата завершается.

Распределение Ньютса является естественным обобщением коксова. Здесь число шагов не ограничивается, и на каждом шаге процесса определяется номер следующего узла. Если этим узлом оказывается $(M+1)$-й, процесс завершается.

Ниже приведены внешние подпрограммы для моделирования многоканальных приоритетных систем. Процедура `ftz` генерирует показательно распределенные случайные числа с различными интенсивностями (второй параметр — тип заявок). В процедуре предусмотрены три сменных оператора `data`, из которых два переводятся в комментарии с помощью восклицательного знака в начальной позиции строки, а третий обеспечивает поддержание заданного суммарного коэффициента загрузки при текущем числе каналов.

```
      real*8 function ftz(tt,j)
         real*8  tt,x,lambda(3)
         integer j
!        data lambda /0.3, 0.45, 0.6/     ! n=1
!        data lambda /0.6, 0.9, 1.2/      ! n=2
         data lambda /0.9, 1.35, 1.8/     ! n=3
         call random_number(x)
         ftz=tt-log(x)/lambda(j)
      end
```

Процедуры генерации длительностей обслуживания являются сменными (к главной программе прилагается комплект, из которых одна рабочая, а остальные временно «закомментированы» описанным выше способом). Поскольку моделируется система с абсолютным приоритетом, здесь согласно гл. 2 формируются *длительности* обслуживания, а не моменты его завершения.

```
      real*8 function flib(j)
! Настройка на E3-обслуживание
        real*8  mu(3),x(3),p
        integer i
        data mu /10.0, 5.0, 3.333/
        call random_number(x);
         p=1.0
         do i=1,3; p=p*x(i); end do
        flib=-log(p)/mu(j)
     end

!     real*8 function flib(j)
! Настройка на регулярное обслуживание
!        real*8  mu(3),x
!        call random_number(x)
!        data mu /0.3,0.6,0.9/
!        flib=mu(j)
!     end
```

Треугольное распределение дается формулой

$$f(x) = -|x-\mu|/T^2 + 1/T, \quad \mu - T \le x \le \mu + T.$$

Сумма двух псевдослучайных чисел, равномерно распределенных между $(\mu - T)/2$ и $(\mu + T)/2$, имеет треугольное распределение. Слагаемые формируются как $S = a + (b-a)U$, соответственно сумма $S = \mu - T + T(U_1 + U_2) = \mu + T(U_1 + U_2 - 1)$. Заменяя U_2 так же распределенным дополнением его до единицы, имеем $S = \mu + T(U_1 - U_2)$.

Напомним, что это распределение является чрезвычайно полезным при реализации мажорант метода режекции.

Нормальное распределение описывается плотностью

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu^2)/2\sigma^2}.$$

Известно,что распределение суммы n одинаково и независимо распределенных случайных величин со средним значением $z^{(1)}$ и дисперсией $d^{(1)}$ при $n \to \infty$ стремится к нормальному распределению с параметрами $z^{(n)} = nz^{(1)}$ и дисперсией $d^{(n)} = nd^{(1)}$. Достаточно хорошая сходимость наблюдается уже при $n > 6$.

Для равномерно распределенных на $[0, 1)$ случайных чисел $z^{(1)} = 1/2$ и $d^{(1)} = 1/12$. Следовательно, случайная величина

$$X(n) = \sqrt{12/n}\left(\sum_{i=1}^{n} U_i - n/2\right) \tag{4.9.2}$$

распределена приближенно нормально, имеет нулевое математическое ожидание и единичную дисперсию. Для формирования нормально распределенной величины со средним x и среднеквадратическим отклонением σ нужно выполнить преобразование $X = x + \sigma X(n)$. Этот метод не является ни точным (распределение упомянутой суммы является нормальным лишь *асимптотически*), ни эффективным. Известно, что такой подход дает плохое представление для больших уклонений от среднего, превышающих по модулю 2σ [73]. Л. Н. Большев показал, что значительно ближе к нормально распределенной величина

$$Y = X(5) - \frac{3X(5) - X^3(5)}{100}.$$

Существует способ получения *пары* стандартных нормально распределенных чисел с помощью пары равномерных. Вспомним, что распределение Релея (4.3.3) описывает радиальное отклонение точки попадания от центра прицеливания при равной дисперсии нормальных отклонений по осям x и y. Метод Бокса и Мюллера дает $N(0, 1)$ случайные числа (распределенные нормально с нулевым средним и единичной дисперсией) согласно

$$\begin{aligned} Z_1 &= W\cos(2\pi U_2), \\ Z_2 &= W\sin(2\pi U_2), \end{aligned}$$

где $W = \sqrt{-2\ln U_1}$. Его недостаток — трудоемкость расчета тригонометрических функций. В «полярном» алгоритме Марсальи отмеченное свойство используется в сочетании с методом режекции:

- Сформировать $U_1,\ U_2$, равномерно распределенные на [-1,1].
- Вычислить $S = U_1^2 + U_2^2$.
- Если $S > 1$, перейти к этапу 1.
- Вычислить $W = \sqrt{-2\ln S/S}$.
- Выдать $X = W \cdot U_1,\ Y = W \cdot U_2$.

Здесь вероятность отбрасывания $p = 1 - \pi/4 \approx 0.215$. Если числа принимаются, то $U_1/\sqrt{S}$ и $U_2/\sqrt{S}$ играют роль синуса и косинуса соответственно.

Предложено также множество трапецеидальных аппроксимаций нормальной плотности для схемы режекции.

Логарифмически нормальное распределение с плотностью

$$f(x) = \frac{\exp[-(\ln(x) - \mu)^2/2\sigma^2]}{x\sigma\sqrt{2\pi}}$$

генерируется как экспонента от нормально распределенного $N(\mu, \sigma^2)$.

Распределение χ^2 задается плотностью

$$f(x) = \frac{(x/2)^{F/2-1}}{2\Gamma(F/2)} e^{-x/2}.$$

Здесь F — число степеней свободы. Искомое число $\xi = \sum\limits_{i=1}^{F} \xi_i^2$, где числа $\{\xi_i\}$ суть стандартизованные нормальные. Аппроксимации этого распределения основаны на идеях:

1. Фишера: $\sqrt{2x} - \sqrt{2F-1} \to N(0,1)$.
2. Уилсона и Хилферти: $(X/F)^3 \to N(1 - 2/(9F), \sqrt{2/(9F)})$.

Нормальные числа генерируются по алгоритму Марсальи.

Стьюдентово распределение имеет плотность

$$f(t) = \frac{\Gamma(\frac{N+1}{2})(1 + t^2/N)^{-(N+1)/2}}{\sqrt{\pi N}\Gamma(N/2)}.$$

Величина $t = z/\sqrt{Y_N/N}$ распределена по Стьюденту, если $z \in N(0,1)$ и Y_N распределена по χ^2. Соответственно в прежних обозначениях

$$\zeta = \xi_0 \Big/ \sqrt{\frac{1}{F}\sum_{i=1}^{F} \xi_i^2}.$$

Фишера распределение с плотностью

$$f(t) = \frac{m^{m/2} n^{n/2} t^{m/2-1}}{\mathrm{B}(\frac{m}{2}, \frac{n}{2})(mt+n)^{(m+n)/2}}$$

широко используется в дисперсионном анализе и описывает отношение нормированных числом степеней свободы дисперсий двух выборок из $N(0,1)$-распределенной популяции с m и n степенями свободы. Соответственно определяется алгоритм генерации

$$\zeta = \sum_{i=1}^{m+1} \xi_i^2 \Big/ \sum_{i=m+2}^{m+n+2} \xi_i^2.$$

Бета-распределение имеет плотность

$$f(x) = x^{p-1}(1-x)^{q-1}/\mathrm{B}(p,q),$$

где p, q — вещественные положительные числа, обычно целые. Для целых и полуцелых p, q $\xi = Y_m/(Y_m + Y_n)$, где Y_m, Y_n распределены по закону χ^2 с соответствующим числом степеней свободы, причем $m = 2p$ и $n = 2q$.

Коши-распределение задается плотностью

$$f(x) = \frac{1}{\pi}\frac{\sigma}{\sigma^2 + (x - x_0)^2}.$$

Для генерации этого распределения используется отношение двух нормально распределенных величин

$$\zeta = \xi/\eta + x_0,$$

где $\xi = N(0,\sigma)$, $\eta = N(0,1)$.

4.9.2 Дискретные распределения

Бернуллиево распределение имеет бинарная $\{0,1\}$ величина, принимающая значение 1 с вероятностью p. Здесь X получает значение 1 при $U \le p$ и 0 — в противном случае.

Геометрическое распределение

$$P_r = p(1-p)^{r-1}$$

описывает число испытаний до первого успеха, который в каждой из независимых попыток достигается с вероятностью p. Бернуллиевы попытки выполняются до получения первого числа $U \le p$; X равен полному числу попыток.

Биномиально распределенное значение определяется вероятностями

$$P_r = \binom{N}{r} p^r (1-p)^{N-r}.$$

Это вероятность точно r успехов в N попытках. Таким образом, следует сформировать N чисел $\{U_i\}$ и подсчитать, сколько из них не превысило p.

Сумма биномиально распределенных величин с общим параметром p распределена биномиально с суммарным параметром n. При $n \to \infty$ и p =const биномиальное распределение стремится к нормальному, при поддержании np=const — к пуассонову. Эти тенденции могут использоваться для полезных аппроксимаций.

Гипергеометрическое распределение задается вероятностями

$$P_r = \binom{M}{n}\binom{N-M}{n-r} \Big/ \binom{N}{n}.$$

Здесь аргумент удовлетворяет условию $\max(0, n-N+M) \le r \le \min(n, M)$, а параметры $n \in [1, N]$, $N, M \ge 1$. Распределение описывает опыт, где элементы извлекаются *без возвращения*. Точнее, имеются N элементов, из которых M обладают некоторым свойством, а остальные — нет. P_r есть вероятность того, что при извлечении ровно n элементов r из них будут обладать упомянутым свойством.

Алгоритм генерации непосредственно реализует это определение:

$\triangledown$

1. Установить счетчики:

$r = 0$ — число выбранных элементов с атрибутом,

$MR = M$ — число оставшихся элементов с атрибутом,

$NR = N$ — общее число оставшихся элементов,

$NP = n$ — общее число выбранных элементов.

2. Сформировать целое число k, равномерно распределенное между 1 и NR.

3. Если $k < MR$, то элемент имеет атрибут; тогда положить $r = r+1$ и $MR = MR - 1$.

4. Принять $NR = NR - 1$ и $NP = NP + 1$.

5. Если $MR > 0$ и $NP \neq n$, перейти к шагу 2.

6. Закончить алгоритм со значением r. $\triangle$

Распределение Пуассона. На интервале $[0, t]$ происходит ровно K событий пуассоновского потока интенсивности λ, если сумма K показательно с параметром λ распределенных случайных чисел меньше t, а сумма $K + 1$ таких чисел — больше t. Таким образом, K есть наибольшее целое число, для которого

$$-\frac{1}{\lambda}\sum_{k=1}^{K}\ln U_k < t$$

или — в более удобной форме —

$$\prod_{k=1}^{K} U_k > e^{-\lambda t}.$$

При известном среднем λt здесь не нужны ни логарифмирование, ни вычисление экспоненты (исключая установку датчика). Возможна работа с предварительно сгенерированным случайным временем T.

Составной пуассоновский поток генерируется с одновременным моделированием объема каждой заявки и накоплением суммарного объема. Последняя проблема характерна, например, для эшелонированных систем управления запасами (спрос в высших звеньях формируется групповыми заявками случайного объема из низших звеньев).

Распределение Пуассона за случайный интервал. При моделировании в схеме однофакторного слежения приходится воспроизводить число K событий, подчиняющееся вероятностному распределению

$$P\{K=k\} = q_k = \int\limits_0^\infty \frac{(\lambda t)^k}{k\,!} e^{-\lambda t}\, dF(t).$$

Этот случай сводится к предыдущему, если предварительно сформировать случайный интервал T, подчиненный вероятностному распределению $F(t)$.

Отрицательное биномиальное распределение задается вероятностями совершить точно r попыток до k успехов

$$P_r = \binom{r+k-1}{k} p^r (1-p)^k$$

при вероятности p успеха в одной попытке. Оно генерируется аналогично бернуллиеву, но подсчет попыток продолжается до получения k успехов.

4.10 Генерация пары коррелированных величин

Если случайные величины X и Y должны иметь коэффициент корреляции ρ и средние и дисперсии $x, y, \sigma_x^2, \sigma_y^2$ соответственно, следует формировать

$$Y = aX + b + z.$$

Здесь $a = \rho\sigma_y/\sigma_x$, $b = y - ax$, $z = \sqrt{1-\rho^2}\sigma_y\eta$ и η — случайное число, распределенное по стандартному нормальному закону ($\bar{\eta} = 0$, $\sigma_\eta = 1$). Обоснование этого правила имеется в книге [28, ч. 1].

4.11 Случайные выборки

Для выборки k элементов из базового множества в схеме *с возвращением* (с повторениями) равномерно распределенное случайное число U каждый раз отображается на диапазон номеров $i = \overline{1,n}$.

В схеме *без возвращения* выбранный элемент меняется местами с последним, после чего n заменяется на $n-1$ и т. д. — пока не будет выбрано требуемое число элементов. Выдача результатов происходит в зависимости от задачи: по мере получения или весь конец списка разом. Эта идея может быть применена при выборе каналов обслуживания: выбранные сдвигаются в конец списка, освободившиеся пристыковываются к началу.

Случайная выборка элементов множества может сочетаться с временным запретом на выбор отдельных элементов — например, при моделировании процессов парковки автомобилей. В таких случаях ведутся списки свободных и занятых элементов, и при каждом изменении пересчитываются вероятности занятия свободных элементов.

4.12 Случайные объекты

К таким объектам относятся графы общего вида, связные, двудольные, деревья (отдельно — бинарные), ациклические пути в графах, случайные разбиения, перестановки. Генерация *путей* позволяет снять требование марковости маршрутизации заявок в ациклических сетях: каждая заявка может снабжаться оформленным в виде магазина списком посещаемых ею вершин. Случайные графы имеют многочисленные применения в программотехнике — главным образом в связи с проблемой тестирования сложных программных комплексов.

Генерация случайных графов начинается с задания числа вершин. Далее последовательно генерируются возможные ребра с учетом предварительно наложенных ограничений: количественных (максимальная степень исхода) и качественных, определяемых видом графа. Дополнительно проверяются связность графа и другие заданные свойства (например, отсутствие циклов).

4.13 Дополнительная литература

Более полный обзор методов формирования как независимых, так и зависимых случайных чисел и векторов с требуемыми вероятностными характеристиками приводится в работах [3, 21, 47, 73, 80, 108]. В частно-

сти, в заключительных разделах главы 3 [47] рассматриваются вопросы генерации случайных векторов, процессов и полей.

Компактная сводка алгоритмов генерации содержится в [1]. Отметим, что при высоких требованиях к точности моделирования и (или) скорости выработки требуемых случайных чисел выбор метода генерации их должен учитывать специфику решаемой задачи, характеристики моделирующей ЭЦВМ (объем оперативной памяти, быстродействие, особенности системы команд) и ее программного обеспечения (наличие и трудоемкость реализации соответствующих стандартных подпрограмм). «Сверхэффективные» ДСЧ обычно программируются на языке ассемблера.

Весьма ценным источником представляется описание библиотеки «Монте-Карло» для машин ЕС ЭВМ [62]. В эту библиотеку включены 27 генераторов случайных чисел; генератор последовательностей Соболя чисел, более равномерных в многомерном кубе; 36 подпрограмм и 5 программ статистической обработки и оценки качества случайных чисел; 44 контрольных примера. Ряд упомянутых алгоритмов — новые. Из упомянутых программ 8 написаны на Ассемблере ЕС, 99 — на *Фортране* IV, 8 — на *ПЛ/1*. По каждой из процедур даются назначение, вид обращения и описание параметров. Указан язык разработки. Тексты процедур поставляются на магнитной ленте в составе упомянутой библиотеки.

В классе *непрерывных* распределений рассмотрены генераторы следующих распределений: с таблично заданной ФР; нормального, логарифмически нормального, χ^2, Стьюдента, Фишера, показательного, гамма-, бета-, Коши, равномерных на $[a, b]$.

В классе *дискретных* распределений обсуждается работа с распределениями: таблично заданным, Бернулли, Пуассона, биномиальным, полиномиальным, Паскаля, гипергеометрическим.

В разделе *статистической обработки* описаны подпрограммы, дающие:

- выборочные значения дисперсии, СКО, среднего;
- минимум и максимум;
- гистограмму;
- квадратичную меру отклонения теоретического распределения от эмпирического;

- ковариации;
- количества серий;
- длины периода в рекуррентной последовательности;
- отрезок апериодичности;
- эмпирическую ФР по гистограмме;
- эмпирическую ФР — интегрированием плотности методом трапеций.

В книге Уолка [108] дан обзор соответствующей части программной библиотеки CERN и собственных («home written») разработок автора. Приводятся основные формулы для распределений и их моментов, асимметрии и эксцесса; идея возможных технологий и сопоставление программ, их реализующих, по точности выдерживания свойств распределений (моментов до 8-го порядка, скошенности, эксцесса) и по трудоемкости. Сообщается, что все программы записаны на *Фортране* 77, реализация проводилась на ЭВМ VAX-11/780. Приводятся конкретные результаты тестирования альтернативных датчиков на последовательностях из миллиона чисел.

Наиболее обстоятельным источником по проблемам генерации случайных величин и объектов является монография [84], по возможности использованная в данной главе. Книга в целом структурирована слабо и в основном построена по *методам*, в связи с чем вопросы генерации заданного распределения рассматриваются в нескольких местах — с разных позиций. Найти конкретные рекомендации помогает предметный указатель. Все предлагаемые генераторы имеют теоретические обоснования — иногда в виде длинной цепочки лемм и теорем. Альтернативные алгоритмы сопоставляются по трудоемкости — в частности, по среднему числу необходимых на одно результирующее значение вспомогательных величин (равномерно, показательно или нормально распределенных). Даются упражнения и методические рекомендации — в зависимости от специализации студентов. К сожалению, в книге встречаются ошибки (опечатки?) в формулах — например, в выражении $H^{-1}(E)$ для распределения Вейбулла из табл. 4.8.

В связи с труднодоступностью ее для отечественного читателя приведем к сведению специалистов, профессионально работающих с генераторами случайных величин, краткий обзор ее содержания.

Гл. 1. Введение. Общий очерк. Определения. Важнейшие плотности. Параметрические семейства. Операции над случайными переменными: преобразования, смеси, порядковые статистики, свертки распределений.

Гл. 2. Общие принципы генерации случайных величин. Метод инверсии. Численная реализация. Аппроксимации. Метод режекции: идея, разработка «хороших» алгоритмов, обобщения, разложение на дискретные смеси.

Гл. 3. Дискретные случайные величины. Метод инверсии (усечение непрерывной переменной, сравнения, управляющие таблицы, обращение коррекцией). Метод просмотра таблиц. Метод алиасов. Геометрические методы. Другие общие принципы (режекция, композиция).

Гл. 4. Специализированные алгоритмы. Метод Форсайта — фон Неймана (общая идея, показательный датчик, Г-генератор). Почти точное обращение. Преобразования «Много — один». Метод рядов (экспонента, Рааба — Грина, Колмогорова — Смирнова). Представление плотностей интегралами (монотонные функции, выпуклые; устойчивые распределения; плотности с характеристической функцией класса Пойа). Отношения равномерных распределений.

Гл. 5. Равномерные и показательно распределенные интервалы. Общая идея. Генерация упорядоченных выборок. Полярный метод (радиально симметричные распределения; генерация: точек, равномерно распределенных в сфере произвольной размерности и в круге, пар нормальных величин, радиально симметричных векторов; метод развертки).

Гл. 6. Пуассоновский процесс. Однородный и неоднородный процесс. Генерация случайных величин с заданной функцией интенсивности (hazard rate) — методы обращения, композиции, утоньшения.

Гл. 7. Универсальные методы. Философия «черного ящика». Логарифмически вогнутые плотности. Неравенства для семейств плотностей. Метод инверсии–режекции.

Гл. 8. Табличные методы для генерации непрерывных переменных. Композиция и режекция. Метод полосок. Сеточные методы. Проблемы избегания.

Гл. 9. Непрерывные плотности одной переменной. Нормаль-

ная, показательная, Г-плотности — отдельно для $\alpha > 1$ и $\alpha < 1$. Распределения Вейбулла, бета, t-распределение Стьюдента; устойчивые и нестандартные распределения (с функцией Бесселя, логистическое, обратное гауссово, фон Мизеса).

Гл. 10. Дискретные распределения одной переменной. Введение (цель главы; производящие функции и факториалы, универсальный метод режекции). Геометрическое, пуассоново, биномиальное распределения. Семейство логарифмических рядов.

Гл. 11. Многомерные распределения. Общие принципы. Линейные преобразования. Многомерное нормальное распределение. Распределения зависимых величин. Частные случаи. Генерация случайных матриц (корреляционной, ортогональной).

Гл. 12. Случайные выборки. Классические задачи. Метод обмена. Выборки с проверкой на принадлежность. Последовательные выборки. Методы интервалов, инверсии, режекции.

Гл. 13. Случайные комбинаторные объекты. Введение (общий подход, метод декодирования, рекуррентная генерация). Случайные перестановки. Случайные двоичные деревья. Случайные разбиения. Случайные графы (с заданными простыми свойствами, связные, двудольные).

Гл. 14. Разное. Максимум из одинаково распределенных переменных. Распределения с заданными моментами. Характеристические функции. Моделирование сумм. Моделирование дискретных событий. Регенеративные процессы.

Гл. 15. Модели со случайными битами.

Для практической работы с распределениями весьма полезен содержательный, доступно написанный, хорошо структурированный и наглядно иллюстрированный справочник [12]. Этот справочник делится на главы, содержащие общие сведения, 13 дискретных и 35 непрерывных распределений. Последние группируются по областям определения (вся числовая ось, положительная полуось и конечные отрезки). Для каждого непрерывного распределения указываются плотность, функция распределения, функция риска (плотность остаточного распределения), мода, медиана, начальные и центральные моменты, асимметрия и эксцесс, имеемые таблицы, способы оценки параметров по статистическим данным, связь с другими распределениями (частные и предельные случаи). Приводятся (без доказательства) блок-схемы алгоритмов генерации

соответствующих случайных последовательностей.

Отдельного внимания заслуживает Приложение к упомянутому справочнику, в котором даны графики для выбора типа выравнивающих статистические данные распределений по методу моментов, аналогичные схеме Пирсона, а также описание работы со статистическими распределениями в статистических пакетах прикладных программ для ПЭВМ *Stadia, Statgraphics, Statistica, SPSS*. Список литературы содержит руководства общего назначения, таблицы, справочники и государственные стандарты по прикладной статистике.

Глава 5

Обработка результатов моделирования

Моделирование вероятностных систем следует рассматривать как статистический эксперимент над моделью системы, отображающей структуру системы и ее причинно-следственный механизм. Выходные данные анализируются для получения выводов о поведении системы. Этот анализ должен основываться на надежных статистических процедурах. В частности, необходимо получение доверительных интервалов, а также установление зависимости между временем моделирования и точностью оценок.

5.1 Точечные оценки

Обозначим через $\hat{a}(N)$ статистическую оценку параметра a по данным N опытов. Оценивание одного и того же параметра, вообще говоря, может быть проведено различными способами. Наилучшими («подходящими») считаются оценки, удовлетворяющие требованиям состоятельности, несмещенности и эффективности.

Оценка называется *состоятельной*, если она при неограниченном увеличении числа опытов сходится по вероятности к искомому значению параметра:

$$\lim_{N\to\infty} P\{|\hat{a}(N) - a| < \varepsilon\} = 1.$$

Оценка является *несмещенной*, если ее математическое ожидание при любом конечном N равно истинному значению параметра:

$$M[\hat{a}(N)] = a.$$

Оценка *эффективна*, если среди всех возможных она обладает наименьшей дисперсией.

Несмещенной оценкой математического ожидания случайной величины X является среднее арифметическое результатов N опытов:

$$\hat{x} = \frac{1}{N}\sum_{i=1}^{N} X_i. \tag{5.1.1}$$

Аналогичными формулами выражаются начальные статистические моменты более высокого порядка:

$$\hat{x}_k = \frac{1}{N}\sum_{i=1}^{N} X_i^k, \qquad k = 1, 2, \ldots \tag{5.1.2}$$

При подсчете по формулам вида (5.1.2) с последовательным накоплением суммы чисел одного знака необходимо иметь в виду характерное для многих типов современных ЭВМ отбрасывание разрядов, выходящих за разрядную сетку, вместо правильного округления младшего из сохраняемых разрядов. Эта особенность при большом N приводит к заметному дрейфу результатов. Для нейтрализации указанного эффекта можно воспользоваться одним из следующих приемов [78]:

- считать средние по сериям наблюдений, а затем определить среднее серий (в этом случае разница в порядках слагаемых каждой операции сложения уменьшается);
- обрабатывать *центрированные* случайные величины (слагаемые будут знакопеременны, и эффект их усечения взаимно компенсируется).

При расчете статистических *центральных* моментов приходится вводить поправку, вызванную заменой истинного математического ожидания X в формулах вида

$$\hat{\mu}_k = \frac{1}{N}\sum_{i=1}^{N} (X_i - x)^k, \qquad k = 1, 2, \ldots$$

статистическим средним $\hat{x}$. В частности, несмещенная оценка для статистической дисперсии

$$\hat{D} = \frac{1}{N-1}\sum_{i=1}^{N} (X_i - \hat{x})^2. \tag{5.1.3}$$

Статистическое среднеквадратическое отклонение

$$\hat{s} = \sqrt{\hat{D}}. \tag{5.1.4}$$

В [47, с. 232–233] приводятся оценки центральных моментов до 4-го порядка включительно и их среднеквадратические отклонения.

Теоретический анализ *существования* моментов выполним не всегда. Однако можно организовать эмпирический контроль сходимости оценок моментов с ростом числа испытаний. В [47, с. 238] дан пример графика оценки эмпирического среднего для показателя, распределенного по закону Коши.

Формирование в процессе моделирования больших массивов необработанных результатов создает проблему их хранения. В частности, для получения оценки статистической дисперсии согласно (5.1.3) необходимо хранить все $\{X_i\}$ до окончания ее расчета, что крайне неудобно при типичном числе испытаний порядка сотен тысяч. Эту проблему обходят, используя *накопленные* данные. Прежде всего отметим, что статистическая оценка среднего по N испытаниям

$$\hat{x}_N = \frac{1}{N}\sum_{i=1}^{N} X_i = \frac{N-1}{N}\hat{x}_{N-1} + \frac{1}{N}X_N. \tag{5.1.5}$$

Далее, на основании (5.1.3) и (5.1.5)

$$\begin{aligned}
(N-1)\hat{D}_N &= \sum_{i=1}^{N}\left(X_i - \frac{N-1}{N}\hat{x}_{N-1} - \frac{1}{N}X_N\right)^2 \\
&= \sum_{i=1}^{N}\left(X_i - \frac{N-1}{N}\hat{x}_{N-1} - \frac{1}{N}\hat{x}_{N-1} + \frac{1}{N}\hat{x}_{N-1} - \frac{1}{N}X_N\right)^2 \\
&= \sum_{i=1}^{N}\left[(X_i - \hat{x}_{N-1}) - \frac{1}{N}(X_N - \hat{x}_{N-1})\right]^2 \\
&= \sum_{i=1}^{N-1}(X_i - \hat{x}_{N-1})^2 + (X_N - \hat{x}_{N-1})^2 \\
&- \frac{2}{N}(X_N - \hat{x}_{N-1})\sum_{i=1}^{N}(X_i - \hat{x}_{N-1}) + \frac{1}{N}(X_N - \hat{x}_{N-1})^2.
\end{aligned}$$

Можно показать, что

$$\sum_{i=1}^{N}(X_i - \hat{x}_{N-1}) \;=\; \sum_{i=1}^{N} X_i - \frac{N}{N-1}\sum_{i=1}^{N-1} X_i = X_N - \hat{x}_{N-1}.$$

Выполнив подстановку, получаем

$$\begin{aligned}(N-1)\hat{D}_N &= (N-2)\hat{D}_{N-1} + (X_N - \hat{x}_{N-1})^2 \\ &- \frac{2}{N}(X_N - \hat{x}_{N-1})^2 + \frac{1}{N}(X_N - \hat{x}_{N-1})^2 \\ &= (N-2)\hat{D}_{N-1} + \frac{N-1}{N}(X_N - \hat{x}_{N-1})^2.\end{aligned}$$

Окончательно

$$\hat{D}_N \;=\; \frac{N-2}{N-1}\hat{D}_{N-1} + \frac{1}{N}(X_N - \hat{x}_{N-1})^2. \qquad (5.1.6)$$

Теперь ясно, что достаточно накапливать и хранить только *суммы* $\{X_i\}$ и их квадратов. Статистические оценки среднего можно вычислять согласно (5.1.5), а дисперсии — по формуле (5.1.6). Применение этих формул дает возможность оперативного завершения счета при достижении заданной точности и надежности оценок, определяемых текущими значениями $\hat{}_N$ и $\hat{D}_N$. Именно такой подход используется в системе *GPSS World* при динамическом обновлении таблиц.

Общие формулы рекуррентных оценок для центральных моментов см. в [47, с. 235].

Оценка корреляционного момента системы двух случайных величин X и Y

$$\hat{K}_{XY} = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \hat{x})(Y_i - \hat{y}), \qquad (5.1.7)$$

а их коэффициента корреляции —

$$\hat{\rho}_{XY} = \hat{K}_{XY}/(\hat{s}_X \hat{s}_Y). \qquad (5.1.8)$$

Для корреляционного момента также можно получить рекуррентную формулу:

$$\begin{aligned}
K_N &= \frac{1}{N-1}\sum_{i=1}^{N}(X_i-\hat{x}_N)(Y_i-\hat{y}_N) \\
&= \frac{1}{N-1}\sum_{i=1}^{N}(X_iY_i-\hat{x}_NY_i-\hat{y}_NX_i+\hat{x}_N\hat{y}_N) \\
&= \frac{1}{N-1}\left[\sum_{i=1}^{N}X_iY_i-\hat{x}_N\sum_{i=1}^{N}Y_i-\hat{y}_N\sum_{i=1}^{N}X_i+N\hat{x}_N\hat{y}_N\right] \\
&= \frac{1}{N-1}\left[\sum_{i=1}^{N}X_iY_i-N\hat{x}_N\hat{y}_N\right].
\end{aligned}$$

Для ее реализации, помимо средних, достаточно пересчитывать сумму парных произведений.

Оценкой вероятности состояния r служит частота его появления

$$\hat{p}_r = N_r/N, \tag{5.1.9}$$

где N_r — количество реализаций (из общего числа N), в которых наблюдалось интересующее нас событие. В моделях с непрерывным временем общее число реализаций N заменяется на системное время T протекания процесса, а N_r — на суммарное время T_r наблюдения состояния r в ходе моделирования

$$\hat{p}_r = T_r/T. \tag{5.1.10}$$

Выборочное среднее (5.1.10) из-за влияния начальных условий будет, вообще говоря, смещенной оценкой истинного значения. Если, например, в начальный момент моделируемая СМО свободна, то несколько первых значений времени ожидания имеют тенденцию быть малыми. Традиционный способ борьбы с этим смещением состоит в отбрасывании данных за период вхождения в стационарный режим (см. разд.2), однако отнюдь не ясно, какова должна быть длина этого периода, и для гарантированной нейтрализации смещения в ходе моделирования непроизводительно затрачивается много машинного времени.

Все перечисленные оценки *состоятельны* и, следовательно, при неограниченном возрастании числа наблюдений N (длительности процесса T) сходятся по вероятности к искомым параметрам. Однако при любом конечном N или T получаемые оценки *случайны*, в связи с чем возникает вопрос об их качестве (погрешности). Простейшим способом определения качества оценки является указание ее дисперсии. Известно ([33, гл. 19]), что дисперсия оценки $\hat{\nu}_r$ r-го начального момента

$$D[\hat{\nu}_r] = (\nu_{2r} - \nu_r^2)/N \tag{5.1.11}$$

(в правую часть входят теоретические значения соответствующих моментов). В частности, дисперсия статистического среднего

$$D[\hat{\nu}_1] = D_X/N. \tag{5.1.12}$$

Воспользуемся этой формулой для вычисления дисперсии оценки $\hat{v}$ среднего времени пребывания заявки в системе $M/M/1$, которое имеет показательное распределение с параметром $(\mu - \lambda)$ [55, 61]. Тогда

$$D[\hat{v}] = \frac{1}{N}(\mu - \lambda)^{-2} = \frac{1}{N\mu^2}\frac{1}{(1-\rho)^2},$$

где

λ — интенсивность входящего потока,

μ — интенсивность обслуживания,

ρ — коэффициент загрузки системы.

Можно ожидать такого же качественного характера зависимости этой дисперсии от ρ и N для многоканальных систем и произвольных распределений. Это объясняет общеизвестную трудность моделирования СМО при большой загрузке ($\rho \to 1$).

Знание дисперсий используется для сопоставления альтернативных оценок одного параметра. В частности, доказано (см. [80, с.104]), что усреднение по системному времени дает оценки с меньшей дисперсией, чем усреднение по числу наблюдений.

5.2 Дисперсия и корреляция

Выходные данные обычно являются автокоррелированными. Предположим, что случайные величины $X_1, X_2, \ldots, X_n$ получены из ковариационно стационарного процесса. Тогда выборочное среднее $\bar{X}(n)$ все еще останется несмещенной оценкой среднего μ; однако выборочная дисперсия $D(n)$ более не является несмещенной. Можно доказать [98, с. 302–304], что

$$D(n) = D\left[1 - \frac{2}{n-1}\sum_{j=1}^{n-1}(1 - j/n)\rho_j\right].$$

Оценки (не слишком хорошие) коэффициентов корреляции

$$\hat{\rho}_j = \frac{\sum\limits_{i=1}^{n-j}[X_i - \bar{X}(n)][X_{i+j} - \bar{X}(n)]}{(n-j)D(n)}.$$

Если $\{\rho_j\} > 0$ (положительная корреляция), что обычно имеет место, выборочная дисперсия будет иметь отрицательное смещение. Наличие смещения приведет к ошибкам при построении доверительного интервала.

Оценка дисперсии выборочного среднего

$$D[\bar{X}(n)] = \frac{D}{n}\left[1 + 2\sum_{j=1}^{n-1}(1 - j/n)\rho_j\right].$$

5.3 Классический подход к интервальным оценкам

Имея оценку и ее дисперсию, можно перейти к построению доверительного интервала. Практические требования к оценкам формулируются в терминах их точности и надежности. При этом под *точностью* понимается половина δ длины доверительного интервала, а под *надежностью* — вероятность P того, что истинное значение параметра окажется принадлежащим упомянутому интервалу (доверительная вероятность). При фиксированных условиях увеличение требований к точности уменьшает доверительную вероятность, а увеличение доверительной вероятности

снижает точность оценок. Обычно ставится задача определения числа испытаний N, при котором будут обеспечены заданные δ и P.

Пусть определяется среднее время w ожидания начала обслуживания, причем дисперсия времени ожидания равна σ_W^2. При характерной для имитационного моделирования кратности наблюдений N разность $\hat{w} - w$ можно на основании теоремы А.М. Ляпунова считать распределенной нормально с дисперсией σ_W^2/N — см. формулу (5.1.12). Следовательно, доверительная вероятность

$$P\{|\hat{w} - w| \leq \delta\} = \Phi\left(\frac{\delta\sqrt{n}}{\sigma_W\sqrt{2}}\right), \tag{5.3.1}$$

где $\Phi(\cdot)$ — функция Лапласа. Переходя к обратной функции, имеем

$$\Phi^{-1}(P) = \frac{\delta\sqrt{N}}{\sigma_W\sqrt{2}},$$

откуда требуемое число наблюдений

$$N \geq 2[\Phi^{-1}(P)]^2\,(\sigma_W/\delta)^2 = k(P)(\sigma_W/\delta)^2. \tag{5.3.2}$$

Коэффициент $k(P)$ выбирается из таблицы 5.1.

Таблица 5.1: Коэффициенты $k(P)$ для расчета числа испытаний

P	0.900	0.950	0.970	0.980	0.990	0.995	0.999
$k(P)$	2.69	3.84	4.71	5.43	6.66	7.90	10.82

Таким образом, необходимое число испытаний обратно пропорционально *квадрату* допустимой погрешности и резко возрастает с повышением доверительной вероятности.

Приведем таблицу 5.2 основных расчетных показателей моделирования среднего времени ожидания в системе $M/M/1$ для получения доверительного интервала в 0.1 от среднего с доверительной вероятностью 0.9 [91].

Таблица 5.2. Зависимость требуемого числа испытаний от загрузки

Загрузка ρ	Среднее w	Теоретическая дисперсия оценки	Требуемое число испытаний
0.10	0.111	$3.75 \cdot 10^{-1}$	$8.20 \cdot 10^3$
0.20	0.250	$1.39 \cdot 10^0$	$6.02 \cdot 10^3$
0.30	0.429	$3.96 \cdot 10^1$	$5.83 \cdot 10^3$
0.40	0.667	$1.06 \cdot 10^1$	$6.43 \cdot 10^3$
0.50	1.000	$2.90 \cdot 10^1$	$7.85 \cdot 10^3$
0.60	1.500	$8.85 \cdot 10^1$	$1.06 \cdot 10^4$
0.70	2.333	$3.35 \cdot 10^2$	$1.67 \cdot 10^4$
0.80	4.000	$1.98 \cdot 10^3$	$3.34 \cdot 10^4$
0.90	9.000	$3.59 \cdot 10^4$	$1.19 \cdot 10^5$
0.95	19.000	$6.08 \cdot 10^5$	$4.55 \cdot 10^5$
0.99	99.000	$3.95 \cdot 10^8$	$1.09 \cdot 10^7$

Немонотонность роста требуемого числа испытаний в верхней части таблицы объясняется заданием доверительного интервала в долях от среднего времени ожидания.

Доверительный интервал для w будет равен $\hat{w} \pm \delta$. Для построения фактического доверительного интервала по заданной вероятности P следует положить $\delta = l(P)\sigma_W/\sqrt{N}$, где $l(P) = \sqrt{k(P)}$.

Во всех формулах этого пункта вместо теоретического значения среднеквадратического отклонения σ_W практически приходится пользоваться его статистической оценкой. Это определяет особую ценность рекуррентного расчета дисперсий оцениваемых величин по формулам типа (5.1.5), (5.1.6). Возможен также двухэтапный эксперимент: на первом этапе грубо определяется оценка $\hat{\sigma}_W$, после чего согласно (5.3.2) вычисляется полное потребное количество испытаний. Вычтя из него количество проведенных на первом этапе, находим потребное количество дополнительных наблюдений.

В случае малого количества наблюдений (порядка нескольких десятков) доверительные интервалы для нормально распределенных величин строят с помощью распределения Стьюдента. Введем случайную величину

$$T = \frac{\hat{w} - w}{\hat{s}}, \tag{5.3.3}$$

где

$$\hat{s} = \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^{N} (W_i - \hat{w})^2}. \tag{5.3.4}$$

Можно доказать, что случайная величина T имеет плотность распределения

$$s_N(t) = \frac{\Gamma(N/2)}{\sqrt{(N-1)\pi}\Gamma((N-1)/2)} \left(1 + \frac{t^2}{N-1}\right)^{-N/2}, \tag{5.3.5}$$

зависящую только от аргумента t и числа наблюдений N. Таким образом,

$$\Pr\{|\hat{w} - w| < t\hat{s}\} = 2 \int_0^t s_N(t)\, dt. \tag{5.3.6}$$

Функция в правой части уравнения (5.3.6) и ей обратная табулированы. С их помощью можно вычислять доверительные вероятности и доверительные вероятности.

При оценке вероятностей в формулу (5.3.2) вместо σ следует подставить $\sqrt{p(1-p)}$. Тогда формула (5.3.2) примет вид

$$N \geq k(P) \frac{p(1-p)}{\delta^2}. \tag{5.3.7}$$

При определении вероятностей обычно задаются их *относительной* точностью, то есть считают $\delta = \varepsilon p$. Тогда для малых вероятностей $1 - p \approx 1$, и можно переписать (5.3.7) в виде

$$N \geq \frac{k(P)}{p\varepsilon^2}.$$

Итак, необходимое число испытаний обратно пропорционально искомой вероятности и *квадрату* допустимой относительной погрешности. Например, для определения вероятности порядка 10^{-6} с относительной погрешностью в 1% и доверительной вероятностью 0.98 должно быть проведено $N > 5.43 \cdot 10^6 \cdot 10^4 \approx 5 \cdot 10^{10}$ испытаний. Это обстоятельство существенно затрудняет непосредственную оценку вероятностей редких событий методом имитационного моделирования.

Глава 6

Уточненная обработка статистики

6.1 Регенерирующий процесс и его обработка

Для применения описанных выше классических процедур построения доверительных интервалов выходные данные должны образовывать набор статистически независимых и одинаково распределенных выборочных значений. Заметим, однако, что если время W_k ожидания k-го требования велико, то W_{k+1} также будет большим с высокой вероятностью. Таким образом, значения W_k и W_{k+1} будут сильно коррелированы, причем независимо от начальных условий. Приведем (табл. 6.1) соответствующие экспериментальные данные, полученные автором для модели $M/D/2$ с помощью программы из главы 2.

Таблица 6.1. Корреляция последовательных времен ожидания

Коэффициент загрузки	0.3	0.5	0.7	0.9
Среднее время ожидания	0.113	0.341	0.917	3.170
Коэффициент корреляции	0.390	0.597	0.807	0.951

Указанное затруднение можно преодолеть, если работать с *регенерирущими* процессами. Такие процессы по определению время от времени возвращаются в некоторую *точку регенерации*, после которой дальнейшее развитие процесса не зависит от его поведения в прошлом и

определяется одним и тем же вероятностным законом. Отрезок процесса между смежными моментами регенерации является независимой «вероятностной копией» любого другого такого отрезка, *вложенной* в исходный процесс.

Регенерирующий процесс строится в зависимости от определяемых характеристик. Для получения оценки среднего времени ожидания начала обслуживания разобьем период моделирования на *циклы*, состоящие из смежных периодов занятости и простоя. Началом цикла будем считать момент прихода заявки в свободную систему (такой подход предлагался, например, в [47]). Поскольку каждый цикл начинается при одних и тех же условиях, результаты последовательных циклов статистически независимы и имеют одинаковые распределения. Пусть всего проведено N циклов. Обозначим

ν_i — число заявок, обслуженных в i-м цикле,

W_i — суммарная длительность ожидания в нем,

$n = \sum\limits_{i=1}^{N} \nu_i$ — общее число обслуженных заявок.

Тогда оценка среднего времени ожидания

$$\hat{w} = \frac{W_1 + W_2 + \ldots W_N}{n} = \frac{(W_1 + W_2 + \ldots W_N)/N}{(\nu_1 + \nu_2 + \ldots \nu_N)/N}. \qquad (6.1.1)$$

При $N \to \infty$ в силу закона больших чисел числитель и знаменатель стремятся к своим математическим ожиданиям, так что

$$M[\hat{w}] = M[W]/M[\nu]. \qquad (6.1.2)$$

Для родственных теории массового обслуживания задач управления запасами (см. [54, 61]) со стратегией восполнения вида (y, Y) и мгновенной поставкой моментами регенерации являются начала периодов, в которые запас восполняется до максимального значения Y.

Описанный подход может быть перенесен на оценку других характеристик случайного процесса. Пусть дана последовательность $\{X_i\}$ случайных векторов размерности k и $f(x)$ — действительная функция компонент случайного вектора. Тогда, определив Y_j как сумму значений f на j-м цикле регенерации

$$Y_j = \sum_{i=\tau_j}^{\tau_{j+1}-1} f(X_i),$$

где $\{\tau_j\}$ — номера наблюдений, открывающие j-й цикл, можно вновь оценить математическое ожидание $f(X)$ по формуле типа (6.1.1).

Для процессов с непрерывным временем считают

$$Y_j = \int\limits_{\tau_j}^{\tau_{j+1}} f(X(t))\,dt,$$

где $\{\tau_j\}$ — уже не номера, а сами *точки* регенерации, а $\{\nu_j\}$ из формулы (6.1.1) суть продолжительности соответствующих циклов.

Процедура построения 100(1-δ)–процентного доверительного интервала для $r = M[f(x)]$ при числе циклов N порядка сотен и более согласно [35] состоит в следующем:

1. Провести моделирование N циклов регенерации.
2. Вычислить $\{Y_j\}$ и $\{\nu_j\}$ для каждого j-го цикла.
3. Вычислить выборочные статистики:
   - статистические средние
   $$\hat{y} = \frac{1}{N}\sum_{j=1}^{N} Y_j, \qquad \hat{\nu} = \frac{1}{N}\sum_{j=1}^{N} \nu_j, \qquad \hat{r} = \hat{y}/\hat{\nu};$$
   - статистические дисперсии
   $$\hat{D}_{11} = \frac{1}{N-1}\sum_{j=1}^{N} Y_j^2 - \frac{1}{N(N-1)}\left(\sum_{j=1}^{N} Y_j\right)^2;$$
   $$\hat{D}_{22} = \frac{1}{N-1}\sum_{j=1}^{N} \nu_j^2 - \frac{1}{N(N-1)}\left(\sum_{j=1}^{N} \nu_j\right)^2$$
   и ковариацию
   $$\hat{D}_{12} = \frac{1}{N-1}\sum_{j=1}^{N} Y_j\nu_j - \frac{1}{N(N-1)}\left(\sum_{j=1}^{N} Y_j\right)\left(\sum_{j=1}^{N} \nu_j\right);$$
   - определить приведенную дисперсию отношения
   $$\hat{D} = \hat{D}_{11} + \hat{r}^2\hat{D}_{22} - 2\hat{r}\hat{D}_{12}.$$

4. Сформировать доверительный интервал

$$\hat{r} = \frac{Z\delta}{\nu}\sqrt{\hat{D}/N}, \tag{6.1.3}$$

где $Z = \Phi^{-1}(\delta/2)$, $\Phi(\cdot)$ — функция Лапласа.

Процедура для траектории с фиксированным временем T идентична процедуре с фиксированным числом циклов регенерации за исключением того, что статистика обрабатывается только по завершившимся циклам.

Для получения требуемого числа испытаний на основе формулы (6.1.3) необходимо предварительное определение оценок в ходе пробного моделирования.

Если существуют две или более последовательностей точек регенерации, соответствующих, например, двум или более возвратным состояниям, то можно выбрать любую из них — доказано, что длины построенных доверительных интервалов при $T \to \infty$ будут асимптотически равны.

6.2 Приближенно регенерирующие модели

Решающими для применения описанной методики являются следующие требования:

1. процесс неоднократно возвращается в некоторое фиксированное состояние (подмножество состояний);

2. среднее время между возвращениями конечно;

3. момент очередного возвращения является точкой регенерации.

Получение на базе данной теории достаточно надежных оценок требует увеличения длительности моделирования — в особенности при большой загрузке, когда случаи освобождения системы становятся редкими. В связи с этим возникает проблема построения процессов, которые *приближенно* могут рассматриваться как регенерирующие, но имеют умеренную длительность циклов.

Очевидным методом ограничения длительности циклов является дискретизация пространства состояний моделируемой системы. Далее, будем рассматривать последовательные возвращения процесса $\{W_n\}$, $n > 1$, в состояние «Т», то есть фиксировать моменты, когда вновь прибывшая заявка ожидает ровно T единиц времени. Поскольку распределение длительностей виртуального времени ожидания непрерывно, среднее время между возвращениями в отмеченное подобным образом состояние (кроме нулевого) в предположении о неограниченной разрядной сетке ЭВМ *бесконечно*. Пусть $t_k(\varepsilon)$ обозначает k-й по счету момент попадания процесса $\{W_n\}$ в промежуток $[T-\varepsilon, T+\varepsilon]$ для фиксированного $\varepsilon > 0$. Теперь среднее время возвращения в этот промежуток конечно и уменьшается с увеличением ε. При таком подходе происходит приближенная регенерация процесса (тем более точная, чем меньше ε), что позволяет применить обсуждавшийся выше статистический аппарат. Значение ε, чтобы наблюдать достаточное число циклов, не должно быть слишком малым. Это обстоятельство, порождая корреляцию *между циклами*, несколько искажает результаты.

Еще одним примером приближенного подхода к построению регенерирующего процесса может служить метод *меченых заявок* при исследовании сети обслуживания. Если в каждый момент времени в сети находится только одна меченая заявка, а сеть достаточно сложна, то времена пребывания в сети последовательных меченых заявок окажутся некоррелированными. Этот выигрыш в «чистоте эксперимента» также приходится оплачивать: кратность замедления процесса моделирования равна среднему числу заявок, находящихся в сети.

В заключение отметим, что ключевым моментом регенеративного подхода является использование отношений сумм случайных величин. Но равенство

$$M\Big[\sum_{i=1}^{N} W_i \Big/ \sum_{i=1}^{N} \nu_i\Big] = M\Big[\sum_{i=1}^{N} W_i/N\Big] \Big/ M\Big[\sum_{i=1}^{N} \nu_i/N\Big]$$

верно лишь асимптотически — при $N \to \infty$. Таким образом, получаемые оценки оказываются *смещенными*.

Просуммируем недостатки регенеративного метода:

- смещенность получаемых оценок, хотя и нулевая асимптотически;
- априорная неизвестность длины цикла, что затрудняет планирование прогонов;

- большая длительность циклов, в особенности для систем с коэффициентом загрузки, близким к единице;
- трудность синхронизации параллельных прогонов при использовании методов уменьшения дисперсии результатов моделирования (см. главу 8).

С другой стороны, его несомненными достоинствами являются:

- независимость наблюдений в различных циклах, позволяющая проводить корректную статистическую обработку результатов;
- отсутствие проблемы начальных условий.

6.3 Метод «складного ножа»

Статистические оценки некоторых величин являются несмещенными лишь асимптотически — при числе испытаний $N \to \infty$. При умеренных значениях N, характерных для регенеративного моделирования и экспериментов с применением метода понижения дисперсии — МПД (см. главу 8), это смещение может быть весьма заметным, и встает вопрос о его уменьшении.

Пусть $\hat{\beta}(N)$ — смещенная оценка параметра β, основанная на N наблюдениях, и ее математическое ожидание представлено как

$$M[\hat{\beta}(N)] = \beta + \sum_{i=1}^{N} \alpha_i / N^i, \tag{6.3.1}$$

где $\{\alpha_i\}$ — некоторые коэффициенты, зависящие, вообще говоря, от β, но не от N. Асимптотически смещение оценки (6.3.1) равно нулю, а его главный член имеет порядок $O(N^{-1})$. Следуя [80], вычислим $\beta^*(N) = M[N\hat{\beta}(N) - (N-1)\hat{\beta}(N-1)]$:

$$\begin{aligned} \beta^*(N) &= N\beta + \alpha_1 + \alpha_2/N + \alpha_3/N^2 + \ldots \\ &- [(N-1)\beta + \alpha_1 + \alpha_2/(N-1) + \alpha_3/(N-1)^2 + \ldots] \\ &= \beta - \alpha_2/(N(N-1)) - \alpha_3(2N-1)/(N(N-1))^2 - \ldots \end{aligned}$$

Здесь главный член смещения имеет порядок уже $O(N^{-2})$.

Название метода «складного ножа» [28, 97] объясняется поочередным исключением из обработки результатов отдельных серий. Перечислим его этапы:

1. вычисляют статистические оценки

   $\bar{\beta}$ — среднее по всем N сериям,

   $\bar{\beta}_j$ — средние по $N-1$ сериям (исключается j-я), $j=\overline{1,N}$;

2. формируют «приведенные» результаты

$$\tilde{\beta}_j = N\bar{\beta} - (N-1)\bar{\beta}_j, \qquad j=\overline{1,N}; \tag{6.3.2}$$

3. в качестве искомой величины принимают их среднее

$$\hat{\beta} = \left[N^2\bar{\beta} - (N-1)\sum_{j=1}^{N}\bar{\beta}_j\right]/N = N\bar{\beta} - (1-1/N)\sum_{j=1}^{N}\bar{\beta}_j;$$

4. вычисляют дисперсию оценки $\hat{\beta}$ через $\{\tilde{\beta}_j\}$ по формуле вида (5.1.3);

5. при необходимости строят доверительный интервал с помощью таблиц распределения Стьюдента.

Глава 7

Эксперименты над моделями

7.1 Введение

7.1.1 Постановка задачи

Задача проектирования, например, сети обслуживания должна решаться с позиций системного подхода, учитывающего ее многоаспектность и иерархичность. Естественно, что те же свойства приобретает и процесс проектирования, проходящий уже на фазе моделирования несколько этапов и требующий применения разнообразного математического аппарата [76]. В процессе имитационного *эксперимента* над моделью с учетом случайных исходов каждого прогона необходимо:

1. классифицировать факторы на существенные и несущественные;
2. разделить и оценить количественно влияния факторов и их комбинаций на целевую функцию;
3. найти наивыгоднейшую комбинацию факторов.

Структура эксперимента (количество прогонов, выбор исходных данных для них и характер обработки результатов) определяется его целью. Первая проблема решается средствами дисперсионного анализа, вторая — регрессионного и факторного (скрининг), третья — методами статистической оптимизации.

7.1.2 Целевая функция

Эксперимент начинается с выбора одной или более *метрик* — целевых функций, которые количественно оценивают полезность системы. Типичной задачей оптимизации СМО является обеспечение минимального среднего времени реакции системы при ограниченных затратах или минимизация затрат при среднем времени реакции не более заданного. Более продвинутый вариант ее — обеспечить обслуживание максимального процента заявок за время не более заданного или минимизация времени, за которое успевает обслужиться заданный процент заявок. Требования к реакции системы могут задаваться дифференцированно по типам заявок.

Затраты следует рассматривать приведенными к году штатной эксплуатации системы. Этот комплекс вопросов применительно к вычислительным системам обсуждается, например, в книге [56], устаревшей в своей конкретике, но не в методическом отношении.

7.1.3 Базовые понятия теории экспериментов

Измерения значений целевой функции в теории планирования экспериментов называются *наблюдениями*, а все их множество — *результатами*. Мы хотим изучить влияние одного или нескольких управляемых *факторов* на результаты наблюдений.

Назначим значения (*уровни*) факторов, которые будут определять условия каждого прогона. Если влияние некоторого фактора зависит от значения другого и не сводится к сумме влияний, мы говорим о *взаимодействии* этих факторов. Взаимодействия могут быть между любым числом факторов.

Будем моделировать много раз, называя каждый *прогоном*. Комбинация уровней определяет условия прогона. Когда условия прогона (исключая рандомизацию) неизменны, мы называем прогоны *репликами*. Результаты в совокупности заполняют матрицу результатов. Протяженность матрицы по измерению должна быть равна числу уровней этого фактора. Каждая размерность матрицы результатов сопоставляется одному из факторов, одно дополнительное измерение может использоваться для реплик. Все установки ДСЧ варьируются от реплики к реплике.

Если используются несколько метрик, должно быть столько же результирующих матриц. После очередного прогона следует *параллельно* заполнить одноименную клетку всех матриц.

7.2 Дисперсионный анализ

К большинству сложных систем применим принцип Парето, согласно которому 20% факторов определяют свойства системы на 80% (аналогичное соотношение известно для авторов научной продукции коллективов [20], наименований предметов снабжения в системах управления запасами [54, 61] и др.). Поэтому первоочередной задачей исследователя имитационной модели является отсеивание несущественных факторов, позволяющее уменьшить размерность задачи оптимизации модели.

Анализ дисперсии (Analysis of Variation — ANOVA) оценивает отклонение наблюдений от общего среднего. Затем вариация разбивается на части, каждая из которых имеет свою причину. Остаточная часть вариации (residual), которую не удается связать с условиями эксперимента, считается его случайной ошибкой. Ошибка оценивается в среднеквадратических отклонениях. Обычно только те эффекты, которые превышают величину СКО, считаются значимыми. Для подтверждения значимости используется специальный тест — F-статистика. В связи с большой актуальностью этой задачи однофакторные (`One-Way`) и многофакторные (`Multi-Way`) подпрограммы `ANOVA` включаются во многие пакеты математических подпрограмм — как общего, так и специально статистического назначения. `ANOVA` является важной составной частью обсуждаемой ниже общецелевой системы моделирования *GPSS World*.

Рассмотрим для простоты двухфакторную задачу, в которой первый фактор был зафиксирован на уровнях $i = \overline{1, I}$, а второй на уровнях $j = \overline{1, J}$, и оценим значимость влияния второго фактора. Суммарное рассеивание откликов

$$S = \sum_i \sum_j (y_{ij} - y_{..})^2 = \sum_i \sum_j (y_{.j} - y_{..})^2 + \sum_i \sum_j (y_{ij} - y_{.j})^2 = S_1 + S_2,$$

где

$$S_1 = \sum_i \sum_j (y_{.j} - y_{..})^2 = I \sum_j (y_{.j} - y_{..})^2 \qquad (7.2.1)$$

есть рассеивание средних для уровней второго фактора относительно общего, а

$$S_2 = \sum_i \sum_j (y_{ij} - y_{.j})^2 = \sum_j \sum_i (y_{ij} - y_{.j})^2 \qquad (7.2.2)$$

представляет рассеивание откликов внутри уровней. В обеих формулах точки вместо индекса означают усреднение по всем возможным его значениям. Величина S_1 имеет $J-1$ степеней свободы (J слагаемых, по которым определено их среднее); S_2 имеет $J(I-1)$ степеней свободы (J сумм по I слагаемых, в каждой из которых определено среднее). Поправка на число связей позволяет получить несмещенную оценку СКО при отсутствии других эффектов.

Если верна нулевая гипотеза о независимости ожидаемых откликов от уровня фактора j, то нормированные числом степеней свободы компоненты рассеивания можно представить в виде

$$\begin{aligned} S_1/(J-1) &= \sigma_Y^2 \chi_{J-1}^2, \\ S_2/(J(I-1)) &= \sigma_Y^2 \chi_{J(I-1)}^2. \end{aligned}$$

Здесь σ_Y^2 — общая дисперсия откликов, по предположению не зависящая от уровней факторов, а χ_M^2 — случайная величина, распределенная по закону χ^2 с M степенями свободы. Отношение R нормированных компонент рассеивания должно быть подчинено известному распределению Фишера $F_{J-1,J(I-1)}$ с указанной парой степеней свободы [66]. Если R больше порога $r(P)$, выбираемого в зависимости от доверительной вероятности P (см., например, табл. 7.1), рассматриваемый фактор является *значимым*. В противном случае верна нулевая гипотеза.

При имитации ошибки опытов можно сделать независимыми, применяя в разных опытах различные последовательности случайных чисел. Эти ошибки не обязаны быть нормально распределенными с общей дисперсией, однако F-критерий мало чувствителен к нарушению нормальности [73].

Ниже дан пример ANOVA-таблицы. Внизу таблицы общая сумма квадратов разделена, и компоненты ее связаны с факторами и их взаимодействиями. Остаточная сумма квадратов показана в предыдущей строке, названной «Ошибка». Средняя сумма квадратов ошибочного члена используется для оценки СКО эксперимента.

Таблица 7.1. Критические значения F-критерия по уровню 95%

Степени свободы ошибки	Степени свободы факторов					
	1	2	3	4	5	6
1	161	200	216	225	230	234
2	18.5	19.0	19.2	19.2	19.3	19.3
3	10.1	9.55	9.28	9.12	9.01	8.94
4	7.71	6.94	6.59	6.39	6.26	6.16
5	6.61	5.79	5.41	5.19	5.05	4.95
6	6.00	5.14	4.76	4.53	4.39	4.28
7	5.59	4.74	4.35	4.12	3.97	3.87
8	5.32	4.46	4.07	3.83	3.69	3.58
9	5.12	4.27	3.86	3.69	3.48	3.37
10	4.97	4.10	3.71	3.48	3.33	3.22
11	4.84	3.98	3.59	3.36	3.20	3.10
12	4.74	3.89	3.49	3.26	3.11	3.00
13	4.67	3.81	3.41	3.18	3.03	2.92
14	4.60	3.74	3.34	3.11	2.96	2.85
15	4.54	3.68	3.29	3.06	2.90	2.79
16	4.49	3.63	3.24	3.01	2.85	2.74
17	4.45	3.59	3.20	2.97	2.81	2.70
18	4.41	3.56	3.16	2.93	2.77	2.66
19	4.38	3.52	3.13	2.90	2.74	2.63
20	4.35	3.49	3.10	2.87	2.71	2.60

Таблица 7.2. ANOVA

Источник вариации	Сумма квадратов	Степени свободы	Средний квадрат	F	Критическое F $(p = 0.05)$
A	28.000	2	14.000	5.600	3.89
B	21.000	3	7.000	2.800	3.49
AB	11.000	6	1.833	0.733	3.00
Ошибка	30.000	12	2.500		
Всего	90.000	23			

Уровень		Счетчик	Среднее	Min	Max	95% интервал
A	B					
1	1	2	3.000	2.000	4.000	[0.564,5.436]
1	2	2	1.000	1.000	1.000	[-1.436,3.436]

Каждая сумма квадратов имеет делитель, равный числу *степеней свободы*. Факторы и их взаимодействия в статистической модели представляются в таблице отдельными строками. В каждой строке имеем сумму квадратов и степени свободы, связанные с этой оценкой. Из них выводятся другие необходимые величины. Простое деление дает средний квадрат; делением этой величины на среднеквадратическую ошибку из нижней строки таблицы получаем F-статистику для этого эффекта. Наша таблица показывает, что эффект фактора A значимый, а эффекты B и AB незначимы.

Иногда эксперимент не обнаруживает эффект, даже если он реально существует. Одна из наших целей — сделать такую ситуацию возможно менее вероятной. Для этого имеются два пути: увеличить F-статистику или уменьшить критическое значение. Желательно удалить части суммы квадратов ошибок, связанные с любым важным фактором, не включенным в анализ — тогда F-статистика станет больше. При естественнонаучных экспериментах это достигается проведением их в максимально однородных условиях. При имитационном моделировании следует учесть дополнительные факторы, которые влияют на результаты эксперимента.

Два других подхода связаны с увеличением числа степеней свободы ошибочного члена. Первый состоит в увеличении числа реплик. Это самый затратный, но простой и надежный путь. Вторая возможность связана с планированием эксперимента и совершенствованием статистической модели, положенной в основу ANOVA. Средний квадрат ошибки фактически есть остаток после удаления других квадратов. Если мы найдем приемлемый путь сохранить больше данных после вычитания эффектов, мы получим оценку СКО с бóльшим числом степеней свободы. Критическое значение F-статистики будет меньше, увеличится мощность критерия. Именно это происходит при игнорировании некоторых взаимодействий.

В многофакторных экспериментах часто удаляют взаимодействия высокого порядка, чтобы улучшить имеемую информацию о главных факторах и взаимодействиях низкого порядка. Если взаимодействия высокого порядка отсутствуют, дополнительные степени свободы используются для лучшей оценки F-статистики. Кроме того, увеличение числа степеней свободы означает меньшее критическое значение F. В противном случае лучше использовать дополнительные реплики.

Не все эксперименты требуют «репличного» измерения. При использовании метода латинских квадратов ANOVA способна получить оценку СКО и без этого. Когда ANOVA не может рассчитать F-

статистику, нужно либо добавить реплики, либо удалить из статистической модели взаимодействия высокого порядка. Оба этих подхода требуют изменения в организации вызова ANOVA. Однако наиболее эффективные методы улучшения статистической значимости результатов — уменьшение их дисперсии (см. главу 8).

7.3 Скрининг-эксперименты

Дисперсионный анализ определяет, есть ли эффект. *Регрессионный* анализ позволяет прогнозировать отклик (значение целевой функции) в некоторой точке пространства параметров.

Для простоты и определенности представим результаты полного двухфакторного эксперимента в виде

$$y = \mu + \alpha_A \cdot A + \alpha_B \cdot B + \alpha_{AB} \cdot AB + \varepsilon. \tag{7.3.1}$$

Здесь
- μ — математическое ожидание отклика,
- A, B — факторы,
- α_A, α_B, α_{AB} — коэф. их влияния порознь и совместно,
- ε — случайная ошибка с нулевым математическим ожиданием и дисперсией, не зависящей от значений факторов.

Равенство (7.3.1) представляет собой *регрессию* y по факторам A и B. Вид регрессии определяется структурой правой части. В данном случае это квадратичная функция двух переменных без «чистых квадратов», позволяющая прогнозировать линейные эффекты и парное взаимодействие факторов. Более сложные виды функции регрессии (логарифмическая, гиперболическая и т. д.) применяются на основе теоретических соображений или достаточных экспериментальных данных. Ожидаемое значение отклика вычисляется по формуле регрессии с известными коэффициентами для значений факторов, лежащих внутри выбранных диапазонов. Непосредственной задачей регрессионного анализа (скрининг-эксперимента) является оценка коэффициентов регрессии.

Число уровней каждого фактора обычно ограничивается двумя (тогда объем полного эксперимента с k факторами составит 2^k прогонов). Кроме того, мы пытаемся уменьшить число прогонов, не жертвуя наиболее важной информацией. Последнее соображение приводит к *дробным* факторным экспериментам. Дробность плана определяется

долей числа требуемых прогонов от объема полного факторного эксперимента и может быть целой отрицательной степенью двойки.

Теория *планирования экспериментов* рассматривает проблему построения наиболее экономичных планов, позволяющих получить упомянутые коэффициенты. Планы характеризуются типом регрессионной зависимости, разрешением (способностью разделения главных эффектов и их комбинаций), размещением исследуемых точек в окрестности опорной, количеством значений каждого параметра и т. д. Рациональный план содержит значительно меньшее число точек, чем полный факторный эксперимент.

Каждый прогон дробного эксперимента кодируется последовательностью латинских букв, обозначающих принявшие максимальные значения факторы. Компоненты, заведомо включаемые в план, называются *генераторами эксперимента*. В качестве генератора обычно выбираются взаимодействия наивысшей степени как средство смешивания главного эффекта со взаимодействиями $(k-1)$-го порядка в надежде, что последними можно пренебречь. Для пятифакторного эксперимента, рассматриваемого нами для конкретности, выбирается генератор I=ABCDE (все факторы принимают максимальные значения). Затем назначается дробность эксперимента m и выбираются $m2^k-1$ факторов и их комбинаций, эффект которых по результатам статистической обработки необходимо различать. В частности, при $k=5$ и $m=1/2$ (план вида 2^{5-1}) их будет 15. В порядке возрастания кратностей взаимодействий получаем A, B, C, D, E, AB, AC, AD, AE, BC, BD, BE, CD, CE, DE. Добавив ADCDE, имеем план эксперимента — табл. 7.3.

Таблица 7.3. План эксперимента 2^{5-1}

Фактор	ABCD	A	B	C	D	E	AB	AC	AD	AE	BC	BD	BE	CD	CE	DE
A	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	-1
B	1	-1	1	-1	-1	-1	1	-1	-1	-1	1	1	1	-1	-1	-1
C	1	-1	-1	1	-1	-1	-1	1	-1	-1	1	-1	-1	1	1	-1
D	1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	1	-1	1	-1	1
E	1	-1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	1	-1	1	1

В этой таблице +1 и -1 означают соответственно максимальный и минимальный уровни фактора (относительно середины диапазона). По компонентам плана строим «алиас-группы» неразличимых эффектов, получаемые умножением расчетного эффекта на выбранный генератор. Степени факторов учитываются по модулю 2 — попросту говоря, четные степени

отбрасываются. Например, для эффекта CE получаем ABCDE · CE=AB · (CC) · D · (EE)=ABD. В итоге имеем 15 групп

```
A  = BCDE          AB = CDE          BD = ACE
B  = ACDE          AC = BDE          BE = ACD
C  = ABDE          AD = BCE          CD = ABE
D  = ABCE          AE = BCD          CE = ABD
E  = ABCD          BC = ADE          DE = ABC
```

Факторы и их комбинации, попавшие в одну группу, по своему эффекту неразделимы: его невозможно распределить между членами группы. Может быть даже хуже: два сильно, но противоположно действующих фактора в одной группе будут компенсировать друг друга, группа покажется несущественной и такой же, но глубоко ошибочный вывод будет сделан относительно упомянутых факторов. Таким образом, нежелательно иметь в алиас-группах более чем по одному главному эффекту. Обычно удается сформировать группы так, что каждый главный эффект и наиболее важные парные взаимодействия оказываются в отдельных группах, которые дополняются только взаимодействиями высокого порядка. Если наиболее важные эффекты не разнесены по разным группам, интерпретация результатов будет затруднительной. Отметим, что любая компонента плана может быть заменена на другой элемент той же алиас-группы, а «максимальный» столбец плана — на минимальный.

Для планов меньшей дробности берут несколько генераторов — обычно тройных и парных взаимодействий с попарно общим фактором. Произведения исходных генераторов дают производные генераторы. Например, для эксперимента 2^{5-2} (четверть от полного) при заявленных эффектах A, B, C, D, E, AD, AE стартовые генераторы I=ABC=CDE дополняются ABDE, что дает алиас-группы

```
A  = BC   = ACDE = BDE            E  = ABCE = CD   = ABD
B  = AC   = BCDE = ADE            AD = BCD  = ACE  = BE
C  = AB   = DE   = ABCDE          AE = BCE  = ACD  = BD
D  = ABCD = CE   = ABE
```

В случае эксперимента 2^{5-3} (восьмая доля) можно рассчитывать на выявление только части главных эффектов. При стартовых генераторах I=ABC=ADE=CD имеем производные BCDE=BAD и алиас-группы

```
A = BC = DE   = ACD = ABCDE = BD
B = AC = ABDE = BCD = CDE   = AD
C = AB = ACDE = D   = BDE   = ABCD
```

Посмотрим, что мы можем сделать при четырех прогонах, составляющих этот план. Эффект E не появляется ни в какой группе, и, даже если он является главным, мы не получим о нем никакой информации. Главные эффекты C и D находятся в одной группе и потому неразличимы. При переходе к дробности 1/4 с 8 прогонами наблюдается улучшение: все главные эффекты — в разных группах. Однако информацию о парных взаимодействиях будет трудно интерпретировать. Дробность 1/2 (16 прогонов) заметно лучше. Здесь изолируются и даже отделяются от парных взаимодействий все главные факторы. Более того, изолируются даже парные взаимодействия нескольких первых главных эффектов. По-видимому, априорно следует выбрать вариант эксперимента 2^{5-1}.

Для каждой компоненты плана с учетом расстановки плюс- и минус-единиц в его матрице записывается линейное уравнение регрессии относительно расчетных эффектов и свободного члена. Правой частью уравнения становится результат соответствующего прогона. Компоненты полученного решения делятся на размах (произведения размахов) отображаемых факторов, что дает коэффициенты их влияния относительно середин диапазонов.

Отчет о скрининг-эксперименте сообщает силу каждого эффекта и, если может, вычисляет F-статистику для тех алиас-групп, членами которых являются главные эффекты. На этой основе можно планировать дальнейшие действия: игнорировать незначащие факторы, более тщательно исследовать существенные, оптимизировать последние.

Теория планирования экспериментов является весьма развитой областью прикладной математики. В ней рассматриваются многоуровневые планы, симметричные, рототабельные (вращаемые), типа латинского квадрата и т. д. — см. [2, 22, 36, 28].

7.4 Оптимизирующие эксперименты

7.4.1 Общий подход

При определении тенденций оптимизации помогает рассмотрение *компонент* целевой функции. Например, в задачах управления запасами [61] минимизируется сумма затрат на хранение избыточного (среднего) запаса, организацию поставок и «штрафы» за неполное или несвоевременное обеспечение спроса. Большой удельный вес одной из упомянутых составляющих указывает на необходимость соответственно:

- уменьшения уровня заказа, предельного уровня запаса или объема заказываемой партии;
- увеличения максимального запаса или объема заказа;
- увеличения порога заказа.

Оптимизация систем обслуживания по данным моделирования имеет существенную специфику:

- дискретность пространства аргументов,
- большую трудоемкость получения каждого очередного значения отклика,
- ограниченную точность вычисления отклика,
- отсутствие аналитического выражения для целевой функции и, возможно, части ограничений.

Дискретность аргументов связана с тем, что выбор решения часто осуществляется из дискретных рядов устройств с фиксированными характеристиками (быстродействием, емкостью памяти, набором алгоритмов функционирования и т. п.) и состоит в назначении *целого* числа устройств или расширяющих блоков. Это обстоятельство определяет необходимость использования при оптимизации методов дискретного программирования вплоть до элементарного перебора сначала на редкой решетке, затем — на более частой в перспективной окрестности оптимума. Крайне желательно, однако, сделать этот перебор направленным. Предварительно надо убедиться в единственности или пологом характере экстремума.

Значительная *трудоемкость* вычисления отклика заставляет Авыбирать метод, позволяющий достичь экстремума за минимальное число шагов, хотя бы и ценой сложного определения направления и длины шага в пространстве параметров.

Ограниченная (в статистическом смысле) точность вычисления отклика оправдывает применение весьма «либеральных» критериев остановки процесса оптимизации. Таким критерием может быть, в частности, статистически незначимое улучшение целевой функции на последнем шаге. С другой стороны, ограниченная точность дает основания использовать достаточно грубые аппроксимации поверхности отклика (в частности, регрессии невысокого порядка). Заметим, однако, что в окрестности экстремума аппроксимация должна быть как минимум квадратической.

Отсутствие точного аналитического выражения для целевой функции (и, возможно, части ограничений) является дополнительным аргументом в пользу уже обсуждавшихся простых аппроксимаций.

Оптимизирующий эксперимент в пространстве непрерывных параметров обычно базируется на одном из вариантов метода скорейшего спуска (подъема) для нахождения оптимума. Это предполагает:

- построение эмпирической аппроксимации поверхности отклика;
- вычисление градиента (его проекции являются коэффициентами чувствительности целевой функции к изменению параметров, то есть коэффициентами регрессии);
- определение направления спуска и величины шага по нему — с учетом ограничений на значения параметров.

Эффективность *процесса оптимизации* в решающей степени зависит от статистической погрешности вычисления целевой функции.

7.4.2 Новый метод вычисления градиента

Ключевой проблемой техники оптимизации является оценка градиента $\hat{F} = \partial Y/\partial\theta$ в пространстве параметров системы θ. Для получения k компонент градиента необходимо провести минимум k дополнительных экспериментов. При малых $\Delta\theta$ дисперсия результата (разности двух зашумленных величин, деленной на $\Delta\theta$) будет велика, а при больших $\Delta\theta$

получается плохая аппроксимация градиента. Новые перспективы в решении этой задачи открывает *метод возмущений* [94, 106], позволяющий (при некоторых ограничениях) оценивать градиент по *одной* реализации процесса.

Рассмотрим процесс моделирования работы одноканальной передачи сообщений вида $GI/G/1$ с обслуживанием по принципу FIFO. Обозначим

θ — темп обслуживания в канале (сек/бит),
H — длина заголовка сообщения,
L_i — длина текста i-го сообщения, $i = 1, 2, \ldots$

Тогда время обработки i-го сообщения

$$B_i = (H + L_i)\theta = \gamma + L_i\theta,$$

где $\gamma = H\theta$. Обозначим через $\{A_i\}$ случайные интервалы между последовательными заявками и проследим динамику изменения числа заявок — рис. 7.1.

Рис. 7.1. Динамика числа заявок в цикле занятости

Время пребывания в системе i-й заявки

$$V_i = \left(t_0 + \sum_{j=1}^{i} B_j\right) - \left(t_0 + \sum_{j=2}^{i} A_j\right) = \sum_{j=1}^{i} B_j - \sum_{j=2}^{i} A_j.$$

Пусть m-й период непрерывной занятости (ПНЗ) начинается с (k_m+1)-й заявки и содержит n_m заявок. Тогда оценка среднего времени пребывания

$$\hat{v}(N) = \frac{1}{N}\sum_{m=1}^{M}\sum_{i=1}^{n_m}\left(\sum_{j=1}^{i} B_{k_m+j} - \sum_{j=2}^{i} A_{k_m+j}\right).$$

При увеличении θ на $\Delta\theta$ прирост каждого времени обслуживания составит $\Delta B_i = (H + L_i)\Delta\theta = (\Delta\theta)/\theta)B_i$. При определении времени пребывания V_i в пределах ПНЗ этот эффект будет накапливаться:

$$\Delta V_i = \sum_{j=1}^{i} \Delta B_i = (\Delta\theta)/\theta)\sum_{j=1}^{i} B_i. \tag{7.4.1}$$

Обозначим $\tau_{m-1}(\Delta\theta)$ дополнительную задержку, переходящую в m-й ПНЗ из $(m-1)$-го. Приращение среднего времени пребывания

$$\begin{aligned}\Delta\hat{v}(N) &= \frac{1}{N}\sum_{m=1}^{M}\sum_{i=1}^{n_m}\left[\tau_{m-1}(\Delta\theta) + \frac{\Delta\theta}{\theta}\sum_{j=1}^{i} B_{k_m+j}\right] \\ &= \frac{1}{N}\left[\frac{\Delta\theta}{\theta}\sum_{m=1}^{M}\sum_{i=1}^{n_m}\sum_{j=1}^{i} B_{k_m+j} + \sum_{m\in D(\Delta\theta)} n_m\tau_{m-1}(\Delta\theta)\right],\end{aligned}$$

где $D(\Delta\theta) = \{m|\tau_{m-1}(\Delta\theta) > 0\}$. Естественно допустить, что

$$\lim_{\Delta\theta\to 0}\lim_{N\to\infty}\frac{1}{N}\sum_{m\in D(\Delta\theta)} n_m\tau_{m-1}(\Delta\theta) = 0.$$

Следовательно,

$$\frac{\partial\hat{v}}{\partial\theta} = \lim_{N\to\infty}\frac{1}{N}\left[\sum_{m=1}^{M}\sum_{i=1}^{n_m}\sum_{j=1}^{i} B_{k_m+j}\right]. \tag{7.4.2}$$

Получим теперь оценку $dT/d\gamma$. Поскольку $B_i = \gamma + L_i\theta$, при увеличении γ на $\Delta\gamma$ каждое B_i возрастает на $\Delta\gamma$. При этом формула (7.4.1) перейдет в

$$\Delta V_i = \sum_{j=1}^{i} \Delta\gamma = i\Delta\gamma = \Delta\gamma \sum_{j=1}^{i} 1.$$

Теперь ясно, что

$$\frac{\partial \hat{V}}{\partial \theta} = \lim_{N\to\infty} \frac{1}{N} \sum_{m=1}^{M} \sum_{i=1}^{n_m} \sum_{j=1}^{i} 1, \tag{7.4.3}$$

и накопление соответствующих сумм легко совместить с расчетом согласно (7.4.2). Таким образом, для оценки производных по нескольким параметрам достаточно обработать одну номинальную (невозмущенную) траекторию процесса.

Дальнейшим шагом является адаптивное моделирование, в котором после оценки градиента по небольшому числу ПНЗ выполняется коррекция параметров. Критерием локальной оптимальности является близость градиента к нуль-вектору. Численные эксперименты [94, 106] подтвердили эффективность этого подхода — трудоемкость оптимизации снижалась в сотни раз.

В заключение этого беглого обзора проблемы напомним общее правило оптимизации сложных систем — недопущение узких мест. Поэтому на каждом шаге оптимизации сети обслуживания нужно контролировать сбалансированность очередного приближения, выравнивая средние времена ожидания в узлах (с учетом вероятностей их посещения).

Глава 8

Понижение дисперсии

8.1 Проблема и методы

Известный недостаток имитационного моделирования — статистическая погрешность результатов — в условиях фантастического роста быстродействия ЭВМ перестал быть критически важным. Однако остаются задачи, в которых проблема понижения дисперсии результатов моделирования остается актуальной:

- анализ сильно загруженных систем массового обслуживания (СМО);
- расчет структурно сложных систем с сильно зависящими от состояния выходными показателями (например, те же СМО с отказами каналов);
- расчет вероятностей редких событий (здесь требуемое число испытаний для получения оценки с заданной относительной точностью обратно пропорционально искомой вероятности и квадрату допустимой погрешности).

Для получения оценок с достаточно узкими доверительными границами в рамках обсуждавшихся выше классических подходов необходимо очень большое число наблюдений. Этот недостаток метода был осознан и отчасти преодолен уже при первых попытках моделирования на ЭЦВМ (фон Нейманом и Уламом в рамках Манхэттенского проекта — при моделированиии рассеивания нейтронов). Более эффективные его формы предложены и описаны достаточно давно — см.

[98, 28, 48, 49, 50, 52, 57, 90], но до сих пор известны лишь узкому кругу теоретиков. Практики о них не знают, и примитивное применение имитационного моделирования в значительной степени дискредитировало этот мощный аппарат. К сожалению, во всех перечисленных источниках не обсуждаются или не раскрываются критически важные детали технологии МПД-моделирования, без которых воспользоваться МПД не удается.

Обычно не используемые возможности имитационного моделирования связаны с отказом от принципа «грубой силы» и применением методов понижения дисперсии (МПД) результатов *единичного наблюдения*. К этим методам относят:

- выбор оценок с наименьшей дисперсией,
- расслоенные выборки,
- значимые выборки,
- контрольные переменные,
- дополняющие выборки,
- параллельные выборки,
- условную имитацию.

Ниже дается краткое описание их с обоснованием (насколько позволяет ограниченный объем книги) расчетных схем. При окончательном суждении о целесообразности того или иного метода необходимо учитывать не только выигрыш от уменьшения дисперсии, но и накладные расходы вследствие усложнения процесса моделирования и обработки результатов, а также необходимости дополнительного программирования.

Заметим, что их общей идейной основой является *использование дополнительных знаний* о моделируемой системе. Предлагаемые некоторыми авторами методы МПД, не привлекающие таких знаний, могут «отметаться с порога»— по аналогии с проектами «вечных двигателей».

8.2 Выбор оценок с наименьшей дисперсией

Прежде всего напомним, что усреднение по модельному времени дает оценки с меньшей дисперсией, чем усреднение по числу наблюдений.

В частности, стационарную вероятность иметь в системе k заявок лучше оценивать по формуле

$$\hat{p}_k = \sum_i t_i(k)/T, \tag{8.2.1}$$

чем согласно

$$\hat{p}'_k = N(k)/N. \tag{8.2.2}$$

Здесь

$t_i(k)$	—	длина i-го отрезка времени, проведенного системой в k-м состоянии,
T	—	системное время моделирования,
$N(k)$	—	число наблюдений системы в k-м состоянии,
N	—	полное число наблюдений.

Разумеется, для возможности сопоставления оценок $\hat{p}_k$ и $\hat{p}'_k$ необходима представимость исследуемой системы вложенной цепью Маркова (см. [55]). Формальное доказательство представимости (8.2.1) для системы $M/G/1$ приводится также в [49].

Оценка среднего времени пребывания заявки в системе

$$\hat{v} = \hat{w} + \hat{b}, \tag{8.2.3}$$

где

$\hat{w}$	—	оценка среднего времени ожидания обслуживания,
$\hat{b}$	—	оценка средней деятельности обслуживания.

Дисперсия нашей оценки

$$D[\hat{v}] = D[\hat{w}] + D[\hat{b}], \tag{8.2.4}$$

поскольку времена ожидания и обслуживания статистически независимы. Заменим оценку $\hat{b}$ в (8.2.3) априорно известным математическим ожиданием ее:

$$\bar{v} = \hat{w} + b. \tag{8.2.5}$$

Дисперсия постоянной величины b равна нулю [1]; следовательно, оценка (8.2.5) будет иметь меньшую дисперсию, чем (8.2.3). Именно этот метод и применялся в лабораторной работе по проверке законов сохранения (разд. 2.3) вместо непосредственного получения моментов распределения времени пребывания заявки из имитационной модели. Выигрыш в точности будет расти с уменьшением загрузки системы по двум причинам:

[1] Это не значит, что длительность обслуживания постоянна!

- при малой загрузке время пребывания в основном определяется чистой длительностью обслуживания,
- уменьшение загрузки уменьшает дисперсию времени ожидания (см. разд. 5.1).

Аналогичный подход можно применить к вычислению длины цикла занятости T, состоящего из периода занятости T_B и периода простоя T_I. Здесь T_B следует оценить на модели, а длительность простоя определить по формуле для среднего остатка интервала между заявками

$$M[T_I] = a_2/(2a_1),$$

где $\{a_i\}, \quad i = 1, 2$ — начальные моменты распределения интервалов между смежными заявками рекуррентного потока. Для простейшего потока интенсивности λ

$$M[T_I] = a_1 = 1/\lambda.$$

Рассмотрим определение средней длины очереди. Формула Литтла $q = \lambda w_1$ связывает среднюю длину очереди q с интенсивностью входящего потока λ и средним временем ожидания w_1. Возникает вопрос о том, какие характеристики предпочтительно измерять в процессе моделирования, а какие получать пересчетом.

Сравним дисперсии времени ожидания в системе $M/M/1$ при прямой имитации и через формулу Литтла. Согласно формуле Полячека—Хинчина [15, 55], среднее время ожидания

$$w_1 = \frac{\lambda b_2}{2(1-\rho)} = \frac{\lambda \cdot 2/\mu^2}{2(1-\rho)} = \frac{1}{\mu}\frac{\rho}{1-\rho}.$$

Разлагая формулу для преобразования Лапласа распределения времени ожидания по моментам соответствующих распределений, находим второй момент распределения времени ожидания

$$w_2 = \frac{\lambda}{1-\rho}\left[\frac{2\,!}{0\,!3\,!}b_3 w_0 + \frac{2}{1\,!2\,!}b_2 w_1\right] = \frac{2\rho}{\mu^2(1-\rho)^2}. \tag{8.2.6}$$

Следовательно, дисперсия результатов наблюдения при прямой имитации

$$D_W = w_2 - w_1^2 = \frac{\rho(2-\rho)}{\mu^2(1-\rho)^2}. \tag{8.2.7}$$

Через формулу Литтла находим математическое ожидание *длины очереди*

$$q_1 = \lambda w_1 = \frac{\rho^2}{1-\rho}. \tag{8.2.8}$$

Математическое ожидание ее квадрата

$$\begin{aligned} q_2 &= \sum_{k=1}^{\infty}(k-1)^2 p_k = \sum_{k=1}^{\infty}(k^2-2k+1)(1-\rho)\rho^k \\ &= (1-\rho)\left[\sum_{k=1}^{\infty}k^2\rho^k - 2\sum_{k=1}^{\infty}k\rho^k + \sum_{k=1}^{\infty}\rho^k\right]. \end{aligned}$$

Известно, что

$$\sum_{k=1}^{\infty}kx^k = \frac{x}{(1-x)^2}, \quad \sum_{k=1}^{\infty}k^2x^k = \frac{x(x+1)}{(1-x)^3}.$$

Выполняя подстановки, находим

$$q_2 = \frac{\rho^2(1+\rho)}{(1-\rho)^2}.$$

Соответственно дисперсия длины очереди

$$D_q = \frac{\rho^2(1+\rho)}{(1-\rho)^2} - \frac{\rho^4}{(1-\rho)^2} = \frac{\rho^2+\rho^3-\rho^4}{(1-\rho)^2}.$$

При вычислении среднего времени ожидания через очередь по формуле Литтла дисперсия оценки составит D_q/λ^2. Ее отношение к дисперсии прямой оценки среднего времени ожидания составит

$$\omega = \frac{(\rho^2+\rho^3-\rho^4)\cdot\mu^2(1-\rho)^2}{\lambda^2(1-\rho^2)(2\rho-\rho^2)} = \frac{1+\rho-\rho^2}{\rho(2-\rho)}.$$

С помощью приведенных выше формул легко установить относительный выигрыш от оценки средней длины очереди через среднее время ожидания по формуле Литтла: $f = D_q/(\lambda^2 D_w)$. Результаты расчета для $\mu = 1$ и $\rho = \lambda/\mu$ сведены в табл. 8.1. Очевидно, что реальный (и не слишком существенный) выигрыш от подобного пересчета возможен только при умеренной загрузке.

Таблица 8.1. Основные показатели системы $M/M/1$

ρ	w	q	D_w	D_q	f
0.45	0.818	0.368	2.306	0.835	1.788
0.50	1.000	0.500	3.000	1.250	1.666
0.55	1.222	0.672	3.938	1.864	1.564
0.60	1.500	0.900	5.250	2.790	1.476
0.65	1.857	1.207	7.163	4.234	1.399
0.70	2.333	1.633	10.111	6.588	1.330
0.75	3.000	2.250	15.000	10.688	1.267
0.80	4.000	3.200	24.000	18.560	1.208
0.85	5.667	4.817	43.444	36.205	1.154
0.90	9.000	8.100	99.000	88.290	1.101
0.95	19.000	18.050	399.000	378.150	1.050

Можно предположить, что эти выводы качественно сохранятся и для СМО других типов. Поэтому формулу Литтла и ее обобщения следует использовать для пересчета от временны́х к «штучным» характеристикам.

Еще один часто обсуждаемый способ этого класса — определение среднего времени пребывания заявки в системе через статистическое среднее времени ожидания и известное априорно среднее время обслуживания. Он может дать заметный эффект лишь в тех случаях, когда дисперсия чистого времени обслуживания соизмерима с дисперсией времени ожидания. В нашем случае дисперсия обслуживания равна 1. Сравнивая ее с числами из четвертого столбца таблицы, убеждаемся в незначительности потенциального выигрыша, что и подтвердил эксперимент.

Итак, для определения средних характеристик СМО посредством имитационного моделирования целесообразно оценить среднее время ожидания $\hat{w}$. Остальные оценки вычисляются по формулам (8.2.5), (8.2.6), а среднее число заявок в системе — согласно

$$\hat{L} = \hat{q} + \lambda b_1.$$

Разумеется, эти рекомендации не исключают независимого получения всех упомянутых характеристик в процессе моделирования для контроля правильности работы модели проверкой выполнения законов сохранения.

В работе Карсона и Лоу [81] отмечается, что использование этих приемов в проведенных экспериментах давало уменьшение дисперсии оценок до 200 раз.

В заключение отметим, что расчет средней длины очереди через среднее время ожидания заявки требует моделирования очереди: запоминания моментов прибытия заявок, их продвижения и т. п. Прямой расчет упомянутого показателя эти затраты исключает. Теперь ясно, чем оплачивается повышение точности оценок через временные показатели.

8.3 Расслоенные выборки

Идея использования расслоенных (стратифицированных) выборок восходит к демографической статистике и методикам обработки результатов опросов общественного мнения. Она состоит в разбивке полной совокупности результатов наблюдений на сравнительно однородные (с малой внутренней дисперсией) слои. Пусть

N — общее число наблюдений,
N_k — число наблюдений в k-м слое,
p_k — вероятность отнесения наблюдения к k-му слою,
X_{kj} — j-е наблюдение k-го слоя,
D_k — дисперсия единичных наблюдений k-го слоя,
x_k — теоретическое среднее для слоя,
x — общее среднее.

Тогда оценка среднего для k-го слоя

$$\hat{x}_k = \left(\sum_{j=1}^{N_k} X_{kj} \right) / N_k, \tag{8.3.1}$$

а стратифицированная оценка общего среднего

$$\hat{x}^S = \sum_{k=1}^{K} p_k \hat{x}_k \tag{8.3.2}$$

(вероятности $\{p_k\}$ предполагаются известными).

Дисперсия стратифицированной оценки

$$D[\hat{x}^S] = \sum_{k=1}^{K} p_k^2 D[\hat{x}_k] = \sum_{k=1}^{K} p_k^2 D_k / N_k.$$

Обычно выбирается $N_k = N p_k$, так что

$$D[\hat{x}^S] = \sum_{k=1}^{K} p_k D_k \Big/ N. \tag{8.3.3}$$

Дисперсия простой оценки (с глобальным усреднением)

$$D[\hat{x}] = D[\hat{x}^S] + \sum_{k=1}^{K} p_k(\hat{x}_k - \hat{x})^2/N \tag{8.3.4}$$

(см. [28, ч. 1]) и возрастает с увеличением разницы в средних для слоев. Считается [90, с. 57], что эффективность этого подхода пропорциональна квадрату числу слоев.

Успех расслоения обусловливается двумя требованиями:

- стратифицирующая переменная должна быть коррелирована с откликом (результатом);
- должны быть известны вероятности попадания в слои.

Последнее затруднение снимается, если проводить апостериорную стратификацию и определять вероятности как $p_k = N_k/N$. При этом число наблюдений в каждом классе должно быть не менее 20, так что малочисленные классы следует объединять.

Наиболее естественной областью применения расслоенных выборок к имитации СМО является учет переменных внешних условий (изменение интенсивностей потоков, длительностей обслуживания, числа действующих каналов и т. п.). В частности, представляется целесообразным моделировать n-канальную СМО с ненадежными каналами как обычную систему с числом каналов от 1 до n и затем взвесить результаты с учетом распределения числа исправных каналов, полученного расчетом или моделированием процесса отказов и восстановлений. Ограничением этого подхода является требование существования стационарного режима в каждом слое.

В работе [50] предложено проводить расслоение выборки при регенеративном моделировании по длительности обслуживания головной заявки периода занятости. Развивая эту идею, естественно расслаивать выборку по числу заявок, прибывших за время обслуживания головной заявки.

8.4 Значимые выборки

Основная идея этого приема состоит в эквивалентном преобразовании задачи путем замены распределения исходной случайной величины. Пусть мы хотим оценить

$$y = \int_0^\infty h(x) f(x)\, dx, \tag{8.4.1}$$

где $f(x)$ — плотность вероятностей, так что $y = M[h(X)]$. Для оценки этой величины мы формируем N значений $\{X_i\}$ в соответствии с плотностью распределения $f(x)$ и вычисляем

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} h(X_i). \tag{8.4.2}$$

Теперь перепишем исходную зависимость в форме

$$y = \int_0^\infty \frac{h(x) f(x)}{g(x)} g(x)\, dx,$$

где $g(x)$ — другая плотность вероятностей, и обозначим

$$\tilde{h}(x) = \frac{h(x) f(x)}{g(x)}. \tag{8.4.3}$$

Нетрудно видеть, что

$$M[\tilde{h}(x)] = \int_0^\infty \tilde{h}(x) g(x)\, dx = y,$$

так что оценка искомой величины

$$\hat{y}' = \frac{1}{N} \sum_{i=1}^{N} \tilde{h}(\tilde{X}_i) \tag{8.4.4}$$

при $\{\tilde{X}_i\}$, сформированных в соответствии с плотностью $g(x)$, является несмещенной.

Известно [28], что дисперсия оценки (8.4.4) минимальна при выборе в качестве новой плотности

$$g^*(x) = \frac{h(x)f(x)}{\int\limits_0^\infty h(x)f(x)\,dx} = \frac{h(x)f(x)}{y}. \tag{8.4.5}$$

Эта плотность имеет существенные значения в точках, для которых достаточно велики либо вычисляемая функция $h(x)$, либо исходная плотность $f(x)$, либо они обе, что и объясняет название метода. При выборе плотности согласно (8.4.5) оценка (8.4.4) теоретически имеет *нулевую* дисперсию.

Трудность реализации данного подхода состоит в том, что знаменатель рекомендуемой плотности (8.4.5) — искомая (и, следовательно, неизвестная) величина. Это препятствие можно обойти, организовав серию экспериментов с последовательным уточнением y. В первом эксперименте, естественно, придется воспользоваться обычной оценкой (8.4.2). Оценки, получаемые на двух последовательных шагах, следует взвешивать пропорционально числу испытаний и обратно пропорционально дисперсиям:

$$\hat{y} = c_1\hat{y}_1 + c_2\hat{y}_2,$$

где

$$\begin{aligned} c_1 &= (N/D)\Big/\left(\sum_{i=1}^{2} N_i/\hat{D}_i\right), \\ c_2 &= 1 - c_1. \end{aligned}$$

Усложняются также расчет значений целевой функции (формула (8.4.3) вместо $h(x)$) и генерация случайных чисел в соответствии с более сложным выражением для плотности. Данные эффекты можно нейтрализовать подбором соответствующих аппроксимаций. В частности, в [28] в этих целях рекомендуется гамма-плотность.

Метод значимых выборок в специфических модификациях широко используется для моделирования высоконадежных систем [34, 48]. В [28] сообщается, что применение значимых выборок позволяло уменьшить дисперсию оценок более чем на два порядка.

8.5 Контрольные переменные

8.5.1 Расчетная схема для одномерного случая

Пусть X — случайная величина с неизвестным математическим ожиданием $M[X] = x$ и $\hat{x}$ — его несмещенная оценка, найденная стандартными методами главы 5. Случайная переменная Y может быть выбрана в качестве *контрольной переменной* для X, если она коррелирована с X и имеет известное математическое ожидание y. Новая оценка формируется как старая плюс линейная комбинация контрольных переменных. Коэффициенты комбинации должны быть выбраны оптимально.

Получим с помощью контрольной переменной несмещенную оценку x с дисперсией меньшей, чем у $\hat{x}$. Введем новую случайную величину $X(b) = X - b(Y - y)$. Для любой константы b выражение

$$\bar{x}(b) = \hat{x} - M[b(Y - y)] \tag{8.5.1}$$

также даст несмещенную оценку x, поскольку $M[Y - y] = 0$. Дисперсия контролируемой случайной величины

$$D[X(b)] = D[X - bY] = D[X] - 2bK_{XY} + b^2 D[Y], \tag{8.5.2}$$

если $b > 0$ и корреляционный момент $K_{XY} > bD[Y]/2$, будет меньше $D[X]$. Приравнивая нулю производную правой части (8.5.2) по b, получаем оптимальное значение поправочного коэффициента

$$b^* = K_{XY}/D[Y]. \tag{8.5.3}$$

Минимальная дисперсия $D[X(b)]$ достигается при подстановке b^* и равна

$$\begin{aligned} D^* &= D[X] - 2K_{XY}^2/D[Y] + K_{XY}^2/D[Y] = D[X] - K_{XY}^2/D[Y] \\ &= D[X](1 - K_{XY}^2/(D[X]D[Y])) = D[x](1 - \rho_{XY}^2). \end{aligned} \tag{8.5.4}$$

Ее величина стремится к нулю по мере приближения к единице модуля коэффициента корреляции ρ_{XY} — независимо от знака последнего.

Эта идея была применена к улучшению оценки среднего времени ожидания в системе $M/M/1$ по известной средней длине очереди. Коэффициент корреляции получался накапливанием произведений $(W - w)(Q - q)$ в момент выборки заявки из очереди. Случайное время

ожидания W определялось как разность текущего значения таймера и вносимого в паспорт заявки момента постановки заявки в очередь. Длина очереди Q определялась на момент начала обслуживания как число заявок в очереди после выбираемой в канал. Это можно делать, так как распределение числа заявок в очереди перед прибытием меченой заявки совпадает с распределением числа прибывших за время ожидания (закон сохранения стационарной очереди). Дисперсии W и Q определялись стандартным способом. Заметим, что подстановка вместо них *теоретических* значений приводила к абсурдным результатам — коэффициентам корреляции, превышающим единицу по модулю.

Результаты применения описанного подхода (вариант 7 табл. 8.4, который можно считать «самым умным»), оказались заметно точнее остальных — в особенности при умеренном числе испытаний. К сожалению, выбор контрольных переменных весьма ограничен, а к попыткам сконструировать их искусственно приходится относиться критически. В частности, работа с *суммарной трудоемкостью* очереди к удовлетворительным результатам не привела. Вероятной причиной является то, что здесь фактически нет *новой* контрольной переменной — обсуждаемая сумма является частью статистики по основной переменной. Видимо, так же обстоит дело и с предлагаемой в [98] средней из 100 предыдущих задержек.

Другим вариантом метода контрольных переменных является переход к случайной величине

$$X(b) = X - b(Y - Z), \tag{8.5.5}$$

где Y и Z — случайные величины с равными (не обязательно известными) математическими ожиданиями. Такую пару для СМО можно подобрать, например, на основе законов сохранения. Легко показать, что в этом варианте вновь верны формулы (8.5.3) и (8.5.4), если заменить контрольную переменную Y из (8.5.1) на $\Delta = Y - X$.

Описанный метод является одним из самых элегантных и многообещающих МПД. Он имеет глубокий естественно-научный смысл — привлечение дополнительных теоретических знаний об исследуемом процессе, что, собственно, и позволяет уменьшить неопределенность в результатах моделирования.

8.5.2 Выбор контрольных переменных

Успешное применение метода контрольных переменных требует:

- подбора сильно коррелированых переменных;
- наличия оценок дисперсий и корреляционных моментов.

Теоретические оценки в редких случаях известны для $D[Y]$ и никогда — для K_{XY}, так что их приходится определять по данным моделирования. Можно надеяться, что неточное определение b^* незначительно уменьшит выигрыш от применения контрольной переменной. Поэтому для получения нужных оценок достаточно ограничиться умеренным числом опытов.

Контрольные переменные можно классифицировать на внутренние и внешние. К *внутренним* относятся те, которые определяются при моделировании исходной системы одновременно с искомыми характеристиками, к *внешним* — получаемые при моделировании упрощенного аналога исходной. Приведем таблицу возможных внутренних контрольных переменных, коррелированных со средним временем ожидания начала обслуживания.

Таблица 8.2. Внутренние контрольные переменные

Переменная	Математ. ожидание
Интервал между заявками	$a_1 = 1/\lambda$
Длительность обслуживания	b_1
Вероятность свободного состояния (для одноканальной СМО)	$1 - \lambda b_1$
Число занятых каналов	$\lambda b_1 = b_1/a_1$
Время ожидания (для $M/G/1$)	$\lambda b_2/[2(1-\lambda b_1)]$
Период занятости (для $M/G/1$)	$b_1/(1-\lambda b_1)$

Все они (кроме пары интервал между заявками — время обслуживания) коррелированы и между собой. В литературе отмечается также заметная корреляция между временем ожидания n-й заявки и разностью накопленных к этому моменту сумм длительностей обслуживания и интервалов между смежными заявками. К сожалению, найти внутреннюю контрольную переменную, достаточно коррелированную с требуемым откликом, удается не всегда.

Внешние контрольные переменные формируются при дополнительном моделировании упрощенного варианта исходной системы. Выбрав в качестве контрольной переменной аналогичную искомой величине и применив при ее формировании те же серии случайных чисел, что в основной модели, можно гарантировать достаточно высокую корреляцию откликов. Заметим к тому же, что для упрощенного варианта можно (например, численными методами) рассчитать более широкий круг характеристик, чем перечисленные в табл. 8.2. В частности, в этих целях применимы известные инварианты ТМО (например, независимость распределения числа заявок в очереди от типа приоритетной дисциплины при условии, что последняя не учитывает *фактические* длительности обслуживания заявок). Однако полученный выигрыш должен окупить затраты на параллельное моделирование, тогда как при использовании внутренних контрольных переменных он достается практически «бесплатно».

8.5.3 Расчетная схема для многомерного случая

Рассмотрим многомерное обобщение теории контрольных переменных. Перепишем (8.5.1) в векторной форме

$$\tilde{\boldsymbol{X}}\,(B) = \boldsymbol{X} - B\,(\boldsymbol{Y} - \boldsymbol{y})\,, \tag{8.5.6}$$

где

$\tilde{\boldsymbol{X}}(B)$ и $\boldsymbol{X}$	—	столбцы $n \times 1$ (контролируемый и исходный векторы),
$\boldsymbol{Y}$ и $\boldsymbol{y}$	—	столбцы $m \times 1$ контрольных величин и их матожиданий,
B	—	пересчетная матрица размера $n \times m$.

Обозначим через $\{B_i\}$ строки матрицы B. Тогда i-й элемент контролируемого вектора

$$\tilde{X}_i(B) = X_i - B_i(\,\boldsymbol{Y} - \boldsymbol{y}\,), \tag{8.5.7}$$

а его дисперсия

$$\tilde{d}_i = M[(\tilde{X}_i - \tilde{x}_i)^2] = M[(X_i - x_i)^2]. \tag{8.5.8}$$

Вычислим выражение в квадратных скобках из (8.5.8). С учетом (8.5.7)

оно равно

$$\begin{aligned}
[X_i - B_i(\boldsymbol{Y} - \boldsymbol{y}) - x_i]^2 &= [(X_i - x_i) - B_i(\boldsymbol{Y} - \boldsymbol{y})]^2 \\
&= [(X_i - x_i) - \sum_{j=1}^{m} b_{ij}(Y_j - y_j)]^2 \\
&= (X_i - x_i)^2 - 2\sum_{j=1}^{m} b_{ij}(X_i - x_i)(Y_j - y_j) \\
&+ \sum_{j=1}^{m}\sum_{k=1}^{m} b_{ij}(Y_j - y_j)(Y_k - y_k)b_{ik}.
\end{aligned} \tag{8.5.9}$$

Введем корреляционные моменты

$$\begin{aligned}
K_{YY} &= M[(\boldsymbol{Y} - \boldsymbol{y})(\boldsymbol{Y} - \boldsymbol{y})^T], \\
K_{X_iY} &= M[(X_i - x_i)(\boldsymbol{Y} - \boldsymbol{y})^T], \\
K_{YX_i} &= M[(\boldsymbol{Y} - \boldsymbol{y})(X_i - x_i)^T] = K_{X_iY}^T,
\end{aligned} \tag{8.5.10}$$

представляющие собой соответственно корреляционную $(m \times m)$ матрицу контрольного вектора, вектор-строку $(1 \times m)$ корреляции X_i и контрольного вектора и транспозицию этой строки. Применяя к формуле (8.5.9) оператор математического ожидания, согласно (8.5.8) получаем с учетом (8.5.10)

$$\tilde{d}_i = d_i - 2K_{X_iY}B_i^T + B_iK_{YY}B_i^T. \tag{8.5.11}$$

В этом выражении от нас зависит выбор элементов строки B_i. Приравнивая нулю производные от развернутой записи (8.5.11) по этим элементам, получаем систему линейных уравнений, в матричной форме имеющую вид $-K_{X_iY} + B_iK_{YY} = 0$, откуда следует оптимальный выбор

$$B_i^* = K_{X_iY}K_{YY}^{-1}. \tag{8.5.12}$$

При этом будет получена минимальная дисперсия i-й составляющей контролируемой оценки

$$d_i^* = d_i - K_{X_iY}K_{YY}^{-1}K_{YX_i}. \tag{8.5.13}$$

Последнюю формулу можно переписать в виде

$$d_i^* = (1 - R_{X_iY}^2)d_i,$$

где $R^2_{X_iY} = K_{X_iY}K^{-1}_{YY}K_{YX_i}/d_i$ — квадрат множественного коэффициента корреляции. Множитель $1 - R^2_{X_iY}$ определяет относительное уменьшение дисперсии стандартной оценки.

Важный частный случай изложенного — использование нескольких контрольных переменных для уменьшения дисперсии скалярной величины ($n = 1$). Здесь пересчетная матрица заменяется оптимальным вектором

$$\boldsymbol{\beta} = K_{XY}K^{-1}_{YY}, \tag{8.5.14}$$

и $D[X(\boldsymbol{\beta})] = (1 - R^2_{XY})D[X]$. Ниже будут рассмотрены два метода построения доверительных интервалов для контролируемой оценки [97].

8.5.4 Доверительный интервал из теории регрессий

Пусть K — число независимых реплик. Для k-й реплики обозначим через X_k, Y_k и Y_{qk} значения оценок X, $\boldsymbol{Y}$, Y_q соответственно, $k = \overline{1, K}$. Предположим, что X и $\{Y_1, \ldots, Y_m\}$ имеют многомерное нормальное распределение. Используемые нами оценки и контрольные переменные — выборочные средние, так что это допущение приемлемо априорно и было подтверждено экспериментальными данными (см. разд. 8.5.7). Тогда условное распределение X при заданном $\boldsymbol{Y}$ будет одномерным нормальным со средним

$$E[X|\boldsymbol{Y} = \boldsymbol{y}] = x + \boldsymbol{\beta}^T(\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{Y}}),$$

где $\boldsymbol{\beta}$ дается (8.5.14), вектор $\boldsymbol{\mu}_Y = M[\boldsymbol{Y}]$ предполагается известным, а минимальная дисперсия

$$D[X|\boldsymbol{Y} = \boldsymbol{y}] = \sigma^2_X(1 - R^2_{XY}). \tag{8.5.15}$$

При условии $\bar{Y}_k = \bar{y}_k,\ k = \overline{1, K}$, имеем классическую проблему регрессии с $\bar{\boldsymbol{X}} = Z^T\boldsymbol{\alpha} + \boldsymbol{\varepsilon}$, где $\boldsymbol{X}^T = \{X_1, \ldots, X_K\}$, $\bar{\boldsymbol{\alpha}} = (x, \boldsymbol{\beta})^T$ — вектор-столбец с компонентами x и $\boldsymbol{\beta}$, а матрица

$$Z^T = \begin{bmatrix} 1 & y_{11} - \mu_1 & \ldots & y_{m1} - \mu_m \\ \vdots & \vdots & & \vdots \\ 1 & y_{1K} - \mu_1 & \ldots & y_{mK} - \mu_m \end{bmatrix}.$$

Здесь ε есть вектор независимых нормально распределенных нормальных переменных с нулевым средним и дисперсией, задаваемой (8.5.15). Пусть оценки по методу наименьших квадратов для $\boldsymbol{\beta}$ и $\boldsymbol{\mu}$ суть $\hat{\boldsymbol{\beta}}$ и $\hat{x}_1$. Тогда

$$D[\hat{x}_1] = s_{11}\sigma^2, \tag{8.5.16}$$

где s_{11} — верхний левый элемент $(YY^T)^{-1}$. Из теории регрессий несмещенная оценка для σ^2 есть

$$\hat{\sigma}^2 = \left[\sum_{k=1}^{K} X_k^2 - \sum_{k=1}^{K}(\hat{x}_1 + \hat{\boldsymbol{\beta}}\,(\,\boldsymbol{y_k} - \boldsymbol{\mu_Y}\,))^2\right]\Big/(K-m-1). \tag{8.5.17}$$

Теперь мы можем построить доверительный интервал, и доверительная вероятность

$$\Pr\{\hat{x}_1 - \sqrt{s_{11}}\hat{\sigma}t_{K-m-1}(1-\delta/2) \le x \le \hat{x}_1 + \sqrt{s_{11}}\hat{\sigma}t_{K-m-1}(1-\delta/2)\} = 1-\delta, \tag{8.5.18}$$

где $t_n(x)$ есть $100x$-процентная квантиль t-распределения с n степенями свободы. Поскольку правая часть (8.5.18) постоянная, эта формула верна безусловно. Далее, поскольку оценки по методу наименьших квадратов безусловны, оценка $\hat{x}_1$ является несмещенной безусловно.

8.5.5 Доверительный интервал для «складного ножа»

Здесь мы уже не будем делать никаких допущений относительно распределений X и $\{Y_q\}$. Если бы вектор $\boldsymbol{\beta}$ был известен, мы могли бы положить $X_k(\,\boldsymbol{\beta}\,) = X_k - \boldsymbol{\beta}^T(\,\boldsymbol{Y_k} - \boldsymbol{\mu_Y}\,)$ для $k = \overline{1,K}$ и оценить x через $\bar{X}(\,\boldsymbol{\beta}\,) = \bar{X} - \boldsymbol{\beta^T}\,(\,\bar{\boldsymbol{Y}} - \bar{\boldsymbol{\mu}}_Y)$, где $\bar{X}$ есть выборочное среднее из X_k, а $\bar{\boldsymbol{Y}}$ — выборочное среднее из $\bar{Y}_k$. Из (8.5.15) следует

$$D[\bar{X}(\,\boldsymbol{\beta}\,)] = (1 - R_{XY}^2)(\sigma_X^2/K). \tag{8.5.19}$$

Поскольку $\boldsymbol{\beta}$ неизвестен, будет разумно оценить x по формуле

$$\hat{x}_2 = \bar{X} - \hat{\boldsymbol{\beta}}_{\mathbf{1}}^{\boldsymbol{T}}\,(\,\bar{\boldsymbol{Y}} - \boldsymbol{\mu}_Y),$$

где $\hat{\boldsymbol{\beta}}_{\mathbf{1}} = \bar{K}_{XY}\bar{K}_{YY}^{-1}$, а $\bar{K}_{XY}$ и $\bar{K}_{YY}$ суть выборочный вектор ковариаций и выборочная ковариационная матрица, элементы которых даются

формулами

$$
\begin{aligned}
(\bar{K}_{XY})_q &= (K-1)^{-1}\sum_{k=1}^{K}(X_k-\bar{X})(Y_{qk}-\bar{Y}_q),\\
(\bar{K}_{YY})_{qr} &= (K-1)^{-1}\sum_{k=1}^{K}(Y_{qk}-\hat{Y}_q)(Y_{rk}-\bar{Y}_r).
\end{aligned}
$$

Вообще говоря, оценка $\hat{x}_2$ получается смещенная, и t-статистика не может использоваться для построения доверительного интервала.

Как показано в разд. 6.3, метод складного ножа будет уменьшать смещение оценки. Обозначим $\hat{x}_{2,k}$ оценку, которая имеет тот же вид, что $\hat{x}_2$, но с исключенными $\{X_k, Y_{1k}, \ldots, Y_{mk}\}$. Псевдооценками будут $\theta_k = K\hat{x}_2 - (K-1)\hat{x}_{2k}$ для $k = \overline{1,K}$, а результирующая оценка $\hat{x}_2$ есть $\bar{\theta}$ — выборочное среднее из $\{\theta_k\}$.

Обозначим $\hat{\sigma}_\theta^2$ выборочную дисперсию θ_k. Тогда при очень общих условиях $\sqrt{K}(\bar{\theta}-x)/\hat{\sigma}_\theta$ имеет t-распределение Стьюдента с $K-1$ степенями свободы, и доверительный интервал может быть построен обычным способом — без предположений, сделанных в регрессионном методе. Можно показать, что $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_1$ и $\hat{x}_2 = \hat{x}_1$.

Все величины, необходимые для реализации метода контрольных переменных, вычисляются по формулам главы 5. При расчете корреляционных моментов для контрольных величин используются их теоретические математические ожидания, а для искомых — статистические оценки таковых.

С помощью формулы (8.5.13) можно определить число опытов K_i^*, необходимое для снижения дисперсии i-го вычисляемого показателя до заданного уровня. Требуемое число испытаний $K^* = \max_i K_i^*$. Опыт практических расчетов [52] показывает, что разумные требования по точности удовлетворяются при значениях числа прогонов K порядка 30. Уменьшение дисперсии в зависимости от исходных данных составляло от двух [96] до 30...50 раз [91, 95].

8.5.6 Потери в уменьшении дисперсии

Физически очевидно, что при увеличении числа m контрольных переменных выигрыш в дисперсии будет увеличиваться, поскольку для уточнения искомых показателей привлекается дополнительная информация.

Однако параллельно возрастают объем вычислительной работы и погрешности, вносимые статистической природой оценок корреляционных моментов.

Ниже приведено простое выражение [93, 96, 97] для увеличения теоретически минимальной дисперсии при выборе $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_1$:

$$D[\hat{x}_1] = [(K-2)/(K-m-2)](1 - R^2_{XY})(\sigma^2_X/K). \qquad (8.5.20)$$

Итак, относительное увеличение дисперсии

$$\delta D = \frac{K-2}{K-m-2} = 1 + \frac{m}{K-m-2}.$$

Для снижения потерь есть два пути: увеличить число реплик K и уменьшить число контрольных переменных m. Первый метод невыгоден тем, что в каждой реплике приходится отбрасывать начальный участок (до входа в стационарный режим). При сохранении общего числа событий это заметно снижает эффективность.

Второй путь состоит в отборе минимального числа переменных по убыванию модуля их корреляции с искомой оценкой. Выбор надо делать последовательно: первая; вторая и последующие — *при условии* ранее сделанных выборов. Этот выбор лучше делать на основе пилотного эксперимента умеренного (но не слишком малого) объема.

Формула (8.5.20) верна в предположении о многомерной нормальности переменных $\{X, Y_1, \ldots, Y_m\}$ и условии $K > m + 2$, но эксперименты [97] показывают хорошую точность ее безотносительно к первой оговорке.

8.5.7 Контрольные переменные и замкнутые сети

Наиболее интересные для приложений компьютерные сети обслуживания (с приоритетами, блокировками, неэкспоненциальными обслуживаниями) не поддаются строгому аналитическому расчету, и для их анализа приходится применять имитационное моделирование. Моделирование сетей намного более трудоемко, чем в случае изолированных систем, и при работе с ними применение методов уменьшения дисперсии особенно актуально. Приведем обзор имитационного эксперимента, описанного в [97]. Рассмотрим модель интерактивной вычислительной сети, представленной на рис. 8.1.

Рис. 8.1. Схема интерактивной сети

В этой модели многоканальный узел 1 представляет терминалы (их количество равно числу заявок, циркулирующих в сети, так что очереди к узлу 1 не образуются). Узел 2 соответствует центральному процессору, узлы с номерами $\overline{3, M}$ — внешним устройствам (накопителям на магнитных дисках). Каждый узел имеет один или много каналов. В сети циркулируют заявки D типов, причем численность d-популяции равна $Z(d)$, а общая численность — Z. При моделировании делаются следующие допущения:

а) последовательность посещаемых узлов для каждого типа — неприводимая
марковская цепь с пространством состояний $I(d)$ и матрицей передач $R(d)$;

б) распределения длительности обслуживания $\{T_i(d)\}$ для всех i, d независимы и имеют средние значения $\{b_i(d)\}$;

в) эти распределения не зависят от пройденных маршрутов.

Основные случайные переменные — маршрутизация и длительности обслуживания. Мы хотим найти функции этих основных переменных, для

которых могут быть определены математические ожидания и которые коррелированы с характеристиками сети (такими, как средние времена ожидания и средние времена ответа для подсетей). Интуитивно ясно, например, что ожидание коррелировано с мерами обслуживания и потока.

Пусть $\boldsymbol{\pi(d)}$ — вектор-строка вероятностей состояний неприводимой марковской цепи для d-заявок. Вектор $\boldsymbol{\pi(d)}$ есть единственное решение системы уравнений $\boldsymbol{\pi(d)}\, R(d) = \boldsymbol{\pi(d)}$, и $\sum\limits_{i \in I(d)} \pi_i(d) = 1$.

Определим как *событие* в сети уход заявки из узла обслуживания (в замкнутой сети это единственный класс первичных событий). Пусть $e_{d,i}(N)$ — число событий для d-заявок в узле i из первых N событий, $T_{d,i}(N)$ — сумма связанных с ними времен обслуживания, $e_d(N) = \sum\limits_{i=1}^{S} e_{d,i}(N)$. Назовем

$T_{d,i}(N)/e_{d,i}(N)$ — *переменная времени обслуживания* (выборочное среднее время обслуживания d-заявок в узле i),

$e_{d,i}(N)/e_d(N)$ — *переменная потока* (доля d-событий, приходящихся на узел i),

$W_{d,i}(N)/e_d(N)$ — *переменная работы* (сумма времен обслуживания заявок типа d в узле i, приходящаяся на одно d-событие).

Следует ожидать, что

$$\lim_{N\to\infty} E[T_{d,i}(N)/e_{d,i}(N)] = b_i(d), \tag{8.5.21}$$

$$\lim_{N\to\infty} E[e_{d,i}(N)/e_d(N)] = \pi_i(d), \tag{8.5.22}$$

$$\lim_{N\to\infty} E[W_{d,i}(N)/e_d(N)] = \pi_i(d) b_i(d). \tag{8.5.23}$$

В эксперименте N выбиралось достаточно большим, чтобы можно было пользоваться этими асимптотическими значениями. Отметим, что асимптотическое значение для отношения $e_d(N)/N$ (доли событий d-типа) неизвестно. Ожидается, что переменные времени обслуживания, потока и работы коррелированы с характеристиками ответа сети. Любая из них может служить контрольной переменной и легко подсчитывается в процессе моделирования. Так как переменные работы отражают и потоки, и времена обслуживания, они представляются наиболее привлекательными.

Эксперимент имел целью сравнить:

- выбор этих переменных;
- два метода получения доверительных интервалов (разд. 8.5.4 и 8.5.5);
- точность вычисления фактора потерь (8.5.20).

Организация эксперимента. Исследовалось около 20 вариантов сетей — в частности, модель рис. 8.1 с центральным вычислителем, дисплеями и несколькими внешними накопителями. Рассматривался также вариант с ограниченной емкостью подсети (при нехватке оперативной памяти). Для оценки работы моделируемой системы определялись:

$w_i(d)$ — среднее время чистого ожидания для d-заявок в узле i,

$\lambda(d)$ — средняя интенсивность ухода d-заявок (возврата к пользователям),

$v(d)$ — среднее время ответа (от генерации d-заявки до получения ответа).

Из условия баланса заявок имеем

$$v(d) = [w_2(d) + b_2(d) + \sum_{i=3}^{S} r_{2,i}(d)(w_i(d) + b_i(d))]/r_{2,1}(d).$$

Здесь $\{r_{2,i}\}$ — элементы второй строки матрицы передач R. Напомним номера узлов: 1 — пользователи, 2 — центральный процессор, остальные — внешние накопители.

По каждой сети было выполнено 20 независимых прогонов ($K = 20$). Всего в одном прогоне было 3000 испытаний. Первая тысяча событий для исключения влияния начальных условий отбрасывалась. В процессе моделирования определялись вышеупомянутые рабочие показатели. Проводился (для определения доверительной вероятности) метаэксперимент: все повторялось 50 раз. В качестве оценки дисперсии принималась средняя (по 50 повторениям внешнего цикла) из внутренних оценок.

Внутренней оценкой $D[\bar{X}]$ была выборочная дисперсия X_k, $k = \overline{1, K}$. Внутренняя оценка $D[\hat{\mu}_1]$ определялась как $s_{11}\hat{\sigma}^2$ — см. (8.5.16).

Наконец, внутренней оценкой $D[\bar{\theta}]$ была $\hat{\sigma}_\theta^2$. Альтернативно можно было бы найти дисперсию точечной оценки по 50 базовым экспериментам. Оценивался также минимальный коэффициент уменьшения дисперсии $1 - R_{XY}^2$ по 1000 независимых испытаний.

Результаты эксперимента. Уменьшение дисперсии оказалось больше в малых сетях (размер сети характеризовался произведением $D * M$). Это объясняется меньшим числом контрольных переменных. Лучшим образом себя показали «рабочие» переменные (максимальное снижение дисперсии от 0.15 до 0.70) — в особенности для оценки среднего ожидания в наиболее загруженных узлах и интенсивности выходящего потока. Метод регрессии дал значительно более узкие интервалы, чем «складной нож». Формула для коэффициента потерь экспериментами подтвердилась полностью.

Авторы [97] считают важнейшим направлением дальнейших исследований априорный выбор контрольных переменных.

8.6 «Противоположные» выборки

Проведем два *зависимых* опыта и возьмем в качестве оценки искомой величины полусумму их результатов

$$\tilde{X} = (X_1 + X_2)/2. \tag{8.6.1}$$

Дисперсия этой оценки

$$D[\tilde{X}] = (D[X_1] + D[X_2] + 2K_{X_1X_2})/4. \tag{8.6.2}$$

Если принять $D[X_1] = D[X_2]$, а этого легко добиться соответствующей организацией эксперимента, то

$$D[\tilde{X}] = D_X(1 + \rho)/2, \tag{8.6.3}$$

где ρ — коэффициент корреляции X_1 и X_2. При отрицательном ρ дисперсия усредненного результата двух опытов может быть существенно уменьшена и в идеальном случае ($\rho = -1$) сведена к нулю. Описанный прием широко используется в измерительной технике в виде разнообразных компенсационных схем.

Обычный способ получения отрицательной корреляции — брать в одном опыте исходное случайное число из [0,1), а в другом его дополнение до единицы (отсюда термин antithetic variables — противоположные переменные). Для этого в датчик равномерно распределенных чисел можно встроить вспомогательную переменную — «кувыркающуюся единицу», знак которой определяет, какое из упомянутых действий должно быть выполнено при данном обращении:

```
...........
if (c.eq.1) then
   u=random()
else
   u=1.0-u
end if
dt =-log(u)/lam ! для EXP закона
c=-c
```

Эта переменная должна сохранять свое значение между обращениями к подпрограмме, для чего в Алголе 60 использовался описатель own, в ПЛ/1 — STATIC. В большинстве реализаций Фортрана 77 и Фортрана 90 это свойство обеспечено автоматически [2]. При совместном использовании в одном прогоне $\{U_i\}$ и их дополнений до единицы можно в полтора раза уменьшить число обращений к равномерному датчику.

Для надежности и в особенности в предвидении перехода на другие платформы можно:

- использовать в операторах описания данных дополнительный атрибут SAVE;
- завершить описание процедуры одноименным оператором сохранения;
- воспользоваться COMMON-блоком.

Последний способ предпочтительнее, поскольку позволяет управлять не только генерацией случайных чисел, но и обработкой результатов моделирования.

[2] Рекомендуем проверить на вашей персональной ЭВМ.

Для пары *прогонов* можно использовать дополнение начальной установки до модуля датчика — получаемые числа $\{U_i\}$ (проверено!) будут тождественными.

К сожалению, последующие нелинейные функциональные преобразования могут «смягчить» запланированную противоположность. В общем случае корреляционный момент последовательных интервалов

$$K = \int_0^1 (F^{-1}(u) - \bar{x})(F^{-1}(1-u) - \bar{x})\, du. \tag{8.6.4}$$

Для распределения, равномерного на $[0, a]$, эта формула сводится к

$$\begin{aligned} K &= \int_0^1 (au - a/2)(a(1-u) - a/2)\, du = a^2 \int_0^1 (u - 1/2)(1 - u - 1/2)\, du \\ &= -a^2 \int_0^1 (u - 1/2)^2\, du = -a^2/12. \end{aligned}$$

Поскольку дисперсия упомянутого распределения равна $a^2/12$, в данном случае коэффицент корреляции равен -1.

Для треугольного (Симпсона) распределения на том же интервале имеем дисперсию $a^2/24$ и функцию распределения

$$F(x) = \begin{cases} 2(x/a)^2, & 0 \le x \le 1/2; \\ 1 - 2(x/a - 1)^2, & 1/2 < x \le 1. \end{cases}$$

Соответственно формула (8.6.4) преобразуется в

$$\begin{aligned} a^2 K &= \int_0^{1/2} (\sqrt{u/2} - 1/2)(1 - \sqrt{u/2} - 1/2)\, du \\ &+ \int_{1/2}^1 (1 - \sqrt{(1-u)/2} - 1/2)(\sqrt{(1-u)/2} - 1/2)\, du \\ &= -\int_0^{1/2} (\sqrt{u/2} - 1/2)^2\, du - \int_{1/2}^1 (1/2 - \sqrt{(1-u)/2})^2\, du = -1/24. \end{aligned}$$

Здесь тоже (и, по-видимому, для всех симметричных распределений) коэффициент корреляции -1 сохраняется.

С помощью формул из раздела 4.3 выполним соответствующие расчеты для распределения Вейбулла, полагая для упрощения обозначений масштабный параметр $T = 1$. Теперь (8.6.4) преобразуется в

$$K = \int_0^1 (\sqrt[k]{-\ln(1-u)} - \Gamma(1+1/k))(\sqrt[k]{-\ln u} - \Gamma(1+1/k))\, du.$$

Результаты были получены с помощью математического пакета $Scientific\ WorkPlace$ — численным интегрированием, а для $k = 1$ — и в символьной форме: $K = 1 - \pi^2/6$. Они сведены в табл. 8.3.

Таблица 8.3: Распределение Вейбулла и метод дополняющих переменных

Показатели	$k = 0.5$	$k = 0.7$	$k = 1.0$	$k = 1.5$	$k = 2.0$	$k = 3.0$
f_1	2.0000	1.2658	1.0000	0.9028	0.8862	0.8930
f_2	24.000	5.0291	2.0000	1.1906	1.0000	0.9028
f_3	720.00	37.234	6.0000	2.0000	1.3293	1.0000
D	20.000	3.4268	1.0000	0.3757	0.2146	0.1053
σ	4.4721	1.8512	1.0000	0.6129	0.4633	0.3246
μ_3	592.00	22.192	2.0000	0.2468	0.0627	0.0057
μ_3/σ^3	6.6188	3.4984	2.0000	1.0720	0.6311	0.1681
K	-3.8583	-1.3648	-0.6449	-0.3232	-0.2033	-0.1049
ρ	-0.1929	-0.3983	-0.6449	-0.8602	-0.9471	-0.9959
1+ρ	0.8071	0.6017	0.3551	0.1398	0.0529	0.0041

В этой таблице $\{f_i\}$ — начальные моменты, μ_3 — третий центральный момент, а μ_3/σ^3 — коэффициент асимметрии. Последняя строка показывает уменьшение дисперсии (и такое же уменьшение потребного числа испытаний) в расчете на *один* опыт [3]. Отметим, что случай $k = 1$ соответствует показательному распределению, а $k = 2$ — распределению Релея. Итак, для показательного закона дисперсия должна уменьшиться втрое, а для распределения Релея — почти в 20 раз. Заметим, что плотность последнего *визуально* очень далека от симметричной. По-видимому, данный метод будет хорошо работать для любых модальных распределений.

[3] Формула (8.6.3) предполагает проведение двух опытов.

Моделирование дало -0.6445 для показательного закона, -0.9475 для закона Релея и практически -1 — для треугольного закона, что и предсказывалось. Заметим, что в случае треугольного закона преобразование нелинейно. Следовательно, сохранению -1 мешает не нелинейность, а *асимметрия* распределения.

В [98] обсуждается также способ преодоления упомянутой нелинейности с помощью правила

$$X_1 = 2\bar{x} - X_1.$$

Оно гарантирует равенство $\rho_{X_1 X_2} = -1$.

Наконец, еще одним способом введения отрицательной корреляции откликов является «перекоммутация» датчиков, используемых для генерации интервалов между заявками и длительностей обслуживания соответственно. Основанием для этого приема является противоположность влияния увеличения $\{U_i\}$ и соответственно случайных реализаций названных переменных на продвижение заявок. При использовании этого способа необходима перекрестная синхронизация ДСЧ.

В таблице 8.4 для модели СМО $M/M/1$ с точным значением среднего времени ожидания $w = 9.0000$ сопоставляются результаты прямого моделирования и «глобальных» усреднений конечных результатов прогона.

Таблица 8.4. Оценки среднего времени ожидания

тыс.	Варианты						
испыт.	1	2	3	4	5	6	7
1	9.2585	7.5550	5.8514	13.5440	10.2668	6.9896	8.4349
2	7.2740	7.6912	8.1083	13.9010	10.1245	6.3481	8.4142
5	10.2430	8.9234	7.6037	11.3100	8.7735	6.2370	9.1977
10	9.8932	8.1200	6.3469	10.1690	8.0996	6.0241	9.1781
20	10.4430	9.2822	8.1213	9.4804	8.6781	7.8758	9.1850
50	8.9508	8.7864	8.6219	9.4447	8.9823	8.5198	9.0985
100	8.3893	8.6678	8.9462	9.5543	9.0949	8.6355	9.0594
200	8.5539	8.6312	8.7084	9.0043	8.9024	8.8004	8.9965
500	8.8487	8.9969	9.1451	8.9167	8.9926	9.0685	8.9968
1000	8.7570	8.8726	8.9882	8.9792	8.9990	9.0187	8.9978
2000	9.0587	9.0742	9.0897	9.1000	9.0414	8.9828	9.0129
5000	8.9634	8.9226	8.8819	9.0829	9.0076	8.9322	8.9898
10000	8.8977	8.8656	8.8335	9.0111	8.9580	8.9049	8.9221

В этой таблице:

- варианты 2 и 4 задают результаты прямого моделирования и моделирования с дополняющей начальной установкой, а вариант 3 — их полусумму;
- варианты 5 и 7 содержат результаты моделирования с раздельными датчиками и их перекоммутацией, а вариант 6 — их полусумму;
- вариант 8 представляет результаты применения контрольной переменной (см. следующий раздел).

Обращает на себя внимание очень медленная сходимость результатов всех видов «сольного» моделирования, что объясняется большой дисперсией времени ожидания (99.0). Чтобы довести среднеквадратическое отклонение статистического среднего до 0.01, требуется примерно $(100/0.01)^2 = 10^8$ испытаний. Эта оценка хорошо согласуется с результатами моделирования.

Усреднение результатов (случаи с дополняющими и перекрестными датчиками, в особенности последний) дает заметно лучшие результаты. Можно полагать, что перекрестное использование датчиков «глубже затрагивает систему». Эффект усреднения вполне окупает удвоение общего числа испытаний, в чем легко убедиться по следующей строке той же таблицы.

Отметим, наконец, неожиданный и нигде не обсуждавшийся результат: локальное усреднение, т. е. чередование основных и «дополняющих» переменных датчика в ходе *одного* прогона, хотя и уменьшает дисперсию результатов, но дает очень большое смещение оценок. Анализ показал, что при упомянутом чередовании среднее логарифмически преобразованных чисел остается единичным, а их дисперсия

$$D = \frac{1}{4}\left[\int_0^1 (-\ln u - 1)^2\,du + \int_0^1 (-\ln(1-u) - 1)^2\,du\right] = \frac{1}{4}[1+1] = 1/2$$

(интегралы считались численно с помощью пакета Scientific WorkPlace). Следовательно, второй начальный момент сгенерированного распределения длительного обслуживания с единичным средним составит 1.5 вместо необходимого при экспоненциальном законе 2.0. Соответственно по формуле Полячека—Хинчина произойдет уменьшение среднего времени

ожидания на четверть: при коэффициенте загрузки 0.9 с 9 до 6.75, что и подтвердил эксперимент по 200 тыс. реализаций с «кувыркающимся» датчиком длительности обслуживания (было получено 6.7067).

Ключевыми элементами описанной технологии являются монотонность преобразования случайных величин от входа к выходу (это обязательное условие отрицательной корреляции результатов «противоположных» опытов) и синхронизация наблюдений. Для последней необходимы:

- отдельный ДСЧ на каждую генерируемую случайную величину (интервал между заявками, время обслуживания, номер следующего узла сети и т. п.);

- одинаковое количество чисел $\{U_i\}$ на генерацию каждого очередного значения с неравномерным распределением (поэтому предпочтительным считается требующий одного U_i метод обратной функции, хотя бы и приближенно реализуемый);

- возможность либо фиксировать числа $\{U_i\}$ одного прогона для их использования в параллельном, либо запоминать начальную установку соответствующих ДСЧ и затем повторно генерировать ту же серию.

На рис. 8.2 приведена схема организации прогона при использовании m дополняющих переменных и повторной генерации случайных чисел в каждом из m туров. Заметим, что в начале прогона восстановление датчиков не происходит, и каждый прогон работает со своими сериями случайных чисел. Генерация серии «противоположных» чисел происходит автоматически, если начать ее с дополнения стартового элемента исходной серии до модуля соответствующего датчика.

Практически различные ДСЧ можно организовать размножением копий одного датчика и приданием им различных начальных значений. Для гарантии непересекаемости вырабатываемых серий нужно определить количество и длину серий потребных псевдослучайных чисел, после чего по программе из разд. 4.2.4 вычислить необходимые начальные установки. При использовании датчиков с дополнительным перемешиванием рестарт затрудняется.

Рис. 8.2. Работа с дополняющими переменными

Известно обобщение метода «противоположных» переменных, в котором проводятся m параллельных опытов со случайными аргументами

$$X_j = F^{-1}(U \cdot m/j \pmod 1), \qquad j = \overline{1, m}.$$

В работе [101] отмечается рост выигрыша в процессе моделирования многоканальных СМО при увеличении m от 2 до 8, хотя в некоторых случаях $m = 4$ работало хуже, чем $m = 2$.

В заключение этого раздела отметим, что соображения, изложенные здесь в связи с проблемой синхронизации, относятся также к методам, обсуждаемым в главе 6 и разд. 8.7.

8.7 Параллельные выборки

Часто приходится сравнивать разные варианты моделируемых систем, различающиеся параметрически или даже структурно. В таких случаях важны не столько абсолютные значения показателей работы сравниваемых систем, сколько их разность, которую желательно оценить возможно точнее. Результатом эксперимента будет разность показателей двух туров

$$\tilde{X} = X_1 - X_2. \tag{8.7.1}$$

Ее дисперсия

$$D[\tilde{X}] = D[\tilde{X}_1] + D[\tilde{X}_2] - 2K_{X_1X_2}$$

в случае близких дисперсий составляющих равна

$$D[\tilde{X}] \approx 2D[X] \cdot (1 - \rho). \tag{8.7.2}$$

Отсюда вытекает очевидная практическая рекомендация: использовать в параллельных турах одни и те же серии случайных чисел $\{U_i\}$ (разумеется, каждую по своему назначению). Схема организации прогона здесь аналогична рис. 8.2 с очевидной модификацией обработки результата.

В некоторых случаях описанную «разностную» схему целесообразно использовать для оценки абсолютного значения показателя. Пусть систему Σ_2 можно исследовать аналитически и получить точное значение искомой характеристики μ_2, а Σ_1 такому расчету не поддается. Оценка

$$\hat{\mu}_1 = \mu_2 + (X_1 - X_2), \tag{8.7.3}$$

полученная при моделировании обеих систем с общими ДСЧ, при $\rho > 0.5$ будет иметь меньшую дисперсию, чем результат прямого моделирования Σ_1, поскольку $D[\mu_2] = 0$.

Рассмотрим случай, когда в качестве оценки вариантов Σ_1 и Σ_2 выступает дисперсия ([64, с. 183]). В этом случае оценка $\Delta^* = \Delta D = D_1 - D_2$ дисперсий критериев q_1, q_2 вычисляется по независимым реализациям вектора (q_1, q_2) с помощью формулы $\Delta^* = \Delta D = \tilde{D}_1 - \tilde{D}_2$, где $\tilde{D}_1, \tilde{D}_2$ — эмпирические дисперсии упомянутых критериев, рассчитываемые по формуле

$$\tilde{D}_q = \frac{1}{N-1}\sum_{i=1}^{N}(q_i - M[q])^2.$$

Для оценки ΔD дисперсия

$$D[\Delta D] = D[\tilde{D}_1] + D[\tilde{D}_2] - 2K[\tilde{D}_1\tilde{D}_2].$$

Дисперсия эмпирических дисперсий вычисляется по формуле

$$D[\tilde{D}_q] = N[(\mu_4 - D_q^2)/N - 2(\mu_4 - 2D_q^2)/N^2 + (\mu_4 - 3D_q^2)/N^3]/(N-1),$$

где $\mu_4 = M[(q - M[q])^4]$ — центральный момент распределения четвертого порядка.

Ковариация дисперсий

$$K[\tilde{D}_1, \tilde{D}_2] \approx M[(q_1 - M[q_1])^2(q_2 - M[q_2])^2]/N.$$

Метод параллельных выборок требует двойного моделирования. Поэтому для получения выигрыша в машинном времени фактически необходимо иметь $\rho > 0.75$. Поскольку данный метод рекомендуется для сравнения *различных* систем, здесь может осложниться проблема синхронизации серий случайных чисел. В [80] с этой точки зрения рассматривается проблема сравнительного моделирования одноканальной и многоканальной систем одинаковой суммарной производительности.

8.8 Условная имитация

Условная имитация, как и контрольные переменные, комбинирует имитацию и дополнительные знания о моделируемом процессе, но является гораздо более гибким инструментом понижения дисперсии.

Пусть искомая характеристика СМО является математическим ожиданием функции двух независимых случайных величин X и Y:

$$z = \int_0^\infty \int_0^\infty g(x,y) f_X(x) f_Y(y)\, dx\, dy, \tag{8.8.1}$$

причем аналитический вид $g(x,y)$ неизвестен. Зафиксируем значение одного из случайных факторов и получим с помощью имитационной модели *условный* отклик

$$\hat{G}(y) = \int_0^\infty g(x,y) f_X(x)\, dx. \tag{8.8.2}$$

Тогда конечный результат можно получить численным интегрированием согласно

$$\hat{z} = \int_0^\infty G(y) f_Y(y)\, dy. \tag{8.8.3}$$

Проведем расслоение выборки имитируемых времен ожидания по значениям определяющих это ожидание факторов и запишем условное математическое ожидание $\bar{w}$ через найденные в слоях $\{w_k\}$:

$$\bar{w} = \sum_k p_k w_k.$$

Применяя к его слагаемым формулу для дисперсии произведения независимых случайных величин, имеем для дисперсии искомой оценки

$$\begin{aligned} D[\bar{w}] &= \sum_k \left\{ D[p_k] D[w_k] + p_k^2 D[w_k] + w_k^2 D[p_k] \right\} \\ &= \sum_k p_k^2 D[w_k] + \sum_k D[p_k](w_k^2 + D[w_k]). \end{aligned}$$

Точное знание вероятностей слоев $\{p_k\}$ сводит к нулю взвешенную сумму послойных вторых моментов длительности ожидания и значительно уменьшает $D[\bar{w}]$. Эта формула подсказывает организацию математического эксперимента: определить (предпочтительно аналитически) вероятности $\{p_k\}$ и — отдельно для каждого слоя — получить имитацией первые и вторые моменты времени ожидания.

Поскольку влияние фактора Y учитывается аналитически, дисперсия z будет меньше, чем в случае полного моделирования по всему пространству $X \times Y$.

Проиллюстрируем эту идею на аналитических моделях систем с очередями. Рассмотрим расчет среднего времени ожидания заявки в системе $M/G/1$ для равномерного на интервале $[0.3, 0.9]$ распределения времени обслуживания через соответствующую систему с регулярным обслуживанием (при переходе к имитационным моделям в последнем случае дисперсия результатов будет меньшей). Интересующий нас результат может быть получен по формуле Полячека—Хинчина

$$w = \frac{\lambda b_2}{2(1 - \lambda b_1)},$$

где λ — интенсивность входящего потока и $\{b_i\}$ — моменты распределения длительности обслуживания. Для упомянутого равномерного распределения $b_1 = 0.6$, $b_2 = 0.39$ и при $\lambda = 1$ среднее время ожидания $w = 0.4875$. Моделирование системы $M/D/1$ при длительности обслуживания 0.3, 0.6 и 0.9 и 5000 испытаний дало соответственно 0.00627, 0.415 и 2.919. Подставляя эти результаты в формулу численного интегрирования по Симпсону, получаем

$$w = \frac{0.9 - 0.3}{6}(0.00627 + 4 * 0.415 + 2.919) = 0.464.$$

Непосредственное моделирование равномерно распределенной длительности обслуживания дало 0.439, то есть заметно худший результат. Разумеется, утроение числа испытаний даст выигрыш в точности и на исходной модели (было получено 0.471). Однако работоспособность предложенного подхода несомненна.

Реализация этой идеи имеет существенное ограничение: все частные варианты обсчитываемых имитационных моделей должны иметь докритический коэффициент загрузки $\rho = \lambda b/n < 1$. Например, при равномерно распределенной на отрезке $[0.3, 1.5]$ длительности обслуживания и $\lambda = 1$ коэффициент загрузки составит 0.9, однако прогоны модели $M/D/1$ для $b >= 1$ будут бессмысленны из-за отсутствия стационарного режима вследствие перегрузки.

Комбинация аналитики и имитации эффективна и в случае *периодических* процессов обслуживания, методы аналитического расчета

которых отсутствуют. Чтобы встроить в датчик интервалов между заявками циклическое изменение интенсивности входящего потока, возможны (в зависимости от системы программирования) следующие варианты:

- объявить таймер модели в общем блоке;
- оформить датчик как внутреннюю процедуру или ее аналог (с назначаемым переходом);
- сделать таймер параметром датчика.

Ясно, что прямая работа с таким датчиком увеличит дисперсию времени ожидания. С другой стороны, можно прогнать модель для интенсивностей потока, соответствующих узлам квадратурной формулы (для составной формулы Симпсона — 5 узлов от минимума до максимума с шагом фазы $h = \pi/4$). Тогда

$$\bar{w} = \frac{1}{12}\sum_{i=1}^{5}(w_1 + 4w_2 + 2w_3 + 4w_4 + w_5)$$

(стандартный множитель перед суммой равен $h/3$, сумма делится на длину π интервала интегрирования по фазе). Опыт подтвердил хорошую точность даже при малом числе испытаний.

В рассматриваемой задаче в наибольшей степени влияют на интегральную сумму два последних слагаемых — из-за гиперболического роста ожидания по коэффициенту загрузки и коэффициента «4» при четвертом слагаемом. Они же обладают наибольшей дисперсией. Поэтому ограниченный общий лимит испытаний следует распределить между точками пропорционально произведениям квадратов весовых коэффициентов на *вторые* моменты длительности ожидания, определенные по данным пробных прогонов. Дополнительно необходимо учитывать два обстоятельства:

- уже упоминавшееся требование отсутствия перегрузки системы для всех узлов интегрирования;
- при перераспределении количества испытаний даже для их минимума отрезок стационарности должен значительно превосходить переходный.

Чтобы минимизировать число точек $\{y_i\}$, для которых выполняется трудоемкое моделирование, их следует выбирать в узлах квадратурных формул наивысшей степени точности (Гаусса или Лагерра — в соответствии с областью определения $f(y)$). Учитывая ограниченную точность вычисления $\hat{G}(y)$, порядок формул можно брать невысоким.

С математической точки зрения условная имитация эквивалентна расслоенным выборкам. Она является единственным средством получения достаточно точных оценок вероятностей редких событий (например, вероятностей отказа высоконадежных систем). Для решения такой задачи искомую вероятность записывают как сумму произведений условных вероятностей отказа или других неблагоприятных событий (предполагается, что эти вероятности существенно больше искомой). Затем вероятности звеньев цепочек определяются на серии «условных» моделей, а при возможности — аналитическими методами. Наконец, они подставляются в исходную формулу. В статье [31] сообщается, что применение этого подхода к сложной системе из 180 элементов уменьшило трудоемкость моделирования на два порядка.

В целом для успеха условной имитации требуется больше знаний о моделируемом процессе и методологии моделирования, а также умений ими воспользоваться.

Глава 9

Языки моделирования

Специфика имитационного моделирования и масштаб потребностей в нем очень рано поставили вопрос о средствах разработки моделирующих программ [27].

9.1 Моделирование на универсальных и специализированных языках

9.1.1 Универсальные языки программирования

Любой алгоритм моделирования может быть записан на универсальном процедурно ориентированном языке. Преимуществами подобной организации работы являются:

- возможность работы на практически любой ЭВМ;
- отсутствие необходимости в специальном программном обеспечении, его приобретении и изучении;
- «прозрачность» логики модели не только для разработчика, но и для широкого круга программистов;
- доступность встроенных функций, библиотечных подпрограмм, средств статистического анализа, редактирования вывода и других возможностей, предоставляемых штатными системами программирования широкого назначения;

- отсутствие ограничений на состав и логику модели, исключая налагаемые конфигурацией моделирующей вычислительной установки и режимом ее использования;
- намного более высокое быстродействие модели, включающей в себя лишь «необходимый и достаточный» набор операторов;
- возможность управления экспериментом и реализации методов уменьшения дисперсии результатов.

Разработка модели на универсальном языке подразумевает написание кода программы. Научившись моделированию, каждой детали которого нужно уделять внимание, можно лучше понимать действие моделей; соответственно уменьшится риск концептуальных ошибок при работе на специализированных языках.

Опыт программирования на языках типа *ПЛ/1* и *Фортран* показал, что составление имитационных моделей на них весьма трудоемко по следующим обстоятельствам:

- постоянно возрастает количество элементов и отслеживаемых процессов в моделируемых системах;
- имитационные модели сложны в отладке из-за богатой логики, взаимной зависимости ветвей алгоритма, большой трудоемкости прогонов и низкой точности результатов, способной в ряде случаев замаскировать ошибку;
- незначительные изменения моделируемой системы требуют внесения в программу многочисленных поправок и проведения отладки практически заново.
- достигнутая на уровне укрупненной блок-схемы наглядность модели теряется после составления программы. Это затрудняет участие проектировщика в отладке сложной программы и фактически требует от программиста почти такого же знания системы, как от ее разработчика.

9.1.2 Системы моделирования

Трудности моделирования на общецелевых языках программирования снимаются при использовании некоторых стандартных схем в качестве

типовых моделей. Такой язык приближает ЦВМ к разработчику и разработчика — к ЦВМ. Введенные в языке понятия задают рамки, гарантирующие возможность машинной реализации, но в то же время ограничивающие класс описываемых моделей. Возможность создания языков моделирования базируется на двух обстоятельствах:

1. Даже в сложных моделях с тысячами элементов количество *видов* действий остается весьма ограниченным. Такие действия естественно описывать как подпрограммы — при необходимости с дополнительными параметрами.

2. Имитационные модели достаточно широких классов реальных процессов имеют сходную логику, определяемую общей математической моделью.

В самом деле, при всем многообразии реальных ситуаций (работа суперЭВМ, комбината бытового обслуживания, проведение ремонтных и регламентных работ на технике, распространение эпидемии и борьба с ней, динамика популяции окуней, система снабжения авиабазы запчастями в период боевых действий и т. п.) бросается в глаза принципиальное сходство *техники* моделирования и ограниченность потребного комплекта изобразительных средств. Эти обстоятельства в условиях постоянно растущих потребностей в имитационных моделях привели к созданию *языков моделирования* на ЭЦВМ, которые существенно ускорили написание и в особенности отладку моделирующих программ. В общем случае моделирование обеспечивается входным языком, компилятором, системой интерпретации отдельных конструкций языка, библиотекой стандартных подпрограмм, банком данных (для проблемно ориентированных систем), средствами отладки и т. д. Поэтому правильнее пользоваться термином «система моделирования».

Языки моделирования позволяют писать имитационные программы в такой форме, которая напоминает описание моделируемой системы, и притом так, что малым изменениям описания системы соответствуют малые изменения в программе. Системы моделирования имеют базовые понятия, значительно более близкие к предметной области; они упрощают описание модели, позволяют работать с ней в диалоговом режиме; автоматически обнаруживают многие ошибки на этапе синтаксического анализа программы; зачастую включают в себя банки данных по предметным областям.

Преобразование состояний элементов системы обычно связано с выполнением арифметических операций над соответствующими переменными. Поэтому язык моделирования должен включать в себя базисные вычислительные средства (как правило, подмножество какого-либо алгоритмического языка). Специфически модельные средства должны обеспечить удобство описания реальных объектов, имитации событий, параллельного выполнения процессов и их взаимодействия. Наиболее характерной особенностью систем моделирования является наличие выделенной переменной — системного времени. Сервисные средства должны обеспечить удобный интерфейс, средства отладки, сбор и обработку статистического материала. Во всех языках моделирования дополнительно к используемым в *Алголе* типам данных имеется тип ref — ссылочный. Ссылка должна *квалифицироваться*, то есть сопровождаться указанием соответствующего класса.

Логика моделирования дискретных процессов в основном встроена в универсальный интерпретатор и при написании программы моделирования *подразумевается*. Естественно, что программа модели оказывается короче, чем на универсальном языке, поскольку в последнем случае моделирующая программа является синтезом описания модели и его интерпретатора. Компиляторы обеспечивают контроль синтаксической правильности программы. Это обеспечивает быстроту и правильность составления программ моделирования, в особенности при возможности интерактивной работы с моделью. Мощный отладчик позволяет:

- отслеживать отдельные объекты по всей модели, чтобы убедиться в правильности их обработки;
- проверять состояние модели при возникновении определенного события;
- после запланированного останова задать определенные значения переменных или атрибутов, чтобы заставить объект до конца продвинуться по маловероятному пути.

Современные системы моделирования снабжаются панелью инструментов с пиктограммами (блоками) для построения модели. Разработчик «мышью» выбирает нужные пиктограммы и помещает их в рабочую область. Затем он соединяет пиктограммы, чтобы обозначить *именованные потоки* в исследуемой системе. Двойным щелчком мышью по пиктограмме можно вывести диалоговое окно, где уточняются параметры

блока. К примеру, для устройства обслуживания это могут быть число каналов, тип и параметы распределений обслуживания, возможность отказа каналов и т. п. Блоки не должны быть ни слишком примитивными, ни чересчур изощренными. В первом случае понадобится слишком много типовых элементов, во втором — с ними будет трудно работать (слишком много параметров; тогда полезно сгруппировать их во вкладки).

Штатными возможностями современных систем моделирования стали также:

- повторное применение частей ранее разработанных моделей с возможностью их модификации, что повышает эффективность разработки;
- импорт данных из других приложений (баз данных) и экспорт их (например, в таблицы Excel);
- автоматическое моделирование процесса с запрограммированным изменением какого-либо параметра (например, числа каналов);
- средства построения графика для выбранного рабочего показателя как функции изменяемого параметра;
- инициализация состояния занятости (для ускорения вхождения в стационарный режим);
- средства самостоятельного освоения системы: руководство, справочник, контекстный Help, множество подробно описанных примеров, диски с демонстрационной версией (без средств запоминания программ и результатов, с количественными ограничениями);
- «инфраструктура» разработчика: информационные бюллетени и конференции пользователей, регулярные обновления версий и льготная подписка на них, экспресс-помощь зарегистрированным пользователям через Интернет.

Особого упоминания заслуживают *средства визуализации* процесса моделирования: анимация и динамическая графика. Пиктограммы модели изменяют свое положение, цвет и форму по мере развития процесса, отображая пространственное и угловое положение объектов, состояние исправности или поломки, занятости, наличного запаса, длины очереди и т. п. Анимация используется в следующих случаях:

- для представления сути модели руководству и новым сотрудникам;
- для отладки;
- для демонстрации правильности модели и последствий ошибок;
- для поиска лучших параметров;
- для обучения персонала.

Типовыми требованиями к анимации являются:

- возможность создания и хранения пиктограмм с высокой разрешающей способностью;
- наличие библиотеки стандартных пиктограмм;
- возможность их плавного движения;
- управление скоростью анимации;
- увеличение или уменьшение масштабов изображения;
- средства просмотра части процесса (прокрутка или иной способ ее выделения);
- векторная графика, облегчающая повороты и масштабирование;
- трехмерная анимация в обучающих системах;
- построение динамических графиков по ходу процесса.

Поскольку анимация сильно замедляет работу модели, должна быть возможность частичного или полного отключения ее.

По данным [98], стоимость различных средств имитационного моделирования колеблется от 500 до 50 тыс. долл. Необходимо учитывать также стоимость дополнительного оборудования, эксплуатации и программного обеспечения, операционную среду, программную совместимость.

Известно множество систем моделирования. В книге Е. Киндлера [27] упоминаются и в той или иной степени описываются свыше 350 различных систем. Более узкий обзор средств моделирования вычислительных систем приведен в книге И. Н. Альянаха [3]. По данным ACM

(Association for Computing Machinery) [74], *Симула* и *GPSS* входили в число 13 наиболее применяемых языков моделирования, в связи с чем ниже дается более подробная характеристика их. Заслуживают также быть упомянутыми созданный на базе *Фортрана* язык *Симскрипт*; алголоподобный язык *СОЛ*; интегрирующий возможности *GPSS* и *ПЛ/1* язык *ПЛИС*; *Паскаль Плюс*, расширенный встроенными средствами моделирования; диалоговая версия *GPSS* — система *СМДП* (ссылки на их описания см. в [?, 70, 65]). В книге Р. Шеннона [73] приводится схема алгоритма выбора языка моделирования в зависимости от сложности системы, цели моделирования, имеемых ресурсов, опыта программирования и т. п.

Ряд языков моделирования (*СЛЭНГ, Недис, Аргон* и др.) был создан в Советском Союзе. Особо отметим идею агрегативного моделирования и ее реализацию. Сложная система как правило состоит из разнородных элементов, описываемых различными математическими формализмами. Следовательно, нужно располагать набором математических моделей элементов и уметь описывать их взаимодействие. Универсальная модель элемента была предложена Н. П. Бусленко и И. Н. Коваленко в форме кусочно-линейного *агрегата* [23]. Координаты (признаки) агрегата делятся на целочисленные и вещественные. Целочисленные переменные между событиями остаются неизменными, а вещественные меняются линейно до тех пор, пока хотя бы одна из них не станет равной нулю. Этот момент и считается наступлением события. В такой момент сигналы с «выходных клемм» сработавшего агрегата мгновенно передаются на входные клеммы остальных агрегатов, изменяя координаты последних в соответствии со схемой взаимодействия. Подобными формализмами описывается весьма широкий класс систем: производственные, массового обслуживания, управления запасами, передачи данных, дорожного движения и т. д. Теория агрегативных систем интенсивно развивается. На ее базе во ВНИИ системных исследований была создана агрегативная имитационная система (АИС) — см. В. В. Калашников [23]. Помимо универсальных, существуют предметно-ориентированные пакеты ИМ для моделирования производственных систем, медицинских учреждений, центров выполнения заказов. (см. список в разд. 3.7 [98]).

«Размножение» языков моделирования не может быть терпимо ни с экономической, ни с технологической, ни с информационной точки зрения. Поэтому усиливается тенденция к созданию универсальных языков. На этом пути возможны две стратегии:

- «статический» язык с обширным набором средств, требующий формулировки прикладных задач в его терминах;
- «динамический» язык, допускающий создание на основе тщательно отобранного ядра специализированных *классов* для различных приложений.

Примером первого подхода является *GPSS*, которая в связи с ее массовым распространением и достигнутым в последних версиях радикальным улучшением обсуждается в главах 10–16. Второй подход реализован в *Симуле* 67.

Максимальное упрощение описания задач и подготовки исходных данных проще всего достигается специализацией систем моделирования. Чем у́же проблемная ориентация, тем меньше работы по программированию. В пределе можно добиться полной параметризации системы, и тогда задание конкретной модели будет состоять лишь в указании числовых значений параметров. Такая система будет иметь очень ограниченную применимость. Поэтому реальные специализированные системы допускают синтез произвольных структур из включенных в базу данных типовых элементов. Примерами могут служить системы *СЛЭНГ* и *NetCracker* моделирования компьютерных сетей.

9.2 СЛЭНГ

К специализированным системам относится *СЛЭНГ*, разработанная в Институте кибернетики Украины. Ориентация языка *СЛЭНГ* на вычислительные системы обеспечивается наличием в нем специальных средств описания устройств и памяти. Устройство можно рассматривать как ресурс, использование которого различными процессами разделено во времени (процессоры, селекторные каналы). Использование *памяти* совмещено во времени и разделено в пространстве.

Вычислительные средства *СЛЭНГ*'а суть подмножество *Алгола* 60. Программа модели включает в себя глобальные описания, оператор ввода и описания процессов. Последние оформляются аналогично процедуре в *Алголе* 60 с помощью описателя `process`. Активация процесса проводится через сообщение (аналог оператора вызова процедуры). На случай возможного прерывания с каждым сообщением связывается точка возврата, которая определяет очередной оператор описания

процесса, подлежащий выполнению при активизации последнего.

Возможности, предоставляемые в *Симуле* 67 (см. ниже) префикс-классом SIMSET, здесь реализованы объектами типа element (ссылка) и list (список). Средства описания процессов практически полностью заимствованы из *GPSS*. Описание устройств имеет вид

facility b, z[k]

Здесь b — одиночное устройство, а z — группа из k устройств. Устройство может быть занято или свободно. Для занятия устройства используется оператор захвата

seize <устройство>,<Prty>

Операнд <Prty> определяет приоритет сообщения — в зависимости от ситуации право прерывания или место в очереди. Например,

seize CPU, time-t

обеспечивает линейный рост приоритета в темпе системного времени. Для вычисления текущего приоритета применяется функция control. Освобождение устройства производится оператором release.

Теперь приведем пример описания *памяти*:

store s,128, drum[k], 2048

Здесь память с именем s имеет объем 128 единиц, а drum — k секций по 2048 единиц (страниц) каждая. Оператор входа задает требование на занятие некоторого объема памяти. Он записывается в форме

enter <память>,<объем>

Если требуемый объем имеется, то сообщение занимает этот объем и сохраняет активность, иначе ставится в очередь к памяти и переводится в режим ожидания. Оператор выхода leave записывается аналогично. Состояние сообщения, покинувшего память, не меняется.

Блок обработки статистики системы *СЛЭНГ* для каждого устройства без дополнительных указаний программиста обеспечивает подготовку и вычисление:

- количества обращений к устройству;
- коэффициента использования (отношения суммарного времени использования к общему времени моделирования);
- среднего времени занятости в расчете на одно обращение;
- количества прерываний.

Для блоков памяти вычисляется их средняя занятость в абсолютном и в процентном исчислении.

Имеются и другие системы моделирования вычислительных систем — см. [40]. Различные варианты их классификации приведены в книгах [27] и [64, с. 129].

9.3 Универсальная система Симула 67

В условиях дифференциации научных исследований каждая отрасль знаний имеет свой круг понятий и методов. Естественно желание исследователя формулировать задачу для ЭЦВМ в привычных терминах, то-есть опираться на языки динамического типа и по мере расширения специальных знаний и накопления опыта моделирования развивать *средства* языков. Именно такова *Симула* 67, вместе со своей предшественницей *Симула* 1 сыгравшая заметную роль в развитии базовых концепций программирования. Более того, она может служить основой построения различных математических и естественно-научных теорий — от геометрии и алгебры до химической технологии и сельского хозяйства [27, с. 237].

9.3.1 Базисные средства

Симула 67 — универсальная система программирования [18], в которой синтезировались и получили дальнейшее развитие основные понятия языка *Алгол* 60, входящего в *Симулу* 67 как подмножество, и языка моделирования *Симула* 1. В дополнение к средствам *Алгола Симула* 67 содержит удобный аппарат описания новых понятий, средства обработки текстовой информации, стандартизованный аппарат ввода-вывода, средства квазипараллельного исполнения компонент программы.

Симула 67 реализована на БЭСМ-6, машинах ЕС ЭВМ и IBM PC. Имеется опыт моделирования на этом языке вычислительных систем, сетей ЭВМ, бортовых систем управления с БЦВМ, робототехнических систем, систем массового обслуживания, систем искуственного интеллекта [4].

Создание сложных программных комплексов возможно лишь при расчленении задачи на относительно простые части, независимом программировании таковых и организации связей между ними по информации и управлению. Такие части в *Алголе* 60 описываются *блоками*.

В частности, блоком является тело процедуры. При каждом исполнении блока порождается его динамический экземпляр, что позволяет трактовать его вызовы как совокупность динамических объектов, следующих заданному описанием образцу.

Объекты *Симула*-программы принадлежит к одному или нескольким множествам родственных объектов, называемым *классами*. Концепция объектно ориентированного программирования, впервые выдвинутая именно в *Симуле* 67, впоследствии была реализована в языках C^{++}, *Паскаль*, *Smalltalk* 80, *Фортран* 90 и может быть применена при написании на этих языках программ имитационных моделей.

Термином *объект* в *Симуле* 67 обозначаются программные компоненты, обладающие собственными локальными данными (атрибутами) и способные выполнять определенные действия. В роли атрибутов могут выступать переменные, массивы, ссылки на другие объекты, процедуры. Действия объекта, записанные в виде операторов *Симулы* 67, образуют сценарий его поведения. Таким образом, понятие объекта по смыслу близко понятию блока. Напомним, что выполнение оператора процедуры семантически эквивалентно подстановке на его место блока — тела процедуры с заменой формальных параметров фактическими.

Каждый объект в динамике получает системный атрибут, указывающий выполняемый пункт его сценария. При исполнении модели в ней могут сосуществовать несколько объектов, из которых в любой момент времени активен только один. Переключение активностей соответствует событию в системе. События считаются происходящими мгновенно. Допускаются одновременные события.

Симула 67 допускает описание сложных объектов древовидной иерархией деклараций классов, что позволяет поэтапно детализировать моделируемые системы в ходе их исследования или проектирования. При этом свойства класса-префикса (в частности, его формальные параметры) присоединяются к свойствам объектов вновь вводимых классов. Предусмотрены средства независимой компиляции описаний классов для последующего их использования при минимальной затрате машинного времени. Классы описываются наподобие процедур в *Алголе* 60 и взаимодействуют посредством ссылок (указателей), дистанционного обращения, квазипараллельного исполнения, сочленения.

Декларация класса определяет образец программы, то есть ее данные и действия. Если в декларации класса не указаны никакие действия,

она просто описывает структуру данных. Тело класса всегда действует как блок. Объекты одного класса имеют общую структуру атрибутов и одинаковые правила действия, но могут различаться конкретными значениями атрибутов. Они создаются генератором объекта вида

```
new <имя_класса> (<список фактических параметров>)
```

в соответствии с декларацией класса. *Ядро* декларации класса включает в себя:

- конструкцию class <имя класса>;
- совокупность формальных параметров;
- список значений;
- спецификацию параметров;
- виртуальные спецификации;
- тело класса (в общем случае составной оператор).

В *Симуле* 67 ссылка может иметь значение none (нет объекта), переменной, указателя ссылочной функции или генератора объектов:

new <имя класса> <совокупность фактических параметров>

Присваивание ссылочного значения записывается символом «:-» (означает). Каждая объявленная в блоке переменная при входе в этот блок в зависимости от своего типа получает начальное значение: real или integer — нуль, boolean — false, ref — none.

Процессы, состоящие из аналогичных активностей, могут быть отнесены к одному классу. В любой момент в системе может протекать несколько таких процессов, находящихся в различных фазах выполнения. Каждый процесс является элементом своего класса и идентифицируется в нем набором атрибутов. Поведение процессов одного класса может быть описано единым набором правил.

9.3.2 Наборы

Процессы «хранятся» в очередях, картотеках, списках и т. п., объединяемых общим понятием «набор». Все ссылки на процессы организуются

стандартным (косвенным) образом. Наборы строятся циклически в соответствии со схемой рис. 2.4. Такая организация позволяет имитировать работу очередей и стеков без физического перемещения соответствующих записей, что существенно экономит время моделирования. Голова набора — особый элемент, ограничивающий набор с обеих сторон. Она заводится при объявлении набора и может быть уничтожена лишь при ликвидации его. Отсутствие процесса обозначается ключевым словом none.

В языке *Симула* 67 имеется специальный *системный* класс SIMSET, который не нуждается в объявлении и содержит средства обработки списковых структур. Процедуры suc и pred позволяют, например, получить ссылки на преемника и предшественника некоторого элемента списка; out удаляет объект из набора, членом которого он является. Процедуры follow и precede включают некоторый набор непосредственно вслед за или перед тем объектом, на который указывает параметр. Процедуру first можно использовать для получения ссылки на первый член набора, а last — на последний. Процедура cardinal подсчитывает количество членов набора.

9.3.3 Средства моделирования

Средства универсального языка *Симула* 67, непосредственно ориентированные на имитационное моделирование дискретных систем, вводятся с помощью системного класса

SIMSET class SIMULATION

Поскольку префиксом этого класса является SIMSET, в классе SIMULATION разрешено пользоваться всеми возможностями SIMSET. Объединенные возможности этих классов предоставляются любой программе с префиксом SIMULATION. В этом классе определены «ось времени», двунаправленные списки для представления очередей и подкласс process. Ядро класса SIMULATION составляет интерпретирующая программа моделирования дискретной системы по схеме «от события к событию». Основным элементом ее является управляющий список, содержащий упорядоченный по системному времени набор событий (моменты и ссылки на процессы). При исполнении своей активной фазы процесс может:

- запланировать событие для других процессов;
- отменить или обусловить часть ранее запланированных событий;

- запланировать для себя самого следующую активную фазу, и т. п.

Пользователь, желающий написать программу моделирования, начинает ее словами

SIMULATION begin ...

Если же он хочет создать специализированный язык моделирования, например, для моделирования систем с кольцевым обслуживанием, то может написать

SIMULATION class ring begin ...

Особую роль в *Симуле* 67 играют выражения ссылочного типа. Объектное выражение имеет тип ref и задает правило получения ссылки. Простыми объектными выражениями считаются:

- none;
- переменная ссылочного типа;
- генератор объектов;
- указатель функции;
- локальный объект;
- переменная ссылочного типа;
- квалифицированный объект.

Генератор объектов описывается как

new <имя класса><совокупность фактических параметров>

Он вызывает порождение и исполнение некоторого объекта, принадлежащего указанному классу и имеющего заданные фактические параметры. Локальный объект имеет вид this <имя класса>. Наконец, квалифицированный объект есть простое объектное выражение, за которым после ключевого слова qua следует имя класса.

Специальные операторы is и in служат для проверки принадлежности первого операнда к указанному вторым операндом классу (первый оператор — непосредственное вхождение, второй — через подкласс).

Между проходящими в системе процессами и их отдельными фазами (активностями) должны быть установлены необходимые связи. Все классы таких процессов представлены в *управляющем списке* (УС), элементы которого указывают:

- имя соответствующего процесса и его фактические параметры;
- системное время ближайшей активной фазы процесса (сообщения УС упорядочены по возрастанию этого показателя);
- соответствующую этой фазе точку входа.

Сообщение, находящееся в начале УС, может быть вызвано через процедуру current (текущий). Временнóй параметр этого сообщения интерпретируется как текущее системное время.

При завершении любого события в системе сообщение переходит из активной фазы в какую-либо другую и происходит обращение к процедуре choice для выбора очередного сообщения, подлежащего активизации, и других изменений в очередях сообщений (например, перевода из режима ожидания в режим готовности). Поиск и перемещение сообщений осуществляются с помощью ссылок — без фактического перемещения заявок из одной очереди в другую. Для покидаемого процесса (исключая выход по оператору перехода или через замыкающий символ end) определяется точка возврата к его следующей активной фазе.

В классе предусмотрены также средства перекрестной связи между процессами по информации. Возможны так называемые «групповые» обращения типа

inspect after when demand do begin...end

Здесь предполагается просмотреть все содержимое объекта after и для относящихся к demand выполнить составной оператор.

Некоторый процесс, *присоединив* другой процесс, может получить доступ к признакам последнего. Присоединяющий оператор имеет общий вид

```
inspect  when       A1 do S1
         when       A2 do S2
         ........   ........
         when       An do Sn
         otherwise  S;
```

Здесь A1, A2, ...,An — имена классов, а S1, S2, ..., Sn, S — операторы. Хотя бы одна конструкция присоединения when A1 do S1 обязательна, otherwise может отсутствовать. Если X есть процесс, принадлежащий классу Ak, то выполняется оператор Sk, а остальные операторы игнорируются.

Оператор `terminate(X)` завершает процесс под названием `X`. Относящееся к этому процессу уведомление о событии, если таковое имеется, исключается из управляющего списка. Точка возврата процесса уничтожается.

Оператор `cancel(X)` делает процесс пассивным, исключая его из УС, но сохраняя точку возврата. Оператор `activate X` <u>`at`</u> `T` планирует активную фазу пассивного процесса на момент `T` системного времени. Имеется возможность активировать процесс с *задержкой* (<u>`delay`</u> `T`) относительно текущего системного времени или немедленно (перед текущим процессом).

В языке имеется ряд встроенных функций для получения случайных объектов по предварительно сформированному $U \in [0,1)$:

1. <u>`Boolean procedure`</u> `draw(a,U)` — доставляет <u>`true`</u> с вероятностью a и <u>`false`</u> с вероятностью $1-a$;

2. <u>`real procedure`</u> `uniform(a,b,U)` — значение, равномерно распределенное на $[a,b)$;

3. <u>`real procedure`</u> `normal(a,b,U)` — нормальное со средним a и среднеквадратическим отклонением b;

4. <u>`real procedure`</u> `negexp(a,U)` — экспоненциальное со средним $1/a$;

5. <u>`real procedure`</u> `Poisson(a,U)` — распределенное по Пуассону со средним a;

6. <u>`real procedure`</u> `Erlang(a,b,U)` — распределенное по Эрлангу со средним $1/a$ и среднеквадратическим отклонением $1/(a\sqrt{b})$.

Для описания правил функционирования процессов язык предоставляет средства, соответствующие современным алгоритмическим языкам высокого уровня. Это дает возможность адекватно определять объекты со сложными правилами функционирования. В реализациях *Симулы* 67 предусмотрен интерфейс с другими системами программирования, что позволяет использовать накопленные в них подпрограммы.

Наконец, в *Симуле* предусмотрены средства статистической обработки результатов моделирования. В частности, определена процедура

histo(A,B,c,d), которая обновляет гистограмму, определенную одномерными массивами A и B в соответствии с наблюденным значением c, имеющим вес d. При этом A[i] увеличивается на d (здесь i — наименьшее целое, для которого B[i]>c).

Оператор accum(A,B,C,D) можно использовать для накопления интеграла по системному времени от ступенчатой функции C последнего. Приведем описание этой процедуры:

```
procedure accum(a,b,c,d);
    name a,b,c;
    real a,b,c,d;
    begin
        a:=a+cx(time-b);
        b:=time; c:=c+d;
end accum;
```

Здесь b — момент последнего изменения состояния, time — текущее системное время, d — приращение функции.

9.3.4 Оценка языка

Внимательный анализ методики описания модели на языке *Симула* 67 позволяет обнаружить ряд аналогий с *GPSS*:

1. Определение объекта функционально совпадает с сегментом *GPSS*-модели,
   задающим типовую траекторию транзакта.

2. Оператор new эквивалентен блоку GENERATE.

3. Управляющий список *Симулы* соответствует совокупности цепей событий *GPSS*, и т. д.

Разумеется, это сходство не случайно: оно объясняется общностью целевого назначения и в определенной степени — преемственностью. Однако *Симула* 67 сочетает в себе достоинства универсальной и специализированной систем и потому свободна от большинства недостатков *GPSS*.

Опыт моделирования на *Симуле* 67 [80] вынуждает, однако, сделать вывод о том, что прекрасные базисные идеи в этой системе программирования реализованы не лучшим образом :

- руководства составлены невразумительно;

- отсутствует раздельная компиляция процедур;
- трудно организовать нестандартный сбор статистики;
- ограничены возможности управления формой вывода.

9.4 Новые зарубежные пакеты

Обзор наиболее популярных пакетов имитационного моделирования ниже приводится по [98] с акцентом на наиболее интересные особенности.

В главе 3 относительно подробно описаны пакеты Arena и Extend. Для пакета Arena даны общая характеристика, примеры диалоговых окон. В этом пакете для типовых фрагментов модели имеются шаблоны (трафареты) с комплектами модулей (аналогично известной графической системе Visio). Допускается иерархическое построение модели методом пошаговой детализации. Пакет позволяет строить доверительные интервалов для разностей показателей двух систем и проводить функционально-стоимостный анализ по ABC-схеме (разбивке объектов на три группы по их относительной важности — в задачах управления запасами по стоимостному спросу). Профессиональная версия позволяет создавать настраиваемые модули и сохранять их в качестве шаблонов.

Пакет Extend (из самых недорогих) включает внутренний язык ModL для настройки существующих блоков и создания новых. При этом возможна иерархия блоков с наследованием потомками основных свойств предков. Библиотека Manufacturing содержит конструкции для моделирования погрузо-разгрузочных устройств (конвейеры, погрузчики, краны и т. п.). В этом пакете также реализована схема ABC.

По-видимому, к Extend идейно близок объектно ориентированный пакет MODSYM III. О последнем сообщается, что он соответствует стандарту оборонных ведомств США по взаимодействию имитационных моделей.

На с. 276 [98] дан список других универсальных пакетов, а на с. 284 — перечень пакетов, специализированных по приложениям: производственные системы, сети связи, процессы модернизации и обслуживания, медицина, центры обработки заказов, автономная анимация (по данным, выводимым в процессе имитации). Наконец, на с. 762 приводится краткая характеристика пакетов AutoMod, ProModel, Taylor Enterprize Dynamics, WITNESS.

Интересно отметить, что авторы [98] моделировали простую производственную систему с помощью шести различных имитационных пакетов и обнаружили разницу во времени прогона в 11 раз (с. 257).

9.5 Моделирование и математические пакеты

Лица, заинтересованные в имитационном моделировании, часто пытаются искать соответствующие инструменты в популярных пакетах математических программ: *Mathematica, Maple, Mathcad, MatLab, Derive, Scientific WorkPlace* [60] (все нижесказанное относится и к самым современным их версиям, что позволяет для краткости опустить номера последних).

Отдельные инструменты имитационного моделирования там действительно имеются. Почти все перечисленные пакеты включают в себя статистические разделы, предоставляющие средства:

- генерации случайных чисел с известными типами распределений (до 20 типов) и заданнными их параметрами,
- выравнивания распределений (подбора параметров по статистическим данным),
- расчета моментов,
- определения квантилей (то есть необходимой для генерации случайных чисел обратной функции распределения),
- проверки гипотез о свойствах выборки, и т. д.

Арсенал подобных средств еще обширнее в специально статистических пакетах типа *Statistica, Statgraphics, SPSS, Stadia* [68]. Однако во всех перечисленных пакетах средства *программирования*, необходимые для описания сложной логики моделирования, если и достаточны в принципе, то весьма неудобны в работе. Поэтому гораздо более естественно составить модель на входном языке системы программирования с достаточно содержательными библиотеками подпрограмм. В частности, такой системой является *Фортран PowerStation*, пакеты IMSL и

`Numerical Recipes` из которой (см. [59]) удовлетворят даже профессионала.

Особого разговора заслуживает $MatLab$. Среди перечисленных пакетов он выделяется большим количеством тулбоксов — «примочек», позволяющих решать некоторые дополнительные классы общематематических проблем и широкий круг задач теории автоматического управления. С учетом тулбоксов $MatLab$ можно определить как инструмент *математического* моделирования технических систем — аналитического или численного и лишь в минимальной степени (при описании задач типа конечных автоматов или сетей Петри) — имитационного. Избыточные надежды на него автора [17] порождаются не соответствующим контексту восприятием часто встречающегося в документации и литературе по пакету термина `Simulation`, и наличие тулбоксов `Simulink` и `Stateflow` положения не меняет.

Глава 10

Введение в GPSS

10.1 Общая характеристика

10.1.1 Рождение системы

Ни один из языков моделирования не оказал на имитацию столь большого воздействия, как *GPSS* (General Purpose Simulation System — система имитационного моделирования общего назначения) [64, 65, 70, 74, 89]. Впервые созданная Джеффри Гордоном в фирме IBM в начале 1960-х гг., она является самым популярным в мире инструментом имитации и наряду с *Фортраном* и системой TEX являет редчайший пример долгожительства в мире программного обеспечения. Она глубоко повлияла на ряд других языков моделирования — см., например, обзор языка *СЛЭНГ* (разд. 9.2), сходство которого с *GPSS* простирается вплоть до заимствования большинства ключевых слов.

В книге Гордона [89] очень четко изложены основные понятия системы моделирования. В исключительно содержательной, но местами утомительно подробной книге Дж. Шрайбера [74] рассмотрена история создания *GPSS* и множество ранних версий для разных платформ. Тон всей книге и характер обсуждаемых приложений задает ее первая глава — «Вопросы моделирования систем массового обслуживания». Создание моделей проиллюстрировано на 27 примерах; предложено свыше 300 задач с решениями либо указаниями. Дополнительно можно рекомендовать полное и удобочитаемое руководство [79] по *GPSS/PC*, а также учебные пособия Б.Я. Советова и С.А. Яковлева [64, 65], в которых введение в *GPSS* излагается как часть дисциплины «Моделирование систем».

10.1.2 Базовые понятия

Модель *GPSS* состоит из сети *блоков*, представляющих необходимые действия или задержки *транзактов*, которые последовательно проходят через блоки. Например, блок `GENERATE` вводит в модель новые транзакты, воспроизводя *рекуррентный* поток заявок с требуемым распределением интервалов между смежными заявками. Транзакты могут представлять собой клиентов, покупателей, ремонтируемые телевизоры, телефонные звонки, электронные сигналы и т. п. В близком *GPSS* языке *DRUGSIM* транзактами являются наркоманы, получающие разнообразное обслуживание в медицинских и юридических органах.

Транзакты перемещаются в системных времени и пространстве, переходя от одного блока модели к другому и воздействуя на них. Транзакты возникают и уничтожаются, могут расщепляться и сливаться. Входя в блок, транзакт вызывает определяемую типом блока подпрограмму, которая обрабатывает соответствующее событие. Далее транзакт в общем случае пытается войти в следующий блок. Продвижение продолжается до тех пор, пока не окажется, что очередной блок должен выполнить одну из следующих функций:

- удалить транзакт из модели,
- временно заблокировать его в предыдущем блоке до выполнения некоторых
  условий,
- задержать его на определяемое моделью время.

Тогда начинается продвижение другого транзакта и т. д. — до завершения моделирования. За один шаг работы управляющей программы производится поочередный просмотр всех имеющихся процессов и имитация тех, которые могут быть запущены. При этом может освободиться ранее занятый ресурс или оказаться порожденным процесс более высокого приоритета. Тогда просмотр списка процессов можно начать сначала.

Описание траектории транзакта содержит порядок и имена используемых им «устройств» (приборов обслуживания) или «памятей»; временны́е задержки; логические условия, управляющие продвижением транзактов; точки маршрута, в которых производится сбор данных б ожидании или контролируется время прохождения, и т. п. Некоторые

из названных объектов должны быть предварительно описаны специальными командами. Параметры транзакта могут менять свои значения на траектории, оказывать влияние на маршрут и собираемую статистику.

Каждая из названных функций имеет специальное графическое изображение — см. главу 13[1]. Латинские буквы означают последовательные операнды блоков. Многие блоки являются парными (сопряженными). Соответственно их графические представления суть взаимные зеркальные отражения, что облегчает их запоминание и интерпретацию. Из этих блоков составляется схема $GPSS$-модели, описывающая траекторию транзактов (рис. 10.1). Разработчик модели должен так подобрать последовательность блоков и команд, чтобы поведение транзактов соответствовало работе реальной или предлагаемой системы.

Блоки и транзакты — это только два из базовых понятий (сущностей) $GPSS$ — см. главу 11. Обычно, когда транзакт входит в блок, некоторая операция выполняется над третьей сущностью. Например, при входе транзакта в блок `SEIZE` он «завладевает» устройством (`Facility`), имитирующим одноканальное устройство обслуживания. Интерпретация сущностей, транзактов, параметров и т. п., выбор единицы времени — дело разработчика. Все это надо фиксировать в словаре модели.

Траектория и задержки могут зависеть от ситуации в системе и от параметров транзакта. Маршруты следования транзактов, временны́е характеристики, условие прекращения моделирования, требуемые показатели эффективности задаются пользователем.

При описании этих процессов и сборе итоговой статистики используются таймер модельного времени, стандартные числовые атрибуты (СЧА) и параметры транзактов, а также определяемые пользователем переменные, выражения и таблично задаваемые функции. Наиболее сложные аспекты модели — планирование предстоящих событий и очередности их обработки — автоматически реализует планировщик событий (интерпретатор) $GPSS$.

[1] Эти обозначения заимствованы из книги Гордона [89], а для новых блоков построены по аналогии.

Рис. 10.1. Схема GPSS-модели

Запись траекторий транзактов напоминает запись процедур в традиционных языках программирования: блок `GENERATE` аналогичен заголовку процедуры, его операнды соответствуют описанию локальных переменных, `TERMINATE` — закрывающему описание ключевому слову `END`, остальное — телу процедуры. Вход транзакта в систему соответствует вызову процедуры моделирования истории транзакта. Имеется и существенная разница: в обычных языках программирования не допускается совмещение во времени нескольких выполнений одной процедуры (исключая рекурсию).

10.1.3 GPSS World

Система *GPSS* прошла длинный путь развития. До недавних пор ее главными недостатками были:

- низкий уровень входного языка,
- невозможность реализации нестандартных вычислений,
- скрытость от программиста внутренней логики поведения модели.

Они резко ограничивали полезность системы. Радикальные изменения произошли одновременно с переходом к *Windows*-версиям. Ниже описана «студенческая» версия *GPSS World* 4.2.1 (2000 г.), далее сокращенно именуемая *GPSS/W*. Она бесплатно распространяется через Интернет (электронный адрес `www.minutemansoftware.com`) и отличается от коммерческой лишь количественными ограничениями (до 150 блоков в составе модели). Только коммерческая версия обеспечивает совместимость с *GPSS/W* программ, ранее составленных на *GPSS/PC*.

Принципиальные особенности *GPSS/W* можно сгруппировать в несколько разделов.

1. Каждая разработка в духе современных технологий программирования рассматривается как *проект*[2]. Проект состоит из объектов четырех типов: исходный модуль, Имитация (продукт компиляции), отчеты и текстовые файлы. Модель может иметь иерархическое строение и тогда включает в себя командой `INCLUDE` ранее разработанные фрагменты. В рамках проекта может существовать множество автоматически

[2]Эта терминология в электронной документации к системе не используется.

нумеруемых версий. Размеры задачи не лимитируются физически доступным объемом оперативной памяти.

Для каждого из шагов проекта есть свои *инструменты*: полноэкранный текстовый редактор, пункт `Create Simulation` в меню `Command`, множество окон контроля и команд управления моделированием, автоматический генератор отчетов, средства настройки отчетов, статистического анализа. Команда `CONDUCT`, начиная *Эксперимент*, берет на себя управление Имитацией.

В системе реализованы элементы объектного программирования в классическом его понимании: *наследование свойств* (установок) Имитации от исходной модели и *полиморфизм* (автоматическое преобразование операндов выражений — в частности, строки символов в число и обратно — к требуемому типу).

2. Интерфейс ориентирован на пользователя. Однострочный текстовый редактор для набора модели заменен на полноэкранный с реализацией стандартных функций *Windows* — табуляция, выделение текста, работа с буфером `Clipboard`, управление типом и размером шрифтов `True Type`, операции с файлами. Есть возможность замены традиционно используемого вместо знака умножения # на привычную *. Компилятор формирует *список* синтаксических ошибок, обеспечивается автоматический подвод курсора к месту очередной ошибки и вывод диагностики в статусную строку.

Динамика модели может отслеживаться как в числовой, так и в графической форме с помощью окон около 20 разных типов (в зависимости от класса отображаемых сущностей), причем в двух режимах: детальном и обзорном. Размер и положение окон изменяются стандартными средствами. Степень загруженности ресурсов показывается цветом, можно наблюдать за перемещением активного транзакта. В окна можно выводить СЧА и накопленную статистику. Отображение внизу главного окна значения *таймера* облегчает диагностику зависания. В особо сложных случаях можно получить `Snapshots` — снимки мгновенного состояния *цепей событий*, используемые для отладки логики модели. Имеются средства интерактивного ввода команд, контрольных остановов и трассировки. Результаты моделирования могут быть автоматически представлены в виде гистограмм соответствующих распределений.

3. Резко расширены вычислительные возможности. Отменены обязательная целочисленность операндов и принудительное округление результатов. Максимальное число измерений матрицы увеличено до 6.

Используемый при инициализации матриц атрибут UNSPECIFIED предупреждает работу с «неприсвоенными значениями». Язык *GPSS* дополнен языком программирования расчетов *PLUS*, близким к традиционным алгоритмическим. На нем можно записывать как выражения для операндов блоков, так и процедуры пользователя, в том числе содержащие циклы и разветвления. Это практически сняло ограничения на формируемые результаты и способы их обработки. Введены средства непрерывного моделирования (численного интегрирования по системному времени). Перечень встроенных математических функций дополнен функциями генерации случайных чисел для 24 законов распределения, что избавляет пользователя от мучительного построения таблиц обратной функции распределения для каждой комбинации параметров. Операции над строками позволяют редактировать стандартный отчет и применять процедуры с текстовыми аргументами. Появились операции с потоками данных (открытие и закрытие файлов, запись и считывание информации из них).

4. Система берет на себя стандартную статистическую обработку результатов моделирования. Появившаяся в *GPSS/PC* команда ANOVA однофакторного дисперсионного анализа заменена многофакторным эквивалентом. Имеются средства автоматической разработки, проведения и разработки статистических экспериментов (включая дробные факторные) по регрессионному анализу и оптимизации моделей.

10.2 Пример программы

Ниже представлен пример *GPSS*-программы моделирования простейшей одноканальной системы массового обслуживания — с показательно распределенными интервалами распределения интервалов между смежными заявками и длительностей обслуживания.

```
*************************************
*            M/M/1 MODEL            *
*************************************

;           Основной сегмент

1      GENERATE   (Exponential(1,0,1))   ; Генерация заявок
2      QUEUE      Wait                   ; Начало ожидания
3      SEIZE      Chan                   ; Попытка занятия канала
```

```
4       DEPART     Wait                    ; Конец ожидания
5       ADVANCE    (Exponential(2,0,0.9)) ; Задержка на время обслуж.
6       RELEASE    Chan                    ; Освобождение канала
7       TERMINATE                          ; Конец истории транзакта

;          Сегмент таймера

8       GENERATE   50000,,,1               ; Момент окончания
9       TERMINATE 1                        ; Конец моделирования
****************************************************************
```

Комментарии в *GPSS*-программе могут открываться звездочкой в начале строки или продолжать текст оператора (после точки с запятой). Здесь комментарии даны по-русски исключительно для первого знакомства с системой: *программа, содержащая кириллические символы (в том числе в комментариях и в выводимых строках), работать не будет.*

Модель состоит из двух сегментов. В каждом сегменте модели имеется отдельная пара блоков GENERATE — TERMINATE, но только один из TERMINATE может задаваться с операндом, вычитаемым из счетчика завершений. Начальное значение последнего загружается «с пульта» командой START — в данном случае с операндом 1. При обнулении счетчика моделирование прекращается.

Первый сегмент описывает «биографию» направляемой на обслуживание заявки. Во втором генерируется единственный (см. четвертый операнд GENERATE) транзакт, который обеспечивает завершение моделирования при значении таймера модели 50000 (первый операнд). Отдельный сегмент таймера убыстряет работу модели, так как не требует проверки числа обслуженных заявок для каждого транзакта. Кроме того, этот вариант позволяет при необходимости организовать *заключительную* обработку, выполняемую однократно после завершения сбора статистики и программируемую непосредственно перед TERMINATE 1.

Теперь обсудим первый сегмент. В нем GENERATE формирует неограниченное число заявок через экспоненциально распределенные моменты времени, доставляемые встроенной функцией EXPONENTIAL. Это новинка *GPSS/W* — раньше приходилось задавать таблицу обратной ДФР для каждого используемого закона. Первый операнд функции указывает номер используемого датчика равномерно распределенных псевдослучайных чисел (1), второй — смещение экспоненты (нулевое), третий — ее

среднее значение (единица).

Вошедшая заявка отмечается в очереди QUEUE по имени Wait и пытается захватить (SEIZE) устройство обслуживания по имени Chan. Если устройство занято, транзакт помещается в связанную с ним цепь задержки, которая просматривается в моменты освобождения устройства.

Если устройство свободно, то блок DEPART фиксирует конец пребывания в очереди Wait. Соответственно накапливается статистика ожидания, связанная с этой очередью. Отметим, что блоки QUEUE и DEPART определяют только необходимость *сбора статистики* по данной очереди: их отсутствие не влияет на реальное существование и динамику очередей.

Захватившая устройство заявка задерживается в нем (блок ADVANCE) на интервал с показательно распределенной длительностью и средней задержкой 0.9. Для его формирования используется другой (второй) ДСЧ. Таким образом, коэффициент загрузки системы $\rho = \lambda b/n = 1.0*0.9/1 = 0.9$. По истечении задержки блок RELEASE освобождает канал Chan, и траектория транзакта заканчивается *без воздействия на счетчик завершений*.

Обращают на себя внимание исключительная простота и логичность обсуждаемого текста в сравнении с программой модели на алгоритмическом языке общематематического назначения (по крайней мере в этом элементарном случае). Отметим, однако, необходимость знания многих технических деталей — в первую очередь связанных с позицией и смыслом аргументов блоков.

По результатам прогона программы автоматически формируется стандартный отчет:

```
GPSS World Simulation Report - Basic3.62.1
   Saturday, September 08, 2001 19:08:19

     START TIME       END TIME  BLOCKS FACILITIES  STORAGES
       0.000          50000.000    9        1         0

          NAME                          VALUE
          CHAN                        10001.000
          WAIT                        10000.000

 LABEL  LOC BLOCK TYPE    ENTRY COUNT   CURRENT COUNT RETRY
         1  GENERATE       49843              0         0
         2  QUEUE          49843              0         0
```

```
        3  SEIZE          49843              0         0
        4  DEPART         49843              0         0
        5  ADVANCE        49843              1         0
        6  RELEASE        49842              0         0
        7  TERMINATE      49842              0         0
        8  GENERATE           1              0         0
        9  TERMINATE          1              0         0

FACIL ENTRIES UTIL.  AV.TIME AVAIL. OWNER PEND INTER RETRY DELAY
CHAN   49843  0.896   0.899    1    49844  0    0     0      0

QUEUE  MAX CONT. ENTRY ENTRY(0) AV.CONT. AV.TIME AV.(-0) RETRY
 WAIT    45  0   49843  5340     5.132    5.148   5.766    0

FEC XN   PRI     BDT     ASSEM  CURRENT NEXT PARAMETER   VALUE
 49844    0   50001.068  49844     5     6
 49845    0   50001.720  49845     0     1
```

После заголовка отчета в нем указываются системное время начала и окончания моделирования и состав системы. Далее следует таблица внутренних адресов, поставленных в соответствие меткам. Затем печатается таблица с количеством входов в блоки, текущим числом транзактов в них и числом повторных попыток. Две последующих выдачи содержат информацию об устройствах и очередях. Отчет не дает раздельной информации по транзактам разных приоритетных классов.

Содержание отчета зависит от состава модели и при необходимости корректируется через `Установки модели`. Нестандартные вычисления могут быть запрограммированы с помощью процедур встроеннного языка программирования и выведены в файл. Более подробно содержание отчета будет обсуждаться в разд. 14.6.

Общим правилом работы с новым программным обеспечением математического характера является его тестирование на задачах с известным решением (предпочтительно аналитическим). Для системы $M/M/1$ теоретическая средняя длина очереди $q = \rho^2/(1-\rho) = 0.9^2/(1-0.9) = 8.1$. Среднее время ожидания w в соответствии с формулой Литтла (закон сохранения стационарной очереди) должно быть равно q/λ и в данном случае ($\lambda = 1$) численно совпадать со средней длиной очереди. Теоретическая вероятность незанятости системы равна $1 - \rho = 0.1$. Следовательно, в среднем десятая часть заявок должна получать обслуживание без ожидания. Сопоставим с этими ожиданиями результаты моделирования — табл. 10.1.

Таблица 10.1. Тестирование GPSS/W на модели $M/M/1$

Показатель	Теория	Число испытаний		
		50 000	200 000	500 000
Коэффициент загрузки	0.900	0.896	0.897	0.899
Средн. время обслуживания	0.900	0.899	0.902	0.901
Число входов		49843	199042	499032
Из них с нулевым ожиданием		5340	21001	50691
Средняя длина очереди	8.100	5.132	5.757	6.690
Среднее время ожидания	8.100	5.148	5.785	6.703

В составлении приведенной программы было трудно ошибиться (к тому же, она отличается от обсуждаемой у Т. Шрайбера [74] на с. 57 и далее только типом исходных распределений). Формула Литтла и доля заявок, принимаемых к обслуживанию без ожидания, подтверждаются с достаточно высокой точностью. Это дает основания доверять интерпретатору $GPSS/W$. Показатели, определяемые только *средними* значениями моделируемых первичных величин (среднее время облуживания и коэффициент загрузки), также вполне приемлемы.

Время ожидания зависит уже от двух моментов распределения времени обслуживания:

$$w = \frac{\lambda b_2}{2(1-\rho)}.$$

Результаты, связанные с ожиданием, оставляют желать лучшего. Причина может быть только одна: недостаточно качественный генератор псевдослучайных чисел[3]. Увеличение числа проведенных испытаний в общем приближает результаты к ожидаемым, но даже при 500 тыс. испытаний погрешность составляет около 17 %. Судя по динамике версий, $GPSS/W$ проходит процесс интенсивной отладки. Можно надеяться, что встроенные в систему ДСЧ будут усовершенствованы. В этом направлении можно приложить и собственные усилия.

В дополнение к приведенным результатам отметим, что для системы $M/M/1$ распределение времени пребывания заявки в системе также подчинено показательному закону со средним $1/(1/0.9 - 1/1) = 9$. При ограничителе таймера 200000 была заказана гистограмма распределения времени пребывания; получены среднее значение 9.264 и среднеквадратическое отклонение 9.147. Требуемое для показательного распределе-

[3] Результаты сопоставления датчиков обсуждаются ниже в разд. 16.1.2.

ния их равенство приближенно выполняется, однако погрешность самих значений составляет около 2%.

10.3 Инсталляция

Для работы с *GPSS/W* необходимо иметь:

- операционную систему *Windows* 95, 98 или 2000;
- Pentium III или лучше (на Pentium I модели умеренной сложности реализуются вполне успешно);
- оперативную память не менее 32 Мб;
- 10 Мб места на винчестере.

От пользователя требуются:

- опыт работы с персональной ЭВМ,
- навыки программирования на языке типа *Паскаль* или *Фортран*,
- знакомство с основными идеями имитационного моделирования — предпочтительно по первой части данной книги.

Заархивированная система с выполняемыми файлами, `Help`'ом и образцами моделей укладывается в 5 дискет. `Tutorial` должен загружаться отдельно. Старые версии *GPSS* рекомендуется удалить предварительно. Инсталляция запускается по `Setup` и по умолчанию идет в папку `Minuteman Software`. Имитации и отчеты помещаются в ту же директорию, где находятся исходные модели. Для обеспечения удобства вызова *GPSS/W* рекомендуется стандартная технология — через `Проводник` вытащить иконку *GPSS World* (с глобусом) на «рабочий стол».

10.4 Методические указания

В состав электронной документации к системе входят примеры тестовых программ, Справочник и Руководство программиста (`Tutorial`). Под Справочником понимается `GPSS World Reference Manual`, входящий в состав электронной документации. Его сокращенное изложение дано в

главе 11. Глава 15 построена на основе `Tutorial`. Предполагается, что читатель перед изучением разделов («уроков») этой главы просмотрит соответствующие разделы упомянутых источников и главы 11. Проработка книги предполагается за включенным компьютером с установленной на ней системой *GPSS World*. Справку по операндам блока удобнее и быстрее всего получать через `Help`.

Начальные уроки не связаны с рассмотрением конкретных блоков. Более того, не все блоки будут рассмотрены на примерах. Потребность в новых блоках определяется спецификой предметной области и будет расти вместе с квалификацией читателя, который с помощью Справочника и уже указанной литературы сам разберется в проблеме.

Для отличения от процесса моделирования продукта компиляции (`Simulation`) этот термин переведен как Имитация — с большой буквы. Также с большой буквы пишутся Эксперимент как комплекс, обеспечивающий многократные прогоны модели (для отличия от эксперимента в широком смысле слова), а также наименования окон, транзактов, матриц и других объектов, выступающие как имена собственные.

Глава 11

Объекты GPSS

11.1 Классы сущностей

Каждая модель формируется из ограниченного набора *сущностей* — `entities`, интерпретация которых определяется предметной областью. Разработчик *GPSS* Дж. Гордон в [89, с. 3] приводит следующие примеры интерпретаций:

Таблица 11.1. Примеры сущностей

Система	Сущности	Атрибуты	Активности
Авиация	Автопилот Элероны Корпус	Ошибка Угол Курс	Сигнал Торможение Поворот
Завод	Детали Станки Работники	Количество Производитель Умение	Заказ Обработка Планирование
Бизнес	Продукты Потребители Рынки	Цена Спрос Объем	Производство Продажа Реклама
Политика	Партии Заявления Социальные группы	Размер Приемлемость Доход	Рост фондов Кампания Миграция

Мир *GPSS* населяют *абстрактные* сущности, к числу которых относятся: транзакты, группы транзактов и числовые группы, блоки, устройства (`Facilities`), памяти (`Storages`), очереди, таблицы,

переменные, матрицы, функции, сохраняемые значения (Savevalues), логические ключи, генераторы случайных чисел и потоки данных. Все сущности могут иметь назначенные программистом имена и автоматически нумеруются на внутреннем уровне (отдельно в каждом классе). Многие сущности создаются автоматически — при обнаружении ссылок на них в программе модели. Блоки включаются в модель $GPSS$ как элементы некоторой структурной схемы.

Для удобства работы мы опишем основные сущности $GPSS$ на двух уровнях. На концептуальном уровне обсуждаются смысловые аспекты и логика взаимодействия. На формальном уровне дается описание синтаксиса команд и блоков и непосредственного действия их. На обоих уровнях группировка материала проведена по родству абстрактных применений.

Перечисленные ниже объекты должны быть обязательно описаны перед их использованием:

- Памяти — операторами STORAGE.
- Арифметические переменные — операторами VARIABLE.
- Булевы переменные — операторами BVARIABLE.
- Матрицы — операторами MATRIX (внутри $PLUS$-процедур — декларациями TEMPORARY MATRIX).
- Таблицы общего назначения — операторами TABLE.
- Q-таблицы ожидания в очередях — операторами QTABLE.
- Табличные функции — операторами FUNCTION и наборами координат.

11.2 Стандартные числовые атрибуты

Стандартные числовые атрибуты (СЧА)[1] позволяют отразить зависимость поведения транзактов от состояния системы: задержки или траектории от параметра транзакта, времени суток, длин очередей у последующих устройств; тарифов — от расстояния, времени суток и дня недели

[1] В оригинале System Numeric Attributes, но, поскольку лишь малая часть их относится к системе в целом, мы предпочли переводить «System» как «стандартный».

и т. п. СЧА могут быть использованы как входы в матрицу — например, расстояний между географическими пунктами. Второй аспект применения СЧА — обеспечить нестандартные сбор и обработку статистики.

Для обозначения класса СЧА используются одна–две буквы и, возможно, цифра. Если СЧА характеризует отдельную сущность (например, устройство), то к типовому имени приписывается номер или имя сущности. В последнем случае имени предшествует знак доллара `$`. Далее мы при указании частных СЧА будем добавлять букву `j`, подразумевая ее возможную замену целым числом или именем с предваряющим знаком `$`. Как СЧА рассматриваются, например, значения переменных — с плавающей точкой (обозначение `Vj`) и булевых (`BVj`).

11.3 Транзакты

Транзакты (заявки, запросы, сообщения), имитирующие обслуживаемые моделируемой системой объекты реального мира, движутся из блока в блок модели, изменяя свое состояние (параметры). Изменение состояния и положения вызывает коррекцию соответствующих списков и отображается в окне блоков. Движущийся транзакт называется *активным*. При движении активного транзакта по блокам собирается стандартная и дополнительная статистика. Если возможности продвижения транзакта исчерпаны (время выхода из блока больше текущего значения таймера или очередной блок отказывается принять транзакт), он переходит в состояние ожидания, а свое движение в модели продолжает другой транзакт.

Логически транзакты представляются записями и находятся в списках, определяющих их текущее положение (блок в схеме модели) и статус каждого из них. Все транзакты нумеруются. Оператор `CLEAR` восстанавливает нумерацию транзактов с единицы. Транзакты могут находиться в одном блоке, одновременно занимать несколько блоков, ждать возможности занять блок. Транзакты окончательно освобождают устройство (память), если они входят в блоки `RELEASE` или `LEAVE`.

11.3.1 Сегменты модели

Модель *GPSS* может состоять из нескольких сегментов, каждый со своим блоком генерации транзактов. Порядок относительного расположения

сегментов безразличен. Активным считается тот из сегментов, в котором находится активный транзакт. Зависимость сегментов реализуется через общие сущности и переходы транзактов между сегментами. К примеру, модель системы управления запасами может иметь два сегмента: удовлетворения спроса и восполнения запаса. Сегменты такой модели «общаются» через сохраняемую величину STOCK — наличный запас: в первом он снижается на величину ежедневного спроса, во втором увеличивается на величину заказа и определяется необходимость нового заказа.

Типовая «биография» каждого транзакта описывается соответствующим сегментом модели. Когда транзакт входит в блок GENERATE, интерпретатор планирует время поступления транзакта путем формирования случайной величины и добавления ее к текущему значению таймера. При достижении этого времени запланированный транзакт начинает свое движение, а GENERATE готовит следующий транзакт.

В качестве отдельного сегмента обычно оформляется модель потока отказов или перерывов в работе, делающих недоступными на некоторое время устройства и памяти, а также сегмент завершения имитации по времени. Из вариантов

```
GENERATE 480              GENERATE  1,,,480
............              ...........
TERMINATE 1               TERMINATE 1
```

правый будет требовать 480 проверок на окончание моделирования, а левый — только одной. Сегменты, задающие временны́е ритмы, могут быть циклическими. Все они (как и прочие блоки модели) должны иметь один и тот же масштаб времени.

11.3.2 Параметры транзактов

Поведение транзакта определяется как внешними обстоятельствами — значениями некоторых СЧА и переменных, так и индивидуальными свойствами самих транзактов, отражаемыми в их *параметрах*. Число параметров не ограничивается. Параметры могут использоваться в качестве операндов блоков, при вычислении функций и переменных — для определения путей следования, управления циклами, временны́х задержек и т. п.

На параметры можно ссылаться по номерам или именам — как правило, после буквы P. Если операнд блока является ответным, указываются только номер или имя параметра. При обращении к параметрам можно использовать косвенную адресацию: *3 означает, что в третий параметр записывается имя имеющего это значение объекта. Значения параметрам присваивают блоки ASSIGN, PLUS, а также MARK или TRANSFER. *При всех ссылках на параметры подразумеваются параметры активного транзакта.*

В обязательный набор параметров транзакта входят:

PR — приоритет (по умолчанию нуль). Приоритет учитывается при организации очередей, установлении текущей последовательности событий, занятии транзактами устройств и памятей. Транзакту с бóльшим значением приоритета отдается предпочтение перед прочими при выделении запрошенных им ресурсов.

MARK TIME — значение абсолютного времени при входе транзакта в модель или при прохождении транзакта через блок MARK без операнда A.

ASSEMBLY SET — номер ансамбля, к которому принадлежит транзакт. Понятие ансамбля связано с операциями типа разборки, сборки и синхронизации и обсуждается в разд. 13.3.

DELAY INDICATOR указывает, что транзакт не смог перейти к следующему блоку. Индикатор задержки используется блоком TRANSFER SIM.

TRACE INDICATOR устанавливает индикатор (флаг) трассировки транзакту, вошедшему в блок TRACE, и его траектория начинает отслеживаться. Флаг убирается блоком UNTRACE.

CURRENT BLOCK — блок, в котором находится активный транзакт.

NEXT BLOCK — номер блока, к которому должен перейти данный транзакт.

CHAINS — в каких списках можно найти данный транзакт.

Транзакты могут быть в *одном* из следующих состояний:

ACTIVE — активный;

`SUSPENDED` — ожидающий в цепи будущих событий или своего перевода в активное состояние — в цепи текущих событий;

`PASSIVE` — находящийся в состоянии покоя в цепях пользователя, задержки или ожидания;

`TERMINATED` — подлежащий удалению из текущей модели.

Из перечисленых параметров доступен программисту только `PR`, который задается в блоках `GENERATE` или `PRIORITY`. Последний может изменить ранее назначенный приоритет. Остальные системные параметры транзакта используются только управляющей программой.

Программист может вводить произвольное число нумерованных или именованных дополнительных параметров (рост, вес, температура, назначение, индекс детали, цена, трудоемкость обработки и т. п.). Параметры транзактов нельзя использовать для управления работой блока `GENERATE`.

11.3.3 Цепи транзактов

Динамика продвижения транзактов основана на *цепях* (упорядоченных списках) событий. Подобные списки в явной или неявной форме существуют в любых программах имитации, в том числе записанных на универсальных языках программирования. Поскольку в *GPSS* общая логика моделирования отделена от описания конкретного процесса обслуживания, для обеспечения универсальности она реализуется в максимально развитой форме.

Ответ на вопрос, где находится транзакт, может быть дан с двух точек зрения:

- в каком блоке,
- в какой цепи.

Фактическая последовательность квазиодновременных событий всегда определяется их цепями. Работа с цепями обязательна при создании *логически* сложных моделей независимо от числа блоков в них. При выполняемой по запросу программиста распечатке содержимого цепей печатаются и параметры входящих в них транзактов.

«Устройства» имеют несколько цепей, остальные сущности — только цепи повторных попыток. Каждый транзакт может входить в любое число цепей; однако некоторые цепи (прерывания и будущих событий) являются взаимно исключающими.

Транзакт может находиться не более чем в одной из следующих цепей:

- `Future Events Chain` — будущих событий (ЦБС),
- `Current Events Chain` — текущих событий (ЦТС),
- `Delay Chain` — задержки на устройстве или памяти (ЦЗд),
- `Pending Chain` — зависания на устройстве (ЦЗв),
- `User Chain` — цепи пользователя (ЦП).

Транзакт может ожидать наступления любого числа событий, состоять в любом числе групп транзактов, может быть прерван на любом числе устройств одновременно. Это означает, что любой конкретный транзакт может одновременно находиться в любом числе цепей прерываний, групповых цепей и цепей повторных попыток — ЦПП.

Цепь текущих событий

содержит транзакты, которым предстоит пройти еще хотя бы один блок до очередного увеличения модельного времени. ЦТС выстроена по убыванию приоритетов; активный транзакт обычно продолжает свое продвижение до «возбуждения» более приоритетных. Когда он задерживается в одной из цепей, активируется транзакт из ЦТС, имеющий наивысший приоритет. При исчерпании ЦТС в нее переносится ближайший по времени транзакт из ЦБС, и системное время обновляется.

Цепь будущих событий

упорядочена по *времени* активации транзактов. Транзакты попадают в эту цепь только через блоки `GENERATE` и `ADVANCE`. Блоки `PREEMPT` и `DISPLACE` могут вывести транзакт из ЦБС; такой транзакт возвращается в ЦБС через новый `ADVANCE`. При совпадении будущих времен для нескольких транзактов все они переносятся в ЦТС в случайном порядке. В связи с отказом в $GPSS/W$ от целочисленности времени вероятность таких совпадений весьма мала.

Цепь повторных попыток

содержит транзакты, которые не удовлетворяют всем условиям входа в блоки GATE, TEST, TRANSFER ALL, TRANSFER BOTH. Каждое устройство имеет свою цепь блокированных транзактов. Транзакт из ЦПП, дождавшись изменения некоторого СЧА, переносится в ЦТС. Когда он становится активным, проверка повторяется. Неудачный выбор условий может привести к избыточным проверкам и замедлить моделирование.

При входе транзакта в блок он автоматически удаляется изо всех ЦПП.

Цепи устройств

Каждое *устройство* имеет четыре цепи транзактов:

Pending — ожидания возможности прервать устройство в режиме прерывания,

Interrupt — прерванных,

Delay — ожидающих возможности занятия,

Retry — ожидающих готовности.

Транзакт, заставший устройство свободным, занимает его. Вошедший в блок PREEMPT в режиме прерывания транзакт с приоритетом выше текущего обслуживания прерывает последнее и занимает устройство, а прерванный транзакт поступает в цепь Interrupt. В *GPSS* вложенные прерывания исключены. Поэтому при непустой цепи Interrupt данного устройства вновь пришедший помещается в цепь Pending. При отсутствии права прерывания, а также при задании блоку PREEMPT режима приоритета, «пришелец» помещается в цепь Delay.

При освобождении устройства заявки последовательно выбираются из цепей Pending, Interrupt, Delay — во всех случаях с учетом своих приоритетов.

Цепь Retry обсуждалась в предыдущем подпункте и дополнительных пояснений не требует.

Цепи для памятей

Памяти имеют цепи только двух видов: `Delay` и `Retry`. Их функции аналогичны рассмотренным для устройств — с той разницей, что условием занятия памяти является достаточность наличного ресурса в сравнении с запрошенным. При выходе транзакта цепь `Delay` просматривается в порядке убывания приоритетов и из нее выбирается первый транзакт, для которого наличный ресурс окажется достаточным.

Цепи пользователей

Пользовательские цепи ПЦ — это списки транзактов, которые удалены из ЦТС блоками `LINK`. Их назначение — реализация сложных дисциплин обслуживания, управляемых значениями параметров транзактов. Кроме того, они могут применяться для повышения эффективности моделирования посредством исключения проверок с заведомо отрицательным результатом.

11.3.4 Планировщик транзактов

Координация процессов в модели обеспечивается интерпретатором *GPSS* — планировщиком транзактов. Его первая задача — выбрать активный транзакт, которым становится головной транзакт непустого списка с наивысшим приоритетом. Если ЦТС пуста, в нее переносится первый по времени транзакт ЦБС и продвигается системное время. Естественно, что перенесенный транзакт и становится активным.

Активный транзакт продвигается по схеме, вызывая подпрограммы очередных блоков. Движение прекращается при необходимости ждать изменения состояния некоторой сущности. Если транзакт пытается войти в блок типа `SEIZE`, `ENTER`, `TEST` или `GATE` и получает отказ, то он помещается в ЦТС в качестве заблокированного. Для каждого транзакта создается ссылка на причину задержки. Включенный индикатор пропуска заставляет переходить к следующему транзакту ЦТС — это быстрее, чем пытаться продвинуть транзакт. Когда условие блокировки будет снято, индикатор выключается, транзакт продвигается по возможности и просмотр ЦТС начинается с начала. Однако даже этот просмотр может отнимать много времени. Кроме того, здесь жестко реализуется принцип FIFO. Другие дисциплины и более экономный просмотр обеспечиваются через пользовательские цепи (ПЦ).

Транзакты входят в ПЦ через блок LINK и остаются там неактивными, пока не удаляются из цепи транзактом, входящим в UNLINK. Разблокированный транзакт можно вернуть в ЦТС на место, определенное программистом по любому правилу. Например, дисциплину LIFO можно реализовать двумя различными способами — ввести в начало и вывести с конца или наоборот. Применяя LINK с упорядочением по значению параметра, можно реализовать, например, выбор по кратчайшей длительности обслуживания. С помощью ПЦ можно построить модель с переходом транзактов из одной очереди в другую — см. упр. 19 и 20 разд. 7.11 [74]. Ответственность за правильное взаимодействие этой пары блоков целиком ложится на программиста.

Блок PREEMPT работает в режиме приоритета или прерывания. В первом случае не могущий захватить устройство новый транзакт поступает в цепь DELAY, во втором — в PENDING. Для прерванного транзакта фиксируется «недоданное» ему время, с которым он в свое время будет автоматически восстановлен на устройстве.

11.4 Ансамбли и группы

Между транзактами, находящимися в модели, может существовать некоторая связь. Например, процессы промышленного производства любого сложного изделия — «железного» или программного — обычно проходят три стадии: «сверху вниз» при проектировании, параллельное изготовление и сборку с комплексной отладкой. Независимые работы, которые выполняются параллельно, не могут обеспечиваться одним транзактом. Запускающий их транзакт должен *расщепляться* с последующей сборкой компонент. Очевидна необходимость распознавания принадлежности последних к единому агрегату. Операции объединения необходимы также для системы с порогом обслуживания: лифт начинает подниматься, когда наберется не менее заданного числа пассажиров, сборка подшипника — при изготовлении требуемого числа шариков, кольца и сепаратора. С другой стороны, желательна возможность производить некоторые *групповые* действия над однородными транзактами.

Все транзакты в моделях принадлежат к множествам, называемым *ансамблями*. Как правило, в модели сосуществуют несколько различных ансамблей. Каждый транзакт является членом только одного ансамбля. Транзакт, введенный в модель блоком GENERATE, получает числовую мет-

ку — номер ансамбля и становится первым и пока единственным членом нового ансамбля. Формируемые при вхождении в блок SPLIT его потомки (и, возможно, потомки этих потомков) включаются в тот же ансамбль.

Ансамбли полезны, когда процесс должен ждать наступления некоторых событий. С понятием ансамбля связаны *цепи парности* — в этом случае транзакты дожидаются в них членов своего анасамбля. У ансамблей нет имен, на которые можно было бы ссылаться при описании логики модели. В блоке ADOPT можно изменить ансамблевую принадлежность транзакта.

Управлять движением транзактов в зависимости от их принадлежности к ансамблям можно с помощью блоков ASSEMBLE, GATHER, MATCH. С помощью блока GATE и логических условий M/NM вход транзакта в любой блок можно поставить в зависимость от наличия элементов того же ансамбля в другом блоке.

Блок ASSEMBLE (соединить) после входа в него первого члена ансамбля начинает подсчет входящих в него членов того же ансамбля, пока их число не достигнет значения аргумента A (счетчика соединения). С этого момента движение продолжает только первый транзакт, сохраняющий все свои прежние свойства. Значение счетчика не обязано быть равным численности ансамбля. Заметим, что:

- для каждого ансамбля в данном блоке ASSEMBLE одновременно может выполняться только одна операция соединения;
- в данном блоке ASSEMBLE может параллельно выполняться операция соединения двух и более ансамблей;
- для каждого ансамбля операции соединения могут одновременно выполняться в нескольких ASSEMBLE.

Блок GATHER (собрать) действует аналогично ASSEMBLE и обладает теми же свойствами. В отличие от ASSEMBLE, в нем при накоплении заданного в первом операнде числа транзактов движение продолжают все они в порядке FIFO.

Блок MATCH (синхронизировать) в своем операнде A ссылается на метку сопряженного с ним (взаимно). При входе транзакта в один из блоков MATCH и обнаружении в сопряженном другого члена того же ансамбля оба транзакта продолжают движение. При отсутствии пары активный транзакт задерживается в цепи парности.

Принадлежность транзактов к ансамблям определяется только *генетически*. С другой стороны, имеется возможность их *произвольной группировки*. Каждый транзакт может одновременно принадлежать любому числу различных групп. Транзакт становится членом группы, пройдя через блок JOIN (присоединить). В отличие от вошедших в цепи, включенный в группу транзакт не делается пассивным и продолжает двигаться, оставаясь в группе до завершения своей траектории; удаление из группы происходит при входе в блок REMOVE данного или другого транзакта. Путь транзакта можно сделать зависящим от его принадлежности к группе, включив в маршрут блок EXAMINE.

Здесь же имеет смысл рассмотреть родственные по выполнямым функциям, но формально не связанные с понятием группы блоки SELECT и COUNT. Блок SELECT выбирает сущность, удовлетворяющую условию — например, память с пустой или минимальной очередью. Если условие не выполнено, транзакт при пустом поле B будет ждать, а при заполненном перейдет к указанному в B блоку. Блок COUNT подсчитывает количество сущностей, удовлетворяющих условию, и помещает результат в свой первый операнд. Наконец, с помощью ALTER можно модифицировать параметр или приоритет группы в целом или ее отдельных членов. При выполнении этих операций местонахождение членов группы несущественно.

Контроль принадлежности транзактов к группе осуществляется через списки их номеров. В блоках JOIN, REMOVE, EXAMINE есть возможность работать в числовой моде — с произвольно назначаемыми числами вместо номеров. Программист может, например, затребовать список всех занятых устройств. Блоки SCAN и ALTER этот режим не поддерживают.

На каждую группу создается FIFO-список. В режиме транзактов в списке фигурируют номера транзактов, в числовом — произвольные числа, назначаемые пользователем. В пределах группы числа должны быть различными. Числовая мода обычно используется для хранения номеров памятей и устройств. Выбор моды определяется первым блоком, ссылающимся на группу в ходе прогона. Блоки JOIN, REMOVE, EXAMINE работают с группой в любом режиме, SCAN и ALTER — только в режиме транзактов. Помимо транзактов, группы могут состоять и из устройств и памятей (порознь), удовлетворяющих условию отбора. Соответственно все сказанное в этом подразделе о транзактах (исключая движение) относится и к ним.

Типичные приложения перечисленных блоков — моделирование асинхронных процессов в операционных системах (для ввода-вывода должны быть доступны канал и дисковод). Если канал занят, приходится ждать некоторое число оборотов диска — см. рис. на с. 273 [89].

11.5 Устройства и памяти

Устройство (`Facility`) есть объект, которым в любой момент времени может *владеть* только один транзакт. Последний занимает устройство при входе в блок `SEIZE`. Блок `PREEMPT` позволяет занять устройство транзакту более высокого приоритета, временно освобождая его от менее приоритетного транзакта. Посредством устройств проще моделируется обслуживание неоднородных заявок, распределение трудоемкостей и объемов которых различно в зависимости от их типа, а также работа неоднородных каналов с отказами и перерывами в работе.

Название `STORAGE`, вообще говоря, обозначает запас, хранилище или склад, но переводится по-разному: в [74] как «многоканальное устройство», в [65] — как «накопитель». Последний вариант особенно неудачен, поскольку представляет ситуацию с точностью «до наоборот»: реально транзакты не накапливают свои запросы, но *уменьшают* наличие свободного ресурса. В связи с широким применением *GPSS* для моделирования вычислительных процессов мы будем переводить упомянутый термин как *память*.

Примерами «памятей» могут служить *бригады* параллельно работающего персонала: парикмахеры, официанты, кассиры универсамов, сберкасс и касс продажи железнодорожных и авиабилетов, контролеры в театрах, банковские и связные операторы, библиотекари, автомеханики, выездные мастера по установке и ремонту мебели и бытовой техники, портовые лоцманы, авиадиспетчеры, бригады скорой помощи, оперативные группы милиции и т. п. Здесь подразумевается и соответствующее техническое обеспечение. К «памятям» можно отнести также:

- буксиры, вводящие корабли в порт и выводящие их из него;
- причалы в порту;
- подъемные краны в цехах;
- электрокары и автопогрузчики на производстве и в складах;

- места на стоянке автомобилей;
- компьютеры в центрах коллективного пользования (например, в библиотеках);
- ресторанные столики;
- выдаваемую в прокат бытовую технику (автомобили, телевизоры, холодильники, видеокамеры), фраки для нобелевских лауреатов, подвенечные платья.

Здесь, как и в первой группе случаев, обычно имеется в виду человеко-машинный комплекс.

Память определяется своей емкостью и должна быть описана управляющим оператором STORAGE. Память может использоваться одновременно несколькими транзактами, суммарная потребность которых в запрашиваемом ресурсе не превышает ее емкости. Емкость и запросы считаются целочисленными. Объем запроса не обязан быть единичным. Запрос выполняется лишь при возможности его *полного* удовлетворения. Подчеркнем, что *заполненная память* означает отсутствие соответствующего ресурса.

Блоки ENTER и LEAVE при входе в них транзакта изменяют содержимое памяти (соответственно увеличивают и уменьшают его), одновременно корректируя связанные с памятью СЧА. Блок SAVAIL переводит память в состояние доступности, SUNAVAIL — в противоположное. Эти изменения учитываются в статистике, собираемой о функционировании памяти.

Память может использоваться как счетчик, собирающий заданное число временных сущностей или дающий предупреждение о превышении порога. Такой счетчик может быть использован для моделирования систем с порогом включения. В [89, разд. 5.12] приведен пример на отправку продукции упакованными партиями по 20 штук. За памятью емкостью 19 единиц в модели следует условный блок TRANSFER, причем 19 транзактов после памяти идут в TERMINATE. 20-й транзакт, найдя память полной, отклоняется в блок LEAVE, который освобождает 19 единиц памяти и идет на TERMINATE, подсчитывающий количество упаковок.

Многоканальные системы обслуживания в принципе могут моделироваться как устройствами, так и памятью. Сравнительная характеристика этих подходов дана в табл. 11.2.

Таблица 11.2. Сравнение устройств и памятей

Факторы	Устройства	Память
Очередь	Раздельная	Общая
Доступность	Раздельная	Общая
Ресурс	Время	Произвольный
Неоднородность	Возможна	Исключена
Идентификация выхода	Есть	Нет
Возможность прерывания	Есть	Нет

При работе с устройствами идентифицируются конкретные транзакты, так что требуется хранить больше информации. Кроме того, каждое устройство должно быть представлено в программе отдельно, и придется управлять распределением заявок между ними. Раздельные очереди к одноканальным устройствам, строго говоря, в совокупности реализуют отличную от FIFO дисциплину ожидания. Наконец, статистика будет дана отдельно по каждой из этих очередей. Все это — аргументы в пользу «памятей».

Устройства и память, помимо степени занятости, характеризуется текущей *готовностью*. В отличие от совокупности устройств, память может оказаться неготовой только *в целом*. Режим готовности вводится блоками `FAVAIL`, `SAVAIL`, режим неготовности — по `FUNAVAIL`, `SUNAVAIL`. Это свойство позволяет моделировать поломки техники, регламентные работы, перерывы у кассиров на сдачу денег и продавцов — на прием товара, работу с перерывами на обед, недельный и годовой ритмы. Времена поломок генерируются аналогично моментам прибытия, интервалы между ними обычно подчинены распределению Вейбулла [89, с. 99].

От готовности устройства или памяти следует отличать *доступность*, регулируемую блоками `GATE`, `TEST` или `TRANSFER` в зависимости от комбинации *внешних* условий.

Обычная ошибка программиста — разрешение транзакту покинуть сущность без снятия потенциально блокирующего условия — выход из устройства (памяти) без `RELEASE` (`LEAVE`) соответственно. Столь же опасна работа с очередью без блока `DEPART`.

11.6 Приоритет транзактов и прерывания

Необходимость предоставления некоторым транзактам приоритетов часто вытекает уже из постановки задачи. При моделировании однопутного отрезка железной дороги с разъездом посередине приоритет может быть предоставлен одному из направлений, пассажирским поездам перед товарными и т. п. На автодорожных перекрестках приоритет определяется известной водителям системой правил и общей ситуацией. При моделировании работы технических систем аварии имеют приоритет выше, чем основные транзакты.

11.6.1 Механизм приоритета

Выше уже отмечалось, что транзакты могут иметь целочисленные приоритеты (по умолчанию — нулевой). Приоритет назначается при генерации (операнд `E` блока `GENERATE`) или блоком `PRIORITY` в произвольном месте модели. Обычно цепи событий просматриваются по убыванию приоритетов с дисциплиной FIFO в каждой приоритетной группе. Соответственно из транзактов, запланированных на один момент модельного времени, более приоритетные обрабатываются раньше. Напротив, выбор из одновременных событий одного приоритета может явиться результатом длинной цепочки разновременных действий и практически оказывается случайным — см. [89, с. 113]. Изменение приоритета меняет статус транзакта и может потребовать повторного просмотра ЦТС. Если в поле `B` блока `PRIORITY` указано `BUFFER`, имитатор начинает просмотр ЦТС сначала. Такое же действие оказывает помещение блока `BUFFER` после блока смены приоритета.

Старшинство в приоритете само по себе не вызывает прерывания начатого обслуживания — реализуется так называемый *относительный* приоритет, касающийся только ожидания. Организация приоритетного обслуживания с прерыванием программируется специально. Другие дисциплины могут быть запрограммированы работой с ЦБС. Вводить приоритеты имеет смысл только при большой загрузке и малом удельном весе приоритетных заявок. В этих условиях он дифференцирует показатели обслуживания — существенно улучшает их для высокоприоритетных заявок за счет некоторого ухудшения для низкоприоритетных. Другое назначение приоритетов — модификация логики модели (см. ниже).

11.6.2 Прерывание обслуживания

Прерывание начатого обслуживания возможно только для «устройств» и практикуется лишь при исключительной важности вновь пришедшего транзакта. Типичными примерами являются аварийные ситуации, потребность в экстренной медицинской или иной помощи. Допущение прерываний усугубляет вышеупомянутый эффект обслуживания с относительным приоритетом.

Прерывание и последующее возобновление начатого обслуживания организуется блоками `PREEMPT` и `RETURN`. *Прерывающая заявка сама прервана быть не может.* Имя устройства указывается в поле `A` блока `PREEMPT`. Прерванный транзакт с записанным остатком времени обслуживания помещается в цепь прерываний и возвращается в ЦБС, когда прерывание закончится.

Блок `RETURN` снимает прерывание с указанного в поле `A` устройства. Если в блоке `PREEMPT` будут транзакты более высокого приоритета, чем прерванный, устройство займет один из них.

11.6.3 Приоритет и временны́е узлы

Назначение приоритетов может потребоваться для обеспечения правильной последовательности обработки квазиодновременных событий («временны́х узлов»):

1. При одновременном прибытии заявки и окончании обслуживания сначала должно завершиться обслуживание. Повышение приоритета обслуживаемого транзакта перед входом в блок `ADVANCE` позволит предотвратить необоснованный отказ в приеме заявки. Так обстоит дело при входе и выходе пассажиров междугородного автобуса — сначала обеспечивается выход. Еще один временно́й узел образуют вход последнего пассажира и отправление автобуса: пассажир должен успеть войти!

2. При одновременном поступления машины на автозаправочную станцию и закрытием станции в конце рабочего дня машина должна быть обслужена [2]. Для этого транзакту-машине при генерации должен быть присвоен более высокий приоритет, чем временно́му транзакту.

[2] Впрочем, это зависит от национальных традиций.

11.7 Логические ключи

Логические ключи наряду с булевыми переменными используются для управления движением транзактов. Состояние ключа представляется СЧА вида LSj, который равен 1, если логический ключ находится в положении «SET», и нулю — в положении «RESET».

11.8 Переменные и сохраняемые значения

11.8.1 Переменные

GPSS сопоставляет с переменными выражения, которые вычисляются при обращении к ним. Фактически переменные играют роль операторных функций *Фортрана* от глобально определенных аргументов. Выражения могут включать в свой состав константы, СЧА, указатели функций, арифметические и логические операторы. Команды VARIABLE описывают арифметические выражения (формулы), в которых действия над входящими в них переменными производятся с плавающей точкой и удвоенной точностью.

Команды BVARIABLE описывают *булевы* выражения. В них используются имеющие логические значения СЧА (занятость устройства, наличие свободной памяти и т. п.), а также связанные операторами отношения арифметические выражения. Булевы подвыражения могут соединяться логическими связками 'OR', 'AND', 'NOT'. «Истина» и «ложь» представляются единицей и нулем соответственно, что позволяет использовать их и в арифметических сравнениях. Типичный пример применения булевых переменных — проверка комбинации условий: судно не может войти в гавань, пока не окажется, что:

- нет шторма;
- есть свободная якорная стоянка;
- имеется свободный буксир.

СЧА вида BVj дает значение логического выражения, а Vj — арифметического выражения (целое или с плавающей точкой).

В связи с ограниченным набором имеющих логические значения СЧА для формирования сложных логических условий могут быть полезны правила де Моргана:

$$\begin{aligned} \overline{x_1|x_2} &= \bar{x}_1\&\bar{x}_2, \\ \overline{x_1\&x_2} &= \bar{x}_1|\bar{x}_2. \end{aligned}$$

Здесь надчеркивание означает операцию отрицания. Эти правила могут читаться справа налево и естественно обобщаются на произвольное число операндов.

11.8.2 Сохраняемые значения

Транзакты не могут непосредственно ссылаться друг на друга. Их общение реализуется через *сохраняемые значения*, которые в этом контексте могут рассматриваться как аналоги «ящиков» электронной почты. Они могут использоваться для сохранения параметров транзактов, значений выражений и СЧА и изменяются только по прямому указанию программиста в блоках `SAVEVALUE` (`MSAVEVALUE`). Все ненулевые значения сохраняемых величин автоматически отражаются в отчете.

Перед началом моделирования интерпретатор устанавливает значения сохраняемых величин в нуль. При необходимости посредством команд `INITIAL` можно задать ненулевые начальные значения (отдельной командой на каждое). Имя матрицы (после `MX$`) сопровождается индексами (в круглых скобках), далее указывается значение.

11.9 Накопление статистики

11.9.1 Очереди

Ожидание транзакта начинается при возникновении в модели любого условия, блокирующего его продвижение. Однако измерение длительности ожидания начинается только при входе в блок `QUEUE` — *регистратор очереди* [3].

Окончание ожидания определяется входом в блок `DEPART`. Связанные с этими блоками увеличение и уменьшение длины очереди, как и в

[3] Отсутствие регистратора не исключает существования очереди.

случае занятия и освобождения памяти, рассматриваются как независимые процессы (длина очереди не обязательно изменяется на одно и то же число). Если поместить DEPART после блока ADVANCE, задающего время пребывания непосредственно на обслуживании, то в счет будет входить и время обслуживания.

Транзакт может находиться одновременно в нескольких очередях (количество определяется числом полей в записи транзакта для имени очереди и времени входа в нее). В цепочке блоков QUEUE — QUEUE — SEIZE — DEPART — ADVANCE — RELEASE — DEPART транзакт последовательно входит в две очереди, из которых первая фиксирует время ожидания, а вторая — полное время пребывания в системе обслуживания. Затем SEIZE имитирует занятие устройства, первый DEPART — окончание ожидания. Блок ADVANCE организует задержку на время обслуживания, RELEASE освобождает устройства, второй DEPART фиксирует окончание времени пребывания в системе обслуживания.

СЧА, характеризующие состояние очереди, могут использоваться не только для сбора статистики, но и для управления программными моделями — например, для направления транзакта к устройству с наименьшей длиной очереди или с наименьшим числом входов.

Статистика очередей накапливается и выдается автоматически. Интерпретатор в любой момент может рассчитать среднее время пребывания в очереди делением накопленного времени пребывания в очереди на значение счетчика входов. Это среднее будет несколько занижено, поскольку транзакты, еще не вышедшие из очереди (если таковые имеются), уже учтены в счетчике входов, но время их ожидания еще не добавлено к накопителю.

При необходимости построения гистограммы распределения времени ожидания в очереди соответствующая таблица должна быть описана управляющим оператором QTABLE. Кроме частот гистограммы, вычисляется среднее арифметическое значение аргумента и его среднеквадратическое отклонение. В отчет (но не в окно таблиц) выводится также статистическая кумулянта.

11.9.2 Таблицы общего назначения

Статистику можно собирать не только по временам ожидания. Интервал времени, в течение которого транзакт находится в модели, называется его *резидентным временем*. Интервал времени перехода от одной точ-

ки схемы до другой называется *транзитным временем* для этих точек. Блок MARK отмечает системное время начала отсчета, TABULATE — его окончание. Транзакт может нести несколько отметок времени, позволяя измерять перекрывающиеся интервалы. Если началом отсчета является время создания транзакта, то блок MARK не нужен. Соответствующие таблицы описываются оператором TABLE.

11.10 Трассировка

При входе транзакта в блок TRACE включается его индикатор трассы. После этого при каждом входе транзакта в очередной блок в журнал выдается трассировочное сообщение. Отслеживаемый участок программы ограничивается блоком UNTRACE (TERMINATE).

При вставке блока TRACE в интерактивном режиме его действие распространяется на единственный транзакт.

11.11 Генераторы случайных чисел

GPSS/W имеет датчики равномерно распределенных псевдослучайных чисел {RNj}, которые используются в блоках GENERATE для генерации интервалов между транзактами, в ADVANCE — для формирования задержек в устройствах и памятях, в блоке TRANSFER (статистический режим) — для случайного выбора направления дальнейшего движения. Распределение ДСЧ по упомянутым типам блоков задается в меню Model Settings/Random.

Потоки равномерно распределенных псевдослучайных чисел генерируются посредством мультипликативного генератора с длиной периода $2^{31}-2$ (нулевое значение исключается). Начальное значение по умолчанию совпадает с номером соответствующего ДСЧ. Начальные установки первых семи ДСЧ с помощью команды RMULT можно задавать произвольно, что позволяет управлять *парами* потоков случайных чисел через подбор начальных установок. Если для одного генератора случайных чисел выбрана установка M, то при выборе множителя для другого генератора $N = 2^{31} - M = 2147483648 - M$ последний будет формировать случайные числа, дополнительные к числам первого ДСЧ.

СЧА класса RN возвращают целые случайные числа в интервале [0,999]. При необходимости их можно привести к любому другому це-

лочисленному диапазону, определив соответствующую переменную с помощью операции вычета по модулю. Например, конструкция RN3@24+40 даст случайные целые числа из интервала $[40, 63]$.

11.12 Потоки данных

Поток данных — это последовательность строк текста. Каждый поток идентифицируется своим номером (целый, положительный). Потоки данных используются для записи в файл и чтения из него.

Основная единица потока — текстовая строка. Имеется библиотека встроенных процедур для работы со строками. Элементарные операции READ, WRITE, SEEK работают с отдельными строками. Концом строки являются символы CR, LF (возврат каретки и перевод строки соответственно), которые при записи добавляются автоматически. Преобразование строковых значений в числовые при считывании и числа в строку — при записи также производятся автоматически.

Потоки данных бывают двух типов:

- ввод/вывод для файлов,
- потоки внутри оперативной памяти — для тестирования и прямого доступа к внутренним данным.

Блоки OPEN и CLOSE начинают и прекращают работу с потоком. READ и WRITE считывают и записывают строки. SEEK A используется для установки строки с заданным номером A. Пример работы с файлом:

```
*****************************************************************
*         READ & MODIFY  MYFILE.TXT                             *
*****************************************************************
     GENERATE   1,,,1
     OPEN       ("MYFILE.TXT"),1,DONE   ; Открыть файл потока 1
     READ       Text_Parm,1,DONE        ; Строку -> в параметр
     SAVEVALUE  Open_line,P$Text_Parm   ; Сохранить параметр
     SEEK       20,1                    ; Файл -> на 20-ю строку
     WRITE      ("New line 20"),1,DONE  ; Заменить строку
DONE CLOSE      Err_Parm,1              ; Записать файл на диск
     SAVEVALUE  File_Err,P$Err_Parm     ; В станд. отчет
     TERMINATE  1
```

Глава 12

Язык GPSS

Разработать модель реальной системы значит отобразить фрагмент реального мира на мир *GPSS*: составить последовательность блоков и команд, которые заставят транзакты вести себя как в реальных или проектируемых системах и обеспечат сбор и обработку необходимых статистических данных.

Программа *GPSS/W* есть последовательность операторов языка. В программу могут также входить описания *PLUS*-процедур, составленных пользователем. Перечисленные конструкции могут считываться и из ранее составленного программного файла командой

`INCLUDE "<спецификация_файла>"`.

12.1 Виды операторов GPSS

В набор базовых средств языка моделирования входят свыше 50 блоков, 25 команд, 35 системных числовых атрибутов и 12 типов операторов языка *PLUS*. Команды определяют объекты *GPSS*, то есть играют роль *описаний* в языках типа *Фортран* или *Паскаль*, либо некоторым образом влияют на состояние формулируемой модели. Операторы блоков приводят к созданию функциональных блоков *GPSS*.

12.1.1 Команды

Приведем алфавитный перечень команд системы *GPSS/W*:

`BVARIABLE` — определить булеву переменную;

`CLEAR` — сбросить статистику и удалить транзакты;

`CONDUCT` — провести $PLUS$-эксперимент;

`CONTINUE` — возобновить прерванное моделирование;

`EQU` — назначить значение пользовательской переменной;

`EXIT` — закончить сеанс $GPSS/W$;

`FUNCTION` — определить функцию;

`HALT` — остановить моделирование и удалить все отсроченные команды;

`INCLUDE` — прочитать и транслировать дополнительный файл;

`INITIAL` — задать начальные или новые значения ключа, сохраняемого значения или матрицы;

`INTEGRATE` — автоматически проинтегрировать переменную по времени;

`MATRIX` — определить матрицу;

`QTABLE` — определить таблицу для очереди;

`REPORT` — задать имя файла для отчета или выдать отчет немедленно;

`RESET` — сбросить статистики моделирования;

`RMULT` — задать установки для первых семи ДСЧ;

`SHOW` — вычислить и высветить выражение;

`START` — установить счетчик завершений и начать моделирование;

`STEP` — сделать установленное число попыток входа в блок;

`STOP` — задать условие останова по числу попыток входа в блок;

`STORAGE` — определить память;

`TABLE` — определить таблицу.

12.1.2 Блоки GPSS

Оператор блока создает блок $GPSS$, который производит определенное действие при входе в него транзактов. Приведем их перечень:

`ADOPT` — изменить ансамбль транзакта;

`ADVANCE` — задержать транзакт на заданное время в цепи будущих событий;

`ALTER` — проверить и модифицировать транзакт в группе;

`ASSEMBLE` — ждать накопления транзактов и разрушить копии;

`ASSIGN` — изменить параметр транзакта;

`BUFFER` — начать сначала просмотр цепи текущих событий;

`CLOSE` — конец потока данных;

`COUNT` — подсчитать объекты и записать их количество в параметр транзакта;

`DEPART` — уменьшить очередь;

`DISPLACE` — изменить транзакту следующий блок;

`ENTER` — занять или ожидать занятия нескольких единиц памяти;

`EXAMINE` — проверить принадлежность к группе;

`EXECUTE` — выполнить действие, определяемое указанным здесь блоком;

`FAVAIL` — снять запрет на вход транзактов в устройство;

`FUNAVAIL` — запретить вход транзактов в устройство;

`GATE` — проверить объект и изменить маршрут транзакта;

`GATHER` — ждать транзакты одного семейства до сбора требуемого их количества;

`GENERATE` — создать транзакт и поместить его в ЦБС;

`INDEX` — изменить параметр транзакта;

`INTEGRATION` — включить или выключить интегрирование переменной;

`JOIN` — ввести элемент в числовую группу или группу транзактов;

`LEAVE` — освободить часть памяти;

`LINK` — перевести транзакт в ПЦ;

`LOGIC` — изменить состояние логического ключа;

`LOOP` — уменьшить значение параметра на единицу и перейти к указанному блоку, если результат больше нуля;

`MARK` — ввести в параметр транзакта значение модельного времени;

`MATCH` — ждать входа связанного транзакта в сопряженный блок `MATCH`;

`MSAVEVALUE` — изменить элемент матрицы;

`OPEN` — открыть поток данных;

`PLUS` — вычислить $PLUS$-выражение и записать его в параметр транзакта;

`PREEMPT` — сменить «владельца» устройства;

`PRIORITY` — изменить приоритет транзакта;

`QUEUE` — увеличить длину очереди;

`READ` — прочитать следующую строку из потока данных;

`RELEASE` — освободить устройство;

`REMOVE` — удалить элемент из числовой или транзактной группы;

`RETURN` — освободить устройство от транзакта, если оно занято через `PREEMPT`;

`SAVAIL` — снять запрет на вход транзактов в память;

`SAVEVALUE` — изменить сохраняемое значение;

`SCAN` — проверить группу транзактов и записать число членов группы в параметр
активного транзакта;

`SEEK` — сменить указатель строки в потоке данных;

`SEIZE` — занять устройство или ждать этой возможности;

`SELECT` — записать в параметр номер выбранного объекта;

`SPLIT` — создать копии транзакта;

`SUNAVAIL` — запретить вход транзактов в память;

`TABULATE` — изменить содержимое таблицы;

`TERMINATE` — вывести транзакт из модели, уменьшить счетчик завершений карты `START`;

`TEST` — проверить арифметическое условие и изменить маршрут транзактов;

`TRACE` —- начать трассировку активного транзакта;

`TRANSFER` — перейти к указанному блоку;

`UNLINK` — вывести транзакт из цепи пользователя;

`UNTRACE` — закончить трассировку активного транзакта;

`WRITE` —- направить значение в поток данных.

12.2 Структура операторов

12.2.1 Формат

Оператор *GPSS* должен целиком[1] размещаться в одной строке длиной до 250 символов.

Блоки и команды разбиваются на *поля*. Максимальный набор полей состоит из номера строки, метки, операции, операндов (в зависимости от типа операции), комментария. Обязательно заполнение только

[1] Исключение — команда описания таблично заданной функции.

поля операции и некоторых (в зависимости от операции) полей операндов. Операторы записываются в свободном формате — в том смысле, что не нужно беспокоиться о соблюдении номеров колонок при их наборе. Каждое поле оператора завершается ограничителем — пробелом, запятой или (перед комментарием) точкой с запятой.

Номера строк в *GPSS/W* обычно не используются и оставлены для совместимости с предыдущими версиями. Если они применяются, то должны начинаться с первой колонки. В сообщении системы об ошибке указывается просто порядковый номер строки. Для поиска строки с ошибкой следует пользоваться командой `Search/Next Error`.

12.2.2 Поле метки

Поле метки позволяет давать объектам символические имена и соответственно на них ссылаться. Имена не могут быть числовыми. Предпочтительно использовать мнемонические имена. При ссылке на символическое имя последнему должен предшествовать знак $.

Компилятор *GPSS* для внутреннего употребления присваивает вновь встреченным именам операторов последовательные целочисленные значения, начиная с 10000. С помощью оператора `EQU` числовые значения можно присвоить именам произвольно. Сущности (блоки) каждого вида нумеруются начиная с 1. Компилятор составляет отдельно списки заданных числовых и символических имен при вводе модели. *Номер* элемента включается в список лишь при его появлении в определительном контексте (в первом приближении это поле имени определяющей команды либо операнд `A` блоков `SEIZE`, `RELEASE`, `ENTER`, `LEAVE`, `QUEUE`, `DEPART`). Эти номера считаются занятыми; символическим именам назначаются числовые эквиваленты из свободных номеров. Одно и то же имя может соотноситься с объектами различных типов. Соответственно ссылка на него будет иметь контекстно обусловленное значение.

12.2.3 Поле операций

Поле операций содержит названия (типы) блоков или команд из приведенных выше списков.

12.2.4 Поле операндов

Процесс моделирования может управляться стандартными числовыми атрибутами модели, определяющими ее текущее состояние. Например, транзакт может направляться в тот из альтернативных блоков, в который было наименьшее число входов, или в имеющий минимальную очередь. Интервал между заявками может зависеть от числа находящихся в системе транзактов, скорость обслуживания — от времени суток. Подобные сведения сообщаются интерпретатору через операнды блоков.

Набор операндов и смысл заносимой в них информации зависит от вида операции; не все поля обязательны для заполнения. Все аргументы блоков являются позиционными. Поля операндов (в том числе пустые, исключая завершающие оператор) должны разделяться запятыми. При наборе команды (блока) рекомендуется использовать контекстную помощь.

12.2.5 Комментарии

Комментарии должны пояснять назначение и условия применения программы и *смысл* отдельных операторов и групп операторов. Они необходимы автору программы и в особенности лицам, которым программа будет передана для использования и сопровождения.

Вся запись рассматривается как комментарий, если начинается с точки с запятой или звездочки. В противном случае поле комментариев команды (блока) следует за полем операндов любого оператора, кроме определителя функции. Оно начинается точкой с запятой. Напомним, что кириллические символы в *GPSS*-программах недопустимы даже в комментариях.

Комментарии внутри *PLUS*-процедур обсуждаются ниже.

12.3 Алфавит и элементарные конструкции

Алфавит языка *GPSS* состоит из латинских букв, цифр и специальных символов. Строчные и заглавные буквы различаются только в строковых константах и комментариях. Рекомендуется служебные слова набирать

заглавными буквами, а индивидуальные имена объектов — строчными, начиная с заглавной.

Специальные символы — это `#, ^, *, &, +, -, /, @, \, ;` .

Типы значений — целые, вещественные с плавающей точкой (в формате двойного слова), строковые. Логические значения «истина» и «ложь» представляются единицей и нулем соответственно. Необходимые преобразования типов выполняются автоматически.

Целые задаются в формате 32-разрядного слова (до 2147483647 по модулю) и при переполнении автоматически переводятся в вещественный формат с плавающей точкой (15 десятичных цифр, модуль десятичного порядка до 308). Вещественные числа могут записываться как с фиксированной точкой, так и в полулогарифмической форме. В целую форму они переводятся функцией `INT()`.

Имена (метки) присваиваются сущностям $GPSS$ для их взаимных ссылок. Имя начинается с буквы, может занимать до 250 латинских букв или цифр и подчеркиваний и не должно совпадать с названиями СЧА, командами или ключевыми словами. «Безопасными» в указанном смысле будут имена, начинающиеся с трех букв с последующими цифрами, а также содержащие символ подчеркивания. Как и в любом языке программирования, предпочтительно использовать *значимые* имена, связывающие объекты $GPSS$ с элементами исходной задачи.

Имена обычно воспринимаются как глобальные, то есть действующие во всей модели. Исключением могут быть внутренние объекты $PLUS$-процедур. Внутри Эксперимента результаты следует помещать в глобальные переменные и матрицы. В частности, это необходимо для работы процедуры `ANOVA`.

Сохраняемые значения могут быть скалярными и матричными. В последнем случае командой `MATRIX` предварительно определяются имя и структура (экстенты) каждой матрицы. Сохраняемые значения изменяются через блоки `SAVEVALUE` (`MSAVEVALUE`), команды `EQU`, $PLUS$-оператор присваивания (внутри процедуры) или в процессе численного интегрирования по команде `INTEGRATE`. Все они автоматически инициализируются нулями; ненулевые элементы данных должны означиваться (каждый в отдельности) командами `INITIAL` или `EQU`.

Строки — массивы ASCII-символов, заключенные в двойные кавычки. Длина строки не ограничивается. Строки используются при указании в команде `INCLUDE` или `REPORT` спецификации файла, а также для записи информации в отчет. Имеется обширный набор операций для работы со строками — см. разд. 12.7.

12.4 Стандартные числовые атрибуты

Текущее состояние модели *GPSS* характеризуется набором СЧА — как относящихся к системе в целом, так и приписанных к ее отдельным сущностям. Общими для модели являются атрибуты системного времени:

`AC1` — абсолютное системное время (таймер),

`C1` — относительное системное время (с последнего `RESET`),

`TG1` — остаток счетчика завершений.

Набор СЧА будет обсуждаться ниже применительно к отдельным группам блоков. СЧА могут иметь вещественные, целые, логические и строковые значения. Тип значения в случае его очевидности не указывается. Все временны́е и усредненные показатели считаются вещественными.

Большинство СЧА можно обозначить одним из нескольких способов, начиная с класса `W` выбранного атрибута:

`Wj` — по номеру ($j > 0$ целое),

`W$<имя>` — по имени,

`W*j` — через параметр j активного транзакта, который содержит нужный номер (косвенная адресация),

`W*$<имя>` — через имя такого параметра (тоже косвенная адресация).

Здесь и далее `j` используется для представления номера или имени. Используемое в электронной документации обозначение `W*parameter` означает, что можно использовать любой из перечисленных способов. Сразу же укажем «классовые» обозначения переменных (`V` для арифметических и `BV` — для булевых) и сохраняемых значений (`X` для скалярных, `MX` — для матричных). Некоторые классы СЧА (`AC1, C1, M1, MP, PR` и `TG1`) являются атомарными и не включают в свое название спецификатор объекта.

Косвенная адресация реализуется с помощью блока `ASSIGN` через запись номера объекта в параметре транзакта. После этого, например, СЧА `Q*2` будет означать длину очереди, номер которой содержится во втором параметре транзакта. Матричный объект может содержать до

трех косвенных адресов (номера матрицы, строки и столбца для элемента).

СЧА поддерживаются интерпретатором автоматически. *Активный* (и только активный) транзакт имеет следующие СЧА:

`XN1` — номер,

`A1` — группа принадлежности,

`PR` — приоритет,

`Pj` — значение указанного параметра,

`M1` — транзитное время (таймер минус отметка времени),

`MPj` — таймер минус значение параметра.

Состояние многих системных сущностей можно охарактеризовать логическими значениями со стандартизованными именами. Перечислим условия их истинности:

`LSj` — логический ключ j включен;

`FVj/FNVj` — устройство j доступно/недоступно;

`Uj/NUj` — устройство j занято/не занято;

`Ij/NIj` — устройство j в состоянии прерывания (отдано транзакту с более высоким приоритетом)/ не прервано;

`SEj/SNEj` — память j пуста/не пуста;

`SFj/SNFj` — память j заполнена/не заполнена;

`SVj/SNVj` — память j доступна/недоступна.

12.5 Переменные и выражения

Выражения в *GPSS/W* могут использоваться в *PLUS*-процедурах и как операнды блоков и команд. В последнем случае они должны быть заключены в круглые скобки — не всегда обязательные, но со скобками

надежнее. Все они могут ссылаться на информацию о текущем состоянии модели (СЧА).

Переменные основной программы описываются командами `VARIABLE` (логические — `BVARIABLE`) и задают именованные правила вычисления выражений соответствующего класса при текущих (на момент ссылки) значениях термов этих выражений.

Выражение есть комбинация математических операторов, указателей встроенных и пользовательских функций, СЧА, числовых констант, переменных и сохраняемых значений модели. Координаты элементов матрицы в основном тексте указываются в круглых скобках [2]. Выражение основного текста, помещенное в круглые скобки, формально считается $PLUS$-выражением.

При записи выражений используются следующие знаки операций (в порядке убывания приоритета):

`^` — возведение в степень;

`#` — умножение, `/` — деление, `\` — деление нацело;

`@` — вычет по модулю;

`+` , `-` — сложение и вычитание;

`>=`, `'GE'` — больше либо равно; `<=`, `'LE'` — меньше либо равно; `>`, `'G'` — больше; `<`, `'L'` — меньше; `=`, `'E'` — равно; `!=`, `'NE'` — не равно;

`&`, `'AND'` — логическое умножение; `|`, `'OR'` — логическое сложение; `'NOT'` — отрицание.

Операция логической эквивалентности в связи с числовым представлением логических значений заменяется проверкой арифметического равенства.

«Двусмысленность» операций равенства и неравенства нарушает привычное старшинство операций и повышает актуальность использования круглых скобок для указания порядка действий — в особенности если учесть, что операция `'NOT'` в электронной документации к системе не упоминается вообще. Пример:

---

[2] Внутри $PLUS$-процедур и выражений индексы заключаются в квадратные скобки.

```
Trinc BVARIABLE (X$Bool1 'AND' ('NOT' (P1>5)))
```

Из математических функций разрешены ABS, ATN (арктангенс), COS, SIN, TAN, EXP, LOG (натуральный), SQR (корень), INT — выделение целого числа. Аргументы математических функций должны быть заключены в скобки. Все углы предполагаются в радианной мере.

Вычисление выражения может привести к переполнению или потере значащих цифр. При потере значимости результату присваивается нулевое значение и вычисления продолжаются. Останов происходит по переполнению, при попытке извлечь корень из отрицательного числа и т. п.

При выдаче на печать зарегистрированной статистики, если числа в отчете не вмещаются в предписанный формат, производится автоматическая корректировка последнего.

12.6 PLUS и процедуры

GPSS/W обогащена средствами языка *PLUS*, который предоставил программисту практически неограниченные возможности обработки результатов моделирования. В *PLUS* входят богатые библиотеки функций и вероятностных распределений; заключенные в скобки выражения *PLUS* могут быть аргументами блоков *GPSS*. В его терминах можно составлять (и затем использовать) вычислительные процедуры внутри модели. Наконец, в нем появились «надмодельные» конструкции класса EXPERIMENT, позволяющие автоматизировать целевое использование отлаженной модели.

12.6.1 Структура и вызов процедуры

Процедура языка *PLUS* в терминологии классического программирования всегда является *процедурой-функцией*. При благополучном завершении компиляции компилятор сообщает о *регистрации* каждой из процедур. Вложенные процедуры недопустимы.

Описание процедуры помещается в начале или в конце текста модели. Оно состоит из заголовка:

```
PROCEDURE <имя>(<список аргументов>)
```

и тела процедуры. Скобки обязательны, но список аргументов может быть пустым.

Тело процедуры в общем случае оформляется как составной оператор, внутри которого при необходимости объявляются локальные объекты. Типы данных в языке *PLUS* явно не назначаются и распознаются по контексту. Необходимые преобразования типов (в частности, строки в число и числа в строку) выполняются автоматически, округление до целых выполняется функцией `INT(.)`.

После описателя `TEMPORARY` может следовать список *скалярных* локальных переменных. Синтаксически имена должны удовлетворять стандартным соглашениям *GPSS* и быть по крайней мере трехбуквенными. Нарушение этого требования компилятором диагностируется неадекватно, что вызывает затруднения при отладке. Необъявленные объекты рассматриваются как глобальные для модели в целом. С другой стороны, внутри процедур можно ссылаться на СЧА и объекты, явно (матрицы, памяти, сохраняемые значения) или неявно определенные в модели. Приставки «`MX$`» к именам матриц в этом случае не применяются.

Каждая *локальная матрица* объявляется отдельным оператором

`TEMPORARY MATRIX <имя> (<список верхних границ>)`

Все границы должны быть положительными целыми константами (динамические массивы исключены), нижние границы предполагаются единичными, размерность матриц допускается до 6 включительно. Тип значений распознается по контексту. Локальные объекты не имеют смысла вне процедуры и, следовательно, не могут использоваться для представления конечных результатов ее выполнения. *Локальные объекты система не инициализирует.*

Далее следуют «выполняемые» операторы, определяющие действия процедуры над глобальными (СЧА, переменные, матрицы и сохраняемые значения модели) и локальными объектами. Тело процедуры может заканчиваться оператором возврата

`RETURN <выражение>;`

который присваивает имени процедуры значение выражения. При отсутствии выражения возвращается нуль. Операторов возврата может быть несколько, и выход на любой из них завершает выполнение процедуры. Отсутствие `RETURN` предполагает, что действие процедуры сводится к изменению глобальных объектов. Тогда выполнение процедуры завершается при выходе на закрывающую операторную скобку `END`.

Вызов процедуры производится аналогично вызову встроеннных функций, то есть по ее указателю:

```
<имя процедуры>(<список аргументов>)
```

Фактическими аргументами могут быть выражения; тогда они вычисляются в начале вызова. Вызов процедуры без RETURN делается через фиктивно используемый указатель, возвращающий нулевое значение.

Описание конструкции EXPERIMENT и работы с ней приводится в разд. 14.5, 15.18.

12.6.2 Подоператоры процедуры

Операторы *PLUS* могут использоваться только внутри *PLUS*-процедур. С другой стороны, в процедурах не могут использоваться блоки и команды *GPSS*.

Операторы записываются в свободном формате (рекомендуется выделение вложенных конструкций отступами), могут занимать несколько строк и должны заканчиваться точкой с запятой. В одной строке можно разместить несколько операторов. Для ссылок на операторы они могут иметь символьные метки (имена), присоединяемые через двоеточие. В текстах процедур допускаются комментарии (в том числе многострочные), ограничиваемые комбинациями /* и */ — как в *ПЛ/1*.

Оператор присваивания записывается в формате

```
<имя> = <выражение>;
```

После вычисления выражения тип полученного значения автоматически приводится к типу левой части.

Оператор перехода имеет вид

```
GOTO <метка>;
```

и вызывает переход к оператору с указанной именной меткой. Разрешены только
локальные переходы — в пределах данной процедуры.

Составной оператор необходим там, где по синтаксису языка требуется единственный оператор, а фактически приходится записать несколько. Его структура

```
BEGIN <подоператоры> END;
```

Составными операторами часто описывают подоператоры условных операторов, тела циклов и процедур.

Условный оператор имеет конструкцию

```
IF (<выражение>) THEN <оператор>;
                 ELSE <оператор>;
```

ELSE-часть может отсутствовать. Условие обязательно заключается в круглые скобки. Каждая ELSE-часть соотносится с ближайшим IF.

Оператор цикла в языке $PLUS$ существует только одного вида:

```
WHILE (<выражение>) DO <оператор>;
```

Это наиболее общая форма цикла с верхним окончанием (проверка предшествует первому выполнению тела цикла). Оператор может быть составным. Цикл выполняется, пока выражение остается истинным. Программист должен позаботиться о предупреждении зацикливания и о задании начальных значений для используемых в теле цикла и его условии переменных.

Часто требуемый цикл по счетчику — очевидный частный случай. Покажем, как организовать накопление последовательных степеней времени ожидания транзакта:

```
Indx=1; Power=1.0;
WHILE (Indx<5) DO
   BEGIN
      Power=Power#Wait;
      TimePow[1,Indx]=TimePow[1,Indx]+Power; Indx=Indx+1;
   END;
```

Массив TimePow следует объявить в модели как глобальную матрицу с единственной строкой.

12.7 Библиотеки процедур

Библиотеки могут быть двух типов: пользователя и встроенная $GPSS/W$. Библиотека пользователя не имеет специальных средств организации и хранится в рабочей или специально выделенной директории как совокупность файлов. Если процедура используется в нескольких моделях, ее не обязательно каждый раз набирать заново: достаточно *включить* ее в модель оператором

```
INCLUDE "<спецификация_файла>"
```

Если в спецификации не указан путь, файл разыскивается в той же директории, где находится включающая модель. Предложение INCLUDE не может находиться внутри $PLUS$-процедуры. В процессе компиляции INCLUDE заменяется содержимым файла, которое затем транслируется.

В упомянутом файле могут находиться описания нескольких процедур, но выборочная вставка из него (с указанием имен нужных процедур) невозможна. При вставках допускается вложенность до пятикратной. Отладку вставляемых процедур надо выполнить предварительно. Средства присоединения к модели *ранее транслированных* процедур в *GPSS/W* отсутствуют, но это обстоятельство, учитывая быстроту компиляции, затруднений не вызывает.

Встроенная библиотека всегда готова к употреблению и содержит: утилиты; запросные, математические и строковые процедуры; процедуры генерации случайных чисел в соответствии с 24 теоретическими законами распределения.

К *утилитам* относится процедура `DoCommand`, обсуждаемая в разделе об организации многократных прогонов — Экспериментов (это слово как термин *GPSS/W* мы будем писать с большой буквы). *Общематематические* процедуры обсуждались выше.

12.7.1 Запросные процедуры

Запросные процедуры дают информацию о транзакте с номером `TrNumb`:

`QueryXNExist(TrNumb)|` — 1, если такой транзакт существует, и 0 — иначе;

`QueryXNParameter(TrNumb)|` — значение параметра;

`QueryXNAssemblySet(TrNumb)|` — номер ансамбля;

`QueryXNPriority(TrNumb)|` — приоритет транзакта;

`QueryXNM1(TrNumb)|` — отметку времени транзакта.

12.7.2 Строковые процедуры

Строки суть последовательности ASCII-символов, заключенные в двойные кавычки. В частности, допустима пустая строка. Отдельные символы строки указываются их индексами (начальный элемент имеет индекс 1, позиции нумеруются слева направо). *Словом* считается часть строки, отделенная пробелами и, возможно, символами табуляции. Над строками допускаются следующие операции:

`Length(SourceStr)` — дает длину строки;

`Trim(SourceStr)` — обрезает начальные и хвостовые пробелы;

`Datatype(<данное>)` — выдает строку, означающую тип аргумента;

`Value(SourceStr)` — дает числовой эквивалент строки;

`String(<данное>)` — переводит данное в строковый эквивалент;

`StringCompare(Str1,Str2)` — сравнивает строки лексикографически по ASCII-кодам символов. Если предшествует первая, результат будет -1, если вторая — то 1, если равны — нуль; если до принятия решения одна из строк исчерпалась, предшествующей считается она.

`Lowercase(SourceStr)` — переводит символы строки в нижний регистр;

`Uppercase(SourceStr)` — переводит алфавитные символы в верхний регистр;

`Left(SourceStr,MaxCount)` — дает подстроку первого операнда заданной длины, считая слева;

`Right(SourceStr,MaxCount)` — дает подстроку первого операнда заданной длины, считая справа;

`Find(SubStr,SourceStr)` — указывает начальную позицию первого операнда, если он входит во второй;

`Substring(SourceStr,Offset,MaxCount)` — выделяет подстроку заданной счетчиком длины, начиная с указанной позиции;

`Word(SourceStr,WordNumber)` — выделяет из строки слово с указанным номером;

`Align(InsStr,SourceStr,Offset` — заменяет первым операндом часть второго, смещение задает самую правую позицию замены;

`Place(InsStr,SourceStr,Offset)` — заменяет первым операндом часть второго, смещение задает самую левую позицию замены;

`Catenate(Str1,Str2)` — склеивает операнды;

`Polycatenate(Str1,...)` — склеивает произвольное число строк в одну;

`Copies(SourceStr,Count)` — склеивает заданное число копий первого операнда.

К сожалению, операции сдвига строки отсутствуют.

12.8 Распределения вероятностей

Здесь рассматриваются $PLUS$-процедуры генерации случайных чисел с типовыми теоретическими распределениями. В связи с тем, что форма задания и даже число параметров (с учетом сдвига) часто отличаются от рассмотренных в главе 2 стандартных, по обсуждаемым распределениям дается вся приведенная в электронной документации $GPSS/W$ информация. Вызов каждой функции оформлен как оператор присваивания с указанием в его левой части типа возвращаемого значения.

Имеются 24 вероятностных распределения:

бета	обратное гауссово	нормальное
биномиальное	обратное Вейбулла	Парето
дискр. равномерное	Лапласа	Пирсона V
показательное	логистическое	Пирсона VI
крайних значений А	логарифм. Лапласа	Пуассона
крайних значений Б	логарифм. логистическое	треугольное
гамма	логарифм. нормальное	равномерное
геометрическое	отрицат. биномиальное	Вейбулла

В этом списке нет распределения Эрланга. Разумеется, оно может быть получено как частный случай гамма-распределения, но с большей трудоемкостью, чем через суммирование показательно распределенных величин.

Все процедуры вызываются по их указателю. Если требуется задать исходный ДСЧ, подставляется только *номер* последнего — без предшествующего `RN`. Во всех случаях через `Min` и `Max` обозначены соответственно наименьшее и наибольшее значения генерируемой случайной величины, m — смещение, s — масштабный параметр (оба положительные), α и β — параметры формы, которые в операторах вызова заменяются их латинскими аналогами. Нулевые значения плотностей и вероятностей

вне указанной области их положительных значений дополнительно не оговариваются. Попутно исправлены обнаруженные ошибки в выражении для дисперсии Γ- и пуассонова распределений, упрощены выражения для логарифмически нормального, логарифмически лапласова и Пирсона V типа распределений.

12.8.1 Непрерывные распределения

Равномерное распределение формируется как

`Real=UNIFORM(a,b)`

Его плотность на интервале $[a,b]$ равна $1/(b-a)$, среднее $(\ +b)/2$, а дисперсия $(b-a)^2/12$.

Треугольное распределение можно получить оператором

`Real=TRIANGULAR(RNj,min,max,mode)`

В обозначениях $\min = a$, $max = b$, `mode=c` плотность распределения

$$f(x) = \begin{cases} \dfrac{2(b-x)}{(b-a)(c-a)}, & a \le x \le c, \\ \dfrac{2(b-x)}{(b-a)(b-c)}, & c \le x \le b. \end{cases}$$

Среднее равно $(a+b+c)/3$, а дисперсия $D = (a^2+b^2+c^2-ab-ac-bc)/18$.

Распределение Парето получается оператором

`Real=PARETO(RNj,m,s)`

и имеет плотность

$$f(x) = sm^s x^{-(s+1)}, \quad x \ge m.$$

Его среднее $v = sm/(s-1)$ существует при $s > 1$. Дисперсия $D = sm^2/(s-1)^2(s-2)$ конечна при $s > 2$.

Показательное распределение вызывается по

`Real=Exponential(RNj,m,s)`

Здесь m — смещение распределения, а s — масштабный параметр. Плотность распределения

$$f(x) = \frac{1}{s} e^{-(x-m)/s}, \quad x \ge m.$$

Среднее $v = m + s$, а дисперсия $D = s^2$.

Распределение Лапласа вызывается оператором

```
Real=LAPLACE(RNj,m,s)
```

и имеет плотность

$$f(x) = \frac{1}{2s} e^{-|x-m|/s}.$$

Его среднее равно m, а дисперсия $D = 2s^2$.

Логарифмическое распределение Лапласа используется в форме

```
Real=LOGLAPLACE(RNj,m,s,a)
```

и дает распределение с плотностью

$$f(x) = 0.5 \exp[-\alpha |\ln((x-m)/s)|] = 0.5 \ln \left| \frac{s}{x-m} \right|^{\alpha}.$$

Логистическое распределение генерируется согласно

```
Real=LOGISTIC(RNj,m,s)
```

и задается плотностью

$$f(x) = \frac{e^{(x-m)/s}}{s(1 + e^{(x-m)/s})^2}.$$

Среднее равно m, дисперсия $\pi s^2/3$.

Логарифмически логистическое распределение генерируется по

```
Real=LOGLOGIS(RNJ,m,s,a)
```

и описывается плотностью

$$f(x) = -1/\exp[\alpha \ln((x-m)/s) + 1].$$

Гамма-распределение можно вызвать оператором

```
Real=GAMMA(RNj,m,s,a)
```

Плотность распределения

$$f(x) = \frac{s^{-\alpha}(x-m)^{\alpha-1}}{\Gamma(\alpha)} e^{-(x-m)/s}, \quad x \geq m.$$

Среднее $v = \alpha s$, дисперсия $D = \alpha s^2$.

При $\alpha = 1$ распределение сводится к показательному, при целых положительных α — к эрлангову соответствующего порядка.

Бета-распределение. Вид обращения:

`Real=BETA(RNj,min,max,a,b)`

Аргументы `a,b` — положительные вещественные параметры. Плотность распределения в обозначениях $\lambda_1 = \min$ и $\lambda_2 = \max$ имеет вид

$$f(x) = \frac{(x-\lambda_1)^{\alpha-1}(1-(x-\lambda_1))^{\beta-1}}{(\lambda_2-\lambda_1)^{\alpha+\beta-1}\mathrm{B}(\alpha,\beta)}, \quad \lambda_1 \le x\lambda_2.$$

Здесь

$$\mathrm{B}(\alpha,\beta) = \int\limits_0^1 t^{\alpha-1}(1-t)^{\beta-1}\,dt$$

есть бета-функция. Среднее значение и дисперсия распределения

$$\begin{aligned} v &= \lambda_1 + (\lambda_2-\lambda_1)\frac{\alpha}{\alpha+\beta}, \\ D &= (\lambda_2-\lambda_1)^2\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}. \end{aligned}$$

При $\alpha = \beta = 1$ распределение вырождается в равномерное. При $\alpha = 1$, $\beta = 2$ оно обращается в левое треугольное, при обратном соотношении — в правое треугольное.

Распределение Пирсона V типа вызывается по

`Real=PEARSON5(RNj,m,s,a)`

и имеет плотность

$$f(x) = \left(\frac{s}{x-m}\right)^{\alpha+1}\frac{e^{-s/(x-m)}}{s\Gamma(\alpha)}, \quad x \ge m.$$

Его среднее $s/(\alpha-1)+m$ существует при $\alpha > 1$. Дисперсия

$$D = \frac{s^2}{(\alpha-1)^2(\alpha-2)}$$

конечна, если $\alpha > 2$.

Распределение Пирсона VI типа вызывается по

```
Real=PEARSON6(RNj,m,s,a,b)
```

и имеет плотность

$$f(x) = \frac{((x-m)/s)^{\alpha-1}}{s\mathrm{B}(\alpha,\beta)[1+(x-m)/s]^{\alpha+\beta}}, \quad x \geq m.$$

Его среднее $v = s\alpha/(\beta-1)+m$ существует при $\beta > 1$. Дисперсия

$$D = \frac{s^2\alpha(\alpha+\beta-1)}{(\beta-1)^2(\beta-2)}$$

конечна, если $\beta > 2$.

Нормальное распределение (оно же гауссово) генерируется посредством

```
Real=NORMAL(RNj,m,s)
```

и имеет плотность

$$f(x) = \frac{1}{s\sqrt{2\pi}} e^{-(x-m)^2/2s^2}.$$

Здесь среднее $v = m$ и $D = s^2$.

Логарифмически нормальное используется в конструкции

```
Real=LOGNORMAL(RNj,m,s,a)
```

и имеет плотность

$$\frac{1}{(x-m)a\sqrt{2\pi}} \exp\left[-\frac{(\ln(x-m)-s)^2}{2a^2}\right], \quad x \geq m.$$

Его среднее значение есть $v = \exp(s+a^2/2)+m$, а дисперсия

$$D = e^{2s+a^2}(e^{a^2}-1).$$

Параметры распределения выражаются через среднее и дисперсию формулами

$$\begin{aligned} s &= \frac{1}{2}\ln\frac{(v-m)^4}{D+(v-m)^2}, \\ a^2 &= 2\ln(v-m)-2s. \end{aligned}$$

Обратное гауссово распределение получается по схеме

```
Real=INVGAUSS(RNj,m,s,a)
```

и имеет плотность

$$f(x) = \sqrt{\frac{\alpha}{2\pi(x-m)^3}} \exp\left[-\frac{\alpha(x-m-s)^2}{2s^2(x-m)}\right], \quad x \geq m.$$

A-распределение крайних значений вызывается оператором

```
Real=EXTVALA(RNj,m,s)
```

В прежних обозначениях плотность распределения

$$f(x) = \exp[-e^{(x-m)/s}].$$

B-распределение крайних значений вызывается оператором

```
Real=EXTVALB(RNj,m,s)
```

В тех же обозначениях плотность распределения

$$f(x) = \exp[-e^{-(x-m)/s}].$$

Распределение Вейбулла вызывается конструкцией

```
Real=Weibull(RNj,m,s,a)
```

Оно описывается плотностью распределения

$$f(x) = \alpha s^{-\alpha}(x-m)^{\alpha-1} \exp[-((x-m)/s)^{\alpha}], \quad x \geq m.$$

Среднее и дисперсия этого распределения равны соответственно

$$\begin{aligned} v &= \frac{s}{\alpha}\Gamma(1/\alpha) + m, \\ D &= \frac{s^2}{\alpha}[2\Gamma(2/\alpha) - \frac{1}{\alpha}\Gamma^2(1/\alpha)]. \end{aligned}$$

Распределение Вейбулла при $\alpha = 1$ превращается в показательное, а при $\alpha = 2$ — в распределение Релея.

Обратное распределение Вейбулла вызывается согласно

```
Real=INVWEIBULL(RNj,m,s,a)
```

и имеет плотность

$$f(x) = \exp[-e^{-\alpha \ln(x-m)/s}] \quad x \geq m.$$

12.8.2 Дискретные распределения

Дискретное равномерное распределение вызывается оператором

`Integer=DUNIFORM(RNj,min,max)`

Вероятности

$$p(x) = 1/(\max - \min + 1), \quad x \in \{\min, \min + 1, \ldots, \max\}.$$

Среднее $v = (\min + \max)/2$, дисперсия $D = [(\max - \min + 1)^2 - 1]/12$.

Геометрическое распределение вызывают по схеме

`Integer=GEOMETRIC(RNj,p)`

где p — вероятность успеха при одной попытке. Вероятность дискретного значения x

$$p(x) = p(1-p)^x, \quad x = 0, 1, \ldots$$

Среднее $v = (1-p)/p$, дисперсия $D = (1-p)/p^2$.

Биномиальное распределение вызывается оператором

`Integer=Binomial(RNj,t,p)`

Аргументы:

`t` — счетчик бернуллиевых попыток,

`p` — вероятность успеха в каждой.

Вероятности этого дискретного распределения

$$p(x) = \frac{t\,!}{x!\,(t-x)!} p^x (1-p)^{t-x}, \qquad x \in \{0, 1, \ldots, t\}.$$

Среднее $v = tp$, а дисперсия $D = tp(1-p)$.

Отрицательное биномиальное распределение вызывается по схеме

`Integer=NEGBINOM(RNj,c,p)`

где `c` — требуемое число успешных бернуллиевых попыток, а `p` — вероятность успеха в попытке. Вероятности этого распределения

$$f(x) = \frac{(c+x-1)!}{x!\,(c-1)!} p^c (1-p)^x, \quad x = 0, 1, \ldots$$

Среднее значение $v = c(1-p)/p$, а дисперсия $D = c(1-p)/p^2$. При $= 1$ распределение вырождается в геометрическое.

Распределение Пуассона генерируется посредством

`Integer=POISSON(RNj,v)`

где `v` — среднее значение. Вероятности значений

$$p(x) = \frac{v^x}{x\,!}e^{-v}, \quad x = 0, 1, \ldots$$

Дисперсия распределения равна v^2.

Глава 13

Функциональные комплексы

Выше основные сущности *GPSS* и принципы их взаимодействия обсуждались последовательно и на концептуальном уровне — с отвлечением от ненужных для первого знакомства подробностей. *Детальную* информацию об СЧА, командах и блоках для работы с ними имеет смысл привести в иной последовательности. Удобно сгруппировать ее по общности абстрактных применений — функциональным комплексам, обеспечивающим определенный класс имитируемых действий. Таких комплексов можно выделить семь: траекторный, атрибутный, ансамблевый, ресурсный, цепной, управления, ввода-вывода. Для каждого из них перечисляются СЧА, команды и блоки и даются графические обозначения последних. Затем даются комментарии к операндам блоков и (при необходимости) варианты совместного использования операндов.

13.1 Траекторный комплекс

СЧА активного транзакта — это `XN1` (номер), `M1` — транзитное время (разность между абсолютным модельным временем и временем входа транзакта в модель), `PR` — приоритет, `Pj` — значение j-параметра. Атрибут `Wj` представляет текущее число транзактов в блоке `j`.

К блокам этой группы относятся блоки:

- порождающие и уничтожающие транзакты `GENERATE` и `TERMINATE`;

- изменяющие траекторию `TRANSFER`, `LOOP`, `TEST`, `GATE`, `DISPLACE`, `EXECUTE`;

- определяющий временные задержки `ADVANCE`.

На рис. 13.1 показаны их графические обозначения.

Рис. 13.1. Траекторные блоки

GENERATE A,B,C,D,E

Блок создает транзакты для последующего ввода в модель. В его операндах нельзя использовать параметры активного транзакта. Операнды:

A — среднее время между транзактами;

B — модификатор (интервал или функция) времени между транзактами;

C — момент прибытия первого транзакта;

D — количество генерируемых транзактов; при отсутствии D количество транзактов не лимитируется;

E — приоритет (целое число); при отсутствии операнда принимается нулевым.

В начале моделирования вызываются все блоки GENERATE. Первые спланированные ими транзакты помещаются в цепь будущих событий ЦБС.

Перед созданием нового транзакта проверяется операнд D. Если заданный предел генерации еще не превышен, генерация транзактов продолжается. Блок GENERATE создает новый транзакт, назначает ему следующий номер, приоритет по операнду E и время старта транзакта по относительным часам системы.

Время прибытия нового транзакта вычисляется по операндам A, B,C. Если специфицирован только операнд A, то он используется как *приращение* времени. Если заданы A и B, причем B — не функция, то как приращение времени используется равномерно распределенное на $[A - B, A + B]$ случайное число. Источником первичного случайного числа U служит RN1 [1]. Если B есть СЧА класса функций, результат получается умножением операнда A на значение функции. Хотя бы один из операндов A или D должен быть задан обязательно.

Если GENERATE представляет внешний источник, независимый от состояния системы, важно обеспечить вход в следующий блок без блокировок. В этом случае вслед за GENERATE помещается ADVANCE с нулевым временем задержки.

[1] При необходимости работы с другим ДСЧ можно переопределить его на вкладке Random в меню Edit/Settings/Model

С помощью отдельного сегмента, генерирующего *один* транзакт в *нулевой* момент времени, можно однократно выполнить сложную инициализацию модели, заданную $PLUS$-процедурой.

TERMINATE A

Блок удаляет активный транзакт из модели и при заданном операнде A вычитает его значение из счетчика завершения. Например, при входе транзакта в блок

TERMINATE 1

транзакт удаляется из модели, а заданный командой START счетчик завершений (СЧА TG1) уменьшается на 1. Затем для продвижения выбирается новый активный транзакт. После завершения моделирования выдается стандартный отчет.

Если необходимо, чтобы прогон закончился в опеределенный момент системного времени, в модель включается GENERATE-блок, который создает в этот момент единственный транзакт. Последний при необходимости инициирует нестандартную обработку статистики и далее направляется в блок TERMINATE с единицей в поле A. Команда запуска START должна иметь в счетчике 1.

TRANSFER A,B,C,D

Блок изменяет движения активного транзакта. Операнды:

A — режим блока, выбираемый из множества {BOTH, ALL, PICK, FN, P, SBR, SIM}; может быть также дробным числом, именем, константой, СЧА.

B — номер или имя блока (в режиме P — имя параметра).

C — номер или имя блока. В режиме функции (FN) или параметра — приращение.

D — приращение номера блока для режима ALL. По умолчанию равен 1.

Блок TRANSFER может действовать в одном из девяти режимов. При необходимости случайного выбора номер ДСЧ устанавливается на вкладке RANDOM.

Режим безусловного перехода. При отсутствии операнда A активный транзакт переходит к адресу, специфицированному операндом B.

Статистический режим. Если операнд A не является ключевым словом, то транзакт переходит по адресу, определенному операндом C, с вероятностью, заданной операндом A. Если последний — целое число, оно интерпретируется как количество тысячных и преобразуется в дробную вероятность. Альтернативное назначение специфицируется операндом B. Его значение по умолчанию — блок, следующий за TRANSFER. При записи

```
TRANSFER    .75,,NEW_PLACE
```

транзакт следует к блоку с именем NEW_PLACE с вероятностью 0.75, иначе к текстуально очередному блоку.

Режим BOTH. Здесь сначала проверяется блок, заданный операндом B. Если он не может принять активный транзакт, то проверяется указанный в C. Если оба блока откажутся принять транзакт, то он останется в блоке TRANSFER, пока не сможет войти в один из проверяемых блоков.

Режим ALL. Проверяется последовательность блоков между операндами B и C включительно до первого успеха. Адрес каждого следующего проверяемого блока вычисляется прибавлением операнда D (по умолчанию 1) к адресу предыдущего блока. Если операнд C не используется, проверяется только один блок. Если ни один из блоков не сможет принять транзакт, он остается в текущем блоке.

Режим PICK выбирает дальнейшее назначение транзакта в интервале от B до C (последовательные номера!) с равными вероятностями *независимо от возможной блокировки.*

Режим функции действует, когда операнд A есть FN. Новое назначение выбирается вычислением заданной операндом B целочисленной функции, которая затем складывается с необязательным смещением из операнда C.

Режим параметра задает операнд A=P. Активный транзакт переходит по адресу, который равен сумме значения параметра и операнда C. Имя или номер параметра указаны операндом B.

Режим подпрограммы реализуется при A=SBR. В этом случае активный транзакт всегда переходит по адресу, указанному операндом B. Адрес текущего блока помещается в параметр транзакта, указанный операндом C. Эта технология позволяет организовать переход транзакта с возвратом, аналогичный передаче управления на подпрограмму с воз-

вратом на продолжение основной программы. Возврат обеспечивается блоком TRANSFER в параметрической моде.

В *режиме* SIM переход зависит от «индикатора задержки» транзакта, непосредственно не доступного программисту. Этот индикатор включается, когда транзакт получает отказ во вход в какой-либо блок, и может быть выключен блоком TRANSFER. При включенном индикаторе (Set) транзакт переходит по адресу, указанному в C, и индикатор переводится в Reset. В противном случае выполняется переход по адресу, указанному в B. Этот режим является рудиментом ранних версий *GPSS*. Координированную оценку состояния нескольких объектов проще организовать через булеву переменную.

LOOP A,B

Блок LOOP (зациклить) модифицирует параметр и управляет движением активного транзакта, исходя из результата этой модификации. Операнд A содержит ссылку на параметр транзакта, играющий роль счетчика, и значение параметра уменьшается на 1. Если новое значение параметра не равно нулю и операнд B специфицирован, то планируется переход транзакта по адресу, указанному в B. Иначе транзакт следует к следующему блоку. Таким образом, блок LOOP можно использовать для задания циклических маршрутов [2] с известным числом повторений. Циклы получаются с отрицательным шагом и нижним окончанием (обязательным прохождением тела цикла). Допускаются вложенные циклы.

С помощью блока LOOP можно моделировать обслуживание заявок с повторными вызовами и ограничением числа попыток. Другие варианты схемы повторных вызовов (повторение до удовлетворения, с вероятностным отказом от повторной попытки) лучше реализовать с помощью блока TRANSFER, проверяющего занятость устройства (памяти). Для реализации последнего варианта ставится дополнительный TRANSFER в «статистическом» режиме. Этот вариант можно сочетать с охватывающим циклом по LOOP для ограничения числа повторений. Во всех случаях внутри цикла должен быть дополнительный блок ADVANCE для имитации интервалов между попытками.

[2] Не вычислений!

TEST O A,B,C

Блок сравнивает значения указанных СЧА и управляет направлением движения транзакта, основываясь на результате сравнения. Его операнды:

O — проверяемое отношение *числовых* значений операндов A и B (в апострофы не заключается),

A,B — сопоставляемые значения,

C — альтернативный блок назначения.

Поскольку логические значения в *GPSS/W* кодируются нулем и единицей, в блоке TEST можно сопоставлять и логические значения.

При невыполнении условия, если операнд C не используется, TEST действует в *режиме отказа* — блокирует вход активного транзакта, а при заданном C делает переход по этому операнду. Пример:

```
TEST G  C1,70000
```

Здесь активный транзакт входит в блок TEST, если значение таймера C1 больше 70000. В противном случае транзакт поступает в цепи повторных попыток для объектов, участвовавших в сравнении. Когда состояние любого из них изменяется, блокированные транзакты активируются и пытаются снова пройти TEST.

При заданном операнде C реализуется режим *альтернативного выхода*. При неуспешном сравнении транзакт планируется к передаче согласно операнду C и помещается в ЦТС в начало своей приоритетной группы.

Блоки TEST и GATE имеют большие возможности, но неуспешные проверки условия могут вызвать большой расход машинного времени. Для сокращения их частоты можно поместить транзакты в ПЦ с помощью блоков LINK и UNLINK.

GATE O A,B

Блок GATE (впустить) изменяет движение транзакта по состоянию проверяемого объекта — логического ключа, устройства или памяти, *но не очереди*. Состояние всегда имеет логическое значение. Операнды:

`O` — вид проверяемого условия (см. список СЧА). При выполнении условия активный транзакт переходит к следующему блоку.

`A` — имя или номер объекта, который должен быть проверен. Тип элемента должен быть согласован с условием.

`B` — номер блока, к которому направляется транзакт при невыполнении условия.

Если операнд `B` не используется, `GATE` действует в режиме отказа и задерживает транзакт до выполнения условия. В сочетании с памятью `GATE` может использоваться для формирования пачек постоянного объема, равного емкости памяти.

Для разветвления маршрута на два направления обычно используются двухпозиционные логические ключи: замок есть/нет, красный либо зеленый сигнал светофора, табличка «Мест нет», «Ушла на базу», «Перерыв». В случае трехцветного светофора придется использовать блок `TEST` и *числовую* переменную (`VARIABLE`).

DISPLACE A,B,C,D

Блок перемещает транзакт в любой блок с сохранением интервала модельного времени. Его операнды:

`A` — номер перемещаемого транзакта;

`B` — блок назначения;

`C` — параметр перемещаемого транзакта, в котором может быть записано остаточное время пребывания в исходном блоке;

`D` — альтернативное назначение для активного транзакта.

EXECUTE A

Блок вызывает выполнение программы блока `A` текущей модели для активного транзакта.

ADVANCE A,B

Блок задерживает продвижение транзакта на заданный интервал модельного времени. Его операнды:

`A` — средний интервал времени;

`B` — половина временного интервала либо модификатор-функция.

Интервал времени может быть вычислен несколькими путями. Если задан только операнд `A`, то он вычисляется и используется в качестве постоянного интервала времени. Если определены операнды `A` и `B`, но `B` не задает функцию, то вычисляются оба операнда `A` и `B` и формируется равномерно распределенное случайное число $X \in [A - B, A + B]$. Источник (номер генератора) первичных случайных чисел выбирается на вкладке `Random` в меню `View/Settings/Model`.

Если `B` есть СЧА класса `FN`, называемый модификатором-функцией, то значение `B` умножается на результат вычисления `A`; результат используется как интервал времени. При получении отрицательного значения происходит останов.

Длительность задержки в общем случае может зависеть от СЧА модели: интервал между заявками в случае замкнутых систем и сетей — от общего числа транзактов, а время, необходимое для удовлетворения запроса — от типа запасной части (в хорошо организованном хозяйстве редко используемые детали хранятся «в дальнем углу»), текущей длины очереди и т. п.

13.2 Атрибутный комплекс

Здесь объединяются средства задания и изменения *значений*, обрабатываемых моделью.

Атрибуты

`Pj` — текущее значение указанного параметра;

`MPj` — транзитное «параметрическое» время транзакта; вычисляется как разность текущего значения абсолютного времени и значения, записанного в параметр;

`LSj` — состояние логического ключа `j`;

`Xj` — `j`-е сохраняемое значение;

`MXj` — элемент сохраняемой `j`-матрицы (с индексами в круглых скобках);

`Vj` — числовая переменная `j`;

`BVj` — булева переменная `j`;

`RNj` — текущее значение ДСЧ `j`.

Команды

Команда

```
<имя> EQU <выражение>
```

создает именованный объект и присваивает ему определяемое выражением числовое значение. Команда

```
<имя> VARIABLE <выражение>
```

задает правило для получения числового значения (`BVARIABLE` — логического). При ссылке типа `V$<имя>` это выражение вычисляется при текущих значениях входящих в выражение параметров активного транзакта, сохраняемых значений, ДСЧ и т. п.

Матрица представляет собой прямоугольный массив чисел. Размер матрицы должен быть объявлен предварительно в предложении `MATRIX`:

```
<имя> MATRIX ,10,4
```

Операнд `A` не используется, последующие (до шести) задают протяженность по измерениям. При необходимости в *векторе* последний объявляется как двумерная матрица с одной строкой (столбцом). Индексирование элементов по всем измерениям начинается с единицы. Команда `MATRIX` должна предшествовать первой ссылке на эту матрицу.

Оператор `INITIAL` присваивает элементам матрицы начальные значения:

```
INITIAL MX$PPP(1,1),100067
```

Последующие изменения значений делаются блоком `MSAVEVALUE`.

Потоки равномерно распределенных псевдослучайных чисел генерируются посредством мультипликативного генератора с длиной периода

$2^{31} - 2$ (нулевое значение исключается). Начальное значение по умолчанию совпадает с номером соответствующего ДСЧ. Закрепление ДСЧ задается в меню `Model Settings/Random`. Команда

```
RMULT A,B,C,D,E,F,G
```

позволяет задать нестандартные начальные значения первых семи ДСЧ.

Табличную функцию $GPSS$ определяет управляющий оператор `FUNCTION`. Операнд `A` задает аргумент функции (это может быть любой СЧА), операнд `B` — тип функции и количество координат (пар точек аргумент–функция).

Различают пять типов функций:

`C` — непрерывная числовая функция. Значение функции находится методом линейной интерполяции и берется как целое число.

`D` — дискретная числовая функция. Значение функции берется равным значению на правом конце интервала в целочисленном виде.

`E` — расширение понятия дискретной функции; в ней можно использовать любой СЧА, кроме матриц.

`L` — списковая числовая функция, в качестве аргумента которой используется отрезок натурального ряда начиная с 1.

`M` — расширеннное понятие списковой функции; значением функции может быть любой СЧА.

Обращение к функции можно записывать в операнде `B` блоков `ADVANCE` или `GENERATE`. В этом случае она называется «функцией–модификатором». Время задержки в зтих блоках вычисляется как произведение операнда `A` на функцию–модификатор. Точность координат функции ограничивается семью десятичными цифрами. При линейной интерполяции с использованием в качестве аргумента случайных чисел последние автоматически берутся в диапазоне от 0 до 0.999999.

С функциями связан СЧА класса `FNj` — результат вычисления соответствующей функции. Приведем версию программы моделирования системы $M/M/1$ с заимствованной из [74] табличной реализацией непрерывной ДФР показательного закона:

```
************************************
*   M/M/1 MODEL, table function      *
************************************
XPDIS FUNCTION RN1,C24
0,0/0.1,.104/0.2,0.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
.75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
.95,2.99/.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/
.999,7/.9998,8

      GENERATE   1,FN$XPDIS
      QUEUE      Wait
      SEIZE      Chan
      DEPART     Wait
      ADVANCE    0.9, FN$XPDIS
      RELEASE    Chan
      TERMINATE

      GENERATE   500000,,,1
      TERMINATE 1
```

В связи с этой функцией полезно вспомнить, как в английском языке называется буква «X». Отметим также, что она инвариантна к масштабу, и полученное с ее помощью случайное число достаточно умножить на требуемое среднее значение.

Этот пример показывает, насколько *GPSS World* благодаря реализации стандартных распределений облегчил работу пользователя.

Обсудим пример использования *дискретной* функции. Часто транзакты с помощью блока TRANSFER должны быть переданы в одно из нескольких возможных мест. Режим PICK этого блока допускает только равномерное распределение. Иначе в поле A блока TRANSFER помещается FN, а в поле B — имя функции. Если используется и C, оно указывает дополнительное смещение. Пусть точки назначения суть FAC1,FAC2,FAC3,FAC4 с вероятностями 0.2, 0.1, 0.4, 0.3 соответственно. Тогда нас устроит

```
        TRANSFER FN,2
      2 FUNCTION RN1,D4
        .2,FAC1/.3,FAC2/.7,FAC3/1.0,FAC4
```

Аргумент функции — ДСЧ №1; указанные в определении значения получены построением кумулянты дискретного распределения.

Команда

```
<имя> INTEGRATE A,B,C,D,E
```

задает получение указанной именем переменной с помощью численного интегрирования по модельному времени методом Рунге — Кутты — Фельберга (4-го порядка). Операнд A есть выражение для производной. Аналогично может интегрироваться система дифференциальных уравнений первого порядка, разрешенных относительно производных. До начала интегрирования переменным должны быть присвоены начальные значения. В процессе интегрирования происходит чередование непрерывной и дискретной (связанной с изменением состояния модели) фаз процесса, причем момент перехода к дискретной фазе определяется линейной интерполяцией.

Каждая команда INTEGRATE может определять один или два порога B и D, при пересечении которых в любом направлении генерируется дополнительный транзакт нулевого приоритета. Этот транзакт направляется в связанный с данным порогом блок, указанный в команде INTEGRATE (операнды C и E соответственно).

Блоки

Блоки рассматриваемой группы имеют обозначения рис. 13.2.

INTEGRATION A,B

Блок включает или выключает интегрирование переменной пользователя A. Режим (Null, ON, OFF) задается операндом B. По умолчанию интегрирование включено.

SAVEVALUE A,B

Блок изменяет значение ячейки A. Операнд A может сопровождаться знаком «+» или «-» для указания прибавления или вычитания от существующего значения. Операнд B указывает значение, которое должно быть занесено, прибавлено или отнято. В примере

```
SAVEVALUE  ACCOUNT+,1000
```

к содержимому ячейки SAVEVALUE с именем ACCOUNT добавляется 1000.

Рис. 13.2. Обозначения атрибутных блоков

MSAVEVALUE A,B,C,D

Блок изменяет значение элемента матрицы (произвольный доступ возможен только по первым двум измерениям, остальные индексы подразумеваются единичными). Операнды блока:

`A` — имя или номер матрицы с необязательным знаком «+» или «-»;

`B, C` — номера строки и столбца соответственно;

`D` — значение, которое должно быть записано, добавлено или отнято.

С матрицами связан атрибут `MX$<имя>(m,n)` — элемент, находящийся на пересечении строки m и столбца n.

Доступ к элементам матриц большей размерности можно организовать через $PLUS$-процедуры. Матрицы, объявленные командой `MATRIX`, будут для них глобальными объектами. Могут быть введены и временные локальные матрицы.

Матрицы могут использоваться для расчета марковских цепей (при исследовании распределения активного населения по социальным группам в [89, разд. 11.11] матрица описывала вероятности профессии сына в зависимости от профессии отца).

ASSIGN A,B,C

Блок используется для задания или изменения значения параметра транзакта. Операнды:

`A` — номер параметра активного транзакта;

`B` — присваиваемое значение;

`C` — номер функции (не обязателен).

В примере `ASSIGN 20+,-3`
плюс, следующий за операндом `A`, показывает, что значение операнда `B` *прибавляется* к операнду `A`. Итак, этот блок прибавляет -3 к значению, содержащемуся в 20-м параметре транзакта.

Если содержимое поля `C` — не параметр, то оно рассматривается как номер функции. Тогда эта функция вычисляется, умножается на значение операнда `B` и прибавляется, вычитается или присваивается указанному в `A` параметру транзакта в зависимости от знака после операнда `A`.

PLUS A,B

Блок оценивает выражение `A` и помещает результат в операнд `B`.

MARK A

Блок считывает абсолютное время с системных часов в параметр `A` активного транзакта.

INDEX A,B

Блок прибавляет к параметру `A` активного транзакта числовое значение, которое указано в `B`. Результат помещается в *первый* параметр активного транзакта.

PRIORITY A,B

Блок устанавливает приоритет A активному транзакту (напомним, что в GENERATE транзакт по умолчанию рождается с нулевым приоритетом). После этого транзакт помещается в ЦТС согласно своему новому приоритету. Если в поле операнда B записано BUFFER, просмотр ЦТС начинается сначала, то есть вновь назначенный приоритет оказывает немедленное действие.

Значения приоритетов должны быть целые и предпочтительно смежные.

INDEX A,B

Блок присваивает значение суммы своих полей A и B параметру P1, используемому LOOP. Это позволяет организовать переадресацию внутри цикла. Если LOOP применяется в сочетании с блоком INDEX, управление циклом осуществляется только по первому параметру активного транзакта. Блок LOOP реализует только убывающую индексацию с шагом -1.

SELECT O A,B,C,D,E,F

Блок выбирает объект по условию и помещает его номер в параметр активного транзакта. Операнды:

O — стандартный логический атрибут или отношение;

A — операнд для записи номера выбранного элемента;

B — нижняя граница диапазона поиска;

C — верхняя граница диапазона поиска;

D — опорное значение (база сравнения) с операндом E;

E — имя класса СЧА, обязателен только для условной формы;

F — номер блока назначения, если не отобран ни один объект.

При входе в SELECT проверяется элемент, специфицированный операндом B. Если такого элемента нет и не требуется управляющего оператора для его определения, создается новый элемент. После этого проверяется каждый элемент ряда от B до C. Счетчик элементов помещается

в параметр входящего транзакта, заданный операндом A. Тип элемента согласуется с логическим оператором. Заданное отношение определяется между операндами D и E, причем D рассчитывается для входящего транзакта, а E — для каждого элемента упомянутого ряда.

Операнды D и E являются обязательными, если O является условным оператором, но не MIN или MAX. В примере

```
SELECT SF  FULL1,BIN1,BIN13
```

в параметре транзакта с именем FULL1 будет сохранен номер первой полностью занятой памяти из тех, номера которых находятся между BIN1 и BIN13, В примере

```
SELECT E  EMPTY1,QUEUE1,QUEUE9,O,Q
```

проверяется каждая очередь от QUEUE1 до QUEUE9 включительно. Отбирается первый элемент очереди, значение которого *равно* нулю. Номер первой пустой очереди запишется в параметр EMPTY1 входящего в блок SELECT транзакта.

COUNT O A,B,C,D,E

Блок заносит число объектов в параметр активного транзакта. Операнды:

O — логический оператор либо указатель отношения;

A — операнд для принятия результата;

B — нижняя граница диапазона проверяемых объектов;

C — верхняя граница диапазона;

D — опорное значение для операнда E, обязательное только при наличии условия;

E — спецификатор класса объектов (для условного режима проверки).

В общем случае проверяется каждая сущность в диапазоне B–C. Класс объектов устанавливается операндом E или логическим оператором O (они должны быть согласованы). Когда используется *логический оператор*, применяются операнды A,B,C. В *условном* режиме применяются A,B,C,D,E. В этом режиме подсчитываются только те объекты, для которых *отношение между числовыми значениями* операндов D и E

удовлетворяет условию. Указатель класса комбинируется со спецификацией сущности для получения СЧА. Значение СЧА сопоставляется со значением D. Пример:

```
COUNT E EMPTYCOUNT,FIRSTQUEUE,LASTQUEUE,O,Q
```

Здесь блок COUNT работает в условном режиме. Операнд E задает СЧА класса Q, который относится к объекту типа «очередь». Проверяется каждый объект-очередь с номером, находящимся между FIRSTQUEUE и LASTQUEUE. Подсчитывается число пустых очередей и заносится в параметр EMPTYCOUNT активного транзакта.

LOGIC O A

Блок предназначен для включения, выключения или инвертирования состояния логического ключа. Если логический оператор O равен S или R, то ключ A устанавливается в состояние «включено» (Set) или «выключено» (Reset) соответственно. Если логический оператор — I, то состояние ключа инвертируется. Блок LOGIC в режиме инвертирования (в терминологии программистов первого поколения — «кувыркающаяся единица») позволяет чередовать альтернативы различного рода — маршруты, функции, числовые значения и т. п.

13.3 Ансамблевый комплекс

Компоненты данного комплекса связаны с формированием и взаимодействием ансамблей и групп транзактов.

Атрибуты

Атрибут A1 есть номер *ансамбля*, к которому принадлежит активный транзакт. Атрибут MBj указывает, есть ли в блоке Bj транзакт, принадлежащий к одному ансамблю с активным.

С *группой* j связываются атрибуты GNj — количество членов в указанной группе чисел, GTj — количество членов в группе.

Блоки

На рис. 13.3 приведены графические обозначения ансамблевых и «групповых» блоков.

Рис. 13.3. Ансамблевые блоки

SPLIT A,B,C

Блок SPLIT (расщепить) создает A копий активного транзакта. «Транзакт-родитель» и его копии образуют единое семейство. Копии заимствуют атрибуты своего «родителя»: каждый из A «потомков» имеет его приоритет, отметку времени и индикатор трассировки. Все копии направляются по адресу, указанному в поле B. Если копий и необходимых адресов несколько, по этому адресу должен находиться распределяющий TRANSFER.

Иногда необходимо, чтобы между родителем и потомками проводилось различие. Для этого в SPLIT предусмотрена возможность упорядочения. Операнд C понимается как номер j ранжирующего параметра. Если до входа в SPLIT у родителя было $Pj = m$, то по-

сле расщепления будет $m+1$, а у потомков — $m+2, m+3, \ldots$. На этой основе можно адресовать каждую копию к своему блоку назначения, который вычисляется по операнду B копии. «Транзакт-родитель» проходит к блоку, следующему за SPLIT.

С помощью SPLIT и упорядоченных потомков можно обеспечить *параллельное моделирование* альтернативных вариантов системы с одним и тем же набором случайных чисел (моменты появления транзакта и время обслуживания, занесенное в параметр родителя). Это сэкономит время на генерацию упомянутых чисел.

ADOPT A

Блок меняет ансамблевую принадлежность активного транзакта на A.

ASSEMBLE A

Блок используется для сборки. Из собранных A транзактов продолжает дальнейшее движение первый. Транзактам, прерванным на устройстве, которые завершили сборку в блоке ASSEMBLE, не разрешается покинуть блок до тех пор,пока все прерывания не будут сняты.

GATHER A

Блок синхронизирует движение транзактов. После накопления в этом блоке A транзактов *все они* (в отличие от ASSEMBLE) могут продолжить движение.

MATCH A

Блоки MATCH — это парные блоки, которые заставляют транзакты синхронизироваться. Каждый из них ссылается в своем операнде на имя другого. Когда транзакт входит в один из них, вычисляется операнд A и находится сопряженный блок MATCH. Если последний содержит транзакт из того же семейства, что и активный, связанный транзакт удаляется из цепи парности и переносится в ЦТС. Активный транзакт продолжает движение. Если сопряженного блока нет, происходит останов с выдачей сообщения об ошибке. Если транзакты в текущий момент вытеснены по приоритету, им не разрешается покинуть блоки MATCH.

JOIN A,B

Блок добавляет активный транзакт к группе транзактов или число — к числовой группе. Операнды:

A — номер группы, в которую будет добавлен новый член;

B — значение, которое включается в числовую группу.

Если B специфицирует число, блок JOIN оперирует с числами, иначе — с транзактами. Работая с числами, JOIN вычисляет операнды A и B. Затем число, заданное в операнде B, включается в числовую группу, указанную A. Оперируя с транзактами, блок JOIN включает входящий транзакт в группу, специфицированную операндом A.

Новые члены помещаются в цепочку группы в порядке их поступления. Группы чисел отличаются от групп транзактов, даже если они имеют одинаковые номера. После блока JOIN активный транзакт переходит в следующий блок.

Транзакт *удаляется* из группы блоком TERMINATE, соединением его с другими транзактами в блоке ASSEMBLE или удалением из группы (блок REMOVE).

REMOVE O A,B,C,D,E,F

Блок удаляет члены из групп. Его операнды:

O — отношение между D и E, при котором выполняется удаление;

A — номер группы, из которой производится удаление;

B — максимальное число удаляемых транзактов (может быть ALL, имя, константа, СЧА);

C — значение, которое должно быть удалено из числовой группы; в режиме транзактов поле C не используется;

D — номер параметра транзакта, который определяет удаляемые члены группы, или PR, если признаком для удаления является приоритет;

E — справочное значение, с которым сравнивается операнд D активного транзакта;

F — альтернативный блок для активного транзакта; когда F пусто, а также при результативном удалении, тразакт продолжает движение по диаграмме.

Блок REMOVE действует в числовом режиме, если используется операнд C. Иначе он работает с группой транзактов.

Полезно рассмотреть наиболее употребительные комплексные варианты:

1. *Самоудаление.* Если B,D,E пусты, транзакт удаляется.

2. *Счетчик.* Если D,E пусты, количество транзактов, указанное в B, удаляется из группы (если не хватает — все). ALL в этом поле удаляет всю группу.

3. *Управление по параметру или приоритету.* Используются поля B,D,E. Поле B указывает количество (возможно, ALL), D — изучаемый параметр или PR, E — сравниваемый СЧА. Условие удаления записывается после REMOVE; по умолчанию подразумевается равенство.

4. *Экстремаль.* Условие имеет вид MAX или MIN. Аналогично предыдущему случаю B,C,E не применяются.

В численной моде REMOVE использует A,C и, возможно, F. В поле C указывается искомое числовое значение. Найденный элемент покидает группу, при отсутствии активный транзакт переходит по адресу F.

EXAMINE A,B,C

Блок применяется для проверки членства в числовой группе или в группе транзактов и определяет движение транзакта, не меняя состава группы. Он работает либо в «числовом» режиме, либо в режиме «транзакта». Если используется операнд B, режим числовой. Операнды:

A — идентификатор проверяемой группы;

B — указывает, какое числовое значение разыскивается;

C — альтернативный блок для активного транзакта, если члены группы не найдены.

Пример:

```
EXAMINE VALID_COLORS,P$COLOR,NOT_CORRECT_COLOR
```

Если числовая группа с именем VALID_COLORS не включает значение, содержащееся в параметре транзакта с именем COLOR, то активный транзакт проследует к блоку с меткой NOT_CORRECT_COLOR. Если значение является членом группы, активный транзакт направляется к следующему по порядку блоку.

В режиме транзактов проверяется принадлежность активного транзакта к группе, указанной операндом A.

SCAN O A,B,C,D,E,F

Блок работает только с *группами* транзактов. Он позволяет вошедшему в блок транзакту (не обязательно члену группы A) выбрать первый элемент группы, параметр или приоритет (B) которого удовлетворяют условию. Операнды:

O — отношение между B и C для выбора члена группы;

A — проверяемая группа;

B — проверяемый атрибут (PR или параметр);

C — справочное значение, с которым будет сравниваться операнд B;

D — PR или номер параметра, значение которого должно быть назначено параметру активного транзакта;

E — номер принимающего значение параметра;

F — новое назначение для вошедшего в SCAN транзакта при неуспехе.

Операнд B указывает, какой атрибут транзакта подлежит проверке. Блок SCAN выбирает из группы первый транзакт, удовлетворяющий условию, и запоминает заданный атрибут в параметре вошедшего в SCAN транзакта. Если активный транзакт не имеет такого параметра, то он создается.

Если используется условие И/ИЛИ для операндов B,C, то проверяется первый член группы. При указании MIN либо MAX отыскивается транзакт с экстремальным значением указанного атрибута. Если таких транзактов несколько, то берется первый из них. По умолчанию операнд C

принимается нулевым. Если условие имеет значение MIN или MAX, то операнд C использовать нельзя. Если не задано никаких условий проверки, берется первый транзакт группы. После выбора транзакта его атрибут, специфицированный операндом D, записывается в параметр вошедшего транзакта, который задан операндом E. Вошедший транзакт следует к очередному блоку.

Операнды B и D всегда относятся к проверяемому члену группы. Здесь может использоваться любой СЧА. Если для вычисления СЧА требуется транзакт, то используется проверяемый. Результат есть номер параметра, значение которого представляет конечный результат. Отсутствие условного оператора подразумевает проверку равенства. Если условие не используется, но заданы B и C, то сравниваемые значения должны быть равны.

Если в группе не будет найдено ни одного транзакта, удовлетворившего условию, то вошедший транзакт далее адресуется либо в соответствии с операндом F, либо к следующему блоку. В примере

```
SCAN MIN INVTORY,PRICE,,PARTNUMBER,100
```

просматриваются все транзакты из группы INVTORY и выбирается имеющий минимальное значение параметра PRICE. Если таких транзактов несколько, то берется первый из них. Для него вычисляется операнд D. Затем параметру с номером 100 активного транзакта присваивается значение параметра PARTNUMBER. Если группа пуста, никакого действия не выполняется. В примере

```
SCAN E   LOT11,P$PARTNUM,127,P$PRICE,PRICE,PHONE
```

при входе транзакта в блок SCAN группа транзактов с именем LOT11 будет просматриваться до обнаружения первого транзакта, с параметром PARTNUM, равным 127. При нахождении такого транзакта значение его параметра PRICE передается соответствующему параметру вошедшего. Затем активный транзакт следует к очередному блоку. Если в группе не будет найдено ни одного такого члена, очередным окажется блок с меткой PHONE.

ALTER O A,B,C,D,E,F,G

Блок изменяет приоритет либо параметр выбранных членов группы транзактов. Операнды:

O — указатель отношения (буквенный!);

A — группа транзактов, члены которой проверяются на необходимость изменения;

B — максимальное число изменений транзактов (по умолчанию ALL);

C — изменяемый атрибут (PR либо номер или имя параметра транзакта — члена группы);

D — заменяющее значение;

E — PR или параметр, который используется для проверки отношения;

F — базовое значение, с которым сравнивается операнд E;

G — альтернативный выход для активного транзакта при отсутствии изменений.

Блок ALTER выбирает транзакты из группы и изменяет один из атрибутов этих транзактов. Указанному в C атрибуту отобранных членов группы присваивается значение D. Измененные транзакты, вообще говоря, не меняют свою траекторию, однако могут быть перенаправлены в соответствии с операндом G.

При отсутствии указателя отношения и операндов E или F безусловно (без проверок) изменяются все транзакты — до верхнего предела (операнд B). Операнд E определяет, какой атрибут будет проверяться. Он может сравниваться с минимумом или максимумом атрибутов всех членов группы при MIN или MAX в качестве указателя отношения. Тогда будут изменяться все проверенные транзакты, которые имеют максимальный или минимальный атрибут. В этом случае не следует использовать операнд F.

Атрибут члена группы можно сравнить с операндом F по указанному отношению. Условием не должен быть MIN или MAX. В операнде E всегда имеется в виду параметр проверяемого транзакта. Однако если любой другой операнд есть СЧА, относящийся к транзакту, то он вычисляется для вошедшего транзакта.

Отношение между атрибутом транзакта (операнд E) и проверочным значением (операнд F) определяет условие изменения транзакта. По умолчанию указателем отношения считается равенство.

Операнд G указывает вошедшему транзакту альтернативный маршрут, когда:

- ни один транзакт не изменен;

- заданное в B значение счетчика изменений не может быть достигнуто.

Если операнд G не используется, вошедший транзакт всегда продвигается к следующему по порядку блоку. Пример: блок

```
ALTER INVENTORY,ALL,PRICE,4095
```

всем транзактам — членам группы INVENTORY устанавливает параметр с именем PRICE, равным 4095. В примере

```
ALTER NE BIN7,10,PRICE,4995,P$PARTNUM,127,OUT
```

в группе с именем BIN7 отыскиваются транзакты, у которых параметр с именем PARTNUM отличен от 127. Первым 10 транзактам, которые встретятся при проверке, параметр с именем PRICE установится в 4995. Если 10 транзактов не могут быть найдены при проверке, вошедший транзакт пытается войти в блок с меткой OUT, иначе проследует к очередному блоку.

13.4 Ресурсный комплекс

Команды и блоки этого пакета не позволяют работать с *диапазонами* номеров устройств и памятей.

Атрибуты

Устройство j характеризуется стандартными атрибутами:

Fj — состояние устройства; если оно занято, то Fj равно 1, иначе 0;

FCj — общее количество входов транзактов в устройство;

FIj — состояние устройства по прерыванию; если устройство занято транзактом,
прервавшим обслуживание другого транзакта, то FIj равно 1, в противном случае — нулю;

FRj — коэффициент использования устройства; вычисляется в тысячных долях как отношение времени занятости устройства к общему времени моделирования;

FTj — среднее время, в течение которого устройство было занято одним транзактом;

FVj — состояние устройства по готовности (1, если устройство доступно, 0 иначе).

С *памятью* j связаны следующие СЧА:

Rj — количество доступных мест в памяти j;

Sj — текущее содержимое памяти j;

SAj — среднее содержимое памяти j за время моделирования;

SCj — общее количество транзактов, входивших в память j;

SEj — состояние памяти по незанятости (1, если память j полностью свободна, и 0 в противном случае);

SFj — состояние памяти по занятости (противоположное предыдущему);

SRj — коэффициент использования памяти, вычисляется как отношение среднего содержимого памяти j к ее полному объему и выражается в частях от 0 до 1000;

SMj — максимальное количество транзактов, находившееся в памяти j за время моделирования;

STj — среднее время пребывания в памяти j одного транзакта;

SVj — состояние памяти по готовности; равно 1, если память j готова, и равно 0, если не готова.

Блоки

В *GPSS* имеются следующие блоки, связанные с устройствами:

SEIZE — захват устройства (вход транзакта в устройство);

RELEASE — освобождение устройства;

PREEMPT — захват устройства более приоритетным транзактом с прерыванием текущего обслуживания менее приоритетного;

RETURN — освобождение устройства, если оно было занято входом транзакта через блок PREEMPT;

FAVAIL — перевод устройства в состояние готовности;

FUNAVAIL — перевод устройства в состояние неготовности.

Занятие и освобождение памяти отображается блоками ENTER и LEAVE соответственно, а перевод памяти в состояние готовность и неготовность — посредством SAVAIL и SAVAIL. Графические обозначения блоков, работающих с устройствами и памятями, показаны на рис. 13.4.

Рис. 13.4. Блоки устройств и памятей

Если транзакт не может занять устройство, он задерживается перед ним, дожидаясь его освобождения. Устройство резервирует место для транзактов, которые ожидают его освобождения. Каждое устройство имеет:

- список задержки для пытающихся в него войти транзактов;
- список прерываний для транзактов, которые уже обслуживались в нем.

Транзакты, находящиеся в упомянутых списках, конкурируют между собой за право занять устройство. Транзакты, ожидающие в списке прерывания, имеют приоритет на вход в устройство перед транзактами, ожидающими в списке задержек. Устройство освобождается при входе транзакта в блоки RELEASE или RETURN. После этого оно занимается тем из конкурирующих транзактов, который имеет на это больше права.

Блоки SEIZE и RELEASE не обязательно использовать строго попарно. Путь занявшего устройство транзакта может ветвиться: в каждой ветви может быть свой RELEASE. Возможна и обратная ситуация — занятие может быть во многих точках, а RELEASE в одной. То же справедливо и для памятей.

Блок FAVAIL делает устройство готовым, FUNAVAIL — неготовым. Поведение транзакта, занимавшего устройство до перевода его в состояние недоступности, определяется операндами блока FUNAVAIL. Подчеркнем принципиальное отличие блокировки доступа к устройству и перевода его в неработоспособное состояние: в первом случае текущее обслуживание будет завершено, а во втором — прервано. Не забудьте включить в модель блоки возврата в готовность и окончания прерывания!

FAVAIL A

Блок устанавливает указанное операндом A устройство в состояние готовности.

FUNAVAIL A,B,C,D,E,F,G,H

Блок переводит устройство в состояние неготовности для входящих транзактов. Операнды:

A — имя или номер устройства.

B — предписание последующих действий для транзакта, занимавшего устройство. Операнд необязательный, может быть RE, CO (REmove — удалить или COntinue — продолжить) либо Null.

C — следующий блок для транзакта, занимавшего устройство.

D — номер параметра для регистрации времени, оставшегося до окончания обслуживания, если транзакт, занимавший устройство, удаляется из цепи будущих событий.

E — предписание последующих действий для транзактов, обслуживание которых было прервано до выполнения блока FUNAVAIL. Операнд должен быть RE, CO или Null.

F — новый блок для транзактов, выполнение которых было прервано ранее.

G — последующие действия для транзактов, ожидавших возможности занять устройство до выполнения блока FUNAVAIL. Операнд должен быть RE, CO или Null.

H — новый блок, к которому перейдут ожидавшие возможности занять устройство до выполнения блока FUNAVAIL транзакты.

Сложность блока FUNAVAIL объясняется тем, что он имеет дело с тремя типами транзактов:

1. занимающими устройство (операнды B–D);
2. ожидающими возобновления обслуживания, прерванного блоком FUNAVAIL (операнды E–F);
3. ожидавшими возможности занять устройство до выполнения FUNAVAIL.

Транзакты, поступившие в период неготовности, будут задержаны и не допущены к устройству. Если в операндах записана опция RE, транзакты перестают претендовать на занятые устройства. Если в поле G для транзактов, ожидавших возможности занять устройство, указано RE, то для дальнейшего перенаправления транзактов надо заполнить поле H.

Если используется опция CO, то транзакты, занимавшие устройство, могут продолжать его занимать даже в период неготовности. В статистических данных по использованию устройства это время учитывается.

Если указывается блок нового назначения, транзакты освобождаются от ранее установленных связей и направляются к новому блоку. Задержанные и прерванные транзакты, управляемые операндами G и H, не могут быть перенаправлены без опции RE. Транзакт, занявший устройство и управляемый операндами B–D, а также прерванные транзакты, управляемые операндами E и F, могут снова претендовать на занятие устройства, даже будучи направленными по новому назначению. Это делается указанием измененного назначения без использования опции RE.

Операнды B–D используются для управления транзактом, владевшим устройством. Если B=CO, этот транзакт не лишается обладания устройством, но может получить новое назначение через операнд C. При B=RE он удаляется и может продолжать путь в модели, не заходя в RETURN или RELEASE для данного устройства. Здесь операнд C обязателен.

SEIZE A

Блок моделирует занятие устройства A или ожидание такой возможности. Если устройство не занято, транзакт получает разрешение занять его, входит в блок SEIZE и пытается войти в следующий блок. В противном случае транзакт попадает в цепочку задержки последним в своем приоритетном классе.

RELEASE A

Блок освобождает устройство A от транзакта или исключает транзакт, прерванный по приоритету, из борьбы за устройство. В первом случае транзакт освобождает это устройство и следует к очередному блоку. Во втором случае он удаляется из цепи прерванных транзактов для этого устройства, причем право его на владение устройством не затрагивается.

SAVAIL A

Блок устанавливает память A в доступное состояние. Если какие-либо транзакты ожидают в цепи задержанных для этой памяти, они получают возможность войти в память согласно уже обсужденному нами правилу. Те транзакты, запросы которых не могут быть удовлетворены, остаются в цепи задержанных.

SUNAVAIL A

Блок переводит память A в состояние неготовности. Все транзакты, претендующие на вход в память, будут помещаться в цепь задержки для этой памяти.

ENTER A,B

Блок управляет использованием «памятей». Операнды: A — имя или номер памяти; B — объем запроса транзакта (число единиц, на которое уменьшается емкость доступной памяти). По умолчанию B=1.

При готовности памяти и наличии достаточных свободных ресурсов транзакт входит в блок, и корректируется текущее состояние памяти. Иначе транзакт помещается в цепь задержки данной памяти с учетом его приоритета.

LEAVE A,B

Блок освобождает ресурс памяти (STORAGE) для других транзактов. Операнды: A — имя или номер памяти, B — освобождаемый объем памяти (по умолчанию 1).

13.5 «Цепной» комплекс

Атрибуты

С цепью пользователя j связаны следующие СЧА:

CAj — средняя длина цепи;

CCj — общее количество транзактов, входивших в цепь;

CHj — текущая длина цепи;

CMj — максимальная длина цепи;

CTj — среднее время пребывания в цепи одного транзакта.

Блоки

На рис. 13.5 показаны графические обозначения блоков, связанных с цепями событий.

Рис. 13.5. Обозначения «цепных» блоков

PREEMPT A,B,C,D,E

Блок действует аналогично SEIZE, если устройство свободно. Иначе он вытесняет транзакт, занимавший устройство, более приоритетным транзактом. Операнды:

A — имя или номер прерываемого устройства (но не памяти)!

B — режим занятия устройства. Операнд (необязательный) должен быть PR или Null.

C — блок нового назначения для транзакта, в настоящий момент владеющего устройством.

D — номер параметра прерванного транзакта, который запоминает недоданное время обслуживания, если прерваннный транзакт удаляется из ЦБС.

E — при значении RE исключает прерванный транзакт из борьбы за устройство (тогда операнд C обязателен).

Если устройство не занято, то блок PREEMPT ведет себя подобно SEIZE. В противном случае в зависимости от операнда B он действует в режиме приоритета или в режиме прерывания. В режиме *приоритета*

только транзакт с более высоким приоритетом может вытеснить другой транзакт из устройства. В режиме *прерывания* (пустое поле B), если устройство уже занято по приоритету, активный транзакт помещается в цепочку зависших транзактов. При предоставлении устройства транзактам из этой цепочки отдается большее предпочтение, чем идущим по приоритету или из цепочки задержки.

Операнды C–E определяют, что нужно сделать с прерванным транзактом. Таким транзактам не разрешается быть в ЦТС. Однако снятые с устройства транзакты, пребывание которых в блоке ADVANCE не завершено, помещаются в ЦТС автоматически, если в PREEMPT опущены операнды C и E.

Транзакту не разрешается входить в блок PREEMPT в режиме прерывания, если устройство в настоящий момент занято по приоритету. Таким образом, *GPSS* не обеспечивает имитацию вложенных прерываний. Кстати, при моделировании подобных процессов необходимо организовывать *магазин*: последняя пришедшая на устройство заявка должна покинуть его первой (дисциплина LIFO).

RETURN A

Блок освобождает устройство A или удаляет прерваннный транзакт из борьбы за устройство. Здесь могут быть два случая. Если транзакт в настоящий момент владеет устройством, то он освобождает устройство и пытается войти в следующий блок. При необходимости решить вопрос о передаче устройства следующему транзакту претендент последовательно разыскивается в цепях зависших, прерванных и, наконец, задержанных транзактов. Во втором случае транзакт отстраняется от устройства по приоритету. Здесь транзакт удаляется из цепочки прерванных транзактов для этого устройства, причем его право на владение устройством не затрагивается.

Отдельного разговора заслуживает вопрос о дообслуживании прерванной заявки. В «педагогической» модели из [74] в качестве транзактов, прерывающих консультации профессора, рассматриваются телефонные звонки. По окончании телефонного разговора прерванная консультация продолжается с дополнительной задержкой. Для реализации этой дисциплины мы, используя операнд D блока PREEMPT, вычисляем оставшееся время и заносим, к примеру, в 7-й параметр прерванного транзакта. При повторном просмотре ЦТС прерванный транзакт входит в блок

ASSIGN, работающий в режиме приращения. Там к нему добавляется дополнительная задержка — фиксированная или случайная. При восстановлении в блоке ADVANCE содержимое упомянутого параметра определит задержку, фиксируемую в ЦБС.

В вышеупомянутом ASSIGN можно задать новую реализацию случайной длительности обслуживания или прежнюю (таковую надо запомнить перед входом в ADVANCE). Соответственно будут реализованы обсуждавшиеся в [32] дисциплины RS и RW.

LINK A,B,C

Блок управляет помещением активного транзакта в пользовательскую цепь. Его операнды:

A — номер (имя) ПЦ;

B — принцип упорядоченности;

C — блок назначения для активного транзакта в случае, когда он не присоединяется к ПЦ (индикатор связи ПЦ установлен в нуль); индикатор связи включается (устанавливается в 1), если он был выключен.

Если операнд B — LIFO или FIFO, транзакт помещается соответственно в начало или конец цепи. Любое другое выражение оценивается для активного транзакта и для каждого транзакта в цепи, и обеспечивается расположение элементов ПЦ по убыванию этого выражения. Исключением является Pk — транзакты расположатся по *возрастанию* значений k-го параметра. Прямой возможности построить цепь по убыванию значений параметра нет, однако опция BACK блока UNLINK при выборке из цепи вполне ее заменяет.

При определении порядка включения в цепь приоритет текущего транзакта в общем случае игнорируется.

В простейшем варианте (C не используется) входящий транзакт всегда помещается в ПЦ A и удаляется изо всех прочих цепей, исключая группы транзактов и цепи прерываний. Сложнее обстоит дело при использовании операнда C. В каждой ПЦ имеется флажок — индикатор связи (ИС). При заданном операнде C входящий транзакт не будет введен в ПЦ при включенном ИС и направится по указанному в C адресу.

Удобно представлять ИС как признак занятости гипотетического ресурса. Блоки LINK и UNLINK могут управлять очередью к такому ресурсу. ИС сбрасывается, когда UNLINK обнаруживает опустошение ПЦ. Транзакт, обнаруживший недоступность ресурса, не должен ждать в ПЦ. Он направляется по адресу, указанному в операнде C, и блок LINK включает ИС.

Рассмотрим теперь два транзакта, последовательно прибывшие в LINK. Если первый не вошел в UNLINK до прибытия второго, второй найдет ИС включенным и будет помещен в ПЦ. Когда первый войдет в UNLINK, второй будет удален из ПЦ. К этому времени ИС еще включен. Когда второй транзакт войдет в UNLINK, и в ПЦ не останется транзактов, ИС сбрасывается.

Все это нужно для реализации сложных диспетчерских алгоритмов.

UNLINK O A,B,C,D,E,F

Блок извлекает транзакты из пользовательской цепи. Его операнды:

O — отношение между числовыми значениями D и E, при котором происходит удаление;

A — номер ПЦ, из которой извлекаются транзакты;

B — блок назначения для извлекаемых транзактов;

C — максимальное число транзактов, подлежащих извлечению (по умолчанию ALL);

D — сравниваемый атрибут (атрибут транзакта, подлежащий проверке, булева переменная для проверки или BACK — для извлечения членов с конца списка пользователя);

E — справочное значение, с которым сравнивается операнд D (не нужен, если D есть булева переменная);

F — блок альтернативного назначения для транзакта, вошедшего в UNLINK.

Операнды D и E и условие необязательны. Если они отсутствуют, извлекаются все транзакты от начала списка до его конца или до достижения лимита (операнд C).

Операнд D возможен в трех вариантах. Если это булева переменная со значением 1, то транзакт из цепочки извлекается. Если D=BACK, то извлекаются транзакты с конца списка и до исчерпания лимита. В остальных случаях операнд D вычисляется применительно к транзакту из цепи и используется в качестве номера параметра этого транзакта, значение которого и есть конечный результат. Этот результат сравнивается с операндом E.

Операнд E используется только вместе с D и с условием. Сканирование списка пользователя выполняется с его начала. Каждый транзакт вплоть до исчерпания лимита (операнд C) извлекается, если между операндами D и E выполняется условие.

Условие можно использовать для извлечения транзакта, рассматривая его как отношение между атрибутом транзакта (операнд D) и справочным значением (операнд E). По умолчанию подразумевается равенство. Если условие не используется, но присутствуют операнды D и E, то для извлекаемых транзактов эти значения должны быть равны.

Операнд F используется для задания активному транзакту альтернативного назначения в том случае, если не достигнут лимит извлечения (операнд C) или когда из ПЦ нельзя извлечь ни один транзакт. Если операнд F не используется, вошедший в блок UNLINK транзакт всегда переходит к следующему блоку. Пример:

```
UNLINK  ONHOLD,REENTRY,1
```

Это самый простой способ использования блока UNLINK. Первый транзакт с начала списка по имени ONHOLD, если таковой имеется, извлекается из списка и направляется к блоку с именем REENTRY. Он помещается в ЦТС за равными по приоритету. Транзакт, вошедший в UNLINK, переходит к следующему блоку.

BUFFER

Блок помещает активный транзакт в ЦТС вслед за транзактами с таким же приоритетом, и просмотр ЦТС начинается сначала. Последнее обстоятельство и является основной целью применения BUFFER.

13.6 Комплекс управления

Атрибуты

`AC1` — абсолютное системное время (с последней команды `CLEAR`);

`C1` — относительное системное время (с последней команды `RESET`);

`TG1` — остаток счетчика завершений.

Команды

INCLUDE "$<$имя файла$>$"

используется для включения в модель ранее отлаженных фрагментов (в частности, $PLUS$-процедур). По умолчанию файл считывается из той же директории, в которой создается модель. Если нужно считать файл из другого каталога, к нему указывается путь либо от корневого каталога логического диска, либо от текущего каталога (переход к надкаталогу задается двумя точками).

START A,B,C,D

начинает моделирование. Операнд `A` содержит исходное значение счетчика завершений, `B=NP` отменяет выдачу стандартного отчета после завершения моделирования, `C` — не используется, `D=1` включает в отчет распечатку ЦТС и ЦБС.

HALT

задается через одно из динамических окон и немедленно прерывает моделирование. Состояние модели сохраняется. После этой команды можно открыть дополнительные динамические окна, проверить состояние устройств, памятей, блоков, очередей, значения выражений и элементов матриц, состояние цепей и выполнить необходимую коррекцию модели.

STOP A,B,C

задает останов при входе в блок B транзакта с номером A. Отсутствие операнда A подразумевает любой транзакт. Эти остановы удобно задавать и отменять в окне блоков. Отработка останова не исключает его из списка запланированных. C=OFF отменяет все запланированные остановы, удовлетворяющие условиям A и B.

SHOW X

позволяет вычислить выражение X и получить его значение в статусной строке главного окна, а также в окне выражений.

REPORT

приводит к немедленному формированию стандартного отчета.

STEP A

заставляет модель отработать A последовательных входов в блоки. В статусную строку и журнал заносятся модельное время, номер активного транзакта, содержащий его и запланированный на следующий шаг блоки.

CONTINUE

продолжает процесс моделирования до очередной контрольной точки, команды HALT или до исчерпания счетчика завершений.

Блоки

Обозначения блоков данной группы показаны на рис. 13.6.

Рис. 13.6. Блоки управления имитацией

TRACE

включает трассировку активного транзакта. При этом в Окно журнала при каждом входе активного транзакта в очередной блок посылается сообщение. Блок `UNTRACE` режим трассировки отключает.

13.7 Комплекс ввода-вывода

Графические обозначения блоков данной группы приведены на рис. 13.7.

Рис. 13.7. Блоки управления потоками данных

OPEN A,B,C

A — описатель потока (имя, целое число, строка и т. д.),

B — номер потока данных,

C — альтернативное назначение транзакта при ненулевом коде завершения.

Создается поток данных, номер строки устанавливается в 1. Номер потока используется для последующих ссылок в блоках READ, WRITE и CLOSE. Пример:

```
OPEN ("Myfile.txt"),3,Error_Block
```

SEEK A,B

устанавливает номер текущей строки для потока данных. Здесь A — заданный номер, B — имя или номер потока (по умолчанию — единица).

READ A,B,C

Блок считывает текстовую строку из потока данных с указанным номером и помещает ее в параметр активного транзакта. Номер текущей строки увеличивается на 1. Операнды:

A — параметр транзакта,

B — номер потока данных,

C — альтернативное назначение транзакта при ненулевом коде завершения операции.

WRITE A,B,C,D

Блок преобразует (при необходимости) операнд A в текстовую строку и помещает ее в поток данных с указанным номером. Номер текущей строки увеличивается на 1. Операнды:

A — записываемая строка;

B — номер потока данных;

`C` — альтернативное назначение транзакта при ненулевом коде завершения операции;

`D` — режим записи (при `ON` и по умолчанию — добавление, при `OFF` — замена текущей строки).

CLOSE A,B,C

Блок закрывает поток данных и освобождает занятые ресурсы. Здесь

`A` — параметр транзакта, в который записывается код завершения;

`B` — номер потока данных;

`C` — альтернативное назначение при некорректном закрытии.

Глава 14

Технология моделирования

14.1 Среда

14.1.1 Меню

Главное окно системы (рис. 14.1) включает в себя Заголовок, Меню, Панель инструментов и Поле клиента для набора текста модели. Внизу окна имеется строка состояния: слева приглашение к вводу команды, середина — диагностика ошибок, справа — таймер модели (может отключаться).

Пункт меню (из разрешенных в данной ситуации, на что указывает *черный* шрифт его названия) выбирается мышью или клавишами управления курсором. Можно также нажать клавишу, соответствующую подчеркнутой букве в названии этого пункта. Например, нажатие [C] вызовет раскрытие меню `Command`. Для некоторых пунктов указаны комбинации «горячих клавиш». Одна и та же комбинация, вообще говоря, может иметь различный смысл — в зависимости от состояния окружения. Сделанный выбор представляется пунктирным подчеркиванием или утапливанием иконки пункта.

Ввод команды обычно заканчивается нажатием [OK] или [Enter]. При работе с боксом `Command/Custom` последний вариант открывает строку для набора следующей команды.

Многие объекты имеют собственные меню, построенные по общему принципу. Обычно организуется диалог, предлагающий кнопки и поля ввода. После входа в меню для выбора его пункта достаточно нажать клавишу, соответствующую подчеркнутой букве.

Рис. 14.1. Главное окно *GPSS World*

Используя меню главного окна, можно создавать объекты *GPSS/W*: модели, Имитации, отчеты и тексты. Каждый из них можно рассматривать в соответствующих дочерних окнах. Типичный проект начинается с создания или модификации модели. Следует иметь в виду, что модель — не просто текстовый файл: она содержит также установки `Settings`, закладки `Bookmarks` и результаты, которые не могут читаться внешним редактором.

14.1.2 Боксы диалогов и Help

Выбор некоторых пунктов меню требует детализации или конкретизации запроса путем заполнения окон (полей) диалога. Переходы между полями организуется щелчком мышью или [Tab]. Клавиша [Enter] обычно равнозначна [OK]. Нажатие [F1] вызывает алфавитный `Help` в его исходном состоянии, а при выделенном предмете запроса — тот же `Help` с набран-

ным ключевым словом. Через Главное Меню доступны путеводитель по меню, Алфавитный указатель терминов и Справочник.

14.1.3 Настройки

Все настройки задаются при создании *модели* через меню `Edit/Settings`. Последующие фазы проекта эти установки только наследуют. Рассмотрим вкладки упомянутого меню:

Simulation: `Poll Count` — число попыток входа в блок до прерывания; `Max Evaluation Depth` — ограничение глубины ссылок (для предупреждения зацикливания); `Stack Size` — то же для вложенности процедур; `Integration tolerance` есть допуск при численном интегрировании дифференциальных уравнений.

Reports: `Standard Report` формирует типовой отчет; `In Windows` обеспечивает вывод результатов в окно вместо файла. Можно указать желаемые дополнительные подотчеты. `Save Plot Points` определяет объем данных, сохраняемых при построении графика (при недостаточном объеме график будет обрезан слева или справа). `Suppress Scientific` исключает «научную» нотацию в отчетах и потоках данных. `Six Places` задает вывод с 6 знаками после десятичной точки вместо стандартных трех.

Random Numbers: распределяет потоки случайных чисел. `Time Ties` позволяет указать номер ДСЧ для определения очередности практически одновременных событий. Проверка на равенство моментов их наступления производится с точностью 10^{-9}; при задании нуля рандомизация будет отключена. Аналогично обстоит дело с ДСЧ для блоков `ADVANCE` и `TRANSFER`.

Function Keys: связывает операторы с функциональными клавишами.

Expressions: позволяет задать список выражений, часто используемых в процессе диалога с моделью, и при необходимости выбирать их из списка через меню `Select` вместо повторного набора.

14.1.4 Окна

Динамические окна могут быть открыты только после создания Имитации (т. е. после компиляции модели) командой

`Window/Simulation Window/<тип_окна>`

Одновременно могут быть открыты несколько окон. Их относительное расположение управляется командами `Cascade` или `Tile`. Щелчок мышью по любой точке окна, заслоненного другими окнами, делает его активным и выводит на передний план. Активное окно можно перемещать, «зацепившись» мышью за его заголовок. Размеры активного окна изменяются протаскиванием его угла или границы в нужном направлении. Режим вставки/замены указывается формой мерцающего курсора, который одновременно определяет место вставки в окне. Режим переключается нажатием клавиши [Ins]. Курсор можно перемещать щелчком мыши и клавишами управления (правая часть клавиатуры). Переход на очередную строку выполняется нажатием [Enter].

14.1.5 Этапы моделирования

Работа начинается с составления $GPSS$-программы. Посредством команды `INCLUDE "<имя_файла>"` в нее могут вставляться ранее отлаженные фрагменты (например, $PLUS$-процедуры). Если эти фрагменты находятся в директории, отличной от директории модели, имя файла-вставки должно указываться *с путем* от корневого дисковода логического диска или от файла модели. Вставки из файлов автоматически нумеруются, и диагностика ошибок в них сопровождается указанием номера вставки.

Компиляцию организуетпо команда `Command/Create Simulation`. Синтаксический контроль достаточно развит (в частности, контролируется парность скобок). При обнаружении ошибок создается циклический список их. Просмотр этого списка обеспечивают команды `Next Error` и `Previous Error` из меню `Search`. Глубина стека ошибок ограничена, поэтому исправление всех указанных в одном прогоне ошибок не гарантирует синтаксическую правильность модели. Каждый выбор элемента списка вызывает сообщение о типе ошибки в статусную строку и помещает точку вставки точно перед ошибочным элементом. Характер «претензий» системы обычно достаточно ясен:

`Line 10, Col 20, Invalid Delimiter. Expecting a GPSS Verb`

При неясной диагностике следует построить список гипотез и экспери-

ментально проверить каждую из них. Сделать последнее легко, поскольку компиляция выполняется практически мгновенно. Наибольшие трудности вызывает использование некорректных имен: слишком коротких или совпадающих со стандартными числовыми атрибутами. Оперативное запоминание внесенных в модель исправлений не требуется: компиляции подвергается правленый текст.

При успехе компиляции активируются интерактивные команды и становятся доступными окна имитации. После этого к модели можно добавить любые предложения: блоки (кроме GENERATE) и описания *PLUS*-процедур. Можно переопределить STORAGE, TABLE, QTABLE, MATRIX, VARIABLE. Интерактивно вставленный блок — временный и разрушается сразу после входа в него транзакта. Такой режим называется «ручным моделированием». При всех его преимуществах надо иметь в виду необходимость внесения показавших полезность изменений в сохраняемый текст модели.

После успешного создания Имитации через меню Command/Start задается начальное значение счетчика, которое вместе с имеющим ненулевой декремент блоком TERMINATE определяет длительность моделирования. Ее можно ограничить также по счетчику сгенерированных транзактов (операнд D блока GENERATE) или по времени — с помощью отдельного сегмента модели. Через Command/Custom можно набрать любое предложение и вставить его в Имитацию. Длинные процедуры и последовательности операторов лучше вставлять с помощью INCLUDE.

О запуске моделирования сообщает текст в статусной строке: Simulation in Progress. За его ходом можно следить по выведенным в статусную строку системным часам (обязательно на начальных этапах отладки для выявления возможного зависания).

14.2 Останов моделирования

Команда START определяет окончание моделирования: ее счетчик должен уменьшиться до нуля. Любой оператор модели TERMINATE с непустым полем операнда A уменьшает счетчик при входе в него транзактов. *Такой оператор в программе должен быть единственным.*

Начатое моделирование можно прервать командой HALT, кнопка которой имеется во всех динамических окнах. После ее выполнения можно выбирать блоки и задавать контрольные остановы в них (кнопка с

красным светофором), удалять заданные остановы (перечеркнутый красный светофор). При отработке очередного останова первоначальное условие останова, если оно задано по времени, удаляется, а все остальные условия сохраняются. Один шаг модели (вход активного транзакта в следующий блок) выполняется по нажатию кнопки с изображением лестницы. Продолжение моделирования с использованием команды STEP позволяет автоматически остановить его точно после заданного числа входов активного транзакта в блоки модели. Продолжить моделирование можно нажатием кнопки с зеленым светофором.

Контролировать промежуточные результаты (и просматривать окончательные) можно через окна Графики (Plot) и Выражений (Expression). Окно Таблиц позволяет наблюдать сходимость исследуемых распределений — гистограммы, среднего значения и среднеквадратического отклонения. В меню Windows поддерживается список всех открытых окон, включая отчеты по прогонам предыдущих версий модели.

Открытые динамические окна (даже для системных часов) существенно замедляют моделирование.

14.3 Сбор статистики

Для типовых имитаций автоматически собираемая статистика представляется вполне достаточной. При необходимости в более подробных сведениях можно определить в командах TABLE или QTABLE таблицы (гистограммы) распределений. Фактический сбор данных в первом случае обеспечивается специальными блоками TABULATE, а во втором — при входе в блок DEPART. Упомянутые таблицы (если отметить их в установке отчета), а также средние значения и среднеквадратические отклонения будут выводиться автоматически и могут просматриваться в окне таблиц.

Еще более детальные сведения можно собирать и обрабатывать с помощью скалярных и матричных сохраняемых значений и $PLUS$-процедур. Затем «историческая» база данных может быть обработана с помощью ANOVA — для получения доверительных интервалов. Результаты нестандартной обработки статистики можно вывести в файл потока данных.

14.4 ANOVA

Повторный прогон той же модели может дать различные результаты вследствие случайности. Процедура `ANOVA (Analysis Of Variance)` позволяет установить, можно ли объяснить разницу в результатах статистическими флуктуациями. Она рассчитывает доверительный интервал и проводит дисперсионный анализ.

Обычно во время эксперимента мы заполняем глобальную матрицу результатов и передаем ее как аргумент процедуре `ANOVA`. Каждое измерение должно задать максимальное число уровней по соответствующему фактору, дополнительное последнее — число реплик внутри каждого уровня. Поскольку матрица может иметь до 6 измерений, удается исследовать влияние до 5 факторов одновременно.

Пусть, например, мы хотим иметь матрицу для исследования влияния числа банковских служащих на время ожидания клиентов. Мы могли бы поместить три реплики моделирования работы переменного числа служащих (от 1 до 3) в элементы второй строки:

`ResMatr[2,1], ResMatr[2,2], ResMatr[2,3]`

Обычно дело обстоит несколько сложнее, и номер уровня не совпадает со значением фактора.

Работа процедуры `ANOVA` базируется на предположении о нормальном распределении среднего значения (центральная предельная теорема). Таблица `ANOVA` автоматически записывается в журнал моделирования. Если данных достаточно, то для каждого уровня рассчитывается описательная статистика, включая 95-процентный доверительный интервал. Уровней должно быть не менее двух, на каждом уровне — по крайней мере две реплики. Система выдает значение F-критерия и число степеней свободы для ошибки и факторов. Кроме того, для каждого уровня сообщаются:

`номер уровня` ,

`Count` — число результирующих значений,

`Mean` — среднее,

`Std.Dev.` — среднеквадратическое отклонение,

`Min` — минимальное наблюдение,

`Max` — максимальное наблюдение,

`95% Conf.` — полуширина доверительного интервала для среднего.

Полученное F программист должен сопоставить с табл. 7.1.

14.5 Синтаксис Эксперимента

После тестирования модели на ней можно поставить *Эксперимент* для получения количественных результатов и оценки их статистической значимости. Язык $PLUS$ имеет особый класс процедур (`EXPERIMENT`) с широкими возможностями, включая исследование поверхности отклика[1]. Эксперимент организуется записанной на языке $PLUS$ процедурой управления прогонами, которая для каждого прогона сама готовит исходные данные и вызывает модель. Управление прогонами реализуется с помощью встроенной процедуры `DoCommand(<строка>)`. Аргумент-строка транслируется в глобальном контексте и выполняется в Имитации. При записи таких аргументов необходимо использовать 4 двойных кавычки для выделения строк внутри строк. Запрещаются ссылки в них на имена локальных (`TEMPORARY`) объектов.

`EXPERIMENT` заполняет глобальную Матрицу Результатов, которая затем передается процедуре `ANOVA`. Программа Эксперимента может быть составлена как вручную, так и автоматически — с помощью диалога, реализуемого через меню `Edit/Insert Experiment`. Для начала рекомендуется автоматическое программирование. Пример Эксперимента подробно обсуждается в разд. 15.18. Когда `EXPERIMENT` транслируется, процедура добавляется к библиотеке имитации и становится доступной для вызова командой `CONDUCT`. Если Имитация имеет только один `EXPERIMENT` без аргументов, для запуска Эксперимента достаточно команды `CONDUCT` без ссылки на его имя.

Список аргументов, передаваемый Эксперименту через `CONDUCT`, используется для создания временных переменных, вызываемых по этим именам. Каждая из них получает вычисленное значение фактического аргумента, заданного для Эксперимента. Аргументы определяют, какая часть поверхности отклика подлежит исследованию в процессе Эксперимента.

[1] Для отличия таких процедур от эксперимента в обычном смысле слова мы будем писать этот термин с большой буквы.

14.6 Стандартный отчет

Система генерирует стандартный выходной отчет, если в качестве операнда B команды START не задано NP. Отчет не выдается, если обнаружена ошибка. Содержание стандартного отчета можно изменить в меню Edit/Settings — установках для Имитации. Для ускорения имитации, экономии места и концентрации внимания разработчика на наиболее интересных выходных данных можно вывести только их.

В стандартный отчет можно включить любые данные, записав их в ячейки SAVEVALUE или матрицы до выдачи стандартного отчета. Для этого надо указать имя ячейки SAVEVALUE или имя матрицы, в которых записаны эти данные. Дополнительную информацию можно накопить и выдать в отчет с помощью потоков данных (разд. 11.12). По умолчанию отчет пишется в файл report.gps (к имени присоединяется номер версии модели). Стандартный отчет можно редактировать. Перечислим основные разделы отчета.

Титул включает ссылку на систему моделирования, содержащий описание модели файл и дату.

Общая информация указывает начало и конец системного времени, число блоков, устройств и памятей.

Имена — это список назначенных программистом имен и их числовые эквиваленты.

Блоки — сведения о месте в модели, типе блока и числе транзактов (общее, к концу моделирования, в цепи RETRY повторных попыток).

Устройства — имена или номера, число входов, коэффициент использования за время с последнего CLEAR или RESET, среднее время пребывания транзакта за тот же период, готовность и занятость на момент окончания; число транзактов в связанных с устройством цепях (см. разд. 11.3.3, 13.5).

Очереди — имя или номер, максимальная зафиксированная длина, текущая длина, общее число входов, число входов с нулевым ожиданием, средняя длина очереди и среднее время ожидания, среднее время ненулевого ожидания и число транзактов, ожидающих очередной попытки.

Памяти — имя или номер; максимальная емкость; неиспользуемый объем к концу моделирования; минимум и максимум используемого объема; общее число входов; состояние готовности на момент окончания моделирования; средний использованный объем в абсолютном и отно-

сительном исчислении; число транзактов, ожидающих попытки входа и задержанных.

Таблицы — имя или номер; среднее и среднеквадратическое отклонение измеряемого значения; далее по частотным классам — число наблюдений и процент попаданий нарастающим итогом.

Кроме того, в отчет включаются данные о цепях (пользователя, текущих и будущих событий), группах транзактов, числовых группах, логических ключах, матрицах и сохраняемых значениях. Для нулевых значений матриц указываются диапазоны индексов.

Если результаты являются средними арифметическими величинами, то для проведения несложного статистического анализа (расчет доверительного интервала и значимости различия результатов моделирования) может быть применена команда `ANOVA`. Предполагается, что каждый уровень фактора имеет несколько реплик, отличающихся друг от друга заданными `RMULT` начальными установками ДСЧ.

Перед проведением статистического анализа следует стереть всю ненужную информацию, то есть результаты пробных прогонов.

14.7 «Горячие клавиши»

Горячие клавиши обычно используются для быстрой реализации разного рода управляющих действий. Они указаны в соответствующих пунктах меню.

Сначала перечислим клавишные комбинации вызова базовых функций:

[Ctrl+Alt+S] — компиляция,

[Ctrl+Alt+R] — рекомпиляция (никаких указаний на ее отличие от компиляции обнаружить не удалось),

[Ctrl+Alt+L] — повторение предыдущей команды,

[Ctrl+Alt+C] — `CONTINUE` (продолжение моделирования),

[Ctrl+Alt+H] — `HALT` (останов),

[Ctrl+Alt+1] — `STEP 1` (шаг моделирования).

При работе с текстом полезны:

[Ctrl+Alt+B] — следующая закладка,

[Ctrl+Alt+G] — переход к строке,

[Ctrl+Alt+N] — следующая ошибка,

[Ctrl+Alt+P] — предыдущая ошибка,

[Ctrl+Alt+F] — найти/заменить,

[Ctrl+Z] — вставить строку,

[Ctrl+D] — удалить строку,

[Ctrl+I] — Undo,

[Ctrl+C] — скопировать в буфер,

[Ctrl+X] — вырезать в буфер.

[Ctrl+V] — вставить из буфера.

Приведем некоторые комбинации общего назначения:

[Ctrl+O] — открыть файл,

[Ctrl+S] — сохранить файл,

[Ctrl+P] — печатать файл,

[Alt+Q] — следующая панель,

[J+Q] — предыдущая панель.

Кроме того, установлены режимы функциональных клавиш:

[F1] — Help,

[F2] — CONTINUE,

[F3] — выход на Рабочий стол *Windows*,

[F4] — HALT,

[F5] — STEP 1

[F6] — STOP „Off

[F7] — STOP

Остальные функциональные клавиши используются по усмотрению пользователя. Выбрав Блокнот Установок `Edit/Settings` и далее — вкладку `Function Keys`, можно задать дополнительные назначения клавиш — см. разд. 14.1.3. При задании в качестве команды `INCLUDE "<имя_файла>"` нажатие клавиши окажется эквивалентным заранее сформированным в упомянутом файле последовательности команд или определению процедуры.

Заданные пользователем установки функциональных клавиш действительны только при работе с *конкретной* моделью и ее производными (Имитациями).

14.8 Проблемы эффективности

14.8.1 Программирование

Приведем некоторые рекомендации по технике программирования.

Параметры следует нумеровать последовательно и использовать предпочтительно *прямую* адресацию. Назначайте смежные значения приоритетов.

Режим отказов в блоках `GATE` и `TEST` лучше заменять другими конструкциями. Если это невозможно, надо формулировать такие условия, которые меняются нечасто. Исключить напрасные проверки можно применением цепей пользователя. Блоки `COUNT` и `SELECT` по возможности заменяйте менее мощными блоками, учитывающими специфику задачи. Вместо работы с операндами `MIN` и `MAX` лучше написать и применить $PLUS$-процедуру выбора.

Для отражения в модели временны́х режимов типа суточных и недельных ритмов обслуживания следует использовать отдельные сегменты модели, переводящие используемые устройства и памяти в состояние неготовности. Длительность периодов неготовности задается в блоках `ADVANCE`.

Интегрирования идут медленно. Если известно аналитическое выражение для интеграла, используйте `VARIABLE` или $PLUS$-процедуру. Увеличение допустимой погрешности (по умолчанию 10^{-6}) ускорит интегрирование.

14.8.2 Окна

Открытие online-окон замедляет моделирование, поскольку система вынуждена ждать их обновления. Эти задержки обычно недооцениваются. Для иллюстрации степени замедления приведем данные по длительности прогонов одной и той же модели:

- все окна закрыты — 4 с;
- выводится таймер — 97 с;
- открыты окно блоков и таймер — 5271 с (почти полтора часа).

Для сравнения укажем, что та же модель на *Фортране* была реализована за 0.05 с.

На начальном этапе отладки, когда нужно убедиться в работоспособности модели, следует ограничить число входящих транзактов (скажем, сотней) и выводить в статусную строку значение таймера. Последнее необходимо для обнаружения зацикливания[2]. В дальнейшем нужно выполнять прогоны с закрытыми окнами до запланированного или аварийного останова и открывать динамические окна лишь для режима пошаговой отладки.

14.8.3 Память

Объем используемой системой виртуальной памяти — до 1 Гб. Необходимая виртуальная память выделяется автоматически. Ее объем устанавливается на вкладке Simulate в меню Model Settings. При малой емкости оперативной памяти слишком частые операции обмена могут замедлить моделирование, так что потребуется увеличить ее емкость. Для сообщений о программных ошибках необходима дополнительная память (по умолчанию — 500 Кб).

При перемещении или изменении размера окна с графиком из-за нехватки памяти может остаться только часть кривой. По умолчанию на графике отображаются 10000 точек. Изменить эту установку можно на вкладке Reports.

[2] Такая ситуация встретилась при разработке авторского ДСЧ, трудно диагностируемая ошибка в котором давала нулевые приращения времени.

14.8.4 Работа с файлами

Моделирование замедляется операциями пересылки данных в файлы. Для минимизации числа пересылок старайтесь *накапливать* результаты, а также использовать строки с большим объемом данных — см. функции для работы со строками.

14.9 Начальные условия и стационарный режим

Моделирование в *GPSS* ориентировано на получение *стационарных* характеристик. В связи с этим первостепенное значение приобретают вопросы о длительности разгонного участка и стационарного режима. Определение времени вхождения в стационарный режим можно выполнить экспериментально — например, наблюдая графики наиболее важных СЧА.

Типичной можно считать управляющую последовательность вида

```
START 10,NP
RESET
START 500
```

Первый короткий прогон загружает первоначально пустую систему, причем NP в поле B команды START подавляет вывод результатов первого прогона. Предполагается, что к его концу установился режим модели, близкий к стационарному. Затем RESET сбрасывает накопленную статистику и относительные системные часы, но не меняет состояние ДСЧ, сохраняемых значений, матриц и логических ключей. В этом случае разгонный участок не исказит результаты основного прогона, начинаемого командой START 500. Длительность последнего определяется из условия получения требуемой точности оценок характеристик стационарного режима. Сопоставление результатов с контрольными данными аналитических расчетов или натурного наблюдения позволяет провести *верификацию* модели [3].

GPSS/W записывает выполняемые операнды и сообщения о моделировании в *журнал сеанса*, который ведется в кодировке ASCII

[3] Согласие результатов подтверждает работоспособность модели для данного частного случая, но, разумеется, не доказывает ее правильность.

с накоплением записей в процессе моделирования. В нем отражаются трассировка транзактов, результаты работы команды `ANOVA`, сообщения об ошибках и т. д.

Особая проблема — учет при моделировании редких событий (отказов) и временны́х ритмов. Здесь вряд ли возможны универсальные рекомендации. Можно полагать, однако, что

- «разгонный» участок должен быть длиннее самого короткого из моделируемых периодов (например, суток);
- требовать на нем наступления редких событий не стоит;
- на стационарном участке событие каждого из запланированных типов должно произойти минимум по одному разу — хотя бы для того, чтобы убедиться в правильности программирования его обработки и в том, что система справляется с его последствиями (часто требуется, чтобы при одиночном отказе с длительным восстановлением ни один узел системы не оказался перегруженным).

Впрочем, о редких событиях см. раздел 5.3.

Глава 15

Практическая работа в GPSS World

15.1 Первая программа

Для открытия Главного Окна системы нужно последовательно нажать кнопки [Start] и [Programs] и щелкнуть по пункту *GPSS World*. Лучше предварительно перетащить иконку файла `gpssw.exe` с помощью Проводника из выбранной в процессе инсталляции корневой директории системы на рабочий стол *Windows*.

Для открытия файла с моделью выберем `File/Open`. После раскрытия папки `Sample Models` и указания типа интересующих нас файлов (с расширением `.gps`) высвечивается список моделей. Для поиска конкретной модели можно воспользоваться горизонтальной прокруткой. Щелчок по выбранному файлу выделяет его. Нажатием [OK] открываем Окно Моделей и в нем — программу выбранной модели. Щелчком по средней из кнопок в верхнем правом углу окна придаем окну максимальный размер.

Конкретно, выберем в папке `Sample Models` файл `sample1.gps`. Операторы программы *не содержат номеров строк*. Однако *GPSS/W* использует порядковые номера строк для указания места ошибок и других функций. По этим номерам работают команды `Search/Go To Line ...` Считанный файл с программой непосредственно не готов к прогону — требуется его компиляция.

Изучим некоторые установки и пункты меню и правила работы с ними. Главное Меню — начальная среда $GPSS/W$. Панель инструментов содержит стандартные иконки текстовых редакторов и дополнительных пояснений не требует.

Пункт `File` можно использовать для вывода текста модели в Окно Модели. Из текста компиляцией создается отдельный объект — Имитация. С этого момента все взаимодействия заносятся в Окно Журнала или в окна различных сущностей. Вставим блок `BUFFER` после `GENERATE`. Для этого, установив мышью курсор на `GENERATE`, через меню `Edit/Insert Line` добавим пустую строку. Курсор будет выставлен в начале новой строки непосредственно перед строкой с `QUEUE Barber`. Напечатаем в новой строке `BUFFER`. Старайтесь размещать типы блоков («глаголы» $GPSS$) один под другим — это не обязательно, но сделает программу более читаемой.

Используем команду `File/Save As...` для сохранения модели в файле с другим именем. Появится окно диалога. Введем в него имя `tmp` вместо текущего имени и нажмем [Save]. Имя файла вверху Окна Модели меняется на `tmp`. Закроем этот файл. В режиме открытия файла убеждаемся в том, что он был сохранен в папке `Samples`.

Не забудьте удалить этот файл после завершения сеанса, чтобы не засорять «образцовую» папку. То же относится ко всем «технологическим отходам» процесса освоения $GPSS/W$. По аналогичным соображениям для собственной практической работы создайте отдельную папку.

Установки модели задаются через `Edit/Settings`. Все Установки — штатные или измененные — привязываются к *конкретной* модели и производным от нее объектам, в частности, к Имитации. Ознакомьтесь с содержанием страницы `Simulation`. Обратите внимание на возможность переключения режимов знаков `*` и `#` — по умолчанию первый из них является признаком косвенной адресации, а второй знаком умножения. Будем иметь эту возможность в виду для *собственных* разработок, но пока оставим все как есть — иначе штатные примеры работать не будут. Заглянем на страницу `Reports` для ознакомления со средствами управления содержанием Отчета. Щелкнув по [OK], выйдем из блокнота Установок.

Выберем `Command/Create Simulation`. По данной команде выполняется компиляция модели — создается *Имитация*. При этом автомати-

чески открывается Окно Журнала. В него записываются все взаимодействия и сообщения $GPSS/W$ — в частности, информация о найденных синтаксических ошибках.

Познакомимся с Окном Блоков. Выберем

`Window/Simulation Window/Blocks Window`.

Появится детальный вид блоков в новом окне — см. рис. 15.1. Состояние окна для большей информативности дано на конец моделирования. После открытия Окна Блоков вы находитесь в *детальном* его режиме. Для перехода к обзорному следует выполнить команду `View/Entity Details`. Режим детальности переключается повторным выбором этой команды независимо от активного окна.

Рис. 15.1. Окно Блоков

Одновременно могут быть открыты несколько окон. Команды `Window/Tile` и `Window/Cascade` позволяют расположить открытые окна

рядом и с частичным наложением соответственно. Щелчок мышью по любому месту окна делает его активным и выводит — при частичном перекрытии окон — на передний план. Вы можете изменить статус, размеры и положение активного окна стандартными средствами *Windows*. Статус меняется щелчком по одной из кнопок в верхнем правом углу окна — сворачивание, частичное и полноэкранное представление; размер регулируется протаскиванием мышью за угол или сторону. Окно перемещается протаскиванием заголовка.

Выберем, наконец, `Command/START`. Когда откроется диалог, заменим 1 на расчетное число обработанных транзактов (скажем, 5000) и нажмем [OK]. Имитация запустится. Щелкнем мышью по любому месту в Окне Блоков, чтобы сделать его активным и наблюдать динамику модели. Свернем окно, щелкнув по [-] в его верхнем правом углу. Внизу Главного Окна (выше статусной строки) появится иконка свернутого окна, щелчок мышью по которой позволяет быстро восстановить его. Восстановим окно и затем закроем его, щелкнув по [x] в верхнем правом углу. Напомним, что открытые (хотя бы и свернутые) окна многократно замедляют моделирование. Поэтому всегда закрывайте окна, которыми не пользуетесь в данный момент.

После закрытия окон моделирование ускорится и быстро завершится. Автоматически откроется Окно Отчета. Для удобства рассмотрения расширьте окно протягиванием мышью его границы. Распечатайте отчет нажатием кнопки с изображением принтера. Изучите отчет. Обратите внимание на расчетное число заявок, прошедших блок `TERMINATE`, и большое скопление заявок в очереди. Последнее объясняется тем, что система при заданных параметрах не справляется с нагрузкой.

Для завершения сеанса выберите команды `File/Exit`. На вопрос о сохранении изменений ответьте `No`.

Обсудим эту простейшую имитацию. Формально она относится к моделированию одноканальной системы обслуживания с неограниченной очередью, а содержательно — к классической задаче моделирования работы парикмахерской, хотя, разумеется, может иметь другие интерпретации. Клиенты прибывают в среднем каждые 300 с модельного времени. Парикмахер тратит на стрижку в среднем 400 с. Значит, теоретический коэффициент загрузки системы $\rho = 400/300 = 1.33$.

Снова запустим *GPSS World*. Выберем команды `File/Open`, откроем папку `Sample Models` и в ней — `Sample1.gps`. Откомпилиру-

ем эту модель:

`Command/Create Simulation`

Перед началом моделирования закажем график одной из характеристик модели в Окне Графиков. *GPSS* обеспечивает большой набор встроенной статистики, или СЧА — см. список в разд. 12.4. В данном случае имеет смысл рассмотреть `Q$Barber`, представляющий длину очереди. Выберем команды

`Window/Simulation Window/Plot Window`

Появится диалоговый бокс — рис. 15.2.

Рис. 15.2. Диалог настройки графика

Последовательно введем информацию в поля высвеченного шаблона:

- в поле `Label` — наименование кривой (легенда);
- в поле `Expression` — выводимая зависимость (в нашем случае `Q$Barber`);
- в поле `Title` — наименование графика в целом;
- в поле `X` — максимальное значение абсциссы (системного времени) 40000, а в поле `Y` — максимальное значение длины очереди (200).

Предельное значение системного времени определялось из расчета завершения обслуживания 100 заявок, приходящих с интервалами 400 единиц времени. Отметим, что в полях наименований допустимо использование кириллических шрифтов.

Закончив ввод, щелкнем по кнопкам [Plot] и [OK]. На экране увидим координатные оси. Собственно график появится после начала моделирования. Он показан на рис. 15.3.

Рис. 15.3. График изменения очереди

Введем команду verb|START 100|. Через несколько секунд (в зависимости от быстродействия компьютера) моделирование закончится. *GPSS/W* выдаст в статусной строке (внизу Главного Окна) сообщения `The Simulation has Ended` и затем — `Report is Complete`. Отметим, что значение `Q$Barber` на графике меняется скачкообразно. Отчет можно анализировать, прокручивая содержимое окна и изменяя его размеры либо выведя на печать. В разд. 14.6 уже обсуждалась его структура. Закроем окно, щелкнув по [x].

Рассмотрим некоторые результаты. Сделаем активным Окно Журнала, выберем `Command/SHOW` и напечатаем `C1`. В статусной строке будет выведено относительное системное время на момент окончания моделирования — то же, что в Журнале.

Аналогичным образом посмотрим СЧА `QM$Barber` — максимальную длину очереди в процессе моделирования. При желании можете проверить значения других СЧА (их списки по функциональным комплексам даны в главе 13).

GPSS/W позволяет просматривать состояние модели многими различными способами. Каждый из основных типов сущностей *GPSS* имеет окно для отображения их динамики. Можно открыть Окно Выражений, которое содержит список заявленных *PLUS*-выражений — в простейших случаях переменных или СЧА. Выберем

`Window/Simulation Window/Expressions Window`

и затребуем системное время, длину очереди и номер активного транзакта. В поле `Label` напечатаем `Clock`, в поле `Expression` — `AC1`. Щелкнув по соответствующим кнопкам, выберем опции `View` и `Memorize`. Первая из них позволит видеть значение выражения только в текущем прогоне, а вторая — использовать в будущих прогонах. Аналогичные операции нужно выполнить для остальных интересующих нас выражений. Соответственно в разделе `New Expression` наберем и введем метки и выражения

```
Barber Queue     Q$Barber
Act Trans        XN1
```

Закроем Окно Графиков и откроем Окно Устройств. Разместим два окна так, чтобы видеть оба. Введем команду `START 20000,NP` и, остановив процесс по [F4], пронаблюдаем результаты моделирования работы сильно загруженного парикмахера.

Закроем Окна Выражений и Устройств и откроем Окно Блоков. Используем это окно для ввода контрольных остановов на траектории транзактов. Для останова выберем блок DEPART. Щелкнем по иконке этого блока, а затем по иконке с изображением красного светофора Place (Поместить) отладочной панели. Сделаем активным Окно Журнала. Передвинем это окно так, чтобы его можно было видеть вместе с Окном Блоков.

Теперь выберем Command/Custom и введем в диалоговое окно команду трассировки Trace. С этого момента каждый вход *данного* активного транзакта в новый блок вызовет появление в журнале (и Окне Журнала) трассировочного сообщения. Дело в том, что блок Trace был создан в режиме ручного моделирования, то есть для разового использования. Он не становится постоянной частью модели и не повлияет на прохождение других транзактов. Выполним несколько шагов моделирования повторным нажатием [F5] и посмотрим в Окне Журнала сообщения трассировки. Журнал можно распечатать в любой момент, сделав активным Окно Журнала и выполнив команду File/Print.

Теперь *удалим* контрольный останов. Для этого сделаем активным Окно Блоков, щелкнем мышью по блоку DEPART, а затем по иконке [Remove] отладочной панели. Один или несколько остановов можно удалить также, выбрав

Window /Simulation/Snapshot/User Stops.

Выделив нужный останов, следует нажать [Remove] и [OK]. Технология удаления *всех* заданных остановов очевидна.

Нажатием [F2] продолжим моделирование. Трассировочные сообщения будут генерироваться только для «меченого» транзакта — пока он не войдет в блок TERMINATE. Естественно, что сообщения о других транзактах, продвигаемых во время задержки меченого в блоке ADVANCE, не поступали.

Прервем моделирование. Через Command/Custom выдадим команду Report, Now. Немедленно будет создан Стандартный Отчет — текстовый файл с именем модели и расширением .gpr. При *завершении* моделирования по исчерпанию счетчика отчет формируется автоматически. Отчет можно рассмотреть в окне или вывести на печать через File/Print. При закрытии Окна Отчета выдается запрос о необходимости сохранения отчета и имени файла.

Закрытие всех окон возвращает нас в Главное Окно. Закрытие этого

окна или выбор `File/Exit` завершает сеанс работы *GPSS World*.

15.2 Технологический инструментарий

Рано или поздно вам потребуются дополнительные справочные сведения. Покажем, как получить доступ к ним. Откроем модель `Sample10`. Щелкнем дважды по слову `GENERATE`, чтобы оно оказалось *выделенным*, и нажмем [F1]. Система перечислит разделы вхождения `GENERATE` в Справочник. Выберем строку `Block Statements`, чтобы видеть определение блока. Прокручивая окно, просматриваем раздел.

Аналогично можно получить сведения по командам, процедурам, классам СЧА, вероятностным распределениям и т. д. Если в тексте нет предмета запроса, напечатайте в окне модели нужное слово (потребуется знать его написание по-английски), выделите его, получите справку, а затем удалите.

Через Главное Меню можно вызвать систематический Справочник (`Reference Manual`) и, последовательно разворачивая пункты иерархического меню, выйти на нужный его раздел. Имеется также алфавитный вход в Предметный указатель, который после набора ключевого слова открывает раздел того же Справочника.

Мини-справка по разделам Главного Меню выдается при задержке указателя мыши на названии раздела. Не отпуская кнопку мыши, нажмите [F1]. Вы войдете в `Help`, связанный с данным разделом меню.

Рассмотрим Установки Моделей. Набор установок привязывается к конкретной модели и может изменяться от одной модели к другой. Напомним, что в процессе компиляции выбранные установки от Модели передаются Имитации.

Для ручного моделирования очень полезна возможность назначения функциональных клавиш. Выберем `Edit/Settings` и в блокноте установок — `Function Keys`. Видно, что за каждой клавишей можно закрепить произвольный командный текст. Места в окне хватит, так как набранный текст при необходимости сдвигается влево за пределы окна. Текст, в частности, может иметь вид

```
INCLUDE "<имя_файла>"
```

Тогда одно нажатие функциональной клавиши приведет в действие произвольную последовательность блоков и команд, записанных в этом

файле. Стандартное закрепление функциональных клавиш описано в разд. 14.7.

Вернемся к нашей парикмахерской. Запишем в поле команд клавиши [F9] `SHOW Q$Barber` и закроем Блокнот Установок. Откомпилируем модель с этими установками и запустим имитацию командой `START 10000`. Прервем моделирование по [F4] и нажмем [F9]. Увидим длину очереди `Barber` в статусной строке Главного Окна и в Окне Журнала. Не удивляйтесь патологическому значению результата: в этой модели парикмахер не справляется с обслуживанием.

Обратим внимание на команду `SHOW`. Ее можно использовать как калькулятор. Посмотрите список математических функций в главе 12. Эти функции, значения СЧА и переменных пользователя можно комбинировать в сложные выражения — операнды `SHOW`. Для начала проверим простую арифметику. Вызовем `Command/SHOW` и наберем в диалоговом боксе `1234+5678/345#(64+94)` (напомним, что роль знака умножения играет #). Результат 3834.3594203 будет высвечен в статусной строке и занесен в журнал.

Изучим работу с остановами по ошибке. Выберем `Command/SHOW` и напечатаем 3/0. Описание ошибки `Division Range Error` появляется в Окне Журнала и в статусной строке. Закройте открытые окна и файл `Sample10`.

Создадим простую модель с очевидной ошибкой. Выберем `File/New Model` и напечатаем в непоименованном Окне Модели

```
TENERATE 10
TERMINATE 1
```

Откомпилируем модель. Открылось Окно Журнала с сообщениями о том, что:

- в первой строке система ждет «глагол» $GPSS$, то есть тип блока или команду;
- в этой строке записано `TENERATE 10`;
- трансляция закончена аварийно.

Вы можете немедленно вернуться в Окно Модели, заменить T на G и выполнить рекомпиляцию по Command/Retranslate.

Для перемещения между ошибками в большой модели можно использовать команды Search/Next Error (Previous Error). Курсор автоматически выставляется в месте допущенной ошибки.

Поговорим о выражениях *GPSS/W*. Выполним небольшую имитацию и исследуем результаты с помощью команды SHOW, ссылающейся на СЧА модели. Откроем Окно Выражений. Вычислим переменную Fancy, определенную как натуральный логарифм системного времени плюс квадрат максимальной длины очереди. Соответственно наберем

```
Fancy LOG(AC1)+(QM$Barber)^2
```

и щелкнем по [View]. Информация из поля New Expression скопируется в поле Window Contents. Нажмем [OK] и увидим в Окне Выражений ответное значение.

Можно создавать логические выражения со значениями 1 и 0 посредством операторов 'AND', 'OR', 'NOT'. Их составными частями могут быть отношения арифметических выражений. Не забывайте об апострофах вокруг логических операторов, не допускайте пробелов между операторами и операндами. Проверьте применимость буквенных и символьных обозначений операций отношения. Проконтролируйте правильность результатов, при необходимости вставьте круглые скобки.

По Edit/Expressions Window снова вызовем настройку Окна Выражений. Наберем в полях метки и выражения соответственно

```
X$Waiting   1+99#(Q$Barber'GE'6)
```

и щелкнем по [View] и [OK]. Это выражение имеет значение 1, если в очереди менее 6 заявок, иначе равно 100.

Теперь поработаем с датчиками случайных чисел. Выберем Command SHOW и напечатаем в диалоговом боксе RN2. СЧА вида RNj выдает случайное целое от 0 до 999 в статусную строку и журнал. Повторное обращение к тому же генератору даст другое число. Закрепим использованную команду вывода случайных чисел за клавишей [F8] и нажмем ее несколько раз. Получим серию *различных* случайных чисел.

15.3 Текстовый редактор

В *GPSS/W* входит мощный полноэкранный текстовый редактор с обычными функциями редакторов среды *Windows* — Cut,Paste,Copy,Search.

Его специфика заключается в средствах быстрого нахождения синтаксических ошибок, обнаруженных компилятором.

При работе с текстами различаются указатель мыши и курсор вставки — мерцающая вертикальная черта. Передвижение курсора вставки производится мышью или с помощью клавиш управления курсором; переход на начало/конец строки — по [Home]/[End] соответственно. Для коррекции набранного текста полезны команды из меню `Edit`: `Undo`, `Delete Line`, `Insert Line`.

Откроем модель `Sample2.gps` и попытаемся редактировать ее текст. Выделим строку, начинаемую с `ADVANCE`, одним из следующих способов:

- поместив курсор в начало строки и протянув указатель мыши поперек страницы при нажатой левой кнопке;
- щелкнув мышью слева от строки при указателе в форме наклонной стрелки.

Выделеннный текст показывается в инверсном виде (белыми буквами на черном фоне). Его можно копировать, вырезать, вставлять из буфера — как в *Word*'е. Щелкнем мышью слева от строки — выделение будет снято. Двойной щелчок мышью по *слову* выделяет это слово.

Теперь скопируем выделенный текст в буфер (`Clipboard`) командой `Edit/Copy` и вставим через `Edit/Paste`. Наконец, удалим вставленную строку по [Ctrl+X]. Функции копирования, вставки и удаления выделенного текста реализуются также клавишными комбинациями [Ctrl+C], [Ctrl+V], [Ctrl+X] соответственно. Повторите те же действия с помощью клавишных комбинаций, но оставьте блок `ADVANCE` перед `TERMINATE`. Замените операнд 400 на 200.

Функция отмены `Undo` — одноуровневая и распространяется только на непосредственно предшествующую команду.

Основой для новой модели может послужить старая — см. папку `Sample Models`. Из модели в модель могут переходить некие типовые фрагменты. Каждый такой фрагмент следует предварительно отладить и сохранить в отдельном файле. Фрагменты автоматически вставляются в модель по команде `INCLUDE "<имя_файла>"` на этапе компиляции. Директория для вставок по умолчанию — та же, в которой создается модель. В противном случае имя файла должно быть дополнено путем к нему.

Изучите другие опции меню Edit. Вставьте пустую строку после строки с курсором. Примените функцию Undo для отмены предыдущего изменения. Выберите опцию Font для смены шрифтов. Установленный по умолчанию шрифт Courier New является равноширинным и потому удобен для выравнивания колонок при наборе текста модели. Это выравнивание не является обязательным, но увеличивает наглядность программы.

Переключите язык страны на кириллицу. Переведите один из комментариев к операторам программы на русский язык и наберите русскоязычную версию комментария. Перекомпилируйте программу и убедитесь, что компиляция проходит без ошибок. Запустите модель по START 10000 — получите сообщение об аварийном завершении программы. Итак, *GPSS/W* вопреки естественным ожиданиям не допускает кириллический текст даже в комментариях.

Теперь займемся командой Search. Режимы Find/Replace и Find/Replace Box похожи на аналогичные команды обычных текстовых процессоров. Вы можете найти фрагмент, изменить его однократно или во всем тексте. Поиск идет от места курсора только в одном направлении и останавливается при достижении конца файла. Поэтому при организации поиска курсор следует помещать в начало файла.

Выполним команду Search/Find/Replace. Закроем рабочий файл Sample2.gps, отказавшись от сохранения изменений.

Поработаем с *закладками* (bookmarks), которые очень полезны при работе с большими кусками текста. Откроем модель bookmrk.gps:

```
; GPSS World Sample File - BOOKMRK.GPS
110 ****************************************************
120 *
130 * Модель телефонной системы                         *
140 *  by Gerard F. Cummings                            *
150
160 ****************************************************
170 *    Моделирование системы с повторными вызовами     *
180 *    Единица времени --- одна секунда                *
200 Sets     STORAGE 2
210 Transit  TABLE M1,100,100,20    ;Гистограмма транзита
220          GENERATE 100,60        ;Прибытие заявки
```

```
230 Again    GATE SNF Sets,Occupied ;Доступна ли линия?
240          ENTER Sets             ;Выполняется соединение
250          ADVANCE 180,60         ;Разговор  3+/-1 мин
260          LEAVE Sets             ;Освобождение линии
270          TABULATE Transit       ;Табулирование
280          TERMINATE 1            ;Удаление транзакта
290 Occupied ADVANCE 300,60         ;Ожидание 5+/-1 мин
300          TRANSFER ,Again        ;Новая попытка
310 ****************************************************
```

Поместим курсор в строку 210 (номера добавлены для удобства ссылок на строки и для работы модели не нужны). Выберем `Search/Mark` (отметить). Выставим курсор в строку 300 и повторим команду. Этим мы установили в файле две закладки.

Теперь поместим курсор в середине строки 260 и выполним команду `Search/ Select to Bookmark`. Выделяется текст от позиции курсора до ближайшей по ходу просмотра закладки. При передвижении курсора вперед посредством клавиши выделение снимается.

Попытаемся двигаться по файлу с применением закладок. Выберем и несколько раз выполним команду `Search/Next Bookmark`. Убеждаемся, что просмотр происходит *по кольцу* — при выходе на конец файла возобновляется с его начала.

Сделанные закладки удаляются командами `UnMark`, `UnMark All`. При *сохранении* файла будут сохранены и оставшиеся в нем закладки.

Обратимся к поиску и устранению *синтаксических ошибок*, выявленных компилятором. Откроем модель `error.gps` и откомпилируем ее. Отрегулируем размеры и разместим Окна Модели и Журнала так, чтобы были видны они оба. Щелкнем по Окну Модели, сделав его активным.

Воспользуемся поиском `Next Error`. Этот пункт меню поместит курсор в строке и на месте ошибки и прокомментирует ошибку в статусной строке Окна Модели. Выберем `Search/Next`. В связи с диагностикой вспомним правило: $PLUS$-выражения могут применяться в любом месте программы, но, находясь вне $PLUS$-процедур, должны заключаться в скобки. Теперь ясно, что мы должны вставить недостающую правую скобку.

Сообщение об ошибке в Журнале говорит нам, что вторая ошибка допущена в строке 13. Используйте `Search/Go To Line` для перехода к

нужной строке, набрав в диалоговом окне 13. Курсор установится в начале этой строки. Устраним ошибку, вставив пропущенную перед началом комментария точку с запятой.

Команда `Search/Previous Error` позволяет просматривать список ошибок в обратном направлении.

Выполним `Command/Retranslate` и запустим Имитацию командой `START 10,NP`. В окне Журнала появляется сообщение об ошибке фазы выполнения: попытка освобождения не захваченного устройства. Активируем Окно Модели. Используем `Search/Go To Line` для нахождения и исправления ошибочного имени устройства, упоминаемого в блоке `RELEASE` — `Robor` вместо `Robot`. Перетранслируем модель и убеждаемся, что имитация успешно завершается.

Ошибки *фазы выполнения* список не образуют. Они выявляются и устраняются по одной с помощью сообщений, идентифицирующих место и характер ошибки.

Закрываем все открытые окна и выходим из системы, не сохраняя внесенные в модель изменения.

15.4 Транзакты, GENERATE и TERMINATE

Каждая Имитация должна иметь по крайней мере один блок `GENERATE`. Пусть единица времени — секунда. Тогда для ежеминутной генерации транзактов можно записать:

```
GENERATE 60 ;Создать новый транзакт
```

При переопределении единицы времени на минуту мы можем представить генерацию транзактов посредством:

```
GENERATE 1 ;Создать новый транзакт
```

Важно, чтобы во всей модели использовался *единый* масштаб времени.

Начнем сеанс *GPSS/W*. Выберем `File/New` и напечатаем предложение `GENERATE` в окне модели. Транслируем нашу однострочную модель. Нажмем [Tab], напечатаем `GENERATE 60` и комментарий — `;Create a new Transaction`. Теперь выберем в меню `File` пункт `Save As`, в диалоговом боксе заменим `Untitled` на `Mymodel` и подтвердим решение о сохранении. Наберем `Command/Create Simulation`, а затем

`Window/Simulation Window/Blocks Window`

Увидим наш блок GENERATE в окне блоков. Блоки в окне нумеруются Транзакты, GENERATE и TERMINATE начиная с 1. В окне блоки обычно размещаются ниже своих предшественников. Если места по высоте нет, блок помещается в следующей колонке справа. Естественно, на размещение влияет размер окна.

Блок TERMINATE удаляет транзакт из модели. Имеется необязательный операнд, на который уменьшается «Счетчик завершений». Когда вы печатаете START 137, в упомянутый счетчик заносится 137. Если была команда TERMINATE 1, моделирование закончится после прохождения через систему 137 транзактов. По его итогам будет выдан стандартный отчет.

Добавим к нашей модели блок TERMINATE. Начнем в окне модели новую строку нажатием [Enter]. Затем напечатаем блок TERMINATE 1 с комментарием

`;Destroy a Transaction`

и выберем Command/Retranslate.

В этой «игрушечной» имитации заявки регулярно приходят и сразу же уходят. Посмотрим, однако, что происходит при моделировании.

Перед началом моделирования вспомним о двух множествах транзактов: ЦТС и ЦБС. Каждый транзакт имеет атрибут, который указывает, когда он должен попытаться войти в следующий блок. Если таймер не достиг этого момента, транзакт должен находиться в ЦБС. Каждый раз, как транзакт входит в блок GENERATE, создается новый транзакт и помещается в ЦБС со сдвигом момента его появления относительно таймера на интервал, заданный GENERATE.

ЦТС содержит множество транзактов, которым предстоит пройти хотя бы один блок при *текущем* значении таймера. Планировщик транзактов берет их из начала ЦТС по одному и пытается продвинуть как можно дальше. Когда в ЦТС транзактов больше нет, головной транзакт ЦБС переносится в ЦТС, и системные часы переводятся.

Посмотрим на ЦТС и ЦБС нашей модели. Сначала зададим условие останова процесса при каждой попытке входа в блок. Убедимся, что открыто Окно Блоков. Видим в нем два блока схемы. Щелкнем мышью по блоку GENERATE. На панели отладки Debug Toolbar в верхней части окна блоков увидим иконку Place (Поместить) в виде красного светофора. Щелчок по ней вставит STOP в блок GENERATE. Моделирование будет

останавливаться при попытке входа транзакта в GENERATE.

Перед стартом посмотрим, что мы имеем в ЦТС и ЦБС. Один из вариантов — выбрать Command/Custom и напечатать Report ,Now. Тогда внизу отчета будут выведены списки цепей. Теперь просмотрим упомянутые цепи через окна Snapshot. Прежде всего откроем эти окна, выбрав

```
Window/Simulation Snapshot/FEC Snapshot
Window/Simulation Snapshot/CEC Snapshot
```

ЦТС пуста, а в ЦБС содержится один транзакт, подготовленный блоком GENERATE. При запуске модели Транзакт 1 будет перемещен в ЦТС, а таймер примет значение 60.

Упомянутые окна в отличие от окон, выбираемых по команде Window/Simulation Window, являются статическими и отражают состояние *остановленной* модели. Их содержимое не меняется в процессе прогона до нового запроса. Закроем окна цепей, щелкнув по [x] в их верхних правых углах.

Теперь дадим команду START 100. Она установит счетчик завершений в 100, передвинет Транзакт 1 в ЦТС и начнет моделирование. Но мы запланировали останов. Поэтому процесс остановится перед первым входом в блок, и в Журнал запишется двухстрочное сообщение.

Дадим Транзакту 1 войти в блок GENERATE командой STEP 1 или нажатием [F5]. Просмотрим вышеописанным способом содержимое цепей ЦТС и ЦБС. В ЦБС содержится информация о транзакте, ожидающем входа в модель, и значении таймера на момент входа. Следующая колонка показывает номер принимающего транзакт блока. В нашем случае первый блок — GENERATE, а второй — TERMINATE. Транзакт 2 ожидает входа в блок GENERATE.

Теперь рассмотрим *расширенное* представление ЦТС. При активном окне ЦТС щелкнем по знаку [+]. Транзакт 1 вошел в блок GENERATE в момент M1=60 с, тем самым создав Транзакт 2, помещаемый в ЦБС и запланированный на 120 с. Работа блока GENERATE состоит в формировании нового транзакта каждые 60 с, что и показывают (с разных точек зрения) наши окна. Позднее мы увидим, что через окна доступны и *параметры* транзактов.

Закроем окна цепей и посмотрим на системные часы (таймер). Выберем Command/SHOW и напечатаем в диалоговом боксе стандартный атрибут AC1. Транзакт 1 в момент 60 будет двигаться в модели так далеко, как сможет, а затем система выведет его из ЦТС. Ему запланирована

ликвидация при входе в блок TERMINATE. Нажмем [F5] — Транзакт 1 уничтожается. Теперь ЦТС пуста. Таймер переводится на время 120 с ближайшего события, Транзакт 2 переносится из ЦБС в ЦТС и готов войти в блок GENERATE. ЦБС опустела; формируется и планируется на момент 180 с Транзакт 3.

Нажатием [F5] дадим Транзакту 2 войти в блок GENERATE. Выберем Window/ Simulation Snapshot/CEC Snapshot и нажмем [+]. Откроем также ЦБС. Подвигаем окна цепей, расположим их одно над другим. Удлиним оба окна настолько, чтобы можно было прочесть всю информацию. Видим, что Транзакт 3 находится в ЦБС и запланирован на момент 180, а транзакт 2 готов ко входу в TERMINATE. Когда он это сделает, то будет уничтожен, и до момента 120 никаких событий в системе не произойдет.

Вспомним однако, что Транзакт 1 уже вошел в блок TERMINATE. Это означает, что счетчик завершений уменьшился на 1. Набрав команду SHOW TG1, убеждаемся, что его текущее значение — 99. Это значение можно прочитать в статусной строке внизу главного окна, а также в Окне Журнала. Если мы удалим остановы и по [F2] продолжим моделирование, оно завершится автоматически при нулевом TG1.

Поупражняйтесь дополнительно в означивании функциональных клавиш, отображении движения транзактов в цепях, контроле значений таймера. Затем закройте все Snapshot-окна.

Теперь откроем Окно Выражений для системного времени, счетчика завершений и номера активного транзакта. Для этого потребуется ввести следующие метки и выражения:

```
Clock         AC1
Term Ct.      TG1
Act. Xact     XN1
```

Ввод выполняется последовательно через панель New Expression по нажатию кнопок [View] и [Memorize]. Выражения, которые вы хотите выводить, можно сохранить через View/Settings, щелкнув по разделу Expressions. Для ускорения событий удалим все остановы через меню

Window/Simulation Snapshot/User Stops

щелкнув по Remove All. Активируем окно блоков и, несколько раз нажав [F5], пронаблюдаем изменение таймера и счетчика завершений. Заметьте, что символ каждого блока мигает при входе в него транзакта. Затем

перейдите к автоматическому продолжению моделирования щелчком по иконке [Continue] или нажатием [F2]. Моделирование практически мгновенно завершится, и будет сформирован отчет.

15.5 Конец сеанса

Щелкнем мышью по [x] в верхнем правом углу окна модели и ответим «Yes» или «No» на вопросы в диалоговых боксах относительно сохранения Имитации и Модели. Для учета добавленных блоков следовало бы провести рекомпиляцию.

В этой модели с транзактами ничего не происходило. Для реализма надо было бы задерживать транзакты на время обработки — см. следующий раздел.

15.6 Блок ADVANCE

Здесь мы научимся использовать `ADVANCE` для моделирования временны́х задержек. Начнем сеанс *GPSS World*. Откройте одну из сохраняемых моделей или выберите `File/ New`. Напечатайте

```
GENERATE 60 ;Create a new Transaction
```

и в следующей строке

```
TERMINATE 1 ;Destroy a Transaction
```

Создайте Имитацию выбором `Command/Create Simulation`. Откройте окно блоков по цепочке `Window/Simulation Window/Blocks Window`.

Зададим останов при каждой попытке входа в блок `TERMINATE`, щелкнув мышью по этому блоку, а затем по иконке [Place] в панели `Debug Toolbar`, и вернемся в Окно Журнала. Выберем `Command/START` и в диалоговом окне заменим 1 на 1000. Счетчик завершения установится на 1000, и начнется имитация. Заданный нами останов сработает при первой попытке входа в блок `TERMINATE`. Посмотрим ЦБС и ЦТС, набрав `Custom`-команду `Report ,Now`. Транзакт `XN=1` находится в ЦТС, а `XN=2` — в ЦБС и готовится войти в блок `GENERATE`, когда часы покажут `BDT=120` с. Транзакт 1 в момент `M1=60` с отработал все свои входы. Часы переводятся, Транзакт 1 подлежит уничтожению в `TERMINATE`.

Продолжим моделирование. Сделаем активным окно блоков. Нажатием [F5] обеспечим вход в очередной блок. Посмотрим на часы, выбрав `Command/SHOW` и в диалоговом окне атрибут `AC1`. В строке состояния

Главного Окна видим, что текущее время есть 120.000 — время входа в модель второго транзакта.

Транзакт 1 ушел. Так как ЦТС опустела, Имитация перескочила на следующее планируемое время 120 с. Транзакт 2 выбирается из ЦБС и помещается в ЦТС, часы переводятся на 120. Заметьте, что Транзакт 1 не провел в модели ни секунды. Все его входы в блоки относились к одному и тому же моменту времени — 60 с.

Чтобы имитировать задержку транзакта в системе, мы должны снова поместить его в ЦБС. Для этого и нужен блок `ADVANCE` с операндом `A` — приращением времени, с которым транзакт будет помещен в ЦБС. Сделаем это для нашего второго транзакта. Прежде всего вставим блок `ADVANCE` после `GENERATE`. В Окне Модели поместим курсор в конец строки с `GENERATE` и нажмем [Enter] (выберем `Edit/Insert Line`). Наберем

```
ADVANCE  61 ;Processing time
```

Загрузим команду `SHOW`. Выберем `Edit/Settings`, перейдем на вкладку функциональных клавиш и справа от [F8] наберем `SHOW AC1`. Нажмем [OK]. Аналогично закрепим за [F9] команду `Report,No`. Убедимся, что Окно Журнала зафиксировало все сообщения. Выполним `Command/Retranslate`.

В предыдущем разделе Транзакт 2 вышел из модели на отметке 120 до прибытия третьего (момент 180). Теперь Транзакт 2 благодаря блоку `ADVANCE` останется в модели 61 с, и Транзакт 3 придет в модель раньше, чем уйдет второй.

В окне блоков ранее (до рекомпиляции) мы задали останов в блоке номер 2 — `TERMINATE`. Остановы связываются с *номерами* блоков, которые при рекомпиляции *не меняются*. Теперь вторым стал блок `ADVANCE`, и останов будет перед ним.

Начнем моделирование по `Command/START 1000`. Моделирование остановится при попытке Транзакта 1 войти в `ADVANCE`. Нажатием [F5] разрешим ему этот вход. Просмотрим ЦТС и ЦБС, запросив немедленный отчет нажатием [F9]. Транзакт 1 запланирован в ЦБС на момент 121, а Транзакт 2 в ЦТС — на момент 120. Нажатие [F8] позволит увидеть в строке состояния новое время: 120.000.

Нажмем еще два раза [F5] и снова [F8]. Поскольку Транзакт 2 был последним в ЦТС, часы были переведены в момент 121 переноса Транзакта 1 из ЦБС в ЦТС. Транзакт 2 находится в ЦБС и не будет активи-

рован до момента 181.

Продолжаем. Нажмем [F5] дважды и посмотрим цепи событий. По шагам Транзакт 1 покинул блок ADVANCE и сгинул в TERMINATE. Транзакт 3, будучи запланирован на момент 180, вошел в GENERATE. По [F8] смотрим текущее время. Транзакт 3 вошел в GENERATE; создан и запланирован на момент 240 Транзакт 4. Транзакт 3 будет пытаться войти в следующий блок — ADVANCE.

Разберемся в том, что случилось. Транзакт 3 должен быть помещен в ЦБС позади 2-го и 4-го на момент 241=180+61. Текущее время равно 180, и второй транзакт станет активным в момент 181. В интервале [180, 181] оба транзакта будут в ЦБС, ожидая части своих 61-секундных ADVANCE-интервалов. Это первая секунда биографии третьего транзакта и последняя — второго.

Снова нажмем [F5], [F8], [F9]. Транзакт 2 стал активным, текущее время — 181. Транзакт 2 готов покинуть модель, поскольку он только что оставил ADVANCE, и следующим блоком ему запланирован TERMINATE.

Повторно нажмем [F5], [F8], [F9]. Транзакт 2 покинул модель: пришел в момент 120, ушел в 181 — то есть, как и было задумано, провел в системе 61 с.

Рассмотренное иллюстрирует контроль положения всех находящихся в модели транзактов, которые ожидают или выполняют различные операции. Каждый из них знает, в какой следующий блок ему предстоит войти.

Рекомендуем поупражняться в прогнозировании состояния модели. Попробуйте менять ADVANCE. Задействуйте аргумент B — тогда задержки будут случайными (см. Справочник). Не забывайте, что после каждого изменения нужна рекомпиляция.

15.7 Устройства. Блоки SEIZE и RELEASE

В блоке ADVANCE могло задерживаться несколько транзактов. Но задержка должна связываться с *обслуживанием*, которое не всегда предоставляется нескольким транзактам одновременно. Клиент должен *монопольно* завладеть устройством (Facility). Для моделирования таких процессов нужны блоки SEIZE и RELEASE, а для их просмотра — Окно Устройств.

Устройство имеет несколько атрибутов, важнейший из которых — владение (ownership). Владельцем устройства может быть только один транзакт. Он вступает в права собственности, входя в блок SEIZE с указанным в операнде A именем (номером) устройства. На время занятости другие транзакты, претендующие на овладение этим устройством, не могут продвигаться по диаграмме модели. Они удаляются из ЦТС и помещаются в связанную с данным устройством цепь задержки (ЦЗ). Транзакт владеет устройством, пока он не будет перенесен из ЦБС в ЦТС и не войдет в блок RELEASE с указанным в аргументе A именем устройства.

Проверим эти блоки в работе. Начнем новый сеанс *GPSS/W* с команды File/Open и выберем из списка моделей seizemod, описывающую работу магазина. В Окне Моделей увидим:

```
*******************************************************
     GENERATE 20       ;Прибывает покупатель
     SEIZE SALESMAN    ;Покупатель захватывает продавца
     ADVANCE 41        ;Делается покупка
     RELEASE SALESMAN  ;Продавец освобожден
     TERMINATE 1       ;Покупатель покидает магазин
*******************************************************
```

Сформируем Имитацию по Command/Create Simulation и запланируем останов в блоке GENERATE. Откроем Окно Блоков:

Window/Simulation Window/Blocks Window

и щелкнем по блоку GENERATE, а затем по иконке [Place] в панели отладки. Нажмем [OK]. Далее выберем Command/START, занесем в счетчик завершений 1000. Для входа в GENERATE первого транзакта нажмем [F5]. Посмотрим в Окно Журнала. Сообщение покажет, что первый транзакт может войти в блок SEIZE — поскольку «устройство» (продавец) свободно.

Проверим ЦТС и ЦБС. Транзакт 1 готов войти в блок SEIZE, Транзакт 2 находится в ЦБС и войдет в модель в момент 40. Активируем Окно Блоков. Нажатием [F5] дадим Транзакту 1 завладеть устройством. Транзакт 1 пока еще остается активным. Откроем Окно Устройств:

Window/Simulation Window/Facilities Window

Найдем заголовок «Owner XN» и убедимся, что владелец — Транзакт 1. Он готов начать 41-секундный интервал владения, предписываемый очередным блоком ADVANCE. Нажатием [F5] сделаем еще шаг моделиро-

вания, по [F8] посмотрим на часы (40). В ЦБС увидим, что устройство будет занято до момента 61.

Перейдем в Окно Блоков. Нажатием [F5] двинем активный (2-й) транзакт в очередной блок. Теперь этот транзакт пытается захватить устройство, но оно еще занято первым. По [F5], [F8] делаем еще шаг и просматриваем цепи событий.

Транзакт 2 пытается войти в блок `SEIZE`, но получает отказ и помещается в цепь задержки «устройства», имитирующего продавца — см. поле `Delay`. В Окне Устройств можно видеть, что в цепи задержки устройства `Delay Chain` находится один транзакт. Выбрав `View/Entity Details`, видим укрупненную иконку, белый квадратик в которой показывает, что очередь мала.

Снова нажмем [F5], [F8]. Удаление Транзакта 2 из ЦТС опустошило ее. Так как Транзакт 3 — первый в ЦБС, системные часы переводятся на момент 60. Обслуживание Транзакта 1 заканчивается в момент 61, когда он готов освободить (`RELEASE`) устройство.

Активируем Окно Блоков и нажмем [F5]. Транзакт 1 отдал право владения устройством, и оно освободилось. Так как Транзакт 2 — первый в цепи задержки этого устройства, ему автоматически отдается право владения, и он входит в блок `SEIZE`. Теперь в цепи задержки находится Транзакт 3, ожидающий возможность занять устройство. В Окне Устройств (режим `View/Entity Details`) видим, что устройством владеет Транзакт 2, и еще один ждет в очереди. Транзакт 2 далее войдет в `ADVANCE`, где потеряет статус активного. Однако предварительно Транзакт 1 покинет модель через `TERMINATE`.

Снова нажмем [F5], [F8]. Транзакт 1 ушел. Текущее время равно 61, Транзакт 2 еще не начал свой 41-секундный интервал. Если бы мы не вставили блок `SEIZE`, Транзакт 2 не задерживался бы перед входом в `ADVANCE`, и оба были бы помещены в ЦБС, то есть фактически одновременно занимали бы одно и то же устройство. Отметим, что когда транзакт находится в цепи задержки, его нет ни в ЦТС, ни в ЦБС. Это уменьшает трудоемкость просмотра указанных цепей.

Перед выходом из системы закройте все открытые окна.

15.8 Очереди и таблицы. Блоки QUEUE, DEPART

Здесь мы рассмотрим средства получения динамических гистограмм в процессе моделирования.

В отличие от обсуждавшихся ранее, блоки QUEUE и DEPART не влияют на продвижение транзактов: они только обеспечивают сбор статистики. Главным атрибутом очереди QUEUE является ее содержимое (текущая длина) Q$<имя>. При входе транзакта в блок QUEUE она увеличивается, при входе в блок DEPART — уменьшается. Статистика ведется автоматически, доступна во время моделирования и включается в Стандартный Отчет. В Окне Очередей можно видеть ее текущее состояние. Дополнительно можно определить объекты QTABLE, которые можно наблюдать как гистограммы в Окне Таблиц.

Начнем сеанс с загрузки модели sampque.gps:

```
; GPSS World Sample File - SAMPQUE.GPS
************************************************************
* Моделирование парикмахерской
* Время - в минутах
************************************************************
      GENERATE (Exponential(1,0,6.5))  ;Очередной клиент
      QUEUE Barber                     ;Начало ожидания
      QUEUE Total_time                 ;Начало пребывания
      TRANSFER Both,Barb1,Barb2        ;Выбор незанятого
Barb1 SEIZE Barber1                    ;Обслуживание первым
      DEPART Barber                    ;Конец ожидания
      ADVANCE 10,2.5                   ;Стрижка
      DEPART Total_time                ;Конец пребывания
      RELEASE Barber1                  ;Освобожд. парикмахера
      TRANSFER ,Next                   ;Смыкание ветвей
Barb2 SEIZE Barber2                    ;Обслуживание вторым
      DEPART Barber                    ;Конец ожидания
      ADVANCE 13,4                     ;Стрижка
      DEPART Total_time                ;Конец пребывания
      RELEASE Barber2                  ;Освобожд. парикмахера
Next  SAVEVALUE Ave_Queue,QT$Barber    ;Сохр. среднего
      TERMINATE 1                      ;Уход клиента
```

Это класическая модель парикмахерской с двумя цырюльниками. Блоки `QUEUE Barber` и `DEPART Barber` определяют начало и конец временнóго интервала пребывания в очереди по имени `Barber`. При «бутерброде» `QUEUE` — `SEIZE` — `DEPART` будет автоматически регистрироваться время ожидания каждого транзакта (до входа в блок `DEPART`). Транзакты, которым удастся завладеть парикмахером немедленно, будут иметь нулевое время ожидания. Средняя длина очереди и среднее время ожидания с учетом и без учета нулевых входов автоматически включаются в стандартный отчет. В нашу модель дополнительные блоки `QUEUE` и `DEPART` включены, чтобы иметь данные о *полном* времени пребывания в парикмахерской — ожидание плюс собственно стрижка.

Выполним компиляцию по `Command/Create Simulation`. Откроем Окно Очередей `Queues Window` (рис. 15.4) и расположим окна так, чтобы можно было одновременно видеть Окно Модели и Окно Очередей. Пока нет необходимости видеть текст в Окне Моделей. Очереди начнут отображаться, когда произойдет первый вход в блок `QUEUE`.

Zero Entry Count	Maximum Content	Average Content	Average Time (+0)	Average Time (-0)	Retry Chain
53	10	2.855	17.601	20.874	0
0	12	4.659	28.726	28.726	0

Рис. 15.4. Окно Очередей — детальный вид

Окно Очередей мы можем просматривать двумя способами. Общий обзор даст качественное представление о длинах всех очередей в модели и хорош для быстрого выявления критических узлов моделируемой системы. Детальный (рис. 15.5) дает всю информацию о состоянии конкретной очереди, которая входит в отчет.

Для начала посмотрим статистики, которые показывает детальное окно. Моделирование идет с двумя цырюльниками, к которым установлена общая очередь. Выберем `View/Entity Details` и, заготовив это окно, свернем его. Остановим моделирование по [F4]. Поместим курсор в конец последней строки модели и нажмем [Enter]. В новой строке наберем:

```
Waittime QTABLE Barber,2,3,20 ;Waiting Times Table
```

Рис. 15.5. Окно Очередей — обобщенный вид

Наличие подобного определяющего предложения обязательно для построения гистограммы. Выполним рекомпиляцию и запустим модель командой `START 111111`. Остановим моделирование нажатием [F4]. Откроем Окно Таблиц через диалог, предлагающий имя таблицы. Расширим окно до удобного размера — рис. 15.6. Нажмем [F2] (продолжение моделирования), чтобы гистограмма могла обновляться. Ее вертикальная ось — число транзактов, горизонтальная — диапазоны времен ожидания. Ожидание не более 2 мин добавит 1 к первому счетчику, учитывающему отрезки времени от 0 до 2.

Теперь посмотрим, как меняются времена ожидания в нашем примере. Заметим, что среднее время ожидания около 18 мин; некоторые клиенты ждут более 56 мин. Затянутый хвост и J-образный вид диаграммы согласуются со значением коэффициента вариации распределения (больше единицы). Когда насмотритесь, закройте окно и переключитесь на Окно Блоков. Наблюдая в пошаговом режиме за текущим содержимым первого блока `TRANSFER` (порядковый номер 4), видим, что число транзактов меняется, поскольку транзакты ждут в этом блоке освобождения одного из двух цырюльников (возможности войти в блок `SEIZE`). Закрываем Окно Блоков. Теперь посмотрим Окно Устройств:

`Window/Simulation Window/Facilities/Window`

GPSS/W проверяет все цепи, связанные с устройством, чтобы видеть, кто будет следующим владельцем. Когда оба парикмахера заняты, транзакты окажутся не в цепях задержки парикмахеров, но в цепи повторных попыток, связанной с блоком `TRANSFER`.

Снова посмотрим на Окно Модели. Расширим его, чтобы видеть текст модели. Заметим, что метки для блоков `SEIZE` и устройств слегка

различаются. Однако мы можем давать очередям, памятям, устройствам и т. д. одинаковые имена — например, использовать очередь и устройство с одним и тем же именем `Barber`.

Посмотрим на другую гистограмму. Прервем моделирование по [F4] и активируем окно моделей. В нашей модели первый парикмахер тратит на стрижку случайное время от 7.5 до 12.5 мин — операнд `A` задает среднее, а `B` определяет полуразмах. Аналогично второй тратит от 9 до 17 мин. В обоих случаях используется равномерное распределение. Его гистограмма должна иметь вид прямоугольника. Проверим это. Вставим блоки `QUEUE` и `DEPART` для сбора статистики о *чистом* времени стрижки. Откроем Окно `Blocks Input`. В Окне Моделей вставим перед первым `ADVANCE` строку

```
QUEUE Haircut ;Length of cut - Barber 1
```

Поместим после `ADVANCE`

```
DEPART Haircut ;Length of cut - Barber 1
```

и, наконец, в конце модели —
`Cuttimes QTABLE Haircut,7.5,0.5,12 ; Barber1 times Histogram`

Это определит нашу гистограмму. Выполним рекомпиляцию, запустим Имитацию командой `START 111111` и через некоторое время остановим модель нажатием [F4]. Откроем Окно Таблиц и увидим гистограмму рис. 15.6. В диалоговом боксе `Cuttimes` уже выбрана. Хотя гистограмма и выглядит не вполне правильной, при достаточно длительном моделировании она станет близка к прямоугольной — см. рис. 15.7. Среднее время стрижки сходится к расчетным 10 мин. Наглядевшись на гистограмму, прекращаем моделирование и закрываем все окна.

Рис. 15.6. Гистограмма распределения времени ожидания

Рис. 15.7. Гистограмма распределения длительности стрижки

15.9 Команды GPSS World

Команды *GPSS/W* используются для управления созданием и выполнением Имитаций, а также для определения объектов модели. Список команд приведен в разд. 12.1.1. Вы должны попытаться использовать каждую команду минимум по одному разу.

Начните сеанс *GPSS/W* и откройте файл `sample2`. Откомпилируйте его. Теперь действуйте по собственному усмотрению, а в случае затруднений обращайтесь к `Help`'у. Вы заметите, что некоторые команды имеют много опций, которые вы не использовали. В последующих разделах мы подробно обсудим команду `INCLUDE` и другие, связанные с процессом отладки.

Рассмотрим еще две идеи. Первая состоит в том, что каждое имя связано со значением. Откроем *новую* модель и напечатаем

`Var1 VARIABLE 3#3/3`.

Этим мы определили арифметическую переменную *GPSS* с именем `Var1`, которая сохраняет значение указанного выражения. Переменные могут состоять из констант или содержать другие переменные и СЧА, которые меняются по ходу имитации. Оттранслируем эту однострочную модель. Затем выберем `Command/SHOW` и в диалоговом боксе напечатаем `V$Var1`. После нажатия [OK] в строке состояния Главного Окна и в Журнале можно видеть значение переменной по имени `Var1`.

Здесь бывает общая ошибка. Наберем `Command/SHOW` и в диалоговом боксе напечатаем `Var1`. Выведено значение 10000!! Дело в том, что вы должны назначить значения *именам*, используемым в операторе `EQU`, прежде чем сошлетесь на них в выражениях. В противном случае *GPSS* сама назначит значение 10000 или более — что и произошло. Если вы хотите оценить значение *переменной*, не забывайте вставить `V$` перед ее именем.

Предупреждение. Если вы изменили имя после определения именованного объекта, это имя уже нельзя использовать для ссылки на старый объект. Вспомните также, что блоки нумерует только сама *GPSS*.

15.10 Ручное моделирование

Ручное моделирование — одно из самых мощных интерактивных средств *GPSS/W*. Любой оператор (кроме `GENERATE`) может быть «с пульта»

вставлен в существующую Имитацию. Имитация пытается заставить активный транзакт войти во временный блок, описываемый вставленным оператором. Такой оператор не становится постоянным элементом программы.

Проверим все это. Откроем файл sample2:

```
; GPSS World Sample File - SAMPLE2.GPS
*****************************************************
* *
* Моделирование парикмахерской
* *
*****************************************************
   GENERATE 300,100 ;Новый клиент
   QUEUE Barber     ;Начало ожидания
   SEIZE Barber     ;Захват или ожидание парикмахера
   DEPART Barber    ;Конец ожидания
   ADVANCE 400,200  ;Стрижка
   RELEASE Barber   ;Освобождение парикмахера
   TERMINATE 1      ;Уход клиента
*****************************************************
```

Мы должны активировать транзакт прежде, чем начнем ручное моделирование. Этого можно добиться заданием контрольного останова. Откомпилируем модель и откроем Окно Блоков. Щелчком мыши выберем блок GENERATE. Поместим в него останов щелчком по иконке [Place] отладочной панели. Теперь мы можем начать моделирование командой START 1000; оно немедленно остановится. В Окне Журнала появится сообщение об останове. Вернемся в Окно Блоков и удалим заданный ранее останов. Для этого щелкнем по блоку GENERATE и затем — по иконке [Remove].

Теперь активный транзакт имеется. Мы можем набрать для него любой блочный оператор. В Окне Модели выберем Command/Custom и в диалоговом боксе напечатаем Assign Price,99.95. Подтвердим команду — [OK]. Теперь активный транзакт имеет параметр Price со значением 99.95. Убедимся в этом с помощью команды SHOW, напечатав в диалоговом окне P$Price (параметр по имени Price). В Журнале и статусной строке увидим 99.95.

Теперь проверим некоторые дополнительные возможности ручного моделирования. Выберем Command/Custom и в диалоговом боксе напеча-

таем TRACE. Это даст тот же эффект, что прохождение активного транзакта через блок TRACE: включение индикатора трассировки активного транзакта. Последовательными нажатиями [F5] продолжите моделирование. Каждый вход в блок регистрируется в журнале. Если изменится время или номер транзакта, появятся соответствующие сообщения. Здесь мы увидим очень короткую трассу, поскольку будет прослеживаться только один транзакт — вставленный блок TRACE сработал однократно и исчез. Если вам нужна трассировка *всех* транзактов, придется вставить TRACE в модель и перетранслировать ее.

Выполните десяток шагов и распечатайте содержимое журнала. В сообщении о начале трассировки вы увидите номер *меченого* транзакта, для которого заказана трассировка. Сопоставляя отдельные выдачи, убеждаемся, что сообщения о текущем блоке делаются только для меченого транзакта.

Предложения, часто используемые при ручном моделировании, могут быть загружены в функциональные клавиши.

Подытожим возможности ручного моделирования:

- Вставка любого блочного оператора, кроме GENERATE, в текущую Имитацию.
- Создание дополнительных транзактов с помощью блока SPLIT.
- Изменение маршрутизации посредством TRANSFER.
- Модифицикация параметров через блок ASSIGN.
- Проверка состава групп с помощью EXAMINE или SCAN.
- Изменение готовности устройств блоками FUNAVAIL и FAVAIL.
- Активация любого блока модели через EXECUTE.

Попытайтесь их использовать и проконтролируйте результаты через динамические окна.

15.11 Графические окна GPSS World

GPSS/W позволяет видеть процесс имитации и взаимодействовать с ним посредством любого из графических окон, многие из которых име-

ют более одного режима. Десять окон обновляются в процессе моделирования, а остальные являются окнами снимков (Snapshots), которые позволяют фиксировать мгновенное состояние Имитации. Имеются следующие динамические окна:

Blocks — блоков;

Expressions — СЧА, переменных и выражений, значение которых пользователь хочет отслеживать;

Facilities — устройств;

Logicswitches — логических ключей;

Matrix — матриц;

Plots — графиков;

Queues — очередей;

Savevalues — сохраняемых значений;

Storages — памятей;

Table — таблиц в форме гистограмм.

При создании Имитации автоматически открывается Окно Журнала, позволяющее видеть протокол работы системы. Все графические окна позволяют взаимодействовать с Имитацией. С большинством перечисленных окон мы уже работали. Теперь познакомимся с ними всеми систематически и более детально.

Загрузим файл sample9.gps с примером, специально подготовленным для демонстрации работы с графическими окнами.

```
*****************************************************************
; GPSS World Sample File - SAMPLE9.GPS Model to graphics windows
*****************************************************************
Pool    STORAGE 400                 ;Определение памяти Matrix1
MATRIX ,5,5                 ;Определение матрицы Transit TABLE
M1,200,200,20                       ;Гистограмма времен
                                    ;ожидания и обработки
        GENERATE (Exponential(1,0,100)); Показат. распределение
        JOIN Maingrp                ;Транзакт -> к Maingrp
        JOIN Numgrp,9999            ;Добавление 9999
                                    ;->  к числовой группе Numgrp
```

```
        SAVEVALUE Addup+,1          ;Всего транзактов в модели
        ASSIGN Param_1,232          ;Означивание параметра
        JOIN Numgrp,P$Param_1       ; к числ.группе Numgrp
                                    ;значения параметра Param1
        LOGIC S Switch_1            ;On - Switch_1
        MSAVEVALUE Matrix1,2,2,QA$Tot_Process ; Ср. длина очереди
                                    ;в клетку матрицы
        QUEUE Tot_Process           ;Начало ожидания обработки
        SEIZE Facility1             ;Захват Facility1
        LINK Chain1,FIFO,Nxtblk     ;->  в ПЦ, если устр-во занято
Nxtblk  SEIZE Facility2             ;Захват Facility2
        SEIZE Facility3             ;Захват Facility3
        QUEUE Process_Time          ;Начало обработки
        ADVANCE 100,(Exponential(1,0,50)) ; Показат. задержка
        DEPART Process_Time         ;Окончание обработки
        TABULATE Transit            ;Время пребыв. в гистогр. Transit
        RELEASE Facility1           ;Facility1
        ADVANCE 20                  ;Задержка для Facilities 2 и 3
        RELEASE Facility2           ;Освобождение Facility2
        ADVANCE 10                  ;Доп. задержка для Facility3
        RELEASE Facility3           ;Освобождение Facility3
        DEPART Tot_Process          ;Конец пребывания
        UNLINK Chain1,Nxtblk        ;Вывод ожидающих транзактов из ПЦ
        ENTER Pool,100              ;Занятие 100 ед. в памяти Pool
        LOGIC R Switch_1            ;Off переключателя
        LEAVE Pool,50               ;Освобождение 50 единиц Pool
        SAVEVALUE Collect-,1        ;Демонстрация отрицат. значения
        REMOVE Maingrp              ;Удаление транзакта из Maingrp
        ADVANCE 10                  ;Задержка на 10 ед. времени
        LEAVE Pool,50               ;Освобождение еще 50 единиц Pool
Finis   TERMINATE 1                 ;Уничтожение транзакта
*******************************************************************
```

Откомпилируем этот файл. В диалоговом боксе Command/Custom напечатаем Stop 5: пятый транзакт будет остановлен, когда станет активным. Откроем Окно Блоков и в меню View/Entity Details снимем птичку, если она была. Увидим обзорный вариант окна блоков — см. рис. 15.1. Это взгляд «сверху» на поток транзактов в целом, с которого и должна начинаться отладка модели. Бóльшие подробности можно видеть при детальном обзоре.

Запустим моделирование, задав START 11111,NP. Оно остановится по вышеуказанному условию. Перейдем к пошаговому режиму. При каждом нажатии [F5] мы сможем наблюдать продвижение транзактов из блока в блок. На первом шаге Транзакт 5 входит в блок GENERATE, который при этом подсвечивается. Теперь пошлем транзакт прямо в блок TERMINATE средствами ручного моделирования. Выберем в Главном Меню Command/Custom и в диалоговом боксе напечатаем TRANSFER,Finis. Нажмем завершающий набор команды [OK]. После очередного нажатия [F5] увидим, что транзакт вошел в блок с меткой Finis (он обводится пунктирной рамкой). На этапах разработки и отладки модели Окно Блоков дает быстрый и легкий путь обнаружения проблем с продвижением транзактов. Можно проверить альтернативные пути и их влияние на поведение модели.

Поскольку траектория 5-го транзакта закончена, условие STOP теперь удалено из системы. Введем условие нового останова в блок ADVANCE (15-й). Для этого щелкнем мышью по этому блоку, а затем по иконке [Place]. Далее нажатием [F2] продолжим моделирование. Оно остановится по заданному условию. Несколько раз нажав [F5], проследим за продвижением транзакта. Откроем Окно Журнала. Выполним еще несколько шагов и пронаблюдаем их запись в журнале.

Снимем все остановы. В Главном Меню выберем

Window/Simulation Snapshot/User Stops

Щелкнем по кнопкам [Remove All] и [OK]. Нажатием [F2] продолжим процесс моделирования и наблюдение за ним в окне блоков. Полезны оба варианта его детальности: иногда достаточно видеть, где скапливаются транзакты, в другие моменты требуется знать число транзактов в различных блоках.

Поработайте еще с некоторыми пунктами Главного Меню. Пункт File позволяет сохранить Имитацию в произвольный момент времени. File/Print обеспечивает распечатку активного окна. Попытайтесь распечатать окно, предварительно остановив моделирование по [F4]. Сохранив Имитацию в этот момент, вы сможете позднее распечатать любое из окон, связанных с текущим состоянием модели.

Если вам нужен быстрый доступ к некоторому окну, откройте его, а затем сверните (кнопка в верхнем правом углу окна). Окно разворачивается при щелчке по его иконке в статусной строке. Выполните эти действия. Помните, что свернутые окна продолжают обновляться, вслед-

ствие чего моделирование остается замедленным. К сожалению, иконки свернутых окон неразличимы, что затрудняет выбор восстанавливаемого окна.

Продолжим моделирование по [F2] или начнем его с начала (с загрузки модели `sample9.gps`). Откроем Окно Устройств. Выбором `View/Entity Details` переключимся с детального вида на обзорный.

Проследим, как меняется состояние устройства в процессе моделирования. Символ устройства меняется от серого, когда оно свободно, до красного, когда занято. Очередь фактически является счетчиком транзактов в связанной с устройством цепи задержки. Длина очереди показывается высотой белого прямоугольника в правой стороне иконки (малая при очереди от 1 до 5, средняя — от 6 до 10, большая при длине очереди более 10).

Остановим моделирование по [F4]. Выберем `Command/Custom` и дадим команду перевода устройства 1 в состояние неготовности:

`FUNAVAIL Facility1`.

Иконка устройства стала *желтой*. Перейдем к режиму детального просмотра: `View/Entity Details`. Просматриваем цепь задержки в Окне Устройств. Выбираем в Главном Меню команду `START` и заменяем счетчик завершений на 150. По [OK] запускаем процесс и наблюдаем очередь транзактов, накапливаемую в цепи задержки устройства. Знакомимся с обзорным представлением состояния устройств.

Позволим устройству обрабатывать ожидающие транзакты. Снова выберем `Command/Custom` и дадим команду `FAVAIL Facility1`. По [F2] продолжаем моделирование, наблюдая уменьшение очереди.

Теперь перейдем к Окну Матриц. Нажатием [F4] остановим моделирование. Закроем Окно Устройств, щелкнув по кнопке [x] в его правом верхнем углу. *GPSS/W* позволяет использовать матрицы, имеющие до 6 измерений. В многомерном случае вы можете выбрать для отображения интересующее вас двумерное сечение. В окне `Command/Custom` определим шестимерную матрицу, напечатав

`Matrix2 MATRIX ,2,3,4,2,2,2`.

Теперь вызовем Окно Матриц. Открывается диалоговый бокс, который запрашивает имя матрицы. Выберем из ниспадающего меню `Matrix2`. Экстент (протяженность) по всем измерениям дается под именем матрицы. В боксе `Cross Section` нужно указать, какие измерения будут выбраны как строки и столбцы в Окне Матриц. Для оставшихся измерений следует задать фиксированные значения индексов в крайней правой

колонке диалога (по умолчанию — единицы). Для многомерной матрицы можно открыть несколько различных сечений (каждое в своем окне).

Загрузим модель и откомпилируем ее сначала. Выберем `Command/START`, заменим 1 на `111111,NP` и нажмем [OK]. Открыв Окно Матриц, войдем в диалог (см. рис. 15.8) выбора выводимого сечения `Matrix1`. Подтвердим вывод заявленного сечения. Проследим за изменением ее элемента [2,2] при прохождении транзактов через блок `MSAVEVALUE`, который помещает в этот элемент среднюю длину очереди. Наблюдаем медленную стабилизацию этой величины, начиная со старших знаков.

Рис. 15.8. Диалог вывода матрицы

Теперь изменим один из элементов матрицы в режиме ручного моделирования. Выберем `Command/Custom` и в диалоговом боксе зададим

```
MSAVEVALUE Matrix1,3,3,1000
```

В сечение выводятся только два смежных столбца матрицы. С помощью линейки горизонтальной прокрутки добиваемся видимости элемента (3,3) и убеждаемся, что команда присваивания ему значения 1000 сработала. Можно провести еще несколько экспериментов в режиме ручного моделирования. Помните действие последнего: активный транзакт извлекается из модели, проводится через заданный блок и возвращается в исходное место. Останавливаем моделирование по [F4] и закрываем Окно Матриц.

Займемся памятями (`Storages`). Начнем моделирование сначала и откроем Окно Памятей (`Storages Window`). После детального просмотра перейдем к обзорному. Остановим моделирование по [F4]. При частичном использовании памяти ее иконка закрашена наполовину, при полном — целиком. Попытайтесь изменить это состояние вручную.

Выведем из готовности память `Pool` командой `SUNAVAIL Pool`. Ее иконка стала желтой. Перейдем к детальному просмотру по команде `View/Entity Details`. В диалоге команды `START` заменим 1 на 150. При пошаговом продолжении моделирования наблюдаем накопление очереди в цепи задержки выбранной памяти.

Аналогичным образом сделаем память работоспособной (команда `SAVAIL`). *GPSS World* немедленно выделяет всем транзактам цепи задержки затребованную ими память, но продвигает их последовательно. Именно поэтому вы видите сразу несколько транзактов, вошедших в блок `ENTER`, и уменьшение длины цепи задержки на то же число. Продолжив моделирование по нажатию [F2], видим, как уменьшается очередь.

Закроем Окно Памятей и откроем Окно Таблиц. Для заказа гистограммы достаточно определить ее командой `TABLE` или `QTABLE`. Начнем моделирование сначала. Загрузим и откомпилируем модель. Видим в окне `Transit`, но гистограммы еще нет.

Начнем моделирование и будем следить за формированием гистограммы. Убедитесь, что размер окна достаточен для отображения столбцов. Проследите, как меняется гистограмма: возрастают счетчики классов, меняются среднее и среднеквадратическое отклонение. Последние в нашей модели практически равны, что свидетельствует о близости распределения транзитного времени к показательному.

Прервем моделирование, закроем Окно Таблиц. Запустим модель сначала или продолжим моделирование. Будем строить графики неболь-

шого числа переменных (допускается до 8 в каждом окне). Начать построение графика можно и по ходу моделирования, но лучше — после останова.

Войдем в Окно Графиков. Открывшееся Окно редактирования графиков рис. 15.2 имеет сходство с Окном Выражений. Для ввода каждой порции информации нужно щелкнуть мышью в начале соответствующей строки. Последовательный переход от бокса к боксу обеспечивается [Tab]. Нажатие [Enter] или [OK] завершает заполнение бокса. Поле метки должно содержать метку, используемую в легенде (пояснении) внизу графика. Занесем туда текст «Используемая память» (кириллица в метках графиков работает). В окне выражений, которое указывает первую отображаемую переменную, напечатаем «Память» и «S$Pool». Нажатием [Plot] перенесем эту информацию в раздел `Window Contents`, нажатием [Memorize] — в раздел запоминаемых настроек.

Теперь введем следующую пару «метка-выражение». Зададим в поле метки «Время обработки», в поле выражения «QT$Process Time». Щелкнем по кнопкам [Plot] и [Memorize] в боксе `New Expression`. Процесс добавления кривых можно продолжить. В поле `Plot Title` выберем имя, которое опишет обе запланированные к отображению переменные, именно размер очереди и среднее время пребывания — от генерации заявки до уничтожения транзакта. Итак, напечатаем «Используемая память и время обработки».

Для каждого графика должен быть задан диапазон аргумента — времени. При малом диапазоне график будет разбит на страницы по горизонтали, при слишком большом — потеряет наглядность. Практически диапазон подбирается методом проб и ошибок. Для не слишком сложных моделей без перегруженных узлов можно оценить максимальное время как средний интервал между транзактами, умноженный на счетчик завершений. Следует учитывать, что для перерисовывания графика хранится ограниченное число точек (по умолчанию — 10000 для каждого выражения). Если вы изменили размер окна и левая часть графика не перерисована, нужно увеличить число точек в меню `Settings`.

Выберем диапазон времени по оси `X` от 0 до 10000, по оси `Y` — от 0 до 200. Запустив моделирование по `START 100`, получаем график рис. ??. Сделав активным Окно Графика, выводим его на печать и убеждаемся в дефектности распечатки — некоторые линии пропадают, будучи неотличимы друг от друга или белого фона. Для практической работы с

графиками $GPSS/W$ мы рекомендуем следующую технологию:

- По [PrtScr] копируем график в `Clipboard` — буфер $Windows$.
- Сворачиваем окно $GPSS/W$.
- Открываем в $Word$'е новый файл и выгружаем в него содержимое буфера.
- При необходимости обрезаем рисунок, редактируем яркость и контраст (см. руководства по $Word$'у).
- Распечатываем файл.

Именно она была применена для изготовления рис. 15.9. К сожалению, в $GPSS/W$ отсутствуют средства управления цветом и структурой линий, что может вызвать затруднения при черно-белом воспроизведении графиков.

Рис. 15.9. Пример графика

Два следующих окна — Ключей и Значений — весьма малы, но могут быть очень полезны. Снова загрузите и откомпилируйте модель sample9, запустите Имитацию и сразу остановите ее. Откройте окна Logicswitches и Savevalues и расположите их так, чтобы оба были видны полностью. Для каждого окна в момент его активности выберите режим: View/Entity Details. Вы увидите, что при переключении режимов в Окне Ключей нет большой разницы. В Окне Значений в детальном режиме выводятся конкретные числа, тогда как в обзорном — только информация о знаке числа (положительное, отрицательное или нуль). После начала моделирования нулевых значений не будет. На рис. 15.10 одно из этих окон представлено в обзорном режиме, а другое в детальном.

Рис. 15.10. Окна Ключей и Значений

Откроем Окно Очередей. Вспомним, что блоки QUEUE и DEPART просто собирают статистику и, в отличие от Устройств и Памятей, не

имеют связанных с ними цепей транзактов. Начнем моделирование и остановим его по [F4]. Продолжим моделирование в пошаговом режиме последовательными нажатиями [F5]. В обзорном режиме длина очереди качественно указывается высотой белого прямоугольника внутри иконки: нулевая, малая, средняя или большая. Если желаемые изменения долго не происходят, продолжите моделирование в автоматическом режиме по [F2]. Закройте Окно Очередей.

Следующее — Окно Выражений. Опять же начните моделирование сначала или продолжите остановленное. Откройте Окно Выражений. Для начала ввода метки и выражения надо щелкнуть мышью в начале соответствующей строки. Поля заполняются

аналогично процессу настройки графика, только кнопка [Plot] заменяется на [View]. Чтобы видеть в окне выражений системные часы (таймер), номер активного транзакта и длину очереди, мы должны занести в соответствующие поля:

```
системное время      AC1
активный транзакт    XN1
длина очереди        Q$Tot_Process
```

В окне `Edit Expressions` можно выбрать лишь часть выражений, значения которых вы хотите отображать. Для этого нужно в окне `Window Contents` выделить и удалить по [Remove] ненужные строки.

Аналогично можно задавать вывод выражений в Блокноте Установок на странице `Expressions`. Они будут действительны для любого вызова *данной* модели. Попытайтесь это проверить. Убедитесь, что все «установленные» выражения после компиляции доступны как запоминаемые. Вы можете выделить те из них, которые хотите отображать в конкретном случае.

Окна Снимков (`Snapshot Windows`) действуют как фотокамера и сохраняют мгновенное состояние остановленной Имитации. Они обновляются только по запросу программиста. Для работы с этими окнами снова загрузите модель `sample9.gps`, откомпилируйте и запустите ее, а затем остановите по [F4]. Далее через меню

`Window/Simulation Windows/Simulation Snapshots`

откройте необходимые окна. На рис. 15.11 показаны с наложением четыре из них. В заголовках всех окон высвечено модельное время на момент

останова Имитации. Имейте в виду, что содержание окон определяется этим моментом и в вашем случае наверняка будет иным.

Рис. 15.11. Снимки цепей и групп

В окне `Future Events Chain Snapshot` мы можем, например, выделить по его номеру конкретный транзакт, определить его приоритет и ближайшее будущее: момент активизации, порядковый номер и сокращенное наименование запланированного блока. Указано также текущее местонахождение. Щелчок по [+] разворачивает информацию о параметрах — имена и значения.

Наберите в диалоговом боксе номер транзакта, выбранного для прослеживания в ЦТС (в данный момент этот транзакт заведомо находится в модели). Щелкнув по [+], войдете в окно, похожее на `Snapshot` ЦТС. Снова щелкнув по [+], увидите два значения, добавленных к Числовой Группе в ходе моделирования. Обычно значения добавляются через параметры транзактов или устанавливается некоторая начальная

совокупность их для использования при разветвлениях. В этой модели для наглядности каждый транзакт добавляет к группе одну и ту же пару значений.

15.12 Работа с внешними файлами

GPSS/W имеет блоки, позволяющие читать и писать информацию во внешние ASCII-файлы. Они полезны для чтения входной информации, редактирования отчета с применением функций обработки строк или вывода полученной при моделировании информации. В этом разделе решаются очень простые проблемы. Каждая модель поставляется с соответствующим файлом данных.

Имена испытательных файлов имеют специальную кодировку. Вот соответствие между буквами и действиями по вводу–выводу:

`CAT = Catenate (String Procedure)`, `SK = Seek`,

`IN = Insert SQ = Sequential`, `R = Replace ST = String`,

`RD = Read`, `W = Write`.

В `OPEN`-блоках этих примеров пути не указываются — во всех случаях подразумевается директория `Sample`.

Файл данных `tstskinw.txt` изначально содержит следующие записи:

```
This is line 1
This is the info on line 2
This is line 3
Line 4
```

Его копиями являются `tstappw.txt`, `tstskrw.txt`, `tstsqr2w.txt`. В таком виде они должны использоваться как входные. Для записи применяйте файл `tsttempl.txt`. Другой способ сохранения измененных файлов — копировать файлы `tst*.txt` в безопасное место перед началом работы. Их «штатное» место — в папке `Sample Models`.

Когда вы выполните предложенные задания, либо обратно скопируйте файлы `tst*.txt`, либо скопируйте файл `tsttempl.txt` в следующие: `tstskinw.txt`, `tstappw.txt`, `tstskrw.txt`, `tstsqrw.txt`, `tstsqr2w.txt`. Затем удалите `tstcatw.txt`, `tststw.txt`.

Файлы tstrd.txt и tststrd.txt используются только для считывания данных. Они не будут меняться, и их не надо инициализировать для повторения работы.

Начнем с последовательного считывания чисел из файла в матрицу *GPSS/W*. Загрузим модель tstrd.gps и будем работать с файлом данных tstrd.txt.

```
; GPSS World Sample File - TSTRD.GPS
************************************************
; Последовательно читает из файла
; значения в матрицу
; При конце файла переходит на метку Finis
************************************************
Total     MATRIX    ,10,1
          GENERATE  ,,,1
          OPEN      ("TSTRD.TXT"),,flag
Again     READ      Numero,,Finis
          SAVEVALUE Nrow+,1
          ASSIGN    Numrow,X$Nrow
          MSAVEVALUE Total,P$Numrow,1,P$Numero
          TRANSFER  ,Again
Finis     CLOSE      Prob,,Flag1
          TERMINATE 1
Flag      TERMINATE 1
Flag1     TERMINATE 1
************************************************
```

В файле tstrd.txt записаны (по одному числу в строке) 1 2 3 4 5 6 7.

Запустим моделирование и откроем Окно Матриц, чтобы наблюдать процесс заполнения матрицы. В диалоговом окне Open Matrix увидите матричное «Total» в ниспадающем боксе — единственном в этой модели. Скажем [OK] и выберем удобный размер окна. Зададим команду Command START 1. Моделирование пройдет очень быстро, заполняя матрицу вводимыми числами от 1 до 7. Осташиеся три позиции будут замещены заданными при инициализации нулями.

Кратко обсудим работу блоков ввода-вывода. Блок OPEN помещается в отдельном сегменте программы, так как файл открывается только один раз. Высокоприоритетный транзакт открывает читаемый файл

— это первое действие при моделировании. Он начинает считывание в цикле, охватывающем блок READ и четыре последующих. Каждое значение помещается в параметр Numero активного транзакта. Три следующих блока обновляют счетчик SAVEVALUE, который переносится в параметр и используется как номер строки матрицы. Затем блок MSAVEVALUE помещает значение, ранее считанное в параметр активного транзакта P$Numero, в соответствующую строку матрицы Total. Когда достигнут конец файла, транзакт посылается в блок CLOSE с меткой Finis, закрывая файл данных. Затем он входит в блок TERMINATE и завершает свою траекторию.

Если при выполнении операций OPEN или CLOSE произойдут ошибки, транзакт пойдет к блокам с метками Flag или Flag1 соответственно. Такую ошибку будет легко распознать, поскольку при нормальной работе эти блоки будут иметь нулевые счетчики входов. В последующих примерах этого раздела данные блоки не комментируются.

После считывания всех данных файл закрывается (CLOSE). Не нужно держать файлы открытыми дольше, чем это необходимо.

Теперь закроем Окно Матриц и загрузим следующую модель tstskrd.gps, которая покажет поиск данных в файле и размещение их в конкретных позициях матрицы.

```
; GPSS World Sample File - TSTSKRD.GPS
************************************************
* Читает две отдельные текстовые строки файла
* с применением SEEK и сохраняет значения в матрице.
* Закрывает файл после считывания и записи
* второго значения
************************************************
Total     MATRIX     ,10,1
          GENERATE   ,,,1
          OPEN       ("TSTRD.TXT"),,Flag
          SEEK       4
          TRANSFER   ,DoRead
Again     SEEK       6
DoRead    READ       Numero,,Finis
          MSAVEVALUE Total,P$numero,1,P$numero
          TEST E     P$Numero,4,Finis
          TRANSFER   ,Again
```

```
Finis     CLOSE      Prob,,Flag1
          TERMINATE  1
Flag      TERMINATE  1
Flag1     TERMINATE  1
************************************************
```

Исходные данные считываются из того же файла, что и в предыдущем примере.

Выполним компиляцию и откроем Окно Матриц. Диалоговое окно имеет правильное имя матрицы. Дадим команду START 1. На этот раз только числа 4 и 6 будут введены в матрицу в строках 4 и 6. Транзакт, который вызывает считывание, входит в блок SEEK, помещающий указатель текущей строки на строку 4. При входе в READ четвертая строка файла считывается. Значение засылается в матрицу и проверяется, нужно ли продолжать чтение. Выполняется следующее чтение, и в матрицу заносится второе число. Файл закрывается, моделирование кончается.

Закрываем Окно Матриц и Окно Модели. Загружаем tstskinw.gps, которая покажет запись в заданную строку с использованием *произвольного* доступа.

```
; GPSS World Sample File - TSTSKINW.GPS
****************************************************
; Тест случайной записи.
; Вставляет 123456 перед третьей строкой
; файла TSTSKINW.TXT
****************************************************
          GENERATE  ,,,1
          OPEN      ("TSTSKINW.TXT"),,Flag
          SEEK      3
          WRITE     123456,,Flag1,On
          CLOSE     Prob,,Flag2
          TERMINATE 1
Flag      TERMINATE 1
Flag1     TERMINATE 1
Flag2     TERMINATE 1
****************************************************
```

Начнем моделирование командой START 1. Посмотрим на измененный файл данных через меню File/Open. В диалоговом боксе последовательно выберем нужные нам тип (текстовый) и имя файла tstskinw и откроем этот файл. Видим, что перед третьей строкой файла вставлено 123456. Блок SEEK использовался для установки указателя текущей строки файла данных перед третьей строкой, куда блок WRITE пишет информацию. Без него данные были бы вставлены перед первой строкой файла.

Блоки OPEN, WRITE и CLOSE имеют альтернативные выходы в случае ошибок ввода-вывода. Тогда последующий CLOSE помещает код ошибки в параметр транзакта.

Теперь на модели tstappw.gps рассмотрим операцию записи в конец файла.

```
; GPSS World Sample File - TSTAPPW.GPS
****************************************************
* Добавляет 123456 к концу файла данных TSTAPPW.TXT.
****************************************************
          GENERATE   ,,,1
          OPEN       ("TSTAPPW.TXT"),,Flag
          SEEK       10000
          WRITE      123456,,Flag1,On
          CLOSE      Prob,,Flag2
          TERMINATE  1
Flag      TERMINATE  1
Flag1     TERMINATE  1
Flag2     TERMINATE  1
****************************************************
```

Выполним моделирование по START 1 и вышеописанным способом просмотрим измененный файл данных. Действительно, строка 123456 добавлена в конец файла.

Здесь мы иначе сделали две вещи. Во-первых, блок SEEK имеет аргумент 10000, заведомо превосходящий количество строк в файле. Во-вторых, блок WRITE находится в режиме вставки (указано On). В режиме вставки, когда текущая позиция находится за концом файла, данные будут помещены в конец файла. При отключенном режиме вставки (Off) блок WRITE работает в режиме замены: *GPSS/W* будет добавлять

пустые строки до номера строки, указанного в SEEK, и занесет данные в эту позицию.

Следующий пример использует режим замены блока WRITE. Обозначение SKRW указывает, что задействованы блоки SEEK, REPLACE и WRITE. Для управления текущей позицией записи применяется блок SEEK. Загрузим модель tstskrw.gps:

```
; GPSS World Sample File - TSTSKRW.GPS
************************************************
; Записывает 123456 во вторую строку файла данных
; tstskrw.txt , заменяя исходную строку
************************************************
          GENERATE   ,,,1
          OPEN       ("TSTSKRW.TXT"),,Flag
          SEEK       2
          WRITE      123456,,Flag1,Off
          CLOSE      Prob,,Flag2
          TERMINATE  1
Flag      TERMINATE  1
Flag1     TERMINATE  1
Flag2     TERMINATE  1
************************************************
```

Откомпилируем и запустим модель по прежней технологии. Рассмотрим измененный файл данных через меню File/Open. Вторая строка заменена на 123456. Здесь мы задали режим замены (Off) для блока WRITE. Блок SEEK выбирает заменяемую строку.

Теперь выясним, как работает режим Replace, если нет номера строки, определенного блоком SEEK. Загрузим модель tstsqrw.gps.

```
; GPSS World Sample File - TSTSQRW.GPS
*****************************************
; Пишет 123456 в первую строку файла данных
; TSTSQRW.TXT, заменяя исходную строку
*****************************************
          GENERATE   ,,,1
          OPEN       ("TSTSQRW.TXT"),,Flag
```

```
          WRITE      123456,,Flag1,Off
          CLOSE      Prob,,Flag2
          TERMINATE  1
Flag      TERMINATE  1
Flag1     TERMINATE  1
Flag2     TERMINATE  1
*******************************************
```

Откомпилируем и запустим модель. Просмотрим измененный файл данных. Результат тот же, что в предыдущей модели, но, поскольку номер строки указан не был, заменилась первая строка.

Теперь заменим две строки: текстовую и с результатом вычислений. Загрузим модель `tstsqr2w.gps`:

```
; GPSS World Sample File - TSTSQR2W.GPS
*****************************************************
; Делает две последовательные записи в режиме замены
*****************************************************
          GENERATE   ,,,1
          OPEN       ("TSTSQR2W.TXT"),,Flag
          WRITE      ("This is fun."),,Flag1,Off
          WRITE      (5+5),,Flag2,Off
          CLOSE      Prob,,Flag3
          TERMINATE  1
Flag      TERMINATE  1
Flag1     TERMINATE  1
Flag2     TERMINATE  1
Flag3     TERMINATE  1
*****************************************************
```

Откомпилируем и запустим модель. В результирующем файле после каждого входа в блок `WRITE` текущий номер строки увеличивается на 1. Следовательно, будут заменены строки 1 и 2.

Теперь применим встроенные функции работы со *строками*. С их помощью мы можем форматировать заголовки Отчета, писать и считывать информацию частично. Загружаем модель `tstcatw.gps`:

```
; GPSS World Sample File - TSTCATW.GPS
```

```
****************************************
* Моделирование парикмахерской
*     Время - в минутах
****************************************
      GENERATE  5,2.5       ;Новый клиент
      SAVEVALUE Custnum+,1  ;Накопление числа прибывших
      ASSIGN    Custno,X$Custnum ;Назначение клиенту номера
      QUEUE     Barber      ;Начало ожидания
      SEIZE     Barber      ;Захват или ожидание парикмахера
      DEPART    Barber      ;Конец ожидания
      ADVANCE   10,2.5      ;Стрижка
      RELEASE   Barber      ;Конец стрижки
; Освобождение парикмахера.
      TEST G    TG1,1,Fin   ;Окончат. средн. время ожидания
      TERMINATE 1           ;Уход клиента
Fin   OPEN      ("TSTCATW.TXT"),,Flg1
      WRITE     (Catenate("Avg. Barber Queue Time Is ",QT$Barber)),,Flg
      CLOSE     Prob,,Flg2
      TERMINATE 1
Flg   TERMINATE 1
Flg1  TERMINATE 1
Flg2  TERMINATE 1
*****************************************
```

Файл с данными `tstcatw.txt` будет создаваться заново.

Загружаем файл с моделью, компилируем его и выполняем моделирование обслуживания 500 клиентов. Открываем результирующий файл. В блоке `WRITE` мы использовали процедуру `Catenate` склейки текста с числовыми данными, полученными при моделировании. Последние автоматически преобразуются в строку.

Теперь посмотрим манипуляции с данными при считывании из файла. Загрузим модель `tststrdw.gps`:

```
; GPSS World Sample File - TSTSTRDW.GPS
*****************************************************
* Анализ торговли в театральном буфете.
* Считывает информацию о возрасте и поле и изучает,
* как состав смеси будет влиять на ожидание.
* Формирует отчет о временах ожидания, длине очереди
* и количестве обслуженных.
*****************************************************
```

```
Fem     EQU 1
Male    EQU 2
        GENERATE  1
        TEST E    X$Finish,0,In     ;Конец файла -> разрушить next xact
        READ      Info,1,Fin        ;Считывание данных о зрителе
        SAVEVALUE Info1,(Substring(P$Info,1,2)) ;выделили возраст
        SAVEVALUE Info2,(Substring(P$Info,3,1)) ;выделили пол
        SAVEVALUE Counter+,1
        TEST G    X$Info1,8,In      ;Старше 8 - может купить
        TEST LE   X$Info1,17,In     ;Старше 17 - не покупает
        TEST E    X$Info2,Male,Ques ;Все мужчины покупают
Ques    TRANSFER  .50,In            ;50% девушек на диете - не покупают
        QUEUE     Wait              ;Начало ожидания
        SEIZE     Salesperson       ;Захват продавца
        DEPART    Wait              ;Конец ожидания
        ADVANCE   6,1               ;Обслуживание
        SAVEVALUE Served+,1         ;Накопление обслуживаний
        RELEASE   Salesperson       ;Освобождение продавца
In      TERMINATE                   ;Вход в область просмотра
*******************************************************
        GENERATE  ,,,1,10           ;Генерация здесь первого транзакта
        OPEN      ("TSTSTRD.TXT"),1,Er ;Файл для чтения
        OPEN      ("TSTSTW.TXT"),2,Er1 ;Файл для записи
        TERMINATE
Fin     SAVEVALUE  Finish,1         ;Сигнал конца файла EOF
        WRITE     (Catenate("Average "," Average ")),2,Er2
        WRITE     (Catenate(" Wait ","Line Size")),2,Er3
        WRITE     (Polycatenate(" ",INT(QT$Wait)," ",INT(QA$Wait))),2,Er4
        WRITE     "Total People Served",2,Er7
        WRITE     (Catenate(" ",X$Served)),2,Er8
        CLOSE     Prob,1,Er9
        CLOSE     Prob1,2,Er10
        TERMINATE 1
Er      TERMINATE 1
Er1     TERMINATE 1
Er2     TERMINATE 1
Er3     TERMINATE 1
Er4     TERMINATE 1
Er5     TERMINATE 1
Er6     TERMINATE 1
Er7     TERMINATE 1
Er8     TERMINATE 1
Er9     TERMINATE 1
Er10    TERMINATE 1
*******************************************************
```

В файле с данными `tststrdw.txt` по одному в строке записаны:

```
161 021 252 162 151 131 172 172 182 352 341
152 152 162 151 131 172 172 182 352 341 202
141 161 171 151 112 142 142 122 132 142 152
```

Первые две цифры каждого числа кодируют возраст, последняя — пол зрителя (см. первые строки программы). Файл `tststw.txt` будет создаваться.

Загрузим модель, откомпилируем ее и запустим на счет по `START 1`.

Не забывайте инициализировать файлы данных, как указано в начале раздела, и удалите вновь сформированные файлы.

15.13 ANOVA

Результаты имитационного моделирования всегда случайны. Достаточно изменить начальные установки ДСЧ, чтобы получить несколько иные результаты. Процедура `ANOVA` позволяет определить, какие эффекты реальны, а какие объясняются статистическим шумом. Мы вправе ею воспользоваться, если результаты получены арифметическим усреднением многих наблюдений — например, времен ожидания. *При моделировании с целью проверки статистических гипотез интерактивное вмешательство недопустимо.*

Вернемся к примеру с парикмахером, который едва справляется с прибывающими клиентами. Как владелец парикмахерской вы хотите знать, даст ли парикмахер со средним временем стрижки 5 мин значительное уменьшение времени ожидания клиентов. Сравниваются два варианта: парикмахеры 1 и 2. Они различаются только средней длительностью стрижки: 6.8 и 5 мин соответственно. Мы выполним по три прогона на каждом уровне (можно и больше).

Сформируем матрицу результатов в специальном формате, необходимом для `ANOVA`: в нашем случае имеются две гипотезы с тремя репликами для каждой. Матрица будет размером 2 x 3. Загрузим модель `ANOVA`:

```
; GPSS World Sample File - ANOVA.GPS
****************************************
* Моделирование парикмахерской
*  Время в минутах, постоянное
****************************************
          GENERATE  5,1.7    ;Новый клиент
          QUEUE     Barber   ;Начало ожидания
          SEIZE     Barber   ;Захват или ожидание парикмахера
          DEPART    Barber   ;Конец ожидания
          ADVANCE   Cut_Time ;Стрижка
          RELEASE   Barber   ;Освобождение парикмахера
          TERMINATE 1        ;Уход клиента
****************************************
```

Главный параметр эксперимента представлен в блоке ADVANCE *переменной* Cut_Time, что позволит менять среднее время стрижки. Мы будем использовать трехзначную начальную установку, меняя только среднюю цифру для указания номера прогона. Во второй серии прогонов используем те же установки ДСЧ при измененном среднем времени стрижки.

Перед началом моделирования рассмотрим *включаемый* файл, который мы будем использовать для многократного моделирования:

```
RESULTS    MATRIX       ,2,3 ; Три реплики на двух уровнях
Cut_Time   EQU          6.8
Treatment  EQU          1
           RMULT        411
           START        100,NP
           MSAVEVALUE   RESULTS,1,1,QT$Barber
           CLEAR        Off
           RMULT        421
           START        100,NP
           MSAVEVALUE   RESULTS,1,2,QT$Barber
           CLEAR        Off
           RMULT        431
           START        100,NP
           MSAVEVALUE   RESULTS,1,3,QT$Barber
           CLEAR        Off
 Cut_Time  EQU          5
Treatment  EQU          2
```

```
RMULT        411
START        100,NP
MSAVEVALUE   RESULTS,2,1,QT$Barber
CLEAR        Off
RMULT        421
START        100,NP
MSAVEVALUE   RESULTS,2,2,QT$Barber
CLEAR        Off
RMULT        431
START        100,NP
MSAVEVALUE   RESULTS,2,3,QT$Barber
```

В этом файле определена глобальная матрица Results, которая будет накапливать результаты для ANOVA. Команда EQU используется для задания среднего времени стрижки 6.8 мин. После трех прогонов и сбора данных среднее время стрижки устанавливается 5 мин. Для каждой серии устанавливается также номер гипотезы.

Команда CLEAR Off сброса статистики между прогонами исключает корреляцию их результатов. Операнд Off необходим для сохранения матрицы результатов. В нашем простом примере мы пренебрегли эффектом начального состояния (клиентов нет). Это обстоятельство можно учесть, сбросив посредством RESET начальную статистику в каждом прогоне (команда RESET ,OFF позволяет сохранить ответную матрицу). Для определения длины переходного периода полезно поработать с Графическим Окном.

В этой программе прогоны организовывались с опцией NP в операнде B команды START для подавления формирования стандартного отчета. В командный файл были вставлены начальные установки ДСЧ — для каждой гипотезы одни и те же, что уменьшает дисперсию результатов.

Откомпилируем модель. Откроется ее журнал. Затем используем оператор INCLUDE для прогонов. Напомним, что *GPSS* ищет вставки там же, где находится модель. Поэтому выберем Command/Custom, напечатаем INCLUDE "ctlanova.txt" и выполним эту команду. Моделирование начнется автоматически, 6 прогонов завершатся за несколько секунд. Посмотрим на матрицу Results в Окне Матриц — табл. 15.1. Когда появится окно диалога выбора сечения, нажмите [OK]. При необходимости окно можно расширить.

Таблица 15.1. RESULTS

	Dim2	
Dim1	1	2
1	84.639	87.058
2	1.608	2.739

Теперь пришло время анализа. Выбираем `Command/Custom` и печатаем

```
SHOW ANOVA(Results,2,0)
```

После подтверждения [OK] процедура `ANOVA` записывает итоговую таблицу 15.2 в Окно Журнала (названия граф даны в переводе) и возвращает значение F-критерия.

Таблица 15.2. ANOVA

Источник вариации	Сумма квадратов	Степени свободы	Средний квадрат	F	Критическое F $(p = 0.05)$
A	10844.768	1	10844.768	720.459	7.71
Ошибка	60.210	4	15.053		
Всего	10904.978	5			

Уровень A	Счетчик	Среднее	Минимум	Максимум	95% интервал
1	3	88.590	84.640	94.073	[-2.656,9.780]

3.8797650

Так как наше F много больше критического уровня 7.71 для этого эксперимента, считаем эффект *статистически значимым*.

В *GPSS/W* имеется мощная процедура `EXPERIMENT` для автоматического управления экспериментом с многими прогонами — см. разд. 15.18.

Если обеспечиваемой `ANOVA` глубины анализа недостаточно, надо создать текстовый файл для последующего считывания информации из него автономными программами статистического анализа.

15.14 Отчет

Система генерации отчетов обладает следующими особенностями:

1. Отчеты создаются всегда, если только эта функция не подавлена NP в операнде B команды START.

2. Каждый отчет получает имя, которое начинается с имени порождающей модели и дополняется номером Имитации и номером отчета, напр. SAMPLE2.1.1, SAMPLE2.1.2 и т. д. Если вы хотите иначе именовать отчеты, их можно копировать в буфер Clipboard и оттуда — в файл с произвольным именем. Отчет имеет расширение .gpr.

3. Отчеты сохраняются после завершения сессии $GPSS/W$.

Команду REPORT в интерактивном режиме можно использовать только в виде Report ,Now — для выдачи отчета в произвольный момент. Установки для определения содержания отчета можно изменить на вкладке REPORT в Блокноте Установок модели. Изменение установок подействует только на Имитацию, созданную после изменения.

Закроем режим Установок, откомпилируем и запустим модель. После завершения автоматически открывается Окно Отчета. Прогоним Имитацию еще раз, чтобы иметь два отчета. Откроется второе Окно Отчета. Закроем одно из них, отказавшись от сохранения (No). Закроем второе окно, но запомним результаты в файле. Открыв этот файл через File/Open, убедимся, что результаты сохранились.

Проверим команду REPORT, NOW. Запустим «долгую» Имитацию по команде START 111111 и остановим ее по [F4]. Выберем Command/ Custom и напечатаем Report,Now. После нажатия [OK] получаем Отчет о текущем состоянии модели.

15.15 Журнал сессии

Журнал воспроизводит совокупность предложений, записанных в окно Journal/ Simulation Window в процессе моделирования. В нем накапливаются сообщения о трассировке, об ошибках, интерактивные команды SHOW и STEP 1 и т. п.

Загрузим модель sample5. Зададим общее STOP-условие через Command/Custom. Запустим моделирование по START 11. Модель остановится немедленно. Выполним несколько шагов по нажатию [F5] и просмотрим Окно Журнала. Расширим его и распечатаем содержимое.

15.16 Непрерывные модели

Если мы можем выразить изменения переменных через их производные по времени, $GPSS/W$ позволяет автоматически обновлять значения этих переменных. Используя команду INTEGRATE, можно встроить в модель систему обыкновенных дифференциальных уравнений любого порядка. Этот метод — весьма общий и допускает нелинейные системы, но обычно медленнее других методов. Его основными инструментами служат: команда INTEGRATE, которая описывает дифференциальное уравнение и пределы, и блок INTEGRATION, который может включать или отключать каждое интегрирование. Интегрируемую переменную нужно предварительно инициализировать.

Откроем *новый* файл. Пусть мы имеем производственную систему, которая производит две детали в минуту. Когда набирается полный контейнер, рабочий отсылает его в один из распределительных центров. Моделирование начинается при нулевом запасе. Напечатаем

```
Inventory  EQU 0                  ; начальное условие
Inventory  INTEGRATE 2,Full,Over  ; производная и пределы.
Full       EQU 50
```

В нашем случае скорость увеличения запаса (производная по времени) равна 2. Эта константа и является первым операндом INTEGRATE. В общем случае для определения производной нужна $PLUS$-процедура. Тогда заключенный в скобки вызов процедуры используется как операнд A команды INTEGRATE. Когда будет произведено 50 деталей, в блок Over будет послан транзакт, используемый для окончания моделирования.

Рассмотренные соображения относились к «непрерывной» части модели. Что касается дискретной, добавим к модели следующие предложения:

```
        GENERATE 100
        TERMINATE
Over    TERMINATE 1
```

Дискретная часть нужна *всегда* — хотя бы для начала и окончания моделирования.

Сохраним модель для использования в дальнейшем. Выполним компиляцию. Перед началом моделирования откроем Окно Графиков. Затем в окне Edit Plot напечатаем в полях Label, Expression, Title

соответственно «Запас», «Inventory» и «Динамика запаса». Остальные значения оставим без изменений. Щелкнем по кнопкам [Plot] и [Memorize]. Отметим порог на графике: в окне `New Expression` в поле `Label` напечатаем «Порог», а в поле `Expression` — «Full». Щелкнем по кнопкам [Plot], [Memorize], [OK]. Расширим Окно Графиков до удобного размера и запустим Имитацию командой `START 25`.

Запас быстро увеличивается до 50, имитация останавливается. В стандартном отчете указано, что потребовалось только 25 мин для выполнения работы. Если мы хотим показать уменьшение запаса после отправки контейнера, нужно изменить значение переменной пользователя. Заметим, что команда `EQU` не может быть приведена в действие транзактом: для этого нужна $PLUS$-процедура.

Обратимся теперь к примеру с известным решением. Рассмотрим систему обыкновенных дифференциальных уравнений

$$\begin{aligned} x' &= y, \\ y' &= -x \end{aligned}$$

с начальными условиями при $t = 0$

$$\begin{aligned} x &= 1, \\ y &= 0. \end{aligned}$$

Решение этой системы есть

$$\begin{aligned} x &= \cos t, \\ y &= -\sin t. \end{aligned}$$

При записи в $GPSS$ во избежание нарушения соглашений об именах включим в имена переменных знаки подчеркивания.

Выберем модель `sample13`, откомпилируем ее. Откроем Окно Графиков. В подокне `Edit Plot` для `Label` напечатаем `x_`, в поле `Expression` — `x_` и в заголовке графика «Sin/Cos». Для диапазона времени зададим 120, для минимальных и максимальных значений `Y` границы -1 и 1.2

соответственно. Щелкнем по кнопкам [Plot] и [Memorize]. Добавим к графику информацию по другой переменной. В подокне `Edit Plot` занесем в поля `Label` и `Expression` имя `y_`. Снова щелкнем по кнопкам [Plot], [Memorize], [OK]. Увеличим Окно Графиков. Теперь можно начать моделирование по `START 1`. Получились, как и следовало ожидать, банальные синусоиды.

Изучите Стандартный Отчет по окончании моделирования. Сравните переменные `x_` и `y_` с сохраняемыми значениями по имени `x_exact` и `y_exact` из аналитического решения (в конце отчета) и качественно оцените степень согласия.

Более содержательная модель интегрирования системы дифференциальных уравнений динамики популяций («хищник — жертва») приводится в файле `predator`.

15.17 Отладка моделей

15.17.1 Организация отладки

Отладка начинается с выявления и устранения *синтаксических* ошибок. Компилятор формирует в Окне Модели закольцованный список ошибок, который просматривается по команде `Search — Next/Previous Error`. Курсор выставляется в месте ошибки, текстовое сообщение появляется в статусной строке.

Вставляемые по `INCLUDE` файлы нужно отладить предварительно. Общая стратегия *семантической* отладки — сверху вниз. Сначала нужно войти в Окно Блоков для просмотра начала имитации. Затем просмотрите Таблицы, Устройства, Памяти, Матрицы через графические окна и убедитесь в качественном соответствии результатов ожиданиям. Помните, что многие окна имеют разнообразные возможности. Имейте в виду «снимки» цепей — инструмент углубленного исследования, эффективный в руках *профессионала*.

Выявите сгущения; ресурсы сильно загруженные (*GPSS/W* обычно выделяет их красным цветом) и недоиспользуемые. Рассчитайте коэффициенты загрузки узлов. Удостоверьтесь в накоплении транзактов в `Retry Chains`. Просмотрите в окне блоков счетчики текущего числа транзактов. Составьте список выявленных «нештатных ситуаций». Затем можно перейти к частным проблемам, изучаемым с помощью гра-

фических окон и команд START, STEP, STOP, SHOW, CONTINUE. Возможно, потребуется вывести что-то на графики. В Окне Блоков мышью можно выбирать и модифицировать блоки и условия останова.

Полезно использовать режим отказа блока TEST для проверки специфического условия. Таким образом можно остановить моделирование при его выполнении. Ниже показано формирование останова, когда в очереди Teller окажется более 20 транзактов:

```
          STOP       ,Stopper
          GENERATE   ,,,1
Tester    TEST   G   Q$Teller,20
Stopper   TERMINATE
```

Режим отказов блоков TEST и GATE отнимают много процессорного времени. Может оказаться более удобным прервать (HALT) имитацию некоторым другим способом незадолго до наступления этого события и затем использовать в режиме ручного моделирования блок SPLIT, чтобы послать испытательный транзакт в блок по имени Tester. Тогда GENERATE не нужен.

Для целей отладки полезны следующие модификации программы:

- Вставить блок SEIZE или ASSEMBLE между JOIN и REMOVE. Это позволит визуализировать цепь задержки или парности (MATCH) в окне данных с помощью команды GROUPS.

- Временно изменить интенсивность и/или распределение потока транзактов.

Иногда проблемы выявляются только при сопоставлении. Поэтому обязательно нужны эталоны — лучше всего аналитические. Сравните результат с практической ситуацией из реального мира: при общих исходных данных результаты расчета должны соответствовать наблюденным значениям показателей.

Убедившись в работоспособности модели, приступайте к эксперименту по выбору организации обслуживания. Пользуйтесь командой ANOVA для контроля значимости различий в результатах. Имейте в виду продвинутые статистические методы — см. главы 6–8. Результаты моделирования часто дают основание для новых экспериментов. Будьте готовы усовершенствовать модель и повторить весь цикл работы с ней, но обязательно сохраните текст исходной модели и зафиксируйте ее особенности.

15.17.2 Пример отладки

Интерактивные возможности *GPSS/W* существенно облегчают отладку моделей. Покажем на примерах простые приемы отладки. Откроем файл barber.gps.

```
; GPSS World Sample File - BARBER.GPS.
************************************************************
* Моделирование работы парикмахерской
************************************************************
Waittime QTABLE    Barber,0,2,15     ;Гистограмма ожидания
         GENERATE  3.34,1.7          ;Новый клиент
         TEST LE   Q$Barber,1,Finis  ;Ждет, если очередь < 1
                                     ;иначе уходит
         SAVEVALUE Custnum+,1        ;Накопление принятых
         ASSIGN    Custnum,X$Custnum ;Назначение номера
         QUEUE     Barber            ;Начало ожидания
         SEIZE     Barber            ;Захват или ожидание
         DEPART    Barber            ;Конец ожидания
         ADVANCE   6.66,1.7          ;Стрижка
         RELEASE   Barber            ;Освобожд. парикмахера
Finis    TERMINATE 1                 ;Уход клиента
************************************************************
```

Откроем блокнот Model Settings на странице Function Keys. Нижеописанные команды закрепим за клавишами [F8] и [F9], щелкнув мышью по соответствующему полю и набрав в окне команду:

```
SHOW P$Custnum
SHOW X$Custnum
```

Нажмем [OK]. Создадим Имитацию. Откроем Окно Блоков. Наложим условие останова на Транзакт 5: выберем Command/Custom и в диалоговом боксе напечатаем Stop 5. Устроим окна так, чтобы были видны одновременно окна Блоков и Журнала. В Главном Меню выберем Command/START и заменим счетчик завершений на 100. «Процесс пойдет». Полученное сообщение указывает, что Транзакт 5 остановился при попытке входа в свой первый блок — GENERATE.

Теперь начнем пошаговое моделирование с помощью клавиши [F5]. Транзакт 5 войдет в блок GENERATE, трассировочное сообщение появится в журнале. Сделаем еще один шаг. Когда транзакт войдет в блок TEST, он

найдет, что длина очереди к парикмахеру больше 1. Этот клиент ждать не захочет и покинет парикмахерскую (ему запланирован блок TERMINATE).

Изучим другие значения, связанные с моделированием. Выберем Command/SHOW и напечатаем в диалоговом боксе Q$Barber. Длина очереди (2) показана в статусной строке Главного Окна и в Окне Журнала; ее можно наблюдать также в колонке Current Count блока QUEUE Окна Блоков.

Теперь применим команды SHOW, ранее закрепленные нами за клавишами. Нажав [F9], увидим в статусной строке 4 — общее число клиентов в парикмахерской. Вспомним, что для контроля за целой серией значений в процессе пошагового моделирования можно использовать Окно Выражений. Продолжим шаги. Нажатием [F5] двинем Транзакт 5 из блока TERMINATE. Можно убедиться, что в парикмахерской находятся 4 клиента. Транзакт 5 ждать не будет и в этот счет не входит. Нажатием [F2] выполняем команду CONTINUE — модель снова работает в автоматическом режиме.

Закроем Окно Блоков, а Окно Журнала свернем — оно понадобится позже. Моделирование быстро завершится, будет записан стандартный Отчет. Далее откроем Окно Таблиц:

Window/Simulation Window/Table Window

В ниспадающем боксе увидим Waittime, так как это единственная Таблица в модели. Среднее время ожидания было 10.709 мин со среднеквадратическим отклонением 2.702 мин. Эта таблица — динамическая и обновляется в процессе моделирования. Прежде чем двигаться дальше, восстановим Окно Журнала, щелкнув мышью по его иконке в нижнем левом углу экрана. Теперь мы можем читать записи в Журнале — несколько более подробные, чем при работе с остановами.

Откроем Окно Блоков и приведем его к удобному размеру. Выберем в нем щелчком мышью блок №9 (RELEASE). Щелкнув по иконке Place в отладочной панели, введем в него команду останова. Начнем моделирование по Command/START, установив в счетчике завершений 100. Моделирование остановится перед входом первого транзакта в блок RELEASE — см. Журнал.

Изменим структуру модели, не закрывая сеанс. Нажмем [F8]. Вспомним, что мы вначале загрузили в [F8] команду SHOW параметра с произвольным номером. В статусной строке показано значение номера — 1. Удалим STOP из блока RELEASE. Предположим, что мы решили не запоминать номер транзакта в параметре. Исключим соответствующий

блок ASSIGN. В Окне Модели установим указатель мыши на начало строки с блоком ASSIGN и щелкнем мышью. Строка выделится. Удалим ее по [Ctrl+X]. Теперь можно ретранслировать модель и заново начать моделирование. Для команды START зададим счетчик завершений 11111. Подождем несколько секунд работы и остановим модель по [F4]. Проверим системное время. Получили сообщение, что параметр не существует, так как он не назначался новым транзактам. Таким образом, диагностика достаточно точна и вполне конструктивна.

Наконец, вспомним, что специально для целей отладки в системе существуют уже обсуждавшиеся средства (блоки и режим) трассировки. Закроем все открытые окна и файл с моделью.

15.17.3 Оперативная семантическая отладка

Посмотрим, как можно делать интерактивные изменения без рекомпиляции, на новой модели дорожного движения.

```
; GPSS World Sample File - SAMPLE7.GPS
****************************************************************
*                          Перекресток автодорог
*                        с односторонним движением
****************************************************************
*       Направление Восток - Запад
     GENERATE  20,10                    ;Новый автомобиль
     QUEUE     Eastwest                 ;Начало ожидания
     TEST E X$EWlight,F$Intersection    ;Блокировка до зеленого
                                        ;и свободного перекрестка
     SEIZE     Intersection             ;Захват перекрестка
     DEPART    Eastwest                 ;Конец ожидания
     ADVANCE   10                       ;Прохождение перекрестка
     RELEASE   Intersection             ;Освобождение перекрестка
     TERMINATE 1                        ;Уход из системы
*       Направление Север - Юг
     GENERATE  30,10                    ;Новый автомобиль
     QUEUE     Northsouth               ;Начало ожидания
     TEST E    X$NSlight,F$Intersection ;Блокировка до зеленого
                                        ;и свободного перекрестка
     SEIZE     Intersection             ;Захват перекрестка
     DEPART    Northsouth               ;Конец ожидания
```

```
     ADVANCE   10                        ;Прохождение перекрестка
     RELEASE   Intersection              ;Освобождение перекрестка
     TERMINATE 1                         ;Уход из системы
*****************************************************************
*                 Циклическое моделирование светофора           *
*****************************************************************
          GENERATE ,,,1
Begin1    SAVEVALUE NSlight,Red   ;Север - Юг     вкл. красный
          SAVEVALUE EWlight,Green ;Восток - Запад вкл. зеленый
          ADVANCE   Greentime     ;Горит зеленый
          SAVEVALUE NSlight,Green ;Север - Юг     вкл. зеленый
          SAVEVALUE EWlight,R7ed  ;Восток - Запад вкл. красный
          ADVANCE   Redtime       ;Горит красный
          TRANSFER  ,Begin1
* Когда свет зеленый (значение 0) и перекресток не занят
* (СЧА F$Intersection=0), машина может проехать через перекресток.
* Эти условия проверяются в блоке \verb|TEST|.
* Когда свет красный (значение 100) или перекресток занят
* (СЧА F$Intersection=1), проезд запрещен.
Green     EQU         0 Greentime EQU       200 Red       EQU 100
Redtime   EQU       300 *
*****************************************************************
```

Это модель перекрестка двух улиц с односторонним движением по каждой. Изменяя продолжительность включения зеленого света, можно влиять на пропускаемый поток.

Загрузим модель SAMPLE7, откомпилируем ее и откроем Окно Графиков для двух переменных: трафика «Север — Юг» и «Восток — Запад». Когда появится Edit Plot Window, введем указанные ниже значения. Поле метки содержит метку «Восток — Запад», используемую в легенде внизу графика «Next», поле Expression указывает выводимую переменную Q$EastWest. Наконец, поле Title дает название графику в целом — в данном случае «Очереди автомобилей в обоих направлениях». Ось Y по умолчанию имеет диапазон от 1 до 100. Мы только изменим максимум до 200.

Должен быть задан также интервал времени между последовательными просмотрами графика — в нашем случае 8000. Экспериментируйте! Окно Графика разделится горизонтально на страницы, когда график достигнет границы окна. Имейте в виду, что по умолчанию для каждого выражения сохраняется только 10000 точек (это количество можно кор-

ректировать через `Settings`).

Теперь мы хотим добавить на график вторую переменную. Введите метку «Север — Юг» и выражение `Q$NorthSouth`. Затем нужно последовательно щелкнуть по кнопкам [Plot], [Memorize], [OK].

Увеличьте размер окна до максимума. Запустите модель командой `START 4000` и понаблюдайте за динамикой очередей. Обе очереди пульсируют с частотой переключения светофора и противоположны по фазе. Максимумы очереди «Север — Юг» малы. Очередь «Восток — Запад» обнаруживает четкую тенденцию к нарастанию.

Остановим модель по [F4]. Выберем `Command/Custom` и напечатаем `Greentime EQU 1000`. Нажмем [OK], [F2] — моделирование продолжится. Ситуация изменилась на противоположную, тенденцию к росту (но более медленную) обнаруживает очередь «Север — Юг». Очевидно, наивыгоднейшее значение `Greentime` лежит между использованными значениями.

Снова остановим моделирование и скомандуем `Greentime EQU 500`. После продолжения видим, что очереди разумны в обоих направлениях. Максимумы очереди «Восток — Запад» практически не изменились, но исчезли периоды с нулевой очередью. Скопившаяся очередь «Север — Юг» постепенно рассасывается, амплитуды колебаний в обеих очередях весьма близки. Видимо, значение `Greentime=500` вполне приемлемо. График описанного процесса представлен на рис. 15.12.

Рис. 15.12. Динамика очередей на перекрестке

15.17.4 Работа с командными файлами

GPSS/W может вызывать командные файлы, состоящие из предложений INITIAL или других. Это делается интерактивно с помощью команды INCLUDE в диалоге Command/Custom либо вставкой по INCLUDE из файла с программой модели. Ниже следует пример вызова ASCII-файла с операторами инициализации.

Выберем Command/Custom и в диалоговом режиме наберем

INCLUDE "init.txt"

Содержимое этого файла:

```
INITIAL X$One,45
INITIAL X$Two,765
```

Помните, что при изменении директории «примерных» файлов в команде INCLUDE должен быть прописан путь.

Теперь проверим сохраненные значения. Выберем Command/SHOW и напечатаем в диалоговом окне X$One. После нажатия [OK] видим в статусной строке 45. Аналогично контролируем вторую переменную. Дожидаемся конца моделирования и закрываем все окна. Добавим в конец модели строку

INCLUDE "init.txt"

и проведем рекомпиляцию. Теперь Имитация автоматически прочтет содержимое упомянутого файла, создаст и означит нужные переменные.

INCLUDE-файлы можно использовать, как это делалось в разделе по ANOVA. Они позволяют легко управлять многократными прогонами. Для каждого законченного прогона будет сформирован отчет (если опция выдачи отчета не подавлена).

15.17.5 Отладка PLUS-процедур

Из *PLUS*-конструкций непосредственно в модель могут входить лишь PROCEDURE и EXPERIMENT. *PLUS*-выражения, заключенные в скобки, могут использоваться как *операнды* блоков и команд *GPSS/W*.

При прерывании модели на *PLUS*-процедуре модель возвращается в блок, предшествующий вызову. Временные объекты процедуры вывести на экран нельзя, и для ее отладки могут потребоваться дополнительные глобальные объекты.

Обсудим определение и интерактивное переопределение процедур и методику их отладки. Начнем с определения процедуры, которая задает значение пользовательской переменной. Выберем `File/New/Model`. Напечатаем

```
PROCEDURE SetPop(Level) Foxes = Level ;
```

Это простейший случай: `Foxes` — глобальная переменная (из задачи о динамике популяций кроликов и лис), которой присваивается значение `Level` (формального аргумента процедуры). Сохраним модель под именем `Fox`: `File/Save As`. Создадим Имитацию. Любой вызов `SetPop` — интерактивный или из программы — заставит выполниться упомянутое присваивание. Это легко проверить, выбрав `Command/SHOW` и напечатав в диалоговом боксе SetPop(100). Возвращаемое значение 0 задается по умолчанию. Вызовем `Command/SHOW` и напечатаем в диалоговом боксе `Foxes`. Убеждаемся, что переменная обновилась.

При отладке семантики $PLUS$-процедур прежде всего надо включить $PLUS$-трассировку в `Settings/Reports Pages`. Тогда каждое обращение вызовет запись в Окно Журнала. Трассы печатаются с отступом, чтобы показать глубину вызова, и сообщают вычисленные аргументы.

Включим $PLUS$-трассировку и определим процедуру с ошибкой в ней. Выберем `File/New`. Напечатаем

```
PROCEDURE RabbitRate() BEGIN
   TEMPORARY BirthRate, DeathRate, TotRate;
   BirthRate = 100;
   DeathRate =  80;
   RETURN TotRate ;
END;
```

При наборе этого текста постарайтесь воспользоваться операциями копирования и вставки из буфера. Не забудьте последнюю точку с запятой!

Сохраним модель под именем `Rates` и создадим Имитацию. Если вы допустили ошибки при наборе, используйте `Search/Next Error` в Окне Модели для их нахождения и исправления.

Выберите `Command/SHOW`, напечатайте в диалоговом боксе текст `RabbitRate()` и нажмите [OK]. Получите сообщение об ошибке. Рассмотрите его внимательно. Оно указывает номер строки с ошибкой и характер ошибки. Поместите курсор в эту точку. Убедитесь, что Окно Модели активно. Выберите `Search/Go To Line`, напечатайте номер строки

и нажмите [OK]. Очевидна попытка использовать значение неинициализированной временной переменной TotRate. Напечатайте выше строки с ошибкой

```
TotRate = BirthRate - DeathRate ;
```

Ретранслируйте модель и убедитесь в правильности результата — 20.000.

Временные имена надо выбирать не совпадающими с глобальными. С другой стороны, чтобы сделать глобальными внутренние переменные, достаточно закомментировать посредством /* */ их объявления внутри процедуры. Для контроля работы процедуры можно вставить дополнительные операторы присваивания и остановы по RETURN. Отладочные выдачи глобальных переменных удобно закрепить за функциональными клавишами.

Для выполнения трассировки *внутри* процедуры создайте одну или несколько пустых процедур, которые ничего не делают, и «пересыпьте» исходную вызовами пустых с интересующими вас аргументами. Эти аргументы будут зафиксированы в журнале.

15.18 Эксперименты в GPSS World

Здесь будет показано, как с помощью $PLUS$ организовать и эффективно провести эксперимент и посредством ANOVA проанализировать его результаты.

Вернемся к основной цели проекта — улучшению характеристик модели реальной ситуации. В этом и состоит смысл Эксперимента. Эксперимент наследует всю мощь языка $PLUS$ и использует ее автоматически. Как заявлено в Tutorial, «Ваш компьютер может перемалывать результаты и исследовать поверхности отклика, даже когда вы спите». Кстати, Эксперименты действительно выполняются очень медленно и могут длиться часами.

Как и в $PLUS$-процедурах, вы можете вставлять в свою модель определения Экспериментов или использовать их в текущих Имитациях через команду Custom. В любом случае вам может быть полезна команда INCLUDE.

Для прогона предварительно определенного Эксперимента в модель нужно включить команду CONDUCT с единственным аргументом — именем Эксперимента.

Синтаксически программа Эксперимента отличается от текста процедуры только ключевым словом в заголовке. Однако в Эксперименте и вызываемых им процедурах (и только в таких процедурах) имеется возможность использовать библиотечную процедуру DoCommand.

Когда Эксперимент начат, ваши возможности взаимодействия с процессом ограничиваются. Для анализа состояния модели (кроме системных часов, вызываемых по View/Clock), вообще говоря, нужна команда HALT. Затем Эксперимент придется начать сначала. Поэтому *модель надо отладить до начала Эксперимента.*

Посмотрим на Эксперимент, определенный в файле telephon.gps. Загрузим модель oneway.gps, которая определяет Эксперимент для модели телефонной системы. Требуется найти наилучшую конфигурацию телефонных линий при заданной загрузке.

```
; GPSS World Sample File - ONEWAY.GPS
************************************************
* Эксперимент с балансируемой телефонной системой
* и однофакторным дисперсионным анализом
* для определения оптимального числа телефонных линий.
*
* BestLines() помещает результаты в глобальную матрицу
* MainResult  и в файл RESULT.TXT и затем выполняет
* One Way дисперсионный анализ.
*
* Для создания EXPERIMENTа, который может быть остановлен,
* сохранен и продолжен, нужно использовать глобальные
* SAVEVALUE  и/или матрицы для управления прогонами.
*
* Команду  CLEAR надлежит использовать с опцией OFF

************************************************
* Определение глобальной экспериментальной
* матрицы результатов и таблицы остатков
************************************************
* До 4 уровней фактора 1; до 3 реплик на каждом
MainResult MATRIX  ,4,3
* Инициализация матрицы для контроля  отсутствующих данных
           INITIAL   MainResult,UNSPECIFIED
* Создание таблицы для записи остатков ANOVA
MainResult_Residuals TABLE ,-5,1,10
```

```
EXPERIMENT BestLines(LeastLinesToRun, MostLinesToRun, Increment) BEGIN
    /* Каждая матрица результатов имеет измерения:
       1. До  5 факторов; каждое измерение указывает число уровней,
          используемых для этого фактора (применяемая здесь
          ONEWAY DoCommand поддерживает только один фактор).
       2. Последнее измерение указывает максимальное число реплик
          при фиксированных уровнях.                            */

    /* Помещаем в Журнал заголовок                              */
    DoCommand("SHOW """" ");
    DoCommand("SHOW (""**** Balanced
       Telephone-System Experiment with One-way ANOVA ****"")");
    DoCommand("SHOW """" ");

    /* Сохраним данные для возможной дальнейшей обработки      */
    DoCommand("OPEN (""RESULT.TXT""),2");
    /* Открываем поток 2 для статистических данных             */
    DoCommand("SEEK 10000,2");    /* Добавляем результаты      */
    DoCommand("WRITE ""**** EXPERIMENT BestLines() ****"",2");
    /* Заголовок для этого Эксперимента                        */
    LevelOfFactor1 = 1;           /* Уровни 1-4 включительно   */
    CurrentLines = LeastLinesToRun;
    /* Установка числа используемых телефонных линий           */
    WHILE ((CurrentLines <=MostLinesToRun) 'AND' LevelOfFactor1<=4))
       DO BEGIN
          Replicate = 1;
          WHILE (Replicate <= 3) DO BEGIN
       /* Прогон модели через вызов определенной ниже PROCEDURE*/
            DoTheRun(CurrentLines,Replicate);
            /* Сохранить первый (и только) результат этого прогона.
               Это будет использовано ниже в ONEWAY DoCommand.
               Множественные результаты потребуют дополнительных
               матриц результатов.                              */

            MainResult[LevelOfFactor1,Replicate] = TB$Transit;
            /* Запись результата в журнал --- не обязательно.  */
            RunDescription = Polycatenate(
               PolyCatenate("Main Result: ",LevelOfFactor1,"
                  ",Replicate),"TB$Transit=",TB$Transit,"");
```

```
            DoCommand("SHOW (RunDescription)");
            DoCommand("SHOW """" ");    /* Space after Result  */
            /* Необязательный пример записи в результ. файл для
               третьей части статистики                        */

            DoCommand("WRITE (RunDescription),2");

            /* Установка следующей реплики                     */
            Replicate = Replicate + 1;
        END;

        /* Переход к следующей клетке эксперимента             */

        LevelOfFactor1 = LevelOfFactor1 + 1;
        CurrentLines = CurrentLines + Increment;
    END;

    /* Помещение статистического анализа в журнал              */
    DoCommand("SHOW """" ");  /* Пробельная строка             */
    ANOVA(MainResult,2,1);    /* Выполнение One-Way ANOVA      */
    DoCommand("SHOW """" ");  /* Пробельная строка             */
    DoCommand("CLOSE ,2");    /* Закрытие файла результатов    */
    /* Теперь все результаты можно видеть в Окне Матриц или
       в файле RESULTS.TXT. ANOVA - в Журнале.                 */
END;

PROCEDURE DoTheRun(NumberOfLines, ReplicateNumber) BEGIN

   /* PROCEDURE может применять  DoCommand(),
      только если ее вызывает EXPERIMENT                       */

    TEMPORARY RandomSeed1;
    /* Использовать установку, отличную от других реплик       */
    RandomSeed1 = 11 # ReplicateNumber;
    DoCommand("CLEAR OFF");     /* Должно использовать OFF.    */
    /* Здесь мы применяем только один RN                       */
    DoCommand(Catenate("RMULT ",RandomSeed1));
    DoCommand(Catenate("Sets STORAGE ",NumberOfLines));
    DoCommand("START 100,NP");  /* Startup Period              */
    DoCommand("RESET");         /* Начало периода измерений    */
```

```
    DoCommand("START 1000,NP"); /* Прогон модели              */

END;

; From GPSS World Sample File - TELEPHON.GPS,
;      by Gerard F. Cummings
************************************************************
*     Модель телефонной системы
*     Единица времени - минута
************************************************************
Sets      STORAGE   2
Transit   TABLE     M1,.5,1,20      ;Транзитные времена
          GENERATE  1.667,1         ;Генерация звонков
Again GATE SNF      Sets,Occupied   ;Проверка занятости линии
          ENTER     Sets            ;Соединение
          ADVANCE   3,1             ;Разговор
          LEAVE     Sets            ;Освобождение линии
          TABULATE  Transit         ;Табулирование длительности
          TERMINATE 1               ;Удаление транзакта
Occupied  ADVANCE   5,1             ;Ждать 5 мин
          TRANSFER  ,Again          ;Новая попытка
************************************************************
```

Эксперимент по имени BestLines вызывает $PLUS$-процедуру DoTheRun. Ее задача — установить условия, необходимые для каждого прогона. В пользовательском Эксперименте мы должны писать ее сами. В следующем разделе мы увидим, что $GPSS/W$ может сделать значительную часть этой работы за нас.

При компиляции этой модели процедуры будут транслированы и зарегистрированы в Имитации. Возможная альтернатива — послать эти $PLUS$-процедуры в существующую Имитацию посредством команд Custom и/или INCLUDE. Когда мы взаимодействуем с этой Имитацией, мы можем игнорировать Эксперимент BestLines, потому что до команды CONDUCT он ничего не делает.

Эксперимент BestLines разработан для заполнения результатами глобальной матрицы и передачи этой матрицы библиотечной процедуре ANOVA, которая проводит статистический анализ и формирует отчет. Элементы матрицы инициализируются как UNSPECIFIED, так что ANOVA может узнать, какие прогоны выполнены. Сгенерированные Эксперименты, обсуждаемые ниже, проверяют матрицу результатов на состояние

UNSPECIFIED, что исключает повторение уже проделанной работы. В дополнение к матрице результатов создается $GPSS$-таблица для получения остатков от выполняемого ANOVA дисперсионного анализа с тем же именем (и добавлением «_Residuals»), что результирующая матрица.

В программе имеются внешний и внутренний WHILE...DO циклы, которые и обеспечивают все необходимые прогоны. Имеется единственный изменяемый фактор — число телефонных линий, рассматриваемое на четырех уровнях. На каждом уровне выполняются три реплики, отличающиеся только повторяемыми на остальных уровнях начальными установками ДСЧ. Это улучшает оценку среднеквадратической ошибки Эксперимента.

В самом внутреннем цикле Эксперимент BestLines вызывает процедуру организации прогонов, именуемую DoTheRun. Последняя использует группу команд для инициализации (предваряет очередной прогон командами CLEAR OFF, RESET и START) и фактического выполнения моделирования. Так как DoTheRun вызывается EXPERIMENT'ом, она имеет право применять библиотечную процедуру DoCommand. При каждом завершении DoTheRun Эксперимент BestLines сохраняет результат моделирования в соответствующем элементе глобальной матрицы MainResult, что делается сразу после вызова BestLines(). Так как мы выполняем One Way ANOVA (однофакторный эксперимент), наша результирующая матрица — двумерная. Первый индекс есть номер уровня (числа телефонных линий), а второй — номер реплики на каждом.

Когда Эксперимент заканчивается, BestLines вызывает библиотечную процедуру ANOVA, которая пишет статистический отчет в Журнал моделирования.

Рассматриваемая $PLUS$-процедура DoTheRun использует опцию OFF команды CLEAR. Если бы это не было сделано, ResultMatrix была бы обнулена, а данные потеряны. Команды SHOW применяются для помещения сообщений в системный журнал. Это простой способ трассировать текущую фазу Эксперимента.

Эксперимент BestLines создает отдельный текстовый файл результатов в форме текстового объекта. Файл может быть прочитан другими программами обработки статистики или использован как сохраняемая запись результатов эксперимента.

После этих вводных рассуждений перейдем, наконец, к проведению Эксперимента. Откомпилируем загруженный файл. Система сооб-

щит о регистрации и, следовательно, доступности процедур `BestLines` и `DoTheRun`. Мы имеем объект `OneWay Simulation` и хотим начать Эксперимент по `CONDUCT BestLines(1,4,1)`. Для удобства загрузим команду в клавишу [F10] как часть модельных установок. Теперь для запуска достаточно нажать [F10].

Откроем Окно Журнала. Эксперимент идет и завершается автоматически. Сообщается модельное время завершения каждого прогона.

Проконтролируем Матрицу Результатов через
`Window/Simulation Window/Matrix Window`.
Так как матрица единственная, дадим сразу [OK]. В Окне Матриц увидим все результаты. Посмотрим на результирующий файл через `File/Open`. В диалоге выберем текстовый тип файлов, затем укажем имя `result` и откроем файл. Снова видим результаты Эксперимента, занесенные командой `WRITE`, с указанием координат каждого значения. Результаты работы `ANOVA` (табл. 15.3) можно посмотреть в журнале. $GPSS/W$ помещает в таблицу критическое значение F — наша база для сравнения. F-статистика настолько велика, что значимость фактора не вызывает сомнений. Из таблицы `ANOVA` следует, что уже добавление третьей линии дает незначительное улучшение. Добавление четвертой, вероятно, окажется экономически не оправданным. Итак, при данной загрузке система должна быть сконфигурирована с тремя линиями.

Таблица 15.3. ANOVA

Источник вариации	Сумма квадратов	Степени свободы	Средний квадрат	F	Критическое F $(p = 0.05)$
A	732008.284	3	244002.761	44165.048	4.07
Ошибка	44.198	8	5.525		
Всего	732052.482	11			

Ур. A	Счетчик	Среднее	Минимум	Максимум	95% интервал
1	3	579.795	575.314	583.590	[576.665,582.924]
2	3	22.797	21.272	25.255	[19.668,25.927]
3	3	3.319	3.267	3.355	[0.190,6.449]
4	3	3.019	3.011	3.029	[-0.110,6.148]

Проверим, выполнены ли в нашем случае предположения, на которых основан дисперсионный анализ. Изучим сформированную `ANOVA`

Таблицу остатков. Имя нашей результирующей матрицы было `MainResult`. Соответственно Таблица остатков получит имя `MainResult_Residuals`. Для просмотра матрицы выберем

`Window/Simulation Window/Matrix Window`

и, так как есть только одна таблица, дадим [OK]. Увидим гистограмму распределения остатков, сравнительно близкую к графику плотности нормального распределения.

15.19 Генерация скрининг-эксперимента

Проведенный выше Эксперимент `BestLines` состоит в простом переборе. Более сложные эксперименты связаны с определением наиболее значимых факторов (скрининг) и оптимизацией модели. В последнем случае нужно получить математическое описание поверхности отклика по экспериментальным наблюдениям. В $GPSS/W$ имеются средства автоматической генерации подобных экспериментов.

Загрузим модель Ethernet-сети `ExperEther.gps` и щелкнем в меню `Edit` по `Insert Experiment/Screening...`. Откроется диалоговое окно, которое мы заполнили информацией, необходимой для генерации скрининг-эксперимента — рис. 15.13.

Поля `Experiment Name` и `Run Procedure' Name` используются как имена процедур в генерируемом Эксперименте. Далее идут факторы Эксперимента. Имя каждого фактора фактически является именем пользовательской переменной и должно удовлетворять принятым в $GPSS/W$ соглашениям.

Для каждого фактора (их может быть от 1 до 5) следует определить их имена и два уровня. Факторы обозначаются последовательными латинскими буквами от A до E. Критическое обстоятельство здесь — выбор уровней. Для их удачного назначения могут потребоваться *предварительные* эксперименты. Выбирайте уровни подальше от точек скачкообразного изменения поведения; избегайте значений, при которых эффект маскируется другими факторами.

Рис. 15.13. Задание скрининг-эксперимента

Следующее поле — `Fraction` (дробность) определяет, какая часть полного 2^k-эксперимента должна быть проведена. Результирующий счетчик прогонов дается справа. Для уменьшения числа прогонов уменьшайте дробность, но помните, что при этом снижается полезность эксперимента.

В группе `Results` задается выражение для *метрики* (целевой функции) моделирования и имя результирующей матрицы. *GPSS World* автоматически сгенерирует команды определения и инициализации последней.

Имеются два окошка для выбора опций. Поставив отметку в первом, можно заставить *GPSS/W* сгенерировать шаблон процедуры

прогона для использования в эксперименте. Его можно редактировать. Во втором окошке можно предложить $GPSS/W$ загрузить соответствующую команду CONDUCT в функциональную клавишу [F11] — тогда после компиляции для начала эксперимента достаточно нажать [F11].

Перед генерацией эксперимента нужно изучить группы алиасов (альтернативных планов эксперимента). Этот вопрос обсуждался в разд. 7.3. Реализуем сделанную там рекомендацию о половинной дробности. Просмотрев алиас-группы, убеждаемся что совпадают и они. После щелчка по [Insert Experiment] открывается панель Run Procedure Generation с предложением сформировать и вставить ее. Нажимаем [OK].

Образец сгенерированного кода приводится ниже.

```
*****************************************************
*                ScreenEthernet
*    Дробно-факторный скрининг-эксперимент
*****************************************************
Global_Delays MATRIX ,2,2,2,2,2
          INITIAL   Global_Delays,UNSPECIFIED

EXPERIMENT ScreenEthernet() BEGIN

/* Run 1 */
    Node_Count = 100;
    Min_Msg = 512;
    Max_Msg = 12144;
    Fraction_Short_Msgs = 600;
    Intermessage_Time = 1.0;
IF (StringCompare(DataType(Global_Delays[1,1,1,1,1]),
       "UNSPECIFIED")'E'0)
    THEN BEGIN
        /* Run Procedure Call */
        DoTheRun(1);
        Global_Delays[1,1,1,1,1] = QT$Global_Delays;
    END;

    /*****    Здесь будут прогоны 3 - 15      ************/

    /* Run 16 */
```

```
    Node_Count = 200;
    Min_Msg = 1024;
    Max_Msg = 24288;
    Fraction_Short_Msgs = 300;
    Intermessage_Time = 1.0;
IF (StringCompare(DataType(Global_Delays[2,2,2,2,1]),
       "UNSPECIFIED")'E'0)
    THEN BEGIN
        /* Run Procedure Call                    */
        DoTheRun(1);
        Global_Delays[2,2,2,2,1] = QT$Global_Delays;
    END;

    /* Эффекты в дробном факторном эксперименте  */
    EFFECTS(Global_Delays,"I=ABCDE");
END;
*****************************************************
*                 Процедура прогона                 *
*****************************************************
PROCEDURE DoTheRun(Run_Number) BEGIN
    DoCommand("CLEAR OFF");
    /* OFF  обязательно */
    /* Расширьте эти операторы, используйте больше RN */
    /* Все они должны получить новые установки        */
    TEMPORARY CommandString;
    /* Вычислить ранее передачи в DoCommand           */
    CommandString = Catenate("RMULT ",Run_Number#111);
    /* DoCommand компилирует строку в глобальном контексте */
    DoCommand(CommandString);
    /* Подставьте свои условия                        */
    DoCommand("START 100,NP");
    /* Разгонный период */
DoCommand("RESET"); /* Начало периода измерения       */
DoCommand("START 1000,NP"); /* Прогон модели          */
END;
```

Исправим обнаруженную при компиляции ошибку — пропущенное имя DoTheRun процедуры прогона. Настало время CONDUCT эксперимент.

Поскольку в диалоге его создания мы загрузили эту команду в [F11], нажмем данную клавишу.

Эксперимент начинается, сообщая о своем текущем состоянии в Окне Журнала. На Pentium-1/166 он занял почти 27 мин. Для каждого из 16 наборов аргументов выполняются разгонный участок и основной прогон, по окончании которого выдаются номер прогона, значение целевой функции и исследуемых факторов.

Результаты эксперимента представлены в табл. 15.4. Часть данных сообщается только для главных эффектов. Они показывают, что факторы C и E имеют значимые эффекты и достойны дополнительных исследований. Оба связаны только с четырехкратными взаимодействиями и являются наиболее вероятными источниками измеренных эффектов. Ни один из других факторов по 2- и 3-кратным взаимодействиям не был найден значимым. Итак, мы за половинное число испытаний отсеяли большую группу незначимых факторов и их взаимодействий.

Таблица 15.4. Результаты скрининг-эксперимента

Алиас-группа	Эффект	Сумма квадратов	Степени свободы	F	Критическое F $(p = 0.05)$
A=BCDE	-0.026	0.003	1	0.000	4.96
B=ACDE	0.100	0.040	1	0.001	4.96
AB=CDE	2.064				
C=ABDE	14.986	898.295	1	26.025	4.96
AC=BDE	-0.026				
BC=ADE	0.100				
DE=ABC	2.063				
D=ABCE	5.485	120.323	1	3.486	4.96
AD=BCE	0.095				
BD=ACE	-0.051				
CE=ABD	-6.905				
CD=ABE	5.483				
BE=ACD	0.095				
AE=BCD	-0.051				
E=ABCD	-6.906	190.757	1	5.526	4.96
Ошибка		345.170	10		
Всего		1554.588	15		
Среднее	7.494				

15.20 Оптимизирующий Эксперимент

На рис. 15.14 показана панель диалога с генератором *оптимизирующих* экспериментов. Действия пользователя после завершения диалога аналогичны случаю скрининг-эксперимента.

Рис. 15.14. Панель диалога оптимизирующего эксперимента

Глава 16

Типовые модели теории очередей

16.1 Модель $M/M/1$ и датчики случайных чисел

16.1.1 Смитовский датчик равномерных случайных чисел

Приведем поучительный с точки зрения программирования пример генератора Смита и его использования для расчета стандартной системы $M/M/1$. Особенностью этой программы является глобальный по отношению к процедуре `SmithRND` статус массива `MMM` целых случайных чисел. Это позволило формировать при каждом обращении к ней только *одно* число (последний элемент) из этого массива, предварительно выполнив циклический сдвиг остальных. Здесь потребовались три процедуры: собственно для генератора, для задания начальных значений и для перехода к экспоненциальному распределению.

```
*************************************
*            M/M/1 MODEL            *
* Smith's Random Numbers generator  *
*************************************
MMM MATRIX  ,1,4
LLL MATRIX  ,1,4
PPP MATRIX  ,1,4
```

```
    INITIAL MX$LLL(1,1),1580
    INITIAL MX$LLL(1,2),5682
    INITIAL MX$LLL(1,3),1373
    INITIAL MX$LLL(1,4),1430
    INITIAL MX$PPP(1,1),100067
    INITIAL MX$PPP(1,2),300023
    INITIAL MX$PPP(1,3),1000919
    INITIAL MX$PPP(1,4),1001003

*************** Settings ************************
      GENERATE    ,,,1
      MSAVEVALUE MMM,1,1,1
      ASSIGN     1,(Settings())
      TERMINATE
*************** Main ****************************
      GENERATE   (SmithExp())         ;
      QUEUE      Wait                 ;
      SEIZE      Chan                 ;
      DEPART     Wait                 ;
      ADVANCE    (0.9#SmithExp())     ;
      RELEASE    Chan                 ;
      TERMINATE                       ;
*************** Timer ***************************
      GENERATE   200000,,,1
      TERMINATE 1
**************************************************
      PROCEDURE Settings() BEGIN
         TEMPORARY Indx,SetR;
         Indx=1;
         WHILE (Indx<4) DO BEGIN
            Indx=Indx+1;
            SetR=MMM[1,Indx-1]#LLL[1,Indx];
            SetR=SetR@PPP[1,Indx];
            MMM[1,Indx]=SetR;
         END;
      END;
      PROCEDURE SmithRND() BEGIN
         TEMPORARY Indx,XSm;
```

```
         Indx=1;
         WHILE (Indx<4) DO BEGIN
            MMM[1,Indx]=MMM[1,Indx+1]; Indx=Indx+1;
         END;
         XSm=MMM[1,3]#LLL[1,4];
         XSm=(XSm)@PPP[1,4];
         MMM[1,4]=XSm;
         RETURN XSm/PPP[1,4];
      END;
      PROCEDURE SmithExp() BEGIN
         TEMPORARY RSmE;
         RSmE=SmithRND();
         RSmE=-LOG(1-RSmE);
         RETURN RSmE;
      END;
************************************************************
```

Процедуры иллюстрируют достаточно богатый набор конструкций языка $PLUS$: в частности, цикл, использование # вместо знака умножения и операцию @ вычета по модулю. Заметьте, что:

- список аргументов процедуры, хотя бы и пустой, обязателен;
- процедура всегда вызывается как *функция* — по указателю;
- используемые в процедурах матричные объекты объявлены и инициализируются в модели *в целом* и являются глобальными по отношению к использующим их процедурам. Именно это свойство и позволяет реализовать ДСЧ данного вида, а также датчики типа Марсальи.

Результаты выполнения программы приведены в табл. 16.1. Время прогона модели до отметки таймера 500 000 составило около 9 мин. Максимальная длина очереди доходила до 83 транзактов.

Таблица 16.1: Расчет системы $M/M/1$ со смитовским мультидатчиком

Показатели	Предельное время		
	50000	200000	500000
Коэффициент загрузки	0.894	0.894	0.898
Среднее время обслуживания	0.902	0.898	0.899
Число вошедших транзактов	49549	199191	499317
Из них с нулевым ожиданием	5217	20908	50968
Средняя длина очереди	7.724	7.430	7.699
Среднее время ожидания	7.795	7.460	7.710

16.1.2 Датчики с произвольными распределениями

Для генерации случайных чисел с неравномерными распределениями можно воспользоваться встроенными вероятностными процедурами языка $PLUS$ (разд. 12.8). К сожалению, в системе отсутствуют средства расчета параметров распределений по их моментам.

Пользователь может написать собственную $PLUS$-процедуру формирования псевдослучайных чисел с требуемым законом распределения. При этом аргумент `RNj` следует умножить на 0.001. Обычно используемая в подобных программах замена $1-U$ на одинаково распределенное с ним случайное число U здесь недопустима, поскольку в среднем одно из тысячи $\{U_i\}$ окажется нулевым. Поэтому, например, при вычислении $\log(U)$ согласно формуле (4.3.2) гарантируется соответствующее число аварийных остановов модели.

```
***********************************************
*            M/M/1 MODEL                      *
*  Авторская процедура для EXP распределения  *
***********************************************

      GENERATE   (EXPOWN(RN1))
      QUEUE      Wait
      SEIZE      Chan
      DEPART     Wait
      ADVANCE    (0.9#EXPOWN(RN2))
      RELEASE    Chan
      TERMINATE
```

```
      GENERATE   500000,,,1
      TERMINATE 1
**********************************************
PROCEDURE EXPOWN(YRand) BEGIN
   TEMPORARY XRand;
   XRand=0.001#YRand;
   XRand=-LOG(1-XRand);
   RETURN XRand;
END;
```

Здесь аргументом процедуры является равномерно распределенное случайное число от ДСЧ RN1 или RN2. Поскольку оно получается не в качестве аргумента таблично заданной функции (см. следующий подраздел), то формируется как целое $U_i \in [0, 999]$. Соответственно перед логарифмированием оно умножается на 0.001. В среднем одно из 1000 таких чисел окажется нулем. Поэтому здесь нельзя экономить на вычитании равномерно распределенного числа из единицы.

GPSS/W поддерживает линейную аппроксимацию статистических (заданных таблично) обратных функций распределения. В этом случае стандартные ДСЧ в качестве аргументов выдают вещественные значения из диапазона [0,0.999999]. Такая аппроксимация для наиболее употребительного экспоненциального распределения с единичным средним обсуждается в разд. 13.2. Отметим, что показательное распределение хорошо описывает интервалы между редкими событиями и потому в задаче моделирования работы порта [74] применялось для интервалов между штормами. При плохом качестве табличной аппроксимации следует изменить масштаб времени.

В разд. 10.2 приведены контрольные результаты для системы $M/M/1$. Сопоставим с этими ожиданиями результаты моделирования при различных способах задания показательных распределений (встроенная функция Exponential, табличная ДФР и теоретически точное обращение $-\ln(1-U)$) и числе испытаний — табл. 16.2. Результаты, связанные с ожиданием, оставляют желать лучшего даже в случае с теоретически точным переходом от равномерного к экспоненциальному распределению. Причина здесь может быть только одна: недостаточно качественный генератор равномерных псевдослучайных чисел. Эти ошибки усугубляются погрешностями перехода к показательному рас-

пределению в `Exponential` и отчасти нейтрализуются (случайный эффект) при использовании табличного преобразования. Увеличение числа проведенных испытаний в общем приближает результаты к ожидаемым. Эти результаты можно сопоставить также с результатами программы, использующей мультигенератор типа Смита.

Обсудим, наконец, временны́е показатели сравниваемых версий ДСЧ. Для достижения отметки таймера 500000 при табличной реализации логарифма и при работе с `Exponential` потребовалось практически одинаковое время — около 135 с (процессор Pentium 1/166). По-видимому, `Exponential` реализована таблично. При точном обращении показательного распределения было затрачено 187 с.

Таблица 16.2. Тестирование GPSS/W на модели $M/M/1$

Показатель	Exponential		Табличная ДФР		$-\ln(1-U)$	
	50 000	500 000	50 000	500 000	50 000	500 000
Коэф. загрузки	0.896	0.899	0.900	0.900	0.909	0.902
Средн. время обслуж.	0.899	0.901	0.899	0.901	0.903	0.896
Число входов	49843	499032	50048	499390	50320	503176
Из них с нулевым ожид.	5340	50691	5022	49823	4764	50498
Средн. длина очереди	5.132	6.690	7.770	7.911	7.163	6.831
Средн. время ожидания	5.148	6.703	7.763	7.920	7.118	6.788

Прогон описанной в разд. 2.2 программы на *Фортране* 77 при тех же исходных данных дал среднюю длину очереди 8.683. Это заметно ближе к теоретической оценке, чем лучший из приведенных в таблице результатов. Прогон занял всего 6.26 с. Столь большая разница является естественной платой за универсальность интерпретатора $GPSS$. Окупается ли она удобством программирования — зависит от опыта и квалификации пользователя, его желания и возможностей освоить $GPSS$, наличных вычислительных мощностей, сложности моделей и частоты потребностей в моделировании.

Перед моделированием реальной ситуации желательно убедиться, что выбранная процедура действительно воспроизводит случайную величину с требуемым распределением. Гистограмма распределения, порожденная функцией `Exponential` и построенная по 200 тыс. транзактов, оказалась близкой к ожидаемой. Среднее значение вместо заказанной единицы составило 1.005, а среднеквадратическое отклонение — 1.006.

16.2 Многоканальная система

Ниже представлена $GPSS$-программа модели, в основном соответствующая блок-схеме рис. 2.1.

```
*****************************************************************
                    GPSS-аналог базовой модели                  *
*****************************************************************
          Сегмент процесса обслуживания
Prob   MATRIX     ,1,20                ; Матрица  вероятностей
Wtp    MATRIX     ,1,4                 ; Вектор моментов ожидания
Dt     VARIABLE   AC1-X$Told           ; Расчет скачка времени
Ind    VARIABLE   X$Dem+1              ; Сдвиг индекса вероятностей
Chans  STORAGE    3                    ; Трехканальная система
       GENERATE   (Exponential(1,0,1)) ; Простейший поток
                                       ; единичной интенсивности
       TEST E     X$Dem,19,Entry       ; Условие отказа в приеме
       SAVEVALUE  Nref+,1              ; Счетчик отказов
       TRANSFER   ,Out                 ; Уход заявки
 Entry MSAVEVALUE Prob+,1,V$Ind,V$Dt   ; Накопление вероятности
       SAVEVALUE  Told,AC1             ; Обновление момента
       SAVEVALUE  Ncount+,1            ; Счетчик принятых заявок
       SAVEVALUE  Dem+,1               ; Счетчик заявок в системе
       MARK       1                    ; Таймер -> в параметр 1
       QUEUE      Wait                 ; Начало ожидания
       ENTER      Chans,1              ; Попытка входа в канал
       DEPART     Wait                 ; Конец ожидания
       SAVEVALUE  Wtp1,(AC1-P1)        ; Время ожидания
       SAVEVALUE  Wtpw,X$Wtp1          ; Запоминание его
       ASSIGN     2,4                  ; Сч. цикла в параметр 2
 Mem   MSAVEVALUE Wtp+,1,(5-P2),X$Wtpw ; Накопление сумм степеней
       SAVEVALUE  Wtpw,(X$Wtpw#X$Wtp1) ; Очередная степень
       LOOP       2,Mem                ; Управление циклом
       ADVANCE    2.7                  ; Задержка на обслуживание
       LEAVE      Chans,1              ; Освобождение канала
       MSAVEVALUE Prob+,1,V$Ind,V$Dt   ; Накопление вероятности
       SAVEVALUE  Told,AC1             ; Обновл. прежнего момента
       SAVEVALUE  Dem-,1               ; Уменьшение числа заявок
 Out   TERMINATE                       ; Ликвидация транзакта
*****************************************************************
```

```
              Сегмент итоговой обработки
       GENERATE   5000,,,1     ; Момент окончания - один транзакт
       OPEN  "Basic.txt"       ; Открытие файла для результатов
       SEEK 1000               ; Выход в конец файла
       WRITE "**** Waiting Time Moments ****"
       ASSIGN     2,4          ; Установка счетчика цикла
 Wmoms MSAVEVALUE Wtp,1,(5-P2),(Wtp[1,5-P2]/Ncount);
       Сумму степеней разделили на число принятых заявок
       WRITE (Wtp[1,5-P2])     ; Запись момента ожидания в файл
       LOOP       2,Wmoms      ; Управление циклом
       WRITE "**** Stationary Probabilities ****"
       ASSIGN     2,20         ; Установка счетчика цикла
 Probs MSAVEVALUE Prob,1,(21-P2),(Prob[1,21-P2]/AC1);
       Суммарное время разделили на значение таймера
       WRITE (Prob[1,21-P2])   ; Записали вероятность в файл
       LOOP       2,Probs      ; Управление циклом

       WRITE "**** Number of Refuses ****"
       WRITE  X$Nref,1         ; Запись в файл числа отказов
       CLOSE 1                 ; Закрытие файла
       TERMINATE 1             ; Конец работы
*************************************************************
```

Завершение программы задается критическим значением таймера 5000 — см. сегмент итоговой обработки. Сегмент *процесса обслуживания* начинается с объявления матриц и переменных. Dt вычисляет скачок времени как разность между текущим значением таймера AC1 и моментом предыдущего скачка X$Told. Переменная Ind необходима в связи с невозможностью нулевой границы массива: величина скачка времени добавляется к элементу матрицы вероятностей, сдвинутому на 1 относительно текущего числа X$Dem требований в системе. Всем переменным и накапливающим счетчикам начальные (нулевые) значения присваиваются по умолчанию. Оператор STORAGE назначает емкость многоканального устройства Chans — в нашем случае 3.

Блок GENERATE формирует рекуррентный псевдослучайный поток заявок, интервалы между которыми имеют показательное распределение со средним значением 1. Поскольку операндом GENERATE является $PLUS$-выражение (указатель встроенной функции из класса распределений), оно заключено в скобки. При этом используется первый датчик

равномерных чисел. Далее текущее число заявок в системе DEMR сравнивается ('E') с предельным значением 19. В случае равенства (то есть при *невыполнении* условия) к счетчику отказов Nref добавляется единица, заявка передается в блок с меткой OUT и покидает систему.

В случае принятия заявки в соответствующем элементе матрицы PROB[1,DEMR1] накапливается (на что указывает плюс после первого операнда) время постоянства текущего состояния. Это время вычисляется при обращении к переменной V$Dt — см. ее определение. Номер столбца матрицы V$Ind=V$Dem+1D. Затем в качестве сохраняемой величины Told фиксируется текущее значение таймера, количество принятых Ncount и находящихся в системе Dem заявок увеличивается на 1. В первом параметре транзакта (блок MARK) запоминается момент прибытия заявки.

Блок QUEUE включает контроль времени ожидания в очереди Wait. Если в «памяти» Chans есть свободные каналы, то ENTER занимает один из них, а DEPART фиксирует окончание ожидания. Длительность ожидания запоминается в Wtp1.

Далее в соответствующих элементах матрицы Wtp накапливаются последовательные степени времени ожидания. Величина Wtpw служит для представления последовательных степеней (сначала — первой). Блок ASSIGN заносит в параметр P2 текущего транзакта требуемое число повторений цикла. Блок MSAVEVALUE накапливает текущие степени в элементах первой строки, причем номер столбца формируется как 5 минус содержимое счетчика. Затем степень накапливается домножением на X$Wtp1. Блок LOOP вычитает 1 из счетчика цикла и передает транзакт для продолжения накопления, пока счетчик не обнулится.

Блок ADVANCE в данном случае задает постоянное время обслуживания 2.7, что в сочетании с операндом GENERATE обеспечивает тип модели M/D/3 и коэффициент загрузки 0.9. Обслуженная заявка покидает «память», освобождая один канал. Происходит накопление времени пребывания в текущем состоянии, обновление момента скачка и коррекция числа заявок Dem — на этот раз в сторону уменьшения. Войдя в TERMINATE, транзакт покидает систему.

Блоки второго сегмента программы выполняются при достижении значения таймера 5000. Здесь открывается файл Basic.txt для записи результатов счета, и потоку данных присваивается (по умолчанию) значение 1. Блок SEEK 1000 имеет значение операнда, заведомо превышающее число строк, уже записанных в файл. Указатель позиции записи проигнорирует пустые строки и установится впритык к заполненной части

файла.

Далее в первый (по умолчанию) поток данных последовательно пишутся: заголовок в виде текстовой строки и — в цикле — нормированные элементы соответствующих массивов. При этом суммы степеней времени ожидания делятся на количество NCount принятых к обслуживанию заявок, а суммы длин временны́х интервалов — на текущее значение таймера. Запоминаемые в матрицах новые значения построены по правилам $PLUS$-языка: индексы указаны в квадратных скобках, перед именем переменной опущен указатель X$, выражение в целом заключено в скобки. Очередной элемент выводится в файл. Оба цикла построены с помощью блоков LOOP уже обсуждавшимся способом.

После выдачи числа зафиксированных отказов файл закрывается, и работа модели завершается.

Приведем Стандартный Отчет по данной задаче:

```
GPSS World Simulation Report - Basic1.66.1
  Saturday, September 08, 2001 17:57:53

    START TIME    END TIME  BLOCKS  FACILITIES  STORAGES
      0.000       5000.000    41        0           1

           NAME                          VALUE
          CHANS                       10006.000
          DEM                         10005.000
          DT                          10002.000
          ENTRY                           5.000
          IND                         10004.000
          MEM                            16.000
          NCOUNT                      10010.000
          NREF                        10008.000
          OUT                            24.000
          PROB                        10000.000
          PROBS                          35.000
          TOLD                        10003.000
          WAIT                        10011.000
          WMOMS                          30.000
          WTP                         10001.000
          WTP1                        10012.000
          WTPW                        10013.000
```

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	4957	0	0
	2	TEST	4957	0	0
	3	SAVEVALUE	14	0	0
	4	TRANSFER	14	0	0
ENTRY	5	MSAVEVALUE	4943	0	0
	6	SAVEVALUE	4943	0	0
	7	SAVEVALUE	4943	0	0
	8	SAVEVALUE	4943	0	0
	9	MARK	4943	0	0
	10	QUEUE	4943	12	0
	11	ENTER	4931	0	0
	12	DEPART	4931	0	0
	13	SAVEVALUE	4931	0	0
	14	SAVEVALUE	4931	0	0
	15	ASSIGN	4931	0	0
MEM	16	MSAVEVALUE	19724	0	0
	17	SAVEVALUE	19724	0	0
	18	LOOP	19724	0	0
	19	ADVANCE	4931	3	0
	20	LEAVE	4928	0	0
	21	MSAVEVALUE	4928	0	0
	22	SAVEVALUE	4928	0	0
	23	SAVEVALUE	4928	0	0
OUT	24	TERMINATE	4942	0	0
	25	GENERATE	1	0	0
	26	OPEN	1	0	0
	27	SEEK	1	0	0
	28	WRITE	1	0	0
	29	ASSIGN	1	0	0
WMOMS	30	MSAVEVALUE	4	0	0
	31	WRITE	4	0	0
	32	LOOP	4	0	0
	33	WRITE	1	0	0
	34	ASSIGN	1	0	0
PROBS	35	MSAVEVALUE	20	0	0
	36	WRITE	20	0	0
	37	LOOP	20	0	0
	38	WRITE	1	0	0
	39	WRITE	1	0	0

```
         40    CLOSE              1         0           0
         41    TERMINATE          1         0           0

QUEUE  MAX CONT. ENTRY ENT RY(0) AVE.CONT. AVE.TIME AVE.(-0) RETRY
 WAIT   16   12   4943   1 086     2.851     2.884   3.695     0

STORAGE  CAP. REM. MIN MAX ENTRIES AVL.  AVE.C. UTIL. RETRY  DELAY
 CHANS     3    0   0   3   4931    1    2.662  0.887   0     12

SAVEVALUE               RETR Y       VALUE
 TOLD                     0        4999.601
 DEM                      0          15.000
 NREF                     0          14.000
 NCOUNT                   0        4943.000
 WTP1                     0          10.068
 WTPW                     0      103453.236

FEC XN  PRI    BDT     ASSEM  CURRENT  NEXT PARAMETER   VALUE
 4944    0   5000.527   4944     19      20     1      4986.353
                                                2         0.000
 4959    0   5000.655   4959      0       1
 4945    0   5000.735   4945     19      20     1      4987.796
                                                2         0.000
 4946    0   5001.141   4946     19      20     1      4988.373
                                                       2  0.000
```

Сопоставим результаты расчет а выведенных в файл стационарных вероятностей состояний, расс читанных тремя различными способами: теоретически и с помощью ими тационных моделей на *Фортране* и $GPSS/W$. Результаты моделирования на $GPSS/W$ согласуются с теоретическими несколько лучше, чем «фортранные».

Таблица 16.3. Вероятности состояний в модели M/D/3

Индекс	Теория	*Фор тран*	*GPSS*
0	.21211E-01	.2796 8E-01	.25882E-01
1	.65259E-01	.8001 1E-01	.71617E-01
2	.10585E 00	.121 81E 00	.11686E 00
3	.12329E 00	.13823E 00	.14255E 00
4	.11890E 00	.13386E 00	.13353E 00
5	.10358E 00	.11452E 00	.10801E 00
6	.86095E-01	.96647E-01	.86455E-01
7	.70270E-01	.82025E-01	.71697E-01
8	.57268E-01	.62608E-01	.56061E-01
9	.46535E-01	.47752E-01	.42398E-01
10	.37813E-01	.27362E-01	.28896E-01
11	.30725E-01	.15980E-01	.24298E-01
12	.24966E-01	.11482E-01	.21242E-01
13	.20287E-01	.99460E-02	.21384E-01
14	.16485E-01	.67521E-02	.15112E-01
15	.13395E-01	.66284E-02	.11503E-01
16	.10884E-01	.57152E-02	.98450E-02
17	.88442E-02	.56420E-02	.61810E-02
18	.71865E-02	.30881E-02	.35149E-02
19	.58395E-02	.19789E-02	.28844E-02

16.3 Приоритетные системы обслуживания

При большом числе типов заявок неоднородного потока удобно генерировать суммарный поток, разыгрывая затем тип заявки обращением к процедуре `Duniform(RN, Min,Max)`. Этот тип записывается в один из параметров транзакта, после чего с помощью соответственно определенных выражений `VARIABLE` можно в блоках `ASSIGN` означить требуемое число дополнительных параметров.

В приведенной ниже программе обслуживания с прерываниями генерируются независимые простейшие потоки заявок трех типов. Программа состоит из преамбулы и четырех сегментов. В преамбуле определяются матричные объекты, используемые для накопления и представления результатов:

Tmoms — моменты распределения времени пребывания в системе (по три момента для трех типов заявок),

Tmops — накопители степеней времени пребывания для упомянутых моментов,

Ntrans — количество обработанных заявок каждого вида.

Рабочая переменная формирует случайное время обслуживания заявки по закону Эрланга третьего порядка (третий аргумент обращения к распределению Gamma). Второй аргумент обеспечит равенство среднего значения распределения параметру P2 активного транзакта. Первый аргумент указывает номер базового ДСЧ.

```
***************************************************************
*                   M/E3/1 PRIORITY MODEL                     *
***************************************************************
Tmoms  MATRIX     ,3,3
Tmops  MATRIX     ,3,3
Ntrans MATRIX     ,1,3
Work   VARIABLE   (Gamma(2,0,P2/3,3))  ; Working time - by P2
***************************************************************
       GENERATE   (Exponential(1,0,5.00)),,,,3 ; 1-st type
       ASSIGN     1,1                  ; Transact's type
       ASSIGN     2,0.3                ; Mean service time

Common MARK                            ; Entrance moment
       QUEUE      Sojourn              ; Waiting begins
       PREEMPT    Chan,PR,Rept,100     ;
       SAVEVALUE  Time,V$Work          ; Working time
       ASSIGN     5,X$Time             ; To 5-th parameter
Wait   ADVANCE    P5                   ; Working time delay
       TRANSFER   ,Out                 ; Service is completing
;Rept  ASSIGN     5,X$Time             ; The former - RW
 Rept  ASSIGN     5,V$Work             ; Renewing   - RS
       TRANSFER   ,Wait                ; New attempt
 Out   DEPART     Sojourn              ; Sojourn ending
       RETURN     Chan                 ; Facility release
       MSAVEVALUE Ntrans+,1,P1,1       ; Counting P1-transactions
       SAVEVALUE  Zpow,M1              ; Starting power=sojourn time
       ASSIGN     3,3                  ; P3:=3
```

```
 Cum   MSAVEVALUE Tmops+,P1,(4-P3),X$Zpow ; Power saving
       SAVEVALUE Zpow,(X$Zpow#M1)      ; Power accumulation
       LOOP      3,Cum                 ; Cycle, P3=P3-1
       TERMINATE                       ; Trajectory ends
*****************************************************************
       GENERATE  (Exponential(1,0,3.33)),,,,2 ; 2-d type
       ASSIGN    1,2
       ASSIGN    2,0.6
       TRANSFER  ,Common
*****************************************************************
       GENERATE  (Exponential(1,0,2.50)),,,,1 ; 3-d type
       ASSIGN    1,3
       ASSIGN    2,0.9
       TRANSFER  ,Common
*****************************************************************
       GENERATE   5000,,,,1
       OPEN       "Prty.txt"
       WRITE "**** SOJOURN TIME MOMENTS ***** "
       SAVEVALUE XXX,(Outmoms())  ; Procedure call
       ASSIGN    3,3              ; P3:=3
iloop  WRITE     (CATENATE("Type of Demands is ",4-P3));
       ASSIGN    4,3              ; P4:=3
jloop  WRITE (Tmoms[4-P3,4-P4])   ;
       LOOP      4,jloop          ; Cycle, P4:=P4-1
       LOOP      3,iloop          ; Cycle, P3:=3-1
       CLOSE 1
       TERMINATE 1
*****************************************************************
PROCEDURE Outmoms()BEGIN
   TEMPORARY Indi,Indj,Text;
   Indi=0;
   WHILE (Indi<3) DO BEGIN
      Indi=Indi+1; Indj=0;
      WHILE (Indj<3) DO BEGIN
       Indj=Indj+1;
         Tmoms[Indi,Indj]=Tmops[Indi,Indj]/Ntrans[1,Indj];
      END;
   END;
END;
*****************************************************************
```

Первый сегмент описывает «биографию» транзактов первого типа — с наибольшим приоритетом (см. последнее поле блока GENERATE). Пути транзактов всех типов смыкаются в точке с меткой COMMON. В этой точке транзакт получает метку момента прибытия и далее входит в очередь Sojourn, которую покидает (блок DEPART) только после завершения обслуживания. Транзакт пытается по праву приоритета (PR) занять устройство Chan. В случае успеха в ячейке Time запоминается требуемое время обслуживания, которое блок ASSIGN записывает в 5-й параметр транзакта. Блок ADVANCE задерживает транзакт в устройстве на это время. Если не будет прерывания, транзакт покинет систему. В противном случае прерванный транзакт запишет в 100-й параметр недоданное ему время обслуживания и перейдет в блок с меткой Rept, который в модели представлен в двух версиях. Рабочая версия (дисциплина RS) заносит в 5-й параметр *новое* случайное время обработки, сформированное по тому же закону, после чего транзакт вновь пытается войти в имитирующий обслуживание блок ADVANCE. Версия RW, закомментированная точкой с запятой, записывает в этот параметр *прежнюю* реализацию случайной длительности обслуживания. В случае прерывания с дообслуживанием метку Rept следует ставить непосредственно перед ADVANCE.

Заключительный сегмент таймера задает окончание моделирования и описывает процесс обработки. В нем открывается файл Prty.txt для записи результатов. Туда заносится общий заголовок «Моменты распределения времени пребывания», после чего через фиктивное присваивание вызывается процедура Outmoms, вычисляющая упомянутые моменты. Далее завершающий транзакт, крутясь в двойном LOOP-цикле, построчно выводит результаты с 16 десятичными цифрами. К сожалению, ни форматом, ни размещением этой информации управлять нельзя. Заметим также, что вывод изнутри процедуры невозможен — там не могут находиться *блоки GPSS*.

Обслуживание с относительным приоритетом (без прерывания) реализуется в *GPSS* по умолчанию без всяких усилий со стороны программиста: достаточно назначить заявкам приоритет в блоке GENERATE или PRIORITY и организовать дифференцированную обработку показателей обслуживания транзактов разных приоритетов.

Обслуживание с *динамическим приоритетом* (простейший и наиболее типичный случай — линейный рост приоритетов по времени ожидания с угловыми коэффициентами, зависящими от приоритетов), в *GPSS* даже с помощью мощного аппарата цепей пользователя обеспечить не

удается. Дело в том, что цепь может быть переупорядочена произвольно только в момент входа в нее новой заявки, тогда как нам для реализации динамического приоритета необходимо переупорядочение в момент выборки. На такой момент максимум возможного — это инверсия, то есть реализация дисциплины LIFO (Last In – First Out). Предложенная в книге Шрайбера [74, с. 391] схема «динамического приоритета» на самом деле реализует *другую* дисциплину выбора из очереди: по возрастанию времени обработки.

В заключение этого раздела обсудим пример технической ситуации, требующей приоритетного обслуживания — моделирование компьютерных операций ввода-вывода. Дисковый накопитель может одновременно выполнять только одну операцию обмена. Доступ к дисководу организуется через селекторный канал, который моделируется устройством $GPSS$. Еще одно устройство представляет центральный процессор. Команда «Начать ввод-вывод» указывает нужную дорожку. Дисковод будет двигать головку считывания на эту дорожку. Пока это делается, процессор и канал могут быть заняты другими делами. Когда дорожка найдена, возникает дополнительная задержка — ожидание начала дорожки (равновероятное от нуля до времени полного оборота). Затем дисковод прерывает ЦП на операцию обмена. Схема соответствующего фрагмента $GPSS$-модели должна быть выстроена в следующем порядке:

```
SEIZE   CHAN
SEIZE   CPU
ADVANCE CPU  ; Выдача команды обмена
RELEASE CPU
RELEASE CHAN
ADVANCE      ; Выбор дорожки
ADVANCE      ; Ожидание доворота
SEIZE   CHAN
PREEMPT CPU
ADVANCE      ; Обмен
RETURN
RELEASE CHAN
```

Еще один пример ситуации с приоритетами — работа в офисе, где клерк принимает документы у посетителей. С определенной вероятностью в документах обнаруживается ошибка. Клиент направляется к

начальнику, который вносит дополнительные сведения и исправления. Закончив работу, начальник может прервать клерка, уже занятого обслуживанием нового клиента, и дать ему указания по дооформлению. Далее идет дооформление, которое можно приплюсовать к длительности прерывания.

16.4 Квантование времени

В простейшей модели этого вида имеется общая очередь. Выбранная в процессор заявка обслуживается требуемое время, но не более длины кванта q. Заявка, успевшая обслужиться, покидает систему, не успевшая — становится в конец очереди на дообслуживание. Приведем основной фрагмент моделирующей эту ситуацию программы:

```
.................................
Intro  SEIZE     Fcty
       TEST G    P5,X$Quantum,Last
       ASSIGN    5,(P5-X$Quantum)
       ADVANCE   X$Quantum
       RELEASE   Fcty
       TRANSFER  ,Intro
Last   ADVANCE   P5
       RELEASE   Fcty
..................................
```

В блоке `TEST` хранимый в параметре P5 остаток обслуживания сравнивается с длительностью кванта `X$Quantum`. Если остаток больше, то:

- эта длительность вычитается из параметра и разность заносится в него же,
- процессор занимается на длительность кванта,
- процессор освобождается,
- заявка повторно адресуется на вход процессора.

В противном случае процессор занимается на оставшееся время обслуживания, после чего освобождается, а заявка уходит на продолжение траектории (например, отметку окончания времени пребывания в вычислительной системе).

16.5 Круговое обслуживание

Во многих компьютерных и связных приложениях возникает проблема поочередного подключения обслуживающего устройства (центрального процессора, передатчика и т. п.) к накопителям заявок. Схема такой организации (Token Ring) представлена на рис. 16.1.

Рис. 16.1. Схема кругового обслуживания

Эта схема может быть реализована в трех вариантах обслуживания очередей:

- по одной заявке из каждой непустой очереди;

- до исчерпания всех заявок в текущей очереди, включая пришедшие за время ее обслуживания;
- до обслуживания всех заявок, накопившихся к моменту *начала* обслуживания данной очереди.

Ниже приводится программа второго варианта. Все заявки считаются однородными, а стандартный отчет — достаточным.

```
**** TOKEN RING MODEL ***************
***  Until exhausting - by Loop  ****
************************************
       INITIAL X$Empty,1
************************************
       GENERATE   (Exponential(1,0,1))
       ASSIGN     1,AC1
       ASSIGN     2,1
       ASSIGN     3,0.25
       TEST E     X$Empty,1,Inq1
       SAVEVALUE  Pointer,P2
       SAVEVALUE  Empty,0
Inq1   QUEUE      *2
       TEST E     X$Pointer,P2
       DEPART     *2
       TRANSFER   ,Server
*************************************
       GENERATE   (Exponential(2,0,1))
       ASSIGN     1,AC1
       ASSIGN     2,2
       ASSIGN     3,0.25
       TEST E     X$Empty,1,Inq2
       SAVEVALUE  Pointer,P2
       SAVEVALUE  Empty,0
Inq2   QUEUE      *2
       TEST E     X$Pointer,P2
       DEPART     *2
       TRANSFER   ,Server
*************************************
       GENERATE   (Exponential(3,0,1))
```

```
       ASSIGN     1,AC1
       ASSIGN     2,3
       ASSIGN     3,0.25
       TEST E     X$Empty,1,Inq3
       SAVEVALUE  Pointer,P2
       SAVEVALUE  Empty,0
Inq3   QUEUE      *2
       TEST E     X$Pointer,P2
       DEPART     *2
************* CPU ********************
Server SEIZE      Fcty
       ADVANCE    P3
       RELEASE    Fcty
       TEST E     Q*2,0,Zzz ; Is the Current Queue Empty?
       ASSIGN     4,3       ; Cycle setup
       ASSIGN     5,1       ; Checked Q
       ASSIGN     6,1       ; Number of Max Q
       SAVEVALUE  Qmax,0    ; Starting Max
Check  TEST GE    Q*5,X$Qmax,Next
       SAVEVALUE  Qmax,Q*5  ; New Max
       ASSIGN     6,P5      ; New number
Next   ASSIGN     5+,1      ; Next Queue
       LOOP       4,Check
Out    TEST E     P6,0,Zpt
       SAVEVALUE  Empty,1   ; System is Empty
       TRANSFER   ,Zzz
Zpt    SAVEVALUE  Pointer,P6; Access to max Q
Zzz    TERMINATE  1
```

Три начальных сегмента однотипны. В каждом из них в первый параметр транзакта записывается значение таймера, во второй — номер источника заявок, в третий — длительность обслуживания. Далее заявка входит в очередь, причем при пустой системе признак пустоты предварительно инвертируется, а указатель Pointer обслуживаемой очереди устанавливается на данную очередь. Если обслуживание очереди разрешено, транзакт покидает ее и поступает на сервер.

Транзакт пытается захватить сервер. При успехе он занимает его на время своего обслуживания, заданное в P2, а затем освобождает. Далее

проверяется, пуста ли текущая очередь. Тогда организуется цикл на три повторения (по числу источников) для выбора очереди максимальной длины. Если этот максимум — нуль, то выставляется признак пустоты системы, в противном случае стрелка переводится на самую длинную очередь.

Реализация дисциплины поочередного обслуживания отличается тем, что в ней опрос очередей является безусловным. Здесь логично выбирать первую непустую очередь с номером, отличным от текущей, и выходом из цикла при обнаружении таковой. Этот вариант можно встроить и в ранее рассмотренную дисциплину.

Для последней из обсуждаемых дисциплин необходима переменная (скажем, `X$TBeg`), запоминающая начало обслуживания текущей очереди. Ее значение наряду со значением указателя `Pointer` учитывается по схеме логического умножения при допуске к серверу очередной заявки. Не пропущенные заявки приходится записывать в цепь пользователя. При завершении обслуживания текущей очереди для каждого источника анализируется суммарное число заявок в очереди и в цепи. Однако после выбора нового положения стрелки блок `UNLINK` требует указания имени блока, в который адресуются освобождаемые заявки. Это имя в нашей модели можно получить с помощью функции `Catenate` — присоединением к стандартному префиксу `Inc` номера выбранного источника. По синтаксису *GPSS* имя блока *не может быть строкой*, и моделирование завершается аварийно. Безуспешными оказались и попытки использовать числовые обозначения блоков с помощью команды `EQU` (было получено сообщение «you can assign your own value to entity label except for block labels»). Таким образом, данная дисциплина — еще один пример ограниченных возможностей интерпретатора *GPSS* [1].

16.6 Замкнутые системы

В замкнутой системе средний интервал между смежными заявками зависит от числа заявок в ней. Отсутствие в *GPSS* системного атрибута, представляющего это число, обходится введением соответствующей сохраняемой величины, на которую можно сослаться в блоке `GENERATE`. Проблема состоит в том, что к моменту изменения числа заявок в системе в цепь будущих событий уже занесен сформированный по *прежней*

[1] Может быть, читатель найдет работающую альтернативу.

интенсивности момент прибытия очередной заявки. Доступа же к содержимому этой цепи программист не имеет. Значит, точное моделирование замкнутых СМО в $GPSS$ с помощью одного `GENERATE` невозможно. Остается громоздкий вариант с отдельными сегментами генерации заявок по максимальному числу активных источников и циркуляцией в каждом сегменте единственной заявки.

16.7 Нетерпеливые заявки

Этим термином обозначаются заявки, для которых ограничено (как правило, случайной величиной) время ожидания начала обслуживания или полная продолжительность пребывания в системе. Примером нетерпеливой заявки является, например, больной, ожидающий донорский орган — печень, сердце, почку. В родстве с рассмотренной проблемой находится задача управления портящимися запасами — продовольствия, крови, медикаментов, модных товаров и т. п.

Момент ухода можно сформировать обычными методами и записать в один из параметров транзакта. Этот момент сопоставляется с моментом выбора заявки из очереди на обслуживание. Если он меньше, заявка учитывается как потерянная и завершает траекторию. При ограничении на полное время пребывания формируется продолжительность занятия устройства как минимум из оставшегося времени терпения и длительности обслуживания. После освобождения устройства по нетерпению учитывается потеря заявки и/или напрасно затраченное время обработки на устройстве.

В книге Гордона [89, с. 293] обсуждается альтернативный подход к моделированию нетерпеливых заявок. Каждый транзакт расщепляется в блоке `SPLIT` на два. Оригинал заносится в цепь пользователя, а копия задерживается в `ADVANCE` на допустимое время терпения, после чего освобождает оригинал, войдя в `UNLINK`. Этот вариант искажает статистику очереди и не доведен до программной реализации. Мы упоминаем о нем потому, что идея связывания с транзактом отсроченных дополнительных событий сама по себе заслуживает внимания. Кстати заметим, что Дж. Шрайбер [74, с. 399] скептически оценивает полезность цепей пользователя и в особенности их индикаторов связи (с. 385). В его книге [74] объемом в 70 авторских листов приведены лишь два не очень убедительных примера работы с цепями пользователя, а в [65] — ни одного.

16.8 Групповые заявки

Прибытие групповых заявок можно моделировать с помощью блока SPLIT, первый аргумент которого задает число дополнительно формируемых заявок. Это число можно сделать случайным ссылкой на встроенную или пользовательскую вероятностную функцию.

16.9 Работа с памятями

В отечественной литературе по *GPSS* термином «память» обычно для простоты обозначают многоканальные устройства (STORAGE). Для входящих в них транзактов не ведется индивидуальная статистика. Памяти неохотно используют для моделирования систем со случайным объемом заявок — главным образом опасаясь попыток уходящего транзакта освободить больше памяти, чем было занято. Проблема легко снимается, если занести требуемый объем в параметр транзакта:

```
****************************************************
**               Storage Using                    **
****************************************************
INV STORAGE 10
    GENERATE 1,0.3
    ASSIGN 1,(DUNIFORM(1,1,6))
    QUEUE QQQ
    ENTER INV,P1
    DEPART QQQ
    ADVANCE 2,0.5
    LEAVE INV,P1
    TERMINATE 1
****************************************************
```

Здесь к памяти объема 10 единиц обращаются заявки с целочисленным объемом, равномерно распределенным между 1 и 6 и записанным в первый параметр транзакта. Покидающая память заявка освобождает ровно столько памяти, сколько запрашивала. Программа отработала успешно, случаев снижения памяти до отрицательного уровня зафиксировано не было.

16.10 Обслуживание с разогревом

Под эффектом разогрева понимается отличие распределения длительности обслуживания головной заявки периода занятости от следующих за ней. Аналитический расчет таких систем оказывается довольно сложным — см. [61]. В простейшем частном случае разогрев сводится к дополнительной постоянной задержке.

Пусть во фрагменте

```
      GATE NU F$Shop,Szf
      ASSIGN 5,V$Warming
Szf   SEIZE F$Shop
      ADVANCE P5
      RELEASE F$Shop
```

при подходе к GATE транзакт с занесенной в P5 длительностью обслуживания обнаруживает, что устройство F$Shop не используется (NU). Тогда параметр P5 пересчитывается по правилу, заданному в определении переменной V$Warming, и транзакт занимает устройство. В противном случае значение P5 не меняется.

16.11 Обслуживание с порогом включения

Автоматические системы обслуживания после исчерпания пришедших заявок обычно выключаются или переходят в дежурный (облегченный) режим. Каждое обратное включение под нагрузку связано не только с дополнительной задержкой, но и с дополнительными затратами, может потребовать привлечения персонала. Кроме того, переключения снижают надежность системы. Частоту включений можно снизить, введя *порог включения*: обслуживание начинается при накоплении в очереди $d > 1$ заявок и прекращается после рассасывания очереди. Аналитический расчет подобной ситуации рассмотрен в [61]. Однако сочетание порога включения с дополнительными усложняющими факторами может потребовать имитационного моделирования.

Логика данной модели состоит в пропуске транзактов на обслуживание лишь при условии скопления в очереди d заявок или уже

установленной занятости устройства. Соответственно можно определить переменную

```
Switch BVARIABLE (Q2>=Ddd 'OR' F2)
```

(предполагается работа с очередью и устройством, имеющими номер 2). Управление доступом транзактов выполняет фрагмент

```
QUEUE   2
TEST E  X$Switch,1
SEIZE   2
DEPART  2
```

16.12 Учет недельного и суточного циклов

Может потребоваться учесть разницу в режимах работы источников заявок и устройств обслуживания. Достаточно типична ситуация с потоком заявок постоянной средней интенсивности и «каналом», закрываемым до утра по окончании рабочей смены в будни и с вечера пятницы до утра понедельника.

```
*****************************************
**        Working cycle                **
*********** Demands ***********************
GENERATE 1,0.3
   QUEUE QQQ
   SEIZE FFF
   DEPART QQQ
   ADVANCE 0.8,0.5
   RELEASE FFF
   TERMINATE 1
********* Everyday Interruptions *************
   GENERATE 24,0,8
   FUNAVAIL FFF
   ADVANCE 16
   TEST E LS2,0,Out
   FAVAIL FFF
Out TERMINATE
********* Weekend Interruptions  *************
```

```
   GENERATE 168,0,103
   LOGIC S 2
   FUNAVAIL FFF
   ADVANCE 65
   FAVAIL FFF
   LOGIC R 2
   TERMINATE
*******************************************
```

Программа состоит из трех сегментов, моделирующих поток заявок с равномерным распределением интервалов между ними, ежедневные и еженедельные прерывания. Само прерывание реализуется переводом устройства `Fff` в состояние неготовности. При этом прекращается и обслуживание заявки, уже занявшей устройство. Предполагается, что работа модели начинается в 8 часов утра в понедельник. Потоки прерываний регулярны (с нулевой вариацией — второй аргумент), третий аргумент в блоках их генерации задает моменты первого прерывания. В недельном цикле учтено пятничное сокращение рабочего дня на один час. Длительность прерываний задается в блоках `ADVANCE`.

Прерывания взаимодействуют друг с другом через логический ключ 2, который устанавливается (S) и сбрасывается (R) с началом и концом уикэнда соответственно. Если ключ не установлен, то по окончании суточного перерыва устройство переводится в рабочее состояние. В противном случае этот перевод выполняется в «недельном» сегменте.

Если предполагается завершение начатого обслуживания независимо от фаз упомянутых циклов, целесообразно ввести в каждом из сегментов прерываний свой логический переключатель и определить булеву переменную как логическую сумму признаков установки ключей. Блок `TEST` должен разрешать занятие устройства только при нулевом значении этой переменной.

16.13 Моделирование цепями правил планирования

Стандартная дисциплина очереди — по убыванию приоритетов с FIFO для каждого. При необходимости в иной дисциплине нужно применять *цепи пользователя*. Поле `C` блока `LINK` позволяет сделать это без допол-

нительного управляющего цикла. В нем (и в поле B) указывается, куда послать транзакт. Если входящий транзакт нашел цепь пустой, транзакт немедленно посылается по адресу из C. Эта опция эффективна только при условии, что заявки выстроены в порядке желаемой выборки.

16.13.1 Вводный пример

Примером такой технологии может служить программа [89, с. 287]. Машина может оперировать с деталями нескольких видов. Собираются заявки на детали различных типов; пачки заявок отправляются, когда машина свободна.

Транзакт генерируется через 60 ± 20 мин. Параметр 1 представляет тип детали, параметр 2 — количество. Имеется пять типов деталей, объем заявки до 10 штук. Распределения для простоты предполагаем равномерными. Транзакты проходят через два ASSIGN-блока для означивания параметров. Блоки ссылаются на две табличных функции. Затем заявки входят в LINK и присоединяются к цепи 1 в порядке FIFO — ждать отправления. После отправления они входят в ADVANCE по имени JOBQ — ждать очереди на машину 1. Предполагается, что каждая деталь требует 10 единиц времени, переменная 1 формирует задержку в ADVANCE как 10, умноженное на записанное в параметре 2 число деталей.

Первая часть программы формирует цикл с одним транзактом, который управляет распределением. GENERATE создает транзакт в начале прогона и вбрасывает его в цикл. Когда машина заканчивает одну пачку заявок, управляющий транзакт входит в UNLINK для выбора следующей пачки. Ниже будут обсуждаться различные правила, обычно основанные на типах заявленных деталей. Управляющий транзакт сначала входит в блок INDEX для установки параметров 1–5. Цикл прокручивается 5 раз, затем делается новая инициализация для дополнительных 5 раз. Цикл не начинается, пока есть заявки (блок TEST проверяет, что число транзактов CH1 в первой цепи больше нуля).

Ожидается (через GATE), что устройство 1 свободно. Тогда транзакт входит в UNLINK для выдачи одного или более заказов. Малая задержка ADVANCE после UNLINK возвращает систему ко второму сегменту программы. Далее управляющий транзакт входит в BUFFER, что позволяет просмотреть ЦТС с начала и учесть приоритет управляющего транзакта (он был сгенерирован с приоритетом 1). Когда система окажется

свободной от транзактов, управляющий транзакт прекратит циркуляцию и возобновит ее, как только транзакт войдет в цепь. Фактически все это нужно для правильной обработки заявки, пришедшей в свободную систему. Приведем текст программы:

```
        GENERATE    ,,,1
 BBB    INDEX       1P,5
 AAA    TEST G      CH1,0
        GATE NU     1
        UNLINK E    1,JOBQ,ALL,1P
        BUFFER
        LOOP        1P,AAA
        TRANSFER    ,BBB
***********************************
        GENERATE    60,20,,,1
        ASSIGN      1,1,1,P

        ASSIGN      2,1,2,P
        LINK        1,FIFO
 JOBQ   ADVANCE
        SEIZE       1
        ADVANCE     V1
        RELEASE     1
        TERMINATE   1
***********************************
 1      VARIABLE    10#2P
***********************************
 1      FUNCTION    RN1,C2
        0,1/1,6
***********************************
 2      FUNCTION    RN1,C2
        0,1/1,11
***********************************
        START       1000
```

16.13.2 Варианты дисциплин обслуживания

Через UNLINK можно реализовать различные управляющие правила. Если правило — заявки фиксированного объема, счетчик должен быть

помещен в C при пустом D:

```
UNLINK  1,JOBQ,4,FIFO
```

Партии будут по 4 заявки, пока не останется меньше 4. Тогда будет заказан остаток. Поскольку индекс типа не учитывается, нет нужды во внешнем цикле первого сегмента.

Положим теперь, что заказываются все заявки на деталь конкретного типа, а типы просматриваются в обратной последовательности. Тогда следует записать

```
UNLINK  1,JOBQ,ALL,1P
```

Здесь D ссылается на параметр, но E пусто.

16.13.3 Ограничение времен ожидания в очередях

Заявки могут отказываться становиться в очередь, если она превышает некоторую длину (может задаваться вероятность в зависимости от длины). Эта ситуация не требует новых средств программирования.

Иначе обстоит дело в случае лимита по *времени* ожидания. «Нетерпеливая» заявка должна быть удалена из очереди или перенаправлена. «Отвязывание» транзакта из цепи — единственный общий метод вывода из множества без индивидуального рассмотрения транзактов. Получив отметку времени входа в очередь (например, в параметре P3), транзакт поступает в SPLIT. Исходный присоединяется к цепи AAA в блоке LINK, тогда как копия идет в ADVANCE, где задерживается на допустимое время ожидания. Затем она входит в

```
UNLINK AAA,OUT,1,3P
```

после чего уничтожается. Совпадение задержек выведет транзакт из цепи на метку OUT, определяющую судьбу нетерпеливой заявки.

Тем же способом можно учесть ограничение по температуре транзакта (например, в прокатном или кузнечном цехе). Здесь проще всего пересчитать допустимое снижение температуры во временной интервал.

16.14 Моделирование процессов сборки

Процессы расщепления и слияния заявок чрезвычайно неудобны для аналитических исследований, но весьма типичны для практических приложений — в частности, организации промышленного производства, пере-

дачи данных по сетям с коммутацией пакетов и т. п. Приведем заимствованную из [89] программу моделирования штучного производства мягких стульев — с расширенным комментарием:

```
*****************************************************
**           Factory model                         **
*****************************************************
         GENERATE   4,3       ;Формирование заявок
         SPLIT      5,Shop,1  ;5 производных заявок
Paper    ADVANCE    8,3       ;Исх. заявка - в бухгалтерию.
Mtch2    MATCH      Mtch1     ;Заявка ждет готового стула
         TERMINATE
*****************************************************
Shop     TRANSFER   FN,1      ;Распред. производных заявок
*
Frame    ADVANCE    4,1       ;Изготовление каркаса с ножками
         TRANSFER   ,Asm
*
Cush     ADVANCE    3,1       ;Изготовление подушек
         TRANSFER   ,Gthr
*
Cvrs     ADVANCE    2,1       ;Изготовление чехлов
Gthr     GATHER 4             ;Получение чехлов и подушек
         ADVANCE    1,0       ;Обтягивание подушек чехлами
Asm      ASSEMBLE   5         ;Окончательное комплектование
         ADVANCE    3,1       ;Сборка и упаковка
Mtch1    MATCH      Mtch2     ;Выдача заказа
         TERMINATE  1
*
1        FUNCTION   P1,L6;    ;Распред. производных заявок
            1,Paper/2,Frame/3,Cush/4,Cush/5,Cvrs/6,Cvrs
```

В первом сегменте отметим блок SPLIT, формирующий 5 производных заявок — одну на каркас и по две на подушки и чехлы [2]. Номера заявок будут занесены в первый параметр каждого транзакта. Исходная заявка задерживается на время оформления и оплаты заказа, после чего будет дожидаться изготовления изделия.

[2] Возможно дополнительное разделение каркаса на 4 ножки и пр.

Второй сегмент начинается с рассылки производных заявок по списку, заданному функцией 1. Аргументом функции является первый параметр транзакта. Отметим принципиальную возможность как генерации случайного числа производных транзактов, так и случайного их распределения.

Блок GATHER собирает чехлы и подушки (обтягивание подушек начинается после получения четырех изделий). Комплектование завершается при наличии обтянутых подушек и каркаса. С этого момента все производные заявки сливаются в один транзакт. Собранный и упакованный стул синхронизируется с оплатой и оформлением документации.

16.15 Прикладные задачи

Интересующимся методами моделирования в конкретных предметных областях мы рекомендуем разбор «образцовых» программ из папки Sample Models — разработанных Дж. Ф. Каммингсом (Gerard F. Cummings) и другими авторами примеров из широкого спектра приложений. Их совокупность имеет целью показать применение техники моделирования к проблемам реального мира. Примеры упрощены в двух смыслах:

1. Прогоны взяты короче: более длинные дали бы результаты с меньшим разбросом.

2. Не рассматривались более изощренные статистические методы.

Объектами моделирования являются:

ASSEMBLY.GPS — процесс сборки насосов;

BICYCLE.GPS — велосипедная фабрика;

ETHERNET.GPS — сеть Ethernet;

EXCHANGE.GPS — телефонный коммутатор с внутренними и внешними линиями связи;

EXPERETHER.GPS — эксперимент с сетью Ethernet;

FMSMODEL.GPS — гибкое автоматизированное производство;

FOUNDRY.GPS — литейный цех;

`GRAECOLATIN.GPS` — планирование Эксперимента с применением греко-латинских квадратов;

`LATINSQUARE.GPS` — планирование Эксперимента с применением латинских квадратов;

`LOCKSIMN.GPS` — канал и шлюз;

`MANUFACT.GPS` — система производства электроники;

`MULTIWAY.GPS` — многофакторный дисперсионный анализ;

`OILDEPOT.GPS` — нефтебаза;

`ORDERPNT.GPS` — запас с критическим уровнем возобновления;

`PERIODIC.GPS` — запас с периодическим контролем уровня;

`POWDER.GPS` — торговля стиральным порошком;

`PREDATOR.GPS` — хищники и добыча (динамика популяций);

`QCONTROL.GPS` — система контроля качества;

`QTHEORY.GPS` — задача теории очередей с известным решением;

`ROBOTFMS.GPS` — гибкое автоматизированное производство;

`SHIPPORT.GPS` — порт;

`STOCKCTL.GPS` — иерархическая система складов;

`SUPERMRK.GPS` — универмаг;

`TAPEPREP.GPS` — подготовка управляющих программ для металлорежущего оборудования с ЧПУ;

`TELEPHON.GPS` — простая телефонная система;

`TEXTILE.GPS` — текстильная фабрика;

`TRAFFIC.GPS` — транспортные потоки на Т-образном перекрестке;

`TURNSTIL.GPS` — турникет на футбольном стадионе;

`TVREPAIR.GPS` — мастерская по ремонту телевизоров.

Все они обсуждаются подробно, имеются в машиночитаемой форме (папка `Sample_Models`) и могут быть запущены на обработку. Каждый пример сопровождается отчетом; описано рекомендуемое взаимодействие с системой. Следует иметь в виду, что разбор профессионально выполненных разработок — лучший путь приобрести собственное умение.

Поскольку эти примеры занимают много места и представляют частный интерес, в данную книгу в соответствующем контексте были включены лишь необходимые для освоения работы с системой.

В той же папке имеются многочисленные примеры моделей из нестареющей книги Т. Шрайбера [74], перепрограммированные с учетом новых возможностей $GPSS/W$. Соответственно в упомянутой книге можно найти развернутые пояснения к этим моделям (за ними сохранены данные Шрайбером номера). Разумеется, надо иметь в виду возможности аналогий (`Barber Shop` может служить моделью не только парикмахерской, но и косметического салона, мастерской по изготовлению ключей, ремонтных служб, служб спасения, пунктов первой медицинской помощи и т. п.). Кроме того, подходящим может оказаться только название проблемной области, но не специфика ситуации. Поэтому предпочтительны абстрактная формулировка задачи (в терминах теории очередей) и инвариантные к приложениям приемы моделирования.

Специалистам в области техники связи и компьютерных систем будут полезны примеры из учебного пособия Б. Я. Советова и С. А. Яковлева [65]. В частности, там разобраны задачи моделирования:

- абстрактной ациклической сети обслуживания,
- участка транспортного цеха автомобильного завода,
- процесса передачи данных в информационно-вычислительной сети с коммутацией пакетов,
- работы концентратора сети интегрального обслуживания,
- испытательного участка сборочного цеха предприятия,
- локальной сети связи ЭВМ со случайным доступом к моноканалу,
- вычислительного центра коллективного пользования,

а также приведены 45 разнообразных заданий на моделирование.

В качестве задания на самостоятельную работу рекомендуем составить модель обслуживания читателей в библиотеке. Библиотекарь пассивен, пока нет заявок. В программе необходимы два сегмента: стол выдачи и библиотекарь. Возможен и третий: перерывы на кофе (на проветривание). При наличии очереди библиотекарь забирает до трех заявок одновременно, соответственно возрастает время поиска книг. Время на собственно выдачу учитывается отдельно для каждого читателя. Число заявок можно задать, пропустив библиотекаря через условный `TRANSFER`.

Библиотекарей может быть несколько; очередную партию заявок берет тот, который раньше всех освободился.

16.16 Оценка GPSS World

GPSS/W является весьма ценным инструментом имитационного моделирования, свободным от ограничений аналитических и численных методов, достаточно «прозрачным», допускающим нестандартную обработку данных и снимающим с программиста множество нетривиальных проблем программирования и отладки моделей. Тем не менее приходится отметить наличие у нее ряда серьезных недостатков:

1. Громоздкость системы и явную перегруженность встроенными возможностями (многообразие примитивов); непомерное разнообразие графических обозначений блоков, не поддерживаемое даже новейшими версиями *Visio*.

2. Медленная работа интерпретатора.

3. Отсутствие концептуального единства. Достаточно указать, например, различие в обращении к элементам матрицы при простой ссылке и изменении значения, круглые индексные скобки в основном тексте и квадратные — в *PLUS*-выражениях; обязательность приставки `MX$` при ссылке на глобальную матрицу в тексте модели и столь же обязательное ее отсутствие внутри процедур; контекстно зависимый вид ссылок на параметры активного транзакта; потерявшее смысл в *GPSS/W* различение переменных `VARIABLE` и `FVARIABLE`; выражение коэффициента использования устройства в тысячных долях.

4. Путаная семантика операндов: в разных операторах одно и то же значение закреплено за разными операндами — было бы гораздо легче

работать с ключевыми операндами, имеющими значимые имена.

5. Неудачные обозначения операторов отношения `L, G, E` (было бы лучше согласовать с фортрановскими); арифметическое `SQR` используется для квадратного корня (в Паскале так обозначается *квадрат*); в связи с «числовым» представлением логических значений и объединением понятий числового равенства и логической эквивалентности нарушено общепринятое старшинство операций; состояние логических ключей описывается как `SET` и `RESET` (буквальный перевод «установлен» и «установлен заново») вместо `ON`, `OFF`; операнд `RE` (традиционный смысл этой приставки — повторение действия) означает удаление.

6. Однократность прерываний устройств и недопустимость прерываний для памятей, которые могут быть использованы для моделирования многоканальных устройств обслуживания и, следовательно, окажутся подвержены прерываниям.

7. Дополнительные установки действий функциональных клавиш для каждой модели приходится задавать заново; было бы удобнее иметь прямой доступ к заданию и корректировке установок по умолчанию.

8. Отсутствуют средства подбора параметров теоретических вероятностных распределений по заданным моментам, а также формулы для моментов логарифмического распределения Лапласа, логарифмически логистического, распределений крайних значений и обратного гауссова. Нет непосредственных возможностей генерации коррелированных случайных величин.

9. Комментарии с кириллическими символами исключают правильную работу Имитации.

10. Невозможно непосредственно определить вектор. Нижняя граница индексов матрицы по любому измерению равна 1, что может нарушить естественность индексации (например, при расчете вероятностей состояний системы). Крайне неудобен просмотр матриц через динамическое окно: одновременно можно видеть только два столбца. Ненулевая инициализация матрицы требует отдельного оператора `INITIAL` на каждый элемент.

11. Отсутствует возможность регулировать темп автоматического моделирования; он оказывается слишком быстрым, чтобы разглядеть нужные эффекты, а пошаговое продвижение транзактов — чересчур медленным.

12. Иконки свернутых динамических окон неразличимы, и при их восстановлении трудно выбрать желаемое.

13. Нельзя менять тип шкал графиков (только линейные) и их разметку, цвет и структуру линий, что может сделать их неотличимыми друг от друга и/или от фона при черно-белом выводе.

14. Аргументом графиков может быть только время, так что распределение вероятностей состояний системы автоматически построить нельзя. Для такого графика обязательна логарифмическая шкала.

15. При работе с `ANOVA` нельзя менять расчетную значимость.

16. Оставляет желать лучшего электронная документация к системе. При вызове `Help`'а по ключевому слову пользователю представляется «слепое» множество альтернатив, так что приходится выполнять их перебор. Часто встречаются пережитки предыдущих версий ($GPSS/PC$). Отсутствует упоминание о наличии в системе операции отрицания `'NOT'`. Заявление о числовом значении «his value has a magnitude limited to 306 decimal digits» можно понять как длину мантиссы, хотя на самом деле речь идет о модуле десятичного порядка. Ничего (кроме факта существования) не сказано о библиотеке пользовательских функций. В разделе определения табличных функций понятие обратной функции заменено кумулянтой. В заголовке определения команды `MATRIX` неверно указано число операндов, не выделен строкой и буквой неиспользуемый первый операнд. Этот перечень можно было бы продолжить.

Работа на $GPSS$ и других подобных языках безусловно оправдана при массовом создании моделей средней сложности. Эпизодическую разработку моделей умеренной сложности и создание проекта уникальной сложности программист с опытом работы на любом языке высокого уровня может вести на этом языке, причем знакомство с основными понятиями $GPSS$ подскажет ему ряд полезных технических приемов.

Глава 17

NetCracker

17.1 Общая характеристика

Сложность состава и алгоритмов работы многих реальных систем и сетей обслуживания усложняет разработку их имитационных моделей и определяет целесообразность создания в этих целях специального инструментария. Таким инструментом для разработки и исследования вычислительных сетей и сетей связи является *NetCracker*. Этот инструмент предоставляет пользователю:

- Обширную базу данных, содержащую информацию о технических характеристиках и поставщиках тысяч *реальных* устройств.

- Возможность соединения этих устройств (с учетом их типов и совместимости) каналами связи с реальными свойствами.

- Современный графический интерфейс, позволяющий по технологии втаскивания `drag and drop` включать в проект необходимые устройства; оснащать их встраиваемыми дополнительными элементами (например, сетевыми картами); объединять однородные устройства в стеки и разъединять последние; задавать установку математического обеспечения различных видов трафика (отдельно для клиентов и сервера); дополнять проект рисунками и текстом, выполненными как встроенными средствами самой системы, так и внешними (*Visio*). В связи с интерфейсом отметим также реализованный в системе синтез англоязычных речевых сообщений.

- Возможность моделирования функциональных характеристик сети с учетом протоколов передачи данных и с управлением множеством факторов: статистическими параметрами потоков заявок и объема сообщений; типом трафика; имитацией отказов и восстановлений устройств и каналов связи с автоматическим перераспределением потоков; отображением результатов моделирования непосредственно в окне проекта.

- Наглядное представление процесса моделирования в форме анимации, показывающей пути и характер передаваемой информации.

- Многоуровневое иерархическое построение проектов, позволяющее исследовать сети от локального до глобального уровня.

- Средства формирования отчетов о составе, стоимости и рабочих характеристиках сети (при необходимости — по отдельным подсетям).

Термин «высокоскоростная интеллектуальная анимация» является торговой маркой разработчика; не вполне ясно, что здесь имеется в виду: то ли использование средств искусственного интеллекта (каких?), то ли реализация графического интерфейса, то ли просто подчеркивается наличие базы данных и многообразие учитываемых факторов. Вообще у разработчиков нет оснований гордиться этим аспектом обсуждаемого (и безусловно полезного в целом) программного продукта.

В составе пакета (в каталоге `bin`) имеются `.doc`-файлы:

- Первые шаги (`Quick Start Guide`).

- Краткое Руководство (`Tutorial`).

- Справочник по системе (`Users Guide and Reference Manual`).

- Спецификация версии `Release Notes` (`Readme`).

Из них первый знакомит с основными операциями, а второй дает семь комплексных уроков освоения различных аспектов работы с системой и в части технологии многократно дублирует первый, что для начального знакомства с системой представляется вполне оправданным. Эти два источника в совокупности занимают около 70 страниц; их целесообразно

распечатать и использовать в указанной последовательности. Оба они входят в состав help'а.

Первая часть третьего источника —- *Справочника по системе* (60 страниц) — включает в себя общую характеристику системы, начало работы, обзор статистических возможностей, приборы, установление связей, генерацию потоков данных, технологию анимации, проектирование многоуровневых сетей, сценарии тестирования, исследование. Остальные 80 страниц описывают команды, меню и диалоги.

NetCracker распространяется с восемью типовыми проектами: tutor, techno, router, hier, circuit_switching, client_server, frame_relay, sna. Все они имеют расширение .net и помещены в директорию Samples.

Вся документация дается на английском языке; то же относится к панелям, кнопкам и содержимому базы данных. В некоторых случаях, однако, прорезывается русификация *Windows*: Close заменяется на Закрыть, Finish — на Готово, Next — на Далее и т. п. По этой причине (а также вследствие неполной отлаженности системы) возможны некоторые несоответствия между приводимыми в Руководствах экранными снимками и фактическим поведением системы.

Приводимый ниже материал не заменяет перечисленные источники и имеет целью дать *общее представление* о возможностях системы и ее использовании. Поэтому мы воздержимся от углубления в специфические проблемы организации вычислительных сетей, перевода и толкования узкоспециальных терминов. Из технологических приемов выполнения операций как правило рассматривается один.

17.2 Инсталляция

Согласно упомянутой спецификации, для нормальной работы системы необходимы:

- процессор Pentium II,
- 50 Мбайт дисковой памяти,
- 64 Мбайта оперативной (рекомендуется 128),
- 2 Мбайта видеопамяти (лучше 4),

- дисплей SVGA с разрешением 800x600 и 16-битовой цветовой палитрой,

- звуковая карта (для аудио-сообщений по-английски),

- операционная система *Windows* 95/98 или *Windows NT* 4.x.

Опыт показал, что достаточна намного более скромная конфигурация: просто Pentium-166, 24 Мб оперативной памяти и разрешение экрана 600x480. Процедура инсталляции реализуется запуском `Setup`. Для установки надо выбрать логический диск и *создать целевую папку*, иначе инсталляция пойдет в корневой каталог и замусорит его. Предоставляется стандартная возможность деинсталляции.

17.3 Панели инструментов

Выбор выводимых в главное окно (рис. 17.1) системы панелей производится через команду `View` главного меню. Назначение каждой конкретной кнопки при задержке на ней курсора высвечивается (по-английски). Поэтому мы ограничимся обзором содержания панелей.

Standard Tools (на рисунке — первые 9 кнопок верхнего ряда, кончая указателем с вопросительным знаком) содержит кнопки типичных для всех *Windows* -систем операций открытия нового и ранее созданного проектов, сохранения проекта, работы с буферной памятью, печати, ее предварительного просмотра и контекстно-зависимого `help`'а. Надобность в этой панели возникает только в начале и конце сеанса работы с проектом.

Zoom Tools также представлена в верхнем ряду иконок рисунка. Она позволяет уменьшать и увеличивать изображение в активном окне: шагами, до возможного максимума в активном окне и на весь экран; задавать масштаб; возвращаться к прежнему и к реальному (1:1) масштабу; увеличивать выделенный фрагмент проекта. Эту панель желательно иметь активной постоянно.

Рис. 17.1. Главное окно *NetCracker*

Database Tools (ниспадающий список типов оборудования/поставщиков и по четыре иконки слева и справа от него) используется при создании и корректировке состава сети для управления сортировкой и отображением данных об устройствах: размером иконок, отображаемым текстом и его редакцией, иерархией — по типам устройств или производителям, для организации поиска. Кнопка `Compatibles` (правый бинокль) отбирает для вывода в окно базы данных только устройства, совместимые с выделенным в окне проекта.

Control Tools начинается с иконки треугольника, повернутого вершиной вправо. Эта панель необходима для управления анимацией. Ее кнопки позволяют начать, завершить, прервать анимацию; ускорить или замедлить ее; вернуться к номинальной скорости; вызвать диалог `Animation Setup`, позволяющий определить объем сообщений, скорость их генерации и размер изображения.

Modes Tools представлена на рисунке в усеченном виде (5 иконок из 10). Она управляет специальными режимами системы: возвращает указателю его прямое назначение; вызывает панель рисования; организует связи; вызывает установку трафика; управляет режимами телефонных линий для разговора и передачи данных; моделирует отказ и восстановление устройств и линий связи; задает трассировку; включает и отключает звуковое общение с пользователем.

Drawing Tools на рисунке не показана. Она используется при необходимости дополнения стандартных изображений приборов и связей простейшими рисунками и надписями. Инструменты этой панели позволяют рисовать прямые линии, прямоугольники, овалы, эллипсы и круги. Выбранный инструмент щелчком мыши приводится в активное состояние (иконка утапливается), после чего курсор помещается в нужное место рабочего окна и его перемещением при нажатой кнопке мыши задаются размеры фигуры. Рисование заканчивается отпусканием кнопки. Стрелки приходится рисовать как совокупность отрезков прямой, причем каждый отрезок ломаной является отдельным графическим объектом. Имеются варианты с заполненными замкнутыми фигурами.

При щелчке мышью по фигуре вблизи ее контура она *выделяется* — появляется охватывающая рамка с прямоугольными маркерами («ручками управления»). Теперь фигуру можно масштабировать, копировать, перемещать и удалять по стандартной *Windows*-технологии. Щелкнув правой кнопкой мыши, вызываем меню `Properties`. Щелчок левой кнопкой открывает новую панель с двумя вкладками: `Outline` (Контур) и `Fill` (Заполнение). Первая из них позволяет выбрать тип линии (пять вариантов), ее толщину в пунктах и цвет (палитра с 20 элементами). Вторая вкладка позволяет выбрать сплошную заливку или один из пяти вариантов штриховки, а также цвет заполнения.

Удерживая нажатой [Shift], а также обведя указателем прямоугольный контур, можно выделить несколько фигур. Однако групповые операции перемещения и редактирования невозможны.

Кнопка [T] позволяет заносить надписи в предварительно очерченное указателем прямоугольное поле (набор кириллического текста возможен, но результат воспроизводится на схеме «по-грузински»). Выделив набранный текст и локальное меню `Properties`, получаем доступ к вкладке `Font`. С ее помощью можно выбрать шрифт (в том числе содержащий кириллические символы), размер, спецэффекты (зачеркнутый, подчеркнутый, оттененный), задать курсив и жирный шрифт (независи-

мо), определить цвет шрифта и оттенения. В окне на упомянутой вкладке все принятые решения отображаются немедленно.

Последняя кнопка (с картинкой) теоретически позволяет вставлять из папки Samples рисунки, отдельно заготовленные в системе *Visio*, однако реально добиться этого не удалось.

17.4 База данных

Создание проекта предполагает синтез сети. Компоненты выбираются в иерархической базе данных, работа с которой аналогична поиску файла посредством Проводника *Windows*. Структурированные элементы базы в списке отмечены плюсами. Щелчок мышью раскрывает этот элемент и дает дальнейшую детализацию, при этом плюс заменяется на минус. Щелчок по минусу вызывает свертку элемента базы. При щелчке по строке базы данных иконки частных типов выбранного устройства появляются в *панели изображений* в нижней части окна (на рис. ?? не показана), откуда иконка конкретного устройства может быть отбуксирована в панель проекта. Щелкнув по стрелке справа от окна со списком устройств и выбрав в нем Vendors, можно отсортировать базу данных в алфавитном порядке фирм-производителей.

Имеется возможность модификации и пополнения базы данных на базе каталога или реально существующих устройств. Можно изменять количество слотов и портов, набор шин, возможные типы связей, названия фирм-производителей и т. п.

17.5 Создание проекта

Проект запускается файлом netcrack.exe или щелчком по его иконке, предварительно вытащенной через проводник на рабочий стол *Windows*. Через меню File или щелчком по иконке стандартной панели открывается окно Top Site верхнего уровня проекта.

Проект может иметь соответствующий географический фон. Щелкнув правой кнопкой мыши на пустом месте в окне проекта, выбираем из локального меню Site Setup/Background/Map. Далее по Browse можно просмотреть папку Maps, разделами которой являются:

Canada — основные районы Канады;

Countries — алфавитный перечень государств;

Global — континенты и регионы (Индия, Средний Восток, Северная Америка, Северная Азия, Северная Атлантика);

States — штаты США;

World — карты мира в различных проекциях.

Все карты являются контурными, цвет заливки можно менять.

Выход из проекта осуществляется через меню `File/Exit`.

17.6 Синтез простейшей сети

17.6.1 Приборы

Простейший пример — две рабочие станции для локальных сетей (`Local Area Networks`) и коммутатор. Дважды щелкнув мышью по папке `LAN workstations` из базы данных, получаем их иконки в панели изображений и перетаскиваем две станции в панель проекта. Затем выбираем в базе данных папку `LAN adapters` и из нее — `Ethernet card`. Иконку последней перетаскиваем из панели изображений в панель проекта на иконку рабочей станции. В момент, когда указатель курсора меняет форму с перечеркнутого круга на плюс, отпускаем кнопку мыши (если плюс не получается, устройства считаются несовместимыми). Те же действия выполняем для второй рабочей станции. Аналогично из папки `Switches` извлекаем коммутатор.

Для выбора в панель `Devices` только совместимых с выделенным устройств следует нажать [Alt+2]. Если пользователь предпочитает оборудование определенной фирмы, он может, щелкнув по стрелке справа от окна броузера, вызвать режим `Vendors`. База данных будет отсортирована в алфавитном порядке наименований фирм-производителей.

Выделенное устройство вместе со встроенными в него можно скопировать командой `Edit/Replicate` и расположить копии по желаемой геометрической схеме. Однотипные устройства система автоматически нумерует. Отметим, что управлять размещением и выравниванием выделенной совокупности объектов можно также через меню `Object/Organize`. Копия устройства, уже использованного в проекте,

вызывается в панель изображений щелчком по вкладке `Recently Used`. Любое *выделенное* устройство можно удалить нажатием кнопки [Del].

Щелчок по изображению устройства *правой* кнопкой мыши позволяет вызвать локальное меню с командой `Properties`. С ее помощью можно получить информацию о текущих свойствах объекта, переименовать устройство или изменить его характеристики: число слотов, вид используемых шин, групп портов и типов связей для них, а также связующую среду (толстый/тонкий коаксиальный кабель, витая пара — см. шестой урок «Краткого руководства»). Заметим, что протокол маршрутизации устанавливается *глобально* — для проекта в целом и всех его составных частей.

В проект можно добавлять встраиваемые (`Plug-Ins`) приборы. Например, выберем из базы устройств по цепочке

`Routers and bridges/Backbones/Cisco Systems/Cisco 7000`

шасси `Cisco 7000` и отбуксируем его в панель проекта. Далее в это шасси можно втаскивать разрешенные к установке в нем дополнительные устройства (см. приведенное выше описание процесса вставки `Ethernet card`).

Многие устройства (концентраторы — habs, коммутаторы, маршрутизаторы) могут быть объединены (stacked). В этом случае:

- все имеющиеся порты рассматриваются как общие,
- трафик считается проходящим через стек в целом,
- перемасштабирование изображения выполняется для стека в целом.

Здесь каждый последующий прибор втаскивается в первый по обсуждавшейся технологии. Результат для большей наглядности надо промасшабировать (укрупнить). Характеристики встроенного или установленного в слоте прибора запрашиваются щелчком по соответствующему фрагменту укрупненного изображения. Стек в целом или его часть, выделяемая на укрупненном изображении прибора, удаляются нажатием [Del].

17.6.2 Функции

Включенным в проект устройствам можно придать некоторые специальные функции. Сделаем, например, одну из рабочих станций сервером. Для этого выполним следующие действия:

- В броузере устройств найдем и раскроем «математический» пункт `Network and enterprise software`. Щелкнем по пункту `Server software`. Доступные варианты последнего появятся в окне изображений.
- Отбуксируем изображение компакт-диска с соответствующей надписью (например, `E-mail server`) к выбранной рабочей станции.
- Аналогичные действия выполним для станции-клиента.

17.6.3 Связи

Для связывания устройств в сеть необходимо:

- В панели инструментов `Modes` щелкнуть по кнопке связывания (с изображением молнии).
- Последовательно щелкнуть указателем по соединяемым устройствам. Откроется окно диалога `Link Assistant`.
- Согласиться с назначением портов по умолчанию, нажать кнопку `Link` и закрыть диалог.
- Выполнить *быстрое связывание* (это альтернативная технология), щелкнув при нажатой [Shift] по коммутатору, а затем по второй станции. Отпустить [Shift].

Тип соединений между устройствами расшифровывается выбором команд `View/ Legends`. Последняя имеет две вкладки: *тип среды* (коаксиальный кабель, витая пара, оптоволоконная линия, многожильный кабель, радиосвязь) и *род пакета* (`ATM Call`, `LAN Packet`, `WAN Packet`, `Physical Layer Data`, `Dial Up Session`).

Теперь организуем трафик между рабочими станциями. Здесь нужно щелкнуть по кнопке [Traffic] панели `Modes`, а затем последовательно по отправителю и адресату. Откроется окно диалога `Profiles`. Из его списка (`CAD/CAM`, `Database`, `Database server`, `E-mail`, `E-mail server`, `FTP client`, `FTP server`, `Small Office`, `Small Office Server` и т. д.) выберем подходящий (в соответствии с установленным математическим обеспечением) тип и щелкнем по [Assign]. В боксе `Color selection` можно выбрать цвет, которым в процессе анимации будет представляться каждый из выбранных видов трафика.

На вкладке Traffic можно задать вид распределения объема пакета (постоянное, равномерное, показательное, нормальное, логарифмически нормальное, гамма-, Эрланга, Вейбулла) и соответствующие единицы измерения (биты, байты, килобайты и т. п.). При выборе закона распределения открываются окна для задания его параметров (каких именно, выяснить не удалось). Аналогично и из того же ряда задаются распределения интервалов между смежными транзакциями.

17.7 Многоуровневый проект

Примером многоуровневого проекта является файл hier.net. Загрузив его в главное окно и щелкнув по кнопке Project Hierarchy, увидим в броузере иерархическую структуру того же типа, что при просмотре базы данных. Щелчком по нужному элементу вызываем его схему в рабочую панель.

Разработку иерархического проекта начинают с задания верхнего уровня (раздел базы данных Buildings, campuses and LAN workgroups). Далее выберем один из упомянутых объектов (например, здание) в панели изображений и отбуксируем его в окно уровня Top. Выделим здание, щелкнем по нему правой кнопкой мыши и в открывшемся локальном меню выберем команду Expand. Теперь мы имеем двухуровневый проект, включающий в себя уровень Top и второй уровень с собственным окном Building. Изображение здания в окне Top заключается в красную рамку, указывающую на «контейнерный» характер этого объекта.

Далее по описанной в разделе о простейшем проекте технологии создадим в здании локальную сеть и подключим ее к концентратору Ethernet. При достаточном увеличении в углу окна вложенного уровня можно рассмотреть соединитель (Connector). Щелкнув по окну Top, вновь сделаем его активным, выберем очередной элемент верхнего уровня сети и приступим к его детальной проработке.

Вновь выйдя на верхний уровень проекта, щелкнем по кнопке Link devices в панели режимов Modes и далее — последовательно по соединяемым элементам уровня Top. Между ними появится *пунктирная* линия, что указывает на незавершенность соединения. Последнее оформляется окончательно на *вложенном* уровне связыванием через Link Assistant концентратора и соединителя.

17.8 Анимация и имитация

17.8.1 Основные понятия

Смысл термина «имитация» достаточно ясен из предшествующего содержания данной книги. Под «анимацией» в программотехнике понимается создание движущихся изображений с целью наглядного представления отдельных процессов и их взаимодействия. Анимация должна углублять понимание этих процессов и подводить к практическим выводам. Анимация в пакете *NetCracker* является (по крайней мере пока) преимущественно декоративным элементом: она показывает лишь вид и маршруты движения сообщений, и ее дидактическая ценность минимальна.

17.8.2 Анимация

Хотя процессом анимации можно управлять, «any changes apply to the on-screen appearance of the packets and do not affect any network or data flow parameters» (любые изменения применяются к изображению на экране и не влияют ни на какие параметры сети или потока данных — цитата из Руководства). В частности, они не влияют на замеряемые статистические показатели.

Для анимации интенсивность (частота генерации) пакетов, скорость их движения и графический размер устанавливаются в диалоге `Animation Setup` порознь на вкладках `Packets/Calls` перемещением соответствующих движков. Цвет единицы данных указывает вид трафика, а форма ее определяется транспортным протоколом. Упомянутые характеристики могут быть изменены пользователем. Движение сообщений в общем случае будет двусторонним.

Дважды щелкнув по «контейнерной» части проекта и задав достаточное увеличение, можно рассмотреть анимацию на более низком иерархическом уровне.

В процессе анимации можно войти в режим паузы (кнопка с двумя вертикальными прямоугольниками). Щелчок правой кнопкой по замершему пакету вызовет локальное меню, через которое можно просмотреть или услышать (режим `Say`) информацию о свойствах пакета (характер трафика, размер пакета, источник, адресат, сетевой и несущий протоколы передачи). Звуковая информация синтезируется

по-английски и сообщается пользователю при наличии в компьютере звуковой карты.

Можно установить *режим трассировки*. Включив его кнопкой панели Mode, щелчками по двум элементам сети выделяем (красным цветом) путь между ними. Для обозначения выхода в другой фрагмент сети (см. раздел об иерархических проектах) используются специальные маленькие иконки.

Анимация наглядно представляет перегрузку сети, возникающую при задании соответствующего трафика. Имеется возможность отображения выхода из строя (Break) устройств и каналов связи с имитацией последствий. Выберем на панели Modes иконку с «инженерным» символом (скрещенные молоток и гаечный ключ). В поле проекта курсор превращается в молоток. Подводим его к отказавшему элементу. На последнем начинает пульсировать красная вспышка, а потоки сообщений перераспределяются, обходя поврежденный элемент. При повторном щелчке по нему курсор обращается в гаечный ключ, и работоспособность элемента восстанавливается.

17.8.3 Имитация

Применительно к имитации предполагается, что:

- скорость передачи данных зависит от полосы пропускания линии связи;
- скорость поступления данных определяется интенсивностью вызовов и объемом пакетов.

Нажав в панели Animation Setup кнопку Advanced, можно войти в режим количественного определения соответствующих характеристик. Затем нужно нажать Device Setup и войти в диалог Properties. Структура последнего зависит от типа прибора. Получив необходимые сведения, нажимаем клавишу Cancel. Подобным же образом можно просмотреть дополнительно установленные (например, в маршрутизатор) приборы.

Для первоисточников сообщений, по-видимому, можно задать требуемые характеристики произвольно и в достаточно широком диапазоне значений. Темп имитации может регулироваться и, в частности, соответствовать реальному масштабу времени (если быстродействие компьютера это позволяет).

Двойной щелчок по связи открывает диалог `Link Assistant`, из которого можно получить информацию о характере связи.

17.8.4 Оценка работы сети

Щелчок *правой* кнопкой по связи или устройству открывает окно команды `Статистика`. Через него можно затребовать информацию об использовании (коэффициенте загрузки), о средней и текущей нагрузке (производительности в натуральных единицах), числе принятых и отклонененных пакетов (по мостам, коммутаторам и маршрутизаторам), задержке пакетов, длительности телефонных разговоров. Перечень характеристик зависит от вида устройства или связи. По заявленным показателям можно запросить: информацию в числовой форме, в виде столбика текущего уровня с указанием числовой меры, график как функцию времени, а также аудио-информацию (только при наличии звуковой карты и лишь по одному из показателей).

Столбик можно растягивать и сжимать по горизонтали и вертикали. Щелчок правой кнопкой по текстовому боксу позволяет менять шрифт. Для графика можно менять единицы измерения, характер разметки и оцифровку — десятичная (с фиксированной точкой), «научная» или «инженерная» (с порядком, кратным трем).

По потокам данные группируются в три раздела. *Статистика приложений* содержит данные о времени доставки пакетов по каждому маршруту. *Статистика передачи данных по телефонным линиям* включает число сделанных вызовов, количество принятых и продолжительность сеанса для источников запросов. Аналогичные данные собираются для телефонных переговоров (`Voice Calls`).

17.9 Режим «Discovery»

NetCracker имеет совершенно уникальную возможность распознавать сетевое окружение пользователя и тем дает ему в руки средство, отчасти оправдывающее название обсуждаемой системы («взломщик сетей»). Собранная информация представляется затем в виде проекта и может исследоваться стандартными средствами *NetCracker*'а.

Процесс авторaспознавания реализуется в четыре этапа:

- установка диапазона IP-адресов сканируемых хостов;

- собственно сканирование (длительность этапа зависит от диапазона);
- редактирование типов распознанных в сетевом окружении устройств для их последующего включения в проект;
- оформление описания сетевого окружения в виде проекта.

Для запуска процесса войдем в меню File и выберем команду Discover. Откроется диалоговое окно Network Discovery. После выбора опции NetCracker Discovery Engine щелкнем по кнопке [Далее] и перейдем в окно NetCracker SNMP Discovery Settings. В нем зададим диапазон IP-адресов сканируемых хостов с помощью спецификации начального и конечного IP-адресов или маски подсети, установим промежутки времени между отправкой тестовых пакетов для каждого из протоколов (для ICMP задается в поле Ping). После нажатия кнопки [Далее] открывается окно готовности к сканированию. Процесс запускается нажатием кнопки [Start]. В окне SNMP Discovery Console высвечивается его протокол. В нем отражаются информация о распознанных устройствах, их IP-адресах и DNS-именах. Отмечается сканирование таблиц интерфейсов и адресов и просмотра кэш-памяти, содержащей таблицы ARP.

Результирующая информация выдается в виде таблицы с указанием общего числа просмотренных устройств и связей между ними. В окне Network Discovery Results ее можно просмотреть более подробно и отредактировать с использованием входящей в *NetCracker* базы данных об устройствах. Щелкнув по кнопке [Close], запускаем этап создания проекта сетевого окружения. Готовая схема высвечивается в главном окне системы.

Описанная система обладает богатейшими возможностями, достаточно удобна в работе и предоставляет пользователю обширную базу реальных данных. Отметим, однако, существенное в контексте этой книги обстоятельство: четкая проблемная ориентация *NetCracker*, радуя разработчиков сетей связи и вычислительных сетей, одновременно делает ее бесполезной в других предметных областях. По-видимому, следует в ближайшем будущем ожидать появления систем подобного рода:

- с иными специализациями (например, для разработки архитектуры вычислительных машин, технологических процессов, структуры больниц, аварийных служб и т. п.),

- интегрированных (сочетание с сетями связи сетей, специализированных в различных направлениях),
- относительно универсальных (предоставляющих пользователям технологический инструментарий разработки специализированных и интегрированных систем).

В заключение укажем, что за любой сервис всегда приходится платить ограничением творческих возможностей.

Глава 18

Конференция ИММОД-2003

Хорошим завершением данной книги будет аналитический обзор материалов состоявшейся в ЦНИИ технологии судостроения (Санкт-Петербург) 23–24 октября 2003 г. 1-й Всероссийской научно-практической конференции «Опыт практического применения языков и программных систем имитационного моделирования в промышленности и прикладных разработках»[41]. Обзор демонстрирует размах, многообразие и практическую ценность имитационного моделирования. Он дается с разных точек зрения, интересных для различных категорий пользователей: по методам моделирования, инструментальным средствам и прикладным задачам. Для экономии места в тексте и в списке литературы ссылки на доклады сведены к минимуму и даются по фамилии первого автора.

18.1 История имитационного моделирования в России и за рубежом

На конференции отмечались давние и хорошие традиции использования и развития моделирования в СССР в период 1960–1980 гг.: созданы научные школы (прежде всего Н.П. Бусленко); получен ряд практических результатов: языки программирования *СЛЭНГ, НЕДИС, СТАМ* и др., система агрегативного моделирования. Семейство *GPSS* и другие языки моделирования были адаптированы к применявшейся в стране вычислительной технике. Все они широко применялись в реальном секторе

экономики.

Развал страны и «демократические» реформы привели к утрате связей между научными коллективами и отдельными учеными, прекращению активной деятельности многими из них, нарушению преемственности поколений, приостановке или прекращению ряда перспективных разработок. Умирающая промышленность утратила интерес к практическому моделированию, что иссушило внебюджетные ручейки финансирования исследований.

18.2 Современное состояние имитационного моделирования

Мировая наука и экономика в трудные для России последние десятилетия не стояли на месте и интенсивно развивались. За рубежом регулярно проводились конференции по теории и практическим аспектам имитационного моделирования. Оно все шире внедрялось в практику проектирования производственных (в самом широком смысле слова) процессов и оперативного управления ими. В повестку дня встал вопрос о сплошном применении цифровых моделей (`Digital Factory`) в процессе проектирования и эксплуатации производственных систем. Люди, участвующие в такой деятельности, получают возможность наблюдать объекты в виде трехмерных изображений (виртуальная реальность — `VR`). Наличие имитационной модели и обоснование с ее помощью выбранного варианта являются обязательными в комплекте документов проектирования или модернизации нового производства либо технологического процесса. Модели используются и для обучения персонала. Эта концепция называется `e-Manufacturing`. Убежденными сторонниками ее являются, в частности, ведущие автомобильные компании: Daimler–Chrysler, Mercedes–Benz, BMW, Audi, Toyota. Тот же подход применяется и на сборке аэробусов A-380 в Гамбурге.

Целый ряд фирм выпускает программные продукты как широкого назначения (модифицируемые), так и специализированные по логистике, отраслям промышленности и социальной сферы. Однако в условиях чрезвычайной сложности и дороговизны этих разработок из европейских компаний только Technomatics и DELMIA претендуют на полное покрытие `e-Manufacturing` своими продуктами. Типичными недостатками систем моделирования в условиях России являются:

- сложность описания;
- сложность проведения экспериментов;
- слабость средств моделирования конфликтов за общие ресурсы;
- отсутствие поддержки русского языка.

Имитационное моделирование часто комбинируется с аналитическими моделями. Поскольку наиболее широкий класс моделей, охватывающий дискретное сборочное производство, транспортные системы, системы логистики, разного вида обслуживающие и коммуникационные системы — это сети массового обслуживания, активно разрабатываются и такие модели и алгоритмы.

18.3 Имитационный ренессанс

Наметившийся рост экономики взбодрил оставшихся (главным образом в вузах) энтузиастов моделирования, оживил их интерес к преподаванию моделирования, теоретическим исследованиям и прикладным разработкам. В настоящее время 150 вузов России ежегодно выпускают более 10 тыс. специалистов, владеющих основами компьютерного моделирования (В.В. Девятков). Аналогично обстоит дело и в странах СНГ. Моделирование введено в учебные планы не только компьютерных, но и ряда экономических специальностей. В сети Интернет появился целый ряд сайтов, посвященных этой проблеме (`gpss.ru`, `simulation.org.ua`, `gpss-forum.narod.ru`). За не истекший еще 2003 г. уже зарегистрировано 25729 посещений первого из них. Замечено, что пики повышения и минимумы частоты запросов точно совпадают со сроками семестров и каникул. Сообщается о разработке исследовательского портала «Имитационное моделирование». Выходят из печати учебники по имитационному моделированию.

18.4 Общая оценка и результаты конференции

Конференция в целом была очень четко организована, прошла на хорошем уровне и при большой заинтересованности участников. Труды кон-

ференции были изданы до ее начала. Было сделано 5 докладов на пленарном заседании и по 31 — на секциях (теоретической и «прикладной»). Кроме того, без выступлений было представлено еще 13 секционных докладов. Практически все доклады были оформлены в виде компьютерных презентаций; многие секционные и стендовые доклады (последних было запланировано 29) сопровождались эффектными и поучительными анимациями.

Основная цель каждой конференции — установление единства взглядов, личных и производственных контактов — была достигнута и в данном случае. На итоговом заседании было принято решение впредь проводить конференции по имитационному моделированию регулярно и изучить вопрос о придании им статуса международных (впрочем, и на этой конференции были представлены докладчики из Латвии, Германии и США, не говоря о странах СНГ).

Теперь перейдем к обзору содержания докладов по основным группам проблем.

18.5 Методология моделирования

18.5.1 Общие принципы

Участники конференции дружно подтвердили свое согласие с известными принципами моделирования сложных систем:

- системный подход при решении задач анализа и синтеза;
- иерархическое многоуровневое моделирование;
- множественность моделей по функциональному назначению;
- множественность моделей по их математической структуре и их комбинирование: аналитико-имитационные, логико-лингвистические, логико-алгебраические.

Иерархия аналитических моделей должна давать упрощенное (агрегированное) описание различных сторон моделируемых явлений. В комплекс моделирования должны входить средства как универсальные (с возможностями настройки), так и специализированные по предметным областям. Для моделирования, помимо совокупности моделей, необходимы:

- информационная подсистема, включающая базу (банк) данных, а в перспективе — базу знаний, основанную на идеях искусственного интеллекта;

- система управления и сопряжения, обеспечивающая взаимодействие всех компонент системы и работу с пользователем в режиме диалога;

- средства визуализации хода и результатов моделирования.

По мнению участников конференции, технологический уровень современной системы моделирования определяется следующими ее свойствами:

- Универсальностью и гибкостью базовой и альтернативных концепций структуризации и формализации моделируемых динамических процессов, заложенных в систему моделирования. Необходимы обмен информацией о параметрах транзактов и между транзактами и определение времени реализации критических событий для непрерывных компонент процесса.

- Наличием средств проблемной ориентации.

- Применением объектно-ориентированных специализированных языков программирования, поддерживающих авторское моделирование и процедуры управления процессом моделирования.

- Наличием удобного и легко интерпретируемого графического интерфейса в сочетании с заданием параметров моделей через подменю.

- Использованием развитой двух- и трехмерной анимации в модельном времени.

- Возможностью реализации нескольких иерархических уровней представления модели и стратифицированных описаний.

- Наличием процедур: анализа входных данных; чувствительности; планирования и организации статистического эксперимента; анализа поверхности отклика; оптимизации и др.

18.5.2 Визуализация моделирования

Визуализация процесса моделирования позволяет:

- проверить адекватность модели реальной ситуации;
- обеспечить контроль пользователя над внутренним механизмом функционирования;
- повысить интерес заказчиков и облегчить внедрение и применение системы.

К примеру, в докладе М.В. Литвинова были описаны расчет процесса пуска боекомплекта системы залпового огня «Смерч» и визуализация поведения пусковой установки.

Ряд докладчиков отмечал как недостаток $GPSS/W$ ее неспособность к «фотореалистичной» анимации. Однако здесь возможны постпроцессорная анимация, основанная на трассировочных данных, и оперативная анимация посредством PLUS-процедур. Оба механизма требуют использования анимационных пакетов сторонних разработчиков или авторских разработок пользователя. Считается, что 3D-анимация не намного сложнее двумерной.

На рынке существует много анимационных пакетов. Часть из них уже интегрирована в системы моделирования (пример — $eM\text{-}Plant$ с 3D-анимацией). Из инструментов визуализации считается предпочтительным $VRML$:

- Язык является открытым, и для начального процесса обучения не нужно покупать дорогие пакеты по созданию $VRML$-анимации (достаточен стандартный Интернет-браузер).
- $VRML$ через $Java\ Script$ допускает сопряжение с другими программными средствами.
- Различные узлы $VRML$ (LOD, SENSOR, INTERPOLATOR), а также методы прототипирования (PROTO) позволяют получать объекты различной степени детализации и повторно использовать объекты (метод порождения), что ускоряет разработку анимации.
- $VRML$ широко применяется в автоматизированном проектировании и, в частности, играет ведущую роль в концепции Virtual Factory.

Можно рекомендовать и комплексную технологию: сложные объекты (вагоны метрополитена, грузовики, самолеты) создаются в пакетах 3D-графики (*3D Max, True Space, LightWave, SPAZZ*_3D), после чего идет редактирование самой модели *VRML* с включением в нее ранее созданных объектов. Компоновать объекты в единые сцены можно с помощью пакетов, подобных *ISA* (*Internet Scene Assembler*).

18.5.3 Распределенные объекты и распределенное моделирование

Объекты имитационного моделирования почти всегда пространственно распределены. В связи с необходимостью придания им достаточной автономности элементы распределенных интеллектуальных информационных систем должны подвергаться обучению (самообучению). Соответственно для их работы (и моделирования таковой) необходимы подсистемы эволюционного развития (адаптивная, с функциями идентификации) и оперативная (для использования в контуре управления).

Новые проблемы возникают при переходе *мультиагентным* задачам. Под интеллектуальным агентом понимается объект, обладающий внутренней структурой, собственным поведением и возможностью взаимодействия с другими агентами. Каждый агент имеет неполную информацию и/или недостаточные возможности для выполнения общей задачи и в отсутствие централизованного управления должен выполнить ее в кооперации с другими агентами. К этим проблемам непосредственно примыкает описание сложного поведения объектов. В частности, в докладе О.Ю. Яковенко были разработаны модели поведения:

- одиночного бойца в специальной операции;
- танка в бою;
- отдельного человека в местах скопления больших масс людей.

С другой стороны, представляет значительный интерес *распределенное моделирование*. Система *SIMNET* использовалась для обучения армейского персонала действиям в чрезвычайных ситуациях. Распределенное моделирование необходимо и при недостаточных вычислительных ресурсов для особо сложных задач (расчет гидродинамических полей и метеоявлений).

18.5.4 Методы моделирования

Здесь мы перечислим редко используемые «продвинутые» методы, которые нашли применение в представленных докладах.

Методы статистики. Однородность данных по дням недели проверялась непараметрическим критерием Вилкоксона, а гипотез о виде распределений — по Колмогорову — Смирнову.

Теория сетей обслуживания. Применялся аналитический расчет сетей обслуживания на основе их потокоэквивалентной декомпозиции.

Методы понижения дисперсии моделирования. Отмечалась обязательность раздельных ДСЧ для каждой случайной величины. В целях понижения дисперсии применялся метод «антитез» (дополняющих переменных), причем для получения таковых в командах `RMULT` $GPSS$-программы использовались дополняющие начальные установки. Наконец, для вычисления вероятностей редких событий использовалось условное моделирование. Все эти технологии не новы, но, к сожалению, неизвестны практикам и совершенно не применяются ими. Хочется надеяться, что «лед когда-нибудь тронется».

Сети Петри по существу являются одной из форм имитации дискретных процессов. Они были в большой моде лет 20 назад, когда с их помощью надеялись *рассчитывать* упомянутые процессы без имитации. В подавляющем большинстве ситуаций от обычных имитационных моделей они отличаются лишь наукообразием и специфической терминологией.

На конференции был предложен метод синтеза сетей Петри на базе функциональных подсетей, реализованный в программном продукте $FMSim$. Имитатор сетей Петри использовался для анализа процесса очистных работ в длинном забое и роботизированных технологий горных работ.

Тензорные методы. К анализу сетей интегрального обслуживания применялся тензорный метод в сопоставлении с $GPSS$. Однако вопрос о том, что *реально* дает применение столь сложной математики, остается открытым.

Генетические алгоритмы оптимизации. Известна сложность оптимизации автоматических систем регулирования. Генетический алгоритм оптимизации формулируется следующим образом:

1. В пространстве поиска случайным образом задается популяция воз-

можных решений (особей).

2. Вычисляются значения вектора функции цели для всех особей.
3. Из популяции отбираются и удаляются, например, 10% худших по значению целевой функции особей. Оставшиеся образуют родительскую группу.
4. Из родительской группы случайным образом выбираются пары особей в количестве тех же 10% от общего числа.
5. Новое поколение особей генерируется в результате обмена участками хромосом родителей (кроссинговер), а также мутаций.
6. Это поколение восполняет популяцию до исходной численности.

Далее этапы начиная со второго повторяются, пока во всех координатных точках значения оптимизируемой функции не станут отличаться друг от друга меньше, чем на заданное малое число. Предложена также комбинация генетического алгоритма с градиентным поиском локальных экстремумов и заменой ими «плохих» точек.

Гибридный генетический алгоритм был применен для автоматизированного проектирования оптимальных (с минимальной длиной) траекторий термической резки металла. При этом в вышеупомянутых генетических операциях были более четко прописаны вероятностные и имитационные элементы. При *скрещивании* один из родителей выбирается с наилучшим показателем целевой функции, тогда как другой — случайным образом. В хромосоме, подвергающейся *мутации*, два случайно выбранных гена меняются местами. Оператор *разнообразия* улучшает особь методом спуска. Оператор *селекции* уничтожает большую часть популяции, заменяя ее лучшим материалом. Стратегия селекции может быть *элитной*: некоторое количество лучших особей переходит в следующее поколение без скрещивания и мутаций.

18.5.5 Достоверность модели

На конференции большое внимание было уделено проблеме обеспечения достоверности имитационных моделей. В докладе В. А. Пепеляева воспроизводится схема О. Балчи упорядоченности отдельных фаз разработки и реализации имитационных моделей и процедур оценки достоверности этих процессов.

Далее определяются важнейшие понятия, связанные с достоверностью моделирования. *Верификация* модели есть подтверждение ее преобразования из одной формы в другую (от формулировки проблемы до получения исполняемого компьютерного приложения) в соответствии с намерениями разработчика и с достаточной точностью. По существу речь идет о корректности многоэтапной трансляции. *Валидация* подтверждает поведение модели в области применений в соответствии с целями и с удовлетворительной точностью.

Концептуальная модель на основе предположений относительно абстракции от реальности определяет иерархическую структуру исследуемой модели и связи между отдельными ее компонентами. *Квалификация* концептуальной модели подтверждает принятые соглашения для выбранной области применения. Имитационная модель рассматривается как образ концептуальной модели на одном из высокоуровневых объектно-ориентированных языков программирования (моделирования).

Верификация модели взаимодействий подтверждает корректность отображения взаимодействий объектов и динамики исследуемой системы в целом.

Валидация данных направлена на доказательство удовлетворительной точности входных данных и корректности их использования.

Тестирование модели есть планируемый интерактивный процесс, направленный главным образом на поддержку процедур верификации и валидации имитационной модели, в том числе на поиск ошибок в программах.

Четкость вышеприведенных определений и даже их правильность оставляют желать лучшего (к примеру, ценность данных нельзя ставить в зависимость от корректности их использования), а названные понятия явно не ортогональны (имеют пересечения).

В докладе В. А. Пепеляева приводится ряд методических рекомендаций по отдельным этапам обеспечения достоверности.

В близком по содержанию докладе И. В. Яцкив используются некоторые дополнительные понятия. В частности, если модель и ее результаты приняты пользователем и реально используются для принятия решений, то модель считается *заслуживающей доверия* (credible). Далее обсуждаются процедуры проверки адекватности: концептуальные, операционные, основанные на выходных данных. *Концептуальные тесты* включают проверку постановки задачи моделирования, входных

данных, структурных допущений, логическую валидацию. *Операционные тесты* анализируют адекватность поведения модели; включают тесты на непрерывность, анализ чувствительности, анализ вырождения, анализ анимации. *Анализ риска* состоит в определении вероятностей редких, но крайне нежелательных событий (ядерный инцидент, экологическая катастрофа, авария энергосистемы, финансовый крах).

Как особая категория тестов рассматриваются основанные на *выходах*: модель при реальных входных данных должна порождать близкие к реальным выходным (предполагается, что исследователь располагает последними). Возможный подход к оценке адекватности модели — тест на статистическую однородность совокупности реальных и модельных выходных данных.

18.6 Языки и системы моделирования

18.6.1 Обзор новинок

Используя возможности визуального моделирования и современные технологии дизайна и анимации, можно существенно ускорить исследования. За рубежом появилось огромное количество систем имитационного моделирования. Коммерческие симуляторы специализированы по отраслям промышленности: *eMPlant* (машиностроение), *DELMIA* (судостроение), *NETRAC* (телекоммуникации и связь).

Решение задач с преобладанием логистических аспектов может быть получено с помощью любых коммерческих симуляторов; чаще всего это *GPSS, Simula, Arena, AutoMod, eM-Plant, Extend, Process Model, QUEST, SIMFACTORY, Taylor ED, WITNESS*. Сколько-нибудь систематизированные сведения по большинству названных пакетов отсутствуют. Сообщается, однако,что *Arena, Extend, ProModel, SimProcess* используют ту же транзакт-парадигму, что и *GPSS*.

В докладе К. А. Аксенова дается характеристика системы *BPsim*, опирающейся на аппарат динамических экспертных систем и в значительной степени свободной от типичных недостатков. В ней определены новые классы объектов: операции, ресурсы, средства, процессы, источники и приемники ресурсов, перекрестки, параметры. Отдельно выделены *информационные* типы ресурсов: сообщения и заявки на выполнение операций. Параметры процесса задаются функцией от характеристик

объектов и делятся на производные (свертка различного типа характеристик) и консолидированные (свертка одноименных характеристик операций процесса). Описание причинно-следственных связей задается специальными объектами.

Другой новый профессиональный инструмент *AnyLogic* объединяет объектно-ориентированный подход, визуальное проектирование, дружественный графический интерфейс, язык *Java*, агентные технологии, теорию гибридных систем. Визуализация динамики процессов и статистическая обработка случайных параметров являются встроенными и выполняются автоматически.

18.6.2 GPSS

GPSS и ее производные по-прежнему остаются самым популярным языком моделирования в преподавании и практическом использовании. Версии *GPSS/PC* были посвящены доклады В. В. Афонина и С. А. Власова. А. Г. Варжапетян использовал язык *GPSS/H* в процедуре бенчмаркинга, Д. В. Турчановский применял *GPSS/H-Proof* со средствами анимации и поддержкой стратифицированных описаний. В докладе Р. Г. Гиниятуллина обсуждалась работа на *GPSS World* с широким использованием средств языка PLUS.

18.6.3 C, C++

Ряд докладчиков отмечал популярность C++, основными достоинствами которого являются объектная ориентация и использование его как инструмента для написания ядра операционных и моделирующих систем. Это облегчает интеграцию с последними вновь создаваемых (как правило, в форме динамических библиотек функций) расширений и надстроек. Таким образом были реализованы модель решения финансовых задач, АСУ технологическим процессом в стиле *Simula*, анализ работы аэропорта Шереметево.

В докладе В. Н. Томашевского пропагандировался язык *SLX*, объявленный как объектно-ориентированный *GPSS*. Новым здесь является написанный на C++ лингвистический процессор, который по интерактивно формируемому заданию строит *GPSS*-программу. В другом своем докладе В. Н. Томашевский отмечает, что современные программные

средства позволяют автоматизировать построение модели благодаря использованию типовых компонент и графического интерфейса. Однако средства имитации, в которых используется *фиксированное* число моделирующих конструкций и нет возможности ввести новые, обязательно окажутся неподходящими для некоторых практических ситуаций. Далее он заявляет, что «данную проблему можно решить путем использования генератора программ, которые создаются на некотором языке моделирования. В этом случае квалифицированный пользователь сможет изменить код программы. В *ISS* 2000 в качестве языка моделирования используется *GPSS*». Нетрудно видеть, что возможности такой системы лимитируются языком нижнего уровня, и *SLX* сохранит все ограничения *GPSS*.

Недостатками C++ считаются неудобство программирования и трудность сопровождения программ. Корень этого и других недостатков данного языка — лежащая в его основе статическая типизация. Те же недостатки присущи языкам *Java, Object Pascal, Delphi*. Поэтому альтернативой может явиться *SmallTalk* — объектно-ориентированный язык с динамической типизацией.

Все C-подобные языки суть языки системного программирования и уже поэтому неудобны для разработки прикладных программ.

18.6.4 Паскаль и Delphi

Система *SIMPAS* разработана в МГТУ им. Баумана на языке *Pascal*. Она отличается простотой и компактностью записи моделей. Помимо паскалевских, в ней предусмотрены специфические типы скалярных и множественных данных. К типу «объекты» отнесены транзакты, приборы, очереди, накопители и т. п. Для действий над ними используются специальные процедуры. Форма записи моделей напоминает *GPSS*.

Events Modeling Language (EML) построен как отдельный программный модуль в *Borland Delphi*. Здесь имеется примерно такой же перечень видов объектов. Объекты организованы в динамические списки. На *EML* была разработана имитационная модель единичного и мелкосерийного производства. Аналогично построена и используется система *SMPL*.

18.6.5 Фортран

Ряд докладчиков предпочитал моделирование на «универсальных» языках программирования численных задач. О работе на *Фортране* докладывал В.В. Миняев. На *Фортране* же Ю. И. Рыжиков проводил сравнительное тестирование *GPSS* и ставил имитационные эксперименты при разработке аналитического метода расчета многоканальных приоритетных систем.

18.7 Моделирование в образовании

Преподавание моделирования было одним из лейтмотивов конференции, но его предмет трактовался по-разному. Выступающие часто не видели разницы между компьютерным моделированием вообще, статистическим и имитационным.

Все указанные виды математического моделирования и их комбинации могут быть полезны по-разному (и в разных ситуациях). Соответственно они имеют право на включение в учебные программы — в пропорциях, определяемых *конкретными специальностями*. На моделирование (и не только на него) не следует вводить жесткие стандарты: полезно вспомнить часто цитируемого в недавнем прошлом классика о том, что успех дела определяется составом лекторов. Однако общие положения о моделировании и специфику его видов (включая дополнительные признаки: непрерывное/дискретное, линейное/нелинейное, стохастическое/детерминированное и т. п.) необходимо предусмотреть обязательно.

В. К. Кумунжиев (Ульяновский университет) в связи с подготовкой специалистов на факультете информационных и телекоммуникационных технологий заявил, что «многие вузы ограничиваются использованием одного языка моделирования, чаще всего типа *GPSS*. Мы сочли такую позицию далеко не лучшей. Освоение общих принципов и технологии должно быть построено на использовании нескольких языков, ориентированных на различные предметные области». Конкретно им предлагались системы моделирования *IMITA* — потоковые и сигнальные схемы, *ДИСПАС* — системы управления, *GPSS World* — системы массового обслуживания. Ясно, однако, что знание языков и технологий не заменяет усвоения ключевых понятий, то есть элементов *фундаментального* образования. Впрочем, и сам К. В. Кумунжиев пишет, что «знания

человека должны быть системой. Это способствует сохранности знаний и их актуализации».

К этому обзору можно добавить предложения других докладчиков о включении нужных разделов моделирования в более специальные дисциплины (внедрение $GPSS/W$ в курс теории телетрафика для изучения характеристик потоков вызовов с такими распределениями, как Вейбулла, Пирсона, логнормальное, а также моделирования неполнодоступных систем). Заметим, что перечисленные задачи о входящем потоке легко решаются численными методами теории очередей.

На заключительном заседании конференции поднимался выходящий за рамки компетенции собравшихся вопрос о расширении преподавания компьютерного моделирования для экономистов и о его введении для будущих медиков. Дело хорошее, но требующее солидной базы по высшей математике и теории вероятностей, без которых преподавание компьютерного моделирования будет бесполезно.

18.8 Применение моделирования

В этом разделе будут перечислены основные области применения разработанных докладчиками имитационных моделей.

Наиболее масштабным проблемам были посвящены доклады об имитационном моделировании социально-экономического развития регионов и о системе прогнозирования такового.

18.8.1 Промышленность

Типичные модели производства обычно включают детальные модели, производства, позволяющие проводить анализ узких мест, комплексное управление логистическими процесами, выбор эффективных стратегий управления запасами, анализ финансово-экономического состояния предприятия, оперативное и календарное планирование. Большинство ответственных решений принимается в результате диалога (с помощью интерактивных процедур, запускающих имитацию реальных процессов для различных альтернатив).

Отметим новые и уже ставшие общими тенденции в применении имитационных моделей. Помимо исследовательских и проектных работ, они используются:

- для комплексной отладки подсистем АСУ ТП;
- для обучения и аттестации управляющего персонала;
- в оперативном контуре управления АСУ ТП для предсказания нежелательного поведения реальной системы, выдачи предупреждений и рекомендаций персоналу.

Центральным элементом модели производства изделия (партии изделий) является сетевой график, который строится по детализированной технологической схеме изделия. Одновременно реализуется множество графиков. Для получения календарного плана мелкосерийного производства применялось имитационное моделирование с использованием системы правил предпочтения. При этом учитывались:

- взаимозаменяемость ресурсов;
- совместное использование ресурсов;
- режим использования ресурса (с участием человека и без него);
- количество одновременно обрабатываемых деталей;
- затраты времени на переналадку оборудования.

Имитационные модели использовались для следующих объектов металлургического производства:

- кислородно-конвертерные цеха с отделениями разливки в слитки и непрерывной разливки;
- электросталеплавильные цеха с непрерывной разливкой;
- миксерные отделения доменных цехов — конвертерные цеха;
- технологические комплексы «конвертеры (электропечи) — агрегаты внепечной обработки стали — разливочные агрегаты — склады — нагревательные агрегаты»;
- технологические комплексы «печи — стан» различных видов;
- технологические комплексы заготовочных и сортовых станов со складским, транспортным и др. оборудованием.

Два доклада были посвящены технологии *судостроения*. А. М. Плотников моделировал судокорпусное производство на Адмиралтейских верфях (Санкт-Петербург) и на Рыбинском судостроительном заводе. Б. К. Елтышев применял *GPSS/W* для проектирования циклограмм автоматизированных гальванических линий на «Адмиралтейских верфях». Здесь модель была включена в систему оперативного управления, в связи с чем потребовался предельно упрощенный интерфейс. Исходные данные вводились через экранные формы *Microsoft Access*, после чего на внутреннем уровне корректировались операнды соответствующих блоков *GPSS* в тексте модели-прототипа.

Л. А. Осипов докладывал о модели производства *санитарно-керамических изделий* (унитазы, сливные бачки, писсуары, умывальники, пьедесталы, урны).

В докладе О. А. Савиной была построена модель *энергопотребления* промышленного предприятия для проведения энергосберегающих мероприятий.

Докладывалась также модель технической эксплуатации средств измерения. Условия эксплуатации определялись:

- количеством рабочих мест по ремонту и поверке;
- вероятностью достаточности ЗИП;
- укомплектованностью метрологических и ремонтных органов обменными фондами, технологическим оборудованием, передвижными лабораториями.

Очевидна возможность ее применения для анализа работы ремонтных служб разнообразного профиля.

18.8.2 Логистика

Под логистикой понимаются разнообразные процессы перемещения материальных объектов. На конференции сообщалось о внедрении моделей:

- сборочных конвейеров;
- систем транспортировки грузов внутри предприятия (краны, погрузчики, трейлеры и т. п.);

- стационарных напольных и подвесных систем транспортировки грузов;
- складских процессов (прием грузов, их перемещение в зоны хранения и обратно, отбор, комплектация, упаковка и отправка);
- технологических операций, выполняемых роботами (кинематические, временные и эргономические);
- кинематические модели перемещения крупных объектов (фюзеляжей самолетов, корпусов судов);
- анализа взаимного пространственного расположения компонент изделия;
- физического изменения изделий при прессовке, гибке и т. п.;
- ситуаций типа Crashtest (последствий удара, столкновения и т. п.).

Подобные модели, в частности, используются при разработке проектов реконструкции металлургических заводов для анализа:

- совместной работы технологического и подъемно-транспортного оборудования на уровне цеха;
- внутризаводских транспортных потоков, включая склады полупродукта и готовой продукции, в связи с пуском новых агрегатов (цехов) и выводом из действия устаревшего оборудования.

При этом учитываются пространственное взаимодействие агрегатов на уровне кранового оборудования и вероятностный характер процессов, в частности, переменная длительность операций, совпадение требований на обслуживание, взаимные помехи при работе на одном уровне, износ оборудования.

Приводится пример реконструкции сталеплавильного цеха ОАО «Северсталь» (замена электродуговых печей на шахтные, установка сортовой машины непрерывной разливки и реконструкция слябовой) при сохранении существующего здания цеха и подъемно-транспортного оборудования.

По разделу железнодорожного транспорта определены: местоположение дополнительных путей, их необходимая длина и места установки

дополнительных стрелочных переводов. Рассчитаны необходимое число вагонов в составе и количество локомотивов. Проведена оценка пропускной способности автомобильных дорог предприятия с учетом задержек на железнодорожных переездах.

18.8.3 Транспорт

Из специфически транспортных проблем моделировались: организация северного завоза грузов — по железной дороге и кораблями,а также подвод экспортного груза к припортовой станции (Новороссийск). Исследовались алгоритмы диспетчирования в междугородных автогрузовых перевозках. Требовалось найти ответы на следующие вопросы:

- какой наибольший суточный поток заявок заданной структуры может обработать фирма при уровне обслуживания не ниже 80 % и существующем парке грузовиков?
- сколько и каких грузовиков надо иметь фирме для обеспечения 80 % уровня обслуживания при поступлении 30–40 заявок за каждые 6 часов?
- имеет ли смысл для повышения уровня обслуживания и прибыльности компании откладывать выполнение заявки до освобождения транспорта?

Р. Г. Гиниятуллин моделировал работу казанского метро. Были поставлены следующие задачи:

- анализ влияния интенсивности пассажиропотоков на загрузку метрополитена и возможные организационные способы устранения очередей в часы «пик»;
- исследование вариантов расписаний движения поездов;
- исследование влияния надежности работы оборудования на пропускную способность системы;
- анализ эффекта оргмероприятий: количества и расстановки турникетов, расстановки ограждений, введения (отмены) льготного проезда;

- экономическая оценка функционирования метро;
- работоспособность служб метрополитена в чрезвычайных ситуациях: резкое увеличение пассажиропотока, выход из строя большого количества оборудования, снижение энергоснабжения.

18.8.4 Авиация

М.С. Неймарк моделировал эффективность применения самолетов в авиакомпаниях. Модель применялась в конкретных авиапредприятиях для оценки результатов реальной эксплуатации ИЛ-62, ИЛ-86 и для прогнозов — по разрабатываемым Ил-96-300 и Ил-114.

А.А. Артамонов моделировал использование воздушного пространства в зоне аэропорта Рига для определения вероятности конфликтных ситуаций и влияния на нее исходных параметров. Данный доклад интересен большим объемом реальных исходных данных.

Серьезные проблемы порождает *наземное* движение воздушных судов в аэропорту. Здесь целевыми характеристиками считались:

- суточная пропускная способность;
- параметры задержек вылетов;
- время руления по летному полю при взлете и посадке;
- время работы аэропорта в режиме перегрузки.

Компонентами модели являлись: редакторы карты летного поля, расписания, сценариев; имитатор; модуль статистики; модуль визуализации.

18.8.5 Горное дело

Имитационное моделирование в горном деле имеет достаточно давнюю историю. В 1958 г. моделирование было применено для планирования бурения, взрывания, погрузки и крепления в забоях угольной шахты. Позже с помощью имитационного моделирования стали решать задачи по анализу работы шахтной транспортной сети, процессов камерной выемки, взаимодействия самосвалов и экскаваторов на открытом разрезе, работы рельсового транспорта на поверхности и т. д.

18.8.6 Медицина

Медицинские доклады, к сожалению, не затрагивали процессы лечения больных. Моделировалось транспортное обеспечение учреждения «семейной медицины». Рассматривались следующие «автомобильные стратегии»:

- предоставить автомобиль каждому врачу (и тем увеличить пропускную способность с возможностью расширения клиентуры),
- выделить один на несколько врачей,
- не предоставлять никому.

Еще одним объектом была система оплаты врача. Наконец, в докладе Ю.В. Шорникова речь шла о работе (и диагностике) системы желчевыделения, описываемой системой дифференциальных уравнений с запаздывающим аргументом.

18.8.7 Финансы

Большинство участников конференции представляли точные и технические науки. Поэтому для них оказались новыми и интересными финансовые приложения.

Э. Б. Песиков оценивал на имитационной модели эффективность и степень риска маркетинговых стратегий предприятия. Учитывались действия поставщиков ресурсов и предприятий-конкурентов, коэффициенты эластичности спроса в зависимости от цены товара и дохода потребителей, а также воздействие рекламы на объемы продаж продукции (с помощью модели Видаля—Вольфа).

И. В. Яцкив применяла моделирование для оценки рисков инвестиционных проектов. Были определены оцениваемые показатели: математическое ожидание, дисперсия, стандартное отклонение и вариация чистой современной (приведенной к дате расчета) стоимости проекта. Выделены детерминированные и стохастические рисковые факторы, влияющие на изменения денежных потоков и упомянутую стоимость проекта. Приведена схема программы имитации. Создан «бизнес-калькулятор», позволяющий настраивать параметры модели под реальные данные и проводить статистический анализ полученных результатов,

а также выяснить чувствительность проекта к размеру банковской ссуды, ее сроку и проценту по ней.

Очень свежей показалась имитационная модель финансовых торгов (К.В. Воронцов). Торговля идет как двойной аукцион. Участники торгов в произвольном порядке подают *заявки* с указанием направленности (покупка или продажа), цены и объема. Приняв заявку, *торговый автомат* просматривает очередь заявок, поданных ранее, пытаясь найти удовлетворяющую ей встречную заявку (цена заявки на покупку должна быть не меньше цены заявки на продажу). Приоритет имеют заявки, поданные по наилучшей цене, а среди имеющих одинаковую цену — поданные раньше. Если встречная заявка имеется, автомат фиксирует *сделку*. Если заявка не удовлетворяется или удовлетворяется частично, остаток ставится в очередь. Участник торгов в любой момент может снять заявку из очереди.

В каждый момент максимальная цена в заявках на покупку меньше минимальной цены в заявках на продажу. Разница между ними называется *спрэдом*.

Модель воспроизводит реальную торговую сессию, имитируя действия отдельных участников. Каждый из них «видит» протокол торгов, текущее состояние очередей, собственную позицию (купленный и проданный объемы) и принимает решения о подаче или снятии своих заявок на основе формализованных *мотивов* (степеней целесообразности возможных действий). Чем больше мотив, тем выше значения параметров сделки.

В докладе рассматриваются специфические методы валидации модели. Модель называется *адекватной*, если в ответ на типовое возмущающее воздействие возникает типичная ответная реакция (например, в ответ на единичную покупку большого объема происходит кратковременное расширение спрэда и повышение цены, а затем откат обратно).

18.8.8 Информатика

Вычислительные системы и средства связи всегда были популярными объектами математического (в частности, имитационного) моделирования. Самые общие проблемы в данной области рассматривались в докладах об оценке живучести распределенных вычислительно-управляющих систем и об информационных потоках в корпоративной распределенной геоинформационной системе.

Два доклада были сделаны по алгоритмам и протоколам маршрутизации. Изучались также коммутационные системы с различными типами потоков вызовов и временные задержки в них; передача телефонного трафика (речи) по сети `Ethernet` в реальном масштабе времени и по серверам мультимедиа. Оценивалась экономическая целесообразность установки оборудования для оказания клиентам кабельного телевидения Интернет-услуг.

Наконец, моделировалась локальная вычислительная сеть с топологией «Общая шина». Сравнивались методы организации дискового пространства файловых серверов.

18.8.9 Коммунальное хозяйство

К данной сфере, проклятой богом и людьми, исследователи обращаются неохотно. По этому поводу процитируем крик души Л. А. Осипова: «Известный навигационный прибор GPS весом 200 г на маршрутах в десятки и сотни километров выводит путешественника к намеченной цели с точностью до одного метра! Неужели в социально-экономических системах вида ЖКХ мы не можем повысить точность расчетов?». Адекватные математические модели проблем ЖКХ не решат, но могут пригодиться. Поэтому упомянем доклад Л. А. Осипова о моделировании службы ремонта теплосетей РТС-2 г. Сургута.

18.9 Недостатки конференции

На качестве конференции отразилось отсутствие «приглашенных» (заказных) докладов по общим проблемам имитационного моделирования. Такой проблемой прежде всего являются сравнительный анализ и на его основе выбор инструментальных средств. Однако ее решение под силу только *коллективу*. Оргкомитет очередной конференции, учитывая опыт первой, мог бы подготовить методику сопоставления (и, возможно, тестовые задачи), разослать «засветившимся» пользователям приглашения по их любимым системам и представить коллективный аналитический обзор. Такой доклад окупил бы все затраты на подготовку конференции. Кстати, для этого не обязательно дожидаться конференции: можно просто издать информационный или тематический сборник.

При общем высоком уровне конференции по содержанию и форме некоторых докладов приходится сделать ряд серьезных замечаний.

Не то моделирование. Часть докладов не соответствовала профилю конференции — *имитационное* моделирование. К примеру, обсуждалось применение системы $Simulink$ из $MatLab$, которая ничего общего с имитацией не имеет. В докладе о моделировании системы «трал — объект лова» имело место не имитационное моделирование, а метод Монте-Карло.

Банальности и переоткрытие хорошо известного. Многие доклады начинались с перечисления преимуществ и недостатков имитационного моделирования. Это естественно для лекции перед студентами, но было излишне «в кругу своих».

Вряд ли интересно сообщение, что «внесение в цикл диспетчирования временных задержек при передаче заявок на обслуживание не приводит к повышению уровня обслуживания и прибыльности».

В некоторых докладах методами имитационного моделирования переоткрывались известные факты теории очередей (расчет СМО для распределений с «толстыми хвостами», зависимость среднего времени пребывания приоритетных заявок при рекуррентном входящем потоке от высших моментов распределений интервалов между смежными заявками).

Реклама непроверенного. В нескольких докладах излагается и развивается бутстреп-метод. Последний предполагает, что имеющиеся N значений с дисперсией d образуют генеральную совокупность, из которой извлекаются (с возвращением) B выборок. По каждой из них строится оценка искомого параметра исходной случайной величины, затем полученные оценки усредняются. Утверждается, что бутстреп-оценка математического ожидания имеет дисперсию $d/(BN)$, и сообщается, что строгое обоснование метода отсутствует.

Идея этого метода вызывает серьезные сомнения уже с позиций «философии науки». В описанной технологии нет привлечения никакой *дополнительной* информации об исследуемом процессе (в отличие от методов расслоенных выборок, контрольных и коррелированных переменных, условной имитации). Здесь явно имеет место самообман.

Явные ошибки. Приводятся схемы алгоритмов, которые никак нельзя признать таковыми (два входа, висячие фрагменты и т. п.).

Некоторые докладчики путают (или, по крайней мере, безответственно употребляют) такие понятия, как среднеквадратическое откло-

нение, дисперсия (вариация) и второй начальный момент распределения.

Утверждалось, что «глобальные модели наиболее полно отображают структурные и функциональные особенности организации исследуемых систем и представляют собой модели с высоким уровнем детализации» . Общепризнано, что глобальная модель представляют исходную систему *укрупненно*, и получать ее простым объединением детальных не следует.

Наконец, вызывают изумление три тезиса одного из докладчиков:

- Определение соответствия схем и характеристик теории СМО не требует особого ума и труда не составляет.
- Процесс познания на основе метода Монте-Карло идет со скоростью работы ЭВМ.
- Метод имитационного моделирования не требует знания высшей математики.

Заключение

Развитие науки и техники заставляет исследователей иметь дело со все более сложными системами, адекватные аналитические модели которых создаются с возрастающим трудом и со значительным отставанием. Следовательно, потребность в имитационных моделях останется всегда. Общими недостатками моделирующих систем являются:

- необходимость освоения дополнительных систем программирования, зачастую плохо документированных, нестабильных и существующих во множестве версий;

- ориентация на ограниченный класс моделей и как следствие — неудобство или невозможность составления моделей, не входящих в этот класс;

- трудность сбора и обработки результатов, не предусмотренных стандартной логикой, а также организации нестандартной обработки;

- повышение трудоемкости прогона модели, которое является неизбежной платой за ее универсальность;

- ограниченные возможности управления экспериментом и реализации МПД-результатов;

- низкая надежность программ, вызванная скрытой от программиста или недостаточно ясной ему логикой работы модели, что практически исключает правильное описание тонкостей функционирования объекта — особенно для временны́х узлов, когда несколько событий должны произойти квазиодновременно. Поэтому с понятиями о

цепях событий и логике интерпретатора следует познакомиться возможно раньше. Ситуация усугубляется и провоцирующими ошибки особенностями языка.

В подтверждение последнего положения отметим, что в языке *Симскрипт* ключевые слова `ELSE`, `ALWAYS` и `OTHERWHISE` могут использоваться как синонимы для `REGARDLESS`, а в заголовках процедур

```
ROUTING EXAMPLE GIVING X
ROUTING EXAMPLE YIELDING X
```

в первом случае `X` является входным, а во втором — ответным параметром ([80, с. 230]. Недостатки *GPSS* обсуждались в главе 16.

Тем не менее имитационное моделирование имеет хорошие перспективы. Их реализация будет облегчаться:

- ростом быстродействия и объема памяти ЭВМ;
- распараллеливанием вычислений на многопроцессорных машинах;
- созданием сетей ЭВМ с возможностью выхода пользователя на наиболее подходящую машину;
- развитием методов понижения дисперсии результатов;
- обеспечением интерактивного режима взаимодействия программиста с моделью во время прогона последней;
- созданием специализированных устройств моделирования.

Для облегчения процессов разработки имитационных моделей ведутся работы в направлении:

- создания интегрированных систем имитационного моделирования, соединяющих простые языки для простых моделей с мощными средствами описания более сложных случаев;
- разработки диалоговых систем написания моделирующих программ;
- использования банков данных по объектам моделирования, исходным данным и программам.

Литература

[1] *Абчук В. А., Матвейчук Ф. А., Томашевский Л. П.* Справочник по исследованию операций. — М.: Воениздат, 1979. — 368 с.

[2] *Адлер Ю. П., Маркова Е. В., Грановский Ю. В.* Планирование эксперимента при поиске оптимальных условий. — М.: 1976.

[3] *Альянах И. Н.* Моделирование вычислительных систем. — Л.: Машиностроение, 1988. — 225 с.

[4] *Андрианов А. Н., Бычков С. Л., Хорошилов Д. И.* Программирование на языке Симула–67. — М.: Наука, 1985. — 228 с.

[5] *Антипов М. В.* Системы генераторов псевдослучайных чисел для персональных компьютеров. — Препринт 910. — Новосибирск: ВЦ СО АН СССР, 1990. 28 с.

[6] *Ахиезер Н. И.* Классическая проблема моментов и некоторые вопросы анализа, связанные с нею. — М.: Физматгиз, 1961. — 310 с.

[7] *Банди Б.* Методы оптимизации. Вводный курс /пер. с англ. — М.: Радио и связь, 1988. — 128 с.

[8] *Бартеньев О. В.* Современный Фортран. — М.: Диалог-МИФИ, 1998. — 397 с.

[9] *Бартеньев О. В.* Фортран для студентов. — М.: Диалог-МИФИ, 1999. — 400 с.

[10] *Берсенев Г. Б., Шкатов П. Н., Морозов А. А.* Построение и анализ моделей вычислительных систем аналитическими, имитационными и аналитико-имитационными методами. /Сб. научных трудов Тульского политехнического ин-та. — Тула: 1981. С. 3–17.

[11] *Бутомо И. Д., Дробинцев Д. Ф., Котляров В. П.* Методы имитационного моделирования вычислительных систем: Учебн. пособие. — Л.: ЛПИ, 1979. — 72 с.

[12] *Вадзинский Р. Н.* Справочник по вероятностным распределениям. — СПб.: Наука, 2001. — 295 с.

[13] *Важенин В. И и др.* Методы моделирования сложных систем на ЭВМ. // Труды Радиотехнического института АН СССР. — 1972, №12. — С. 5–244.

[14] *Вальд А.* Последовательный анализ /пер. с англ. — М.: Физматгиз, 1960. — 328 с.

[15] Вычислительные системы и их программное обеспечение: модели, методы и средства исследования: Учебник. / Под ред. Ю. И. Рыжикова и А. Д. Хомоненко. МО РФ, 1995. — 312 с.

[16] *Городецкий В. И., Рыжиков Ю. И.* Математическое программирование и массовое обслуживание: Учебн. пособие. — Л.: ВИКИ им. А.Ф.Можайского, 1975. — 182 с.

[17] *Гультяев А. К.* Имитационное моделирование в среде Windows. — СПб.: КОРОНА принт, 1999. — 288 с.

[18] *Дал У., Мюрхауг Б., Нюгорд К.* Симула-67. Универсальный язык программирования / Пер. с англ. — М.: Мир, 1969. — 100 с.

[19] *Двайт Г. Б.* Таблицы интегралов и другие математические формулы / Пер. с англ. — М.: Наука, 1964. — 228 с.

[20] *Добров Г. М.* Наука о науке. — Киев: Наукова думка, 1966. — 270 с.

[21] *Ермаков С. М.* Метод Монте-Карло и смежные вопросы. — М.: Наука, 1975. — 471 с.

[22] *Зедгинидзе И. Г.* Планирование эксперимента для исследования многокомпонентных систем. — М.: 1976.

[23] *Калашников В. В.* Организация моделирования сложных систем. — М.: Знание, 1982. — 64 с.

[24] *Каханер Д., Моулер К., Нэш С.* Численные методы и программное обеспечение / Пер. с англ. — М.: Мир, 1998. — 575 с.

[25] *Кельтон В., Лоу А.* Имитационное моделирование / Пер. с англ., 3-е изд. — СПб.: Питер, Киев, БХВ, 2004. — 847 с.

[26] *Кендалл М. Дж.,Стьюарт А.* Теория распределений / Пер. с англ. — М.: Наука, 1966. — 587 с.

[27] *Киндлер Е.* Языки моделирования / Пер. с чешского. — М.: Энергоатомиздат, 1985. — 288 с.

[28] *Клейнен Дж.* Статистические методы в имитационном моделировании / Пер. с англ. — М.: Статистика, 1978. — Вып. 1, 221 с.; вып. 2, 386 с.

[29] *Клейнрок Л.* Теория массового обслуживания / Пер. с англ. — М.: Машиностроение, 1979. — 432 с.

[30] *Кнут Д.* Искусство программирования для ЭВМ. Т. 2. Получисленные алгоритмы / Пер. с англ. — М.: Мир, 1977. — 724 с.

[31] *Коваленко И. Н., Кривуца В. Г., Кузнецов Н. Ю.* Опыт практического применения методов статистического моделирования в теории надежности. // Кибернетика. — 1987, №5. — С. 111–117.

[32] *Конвей Р. В., Максвелл В. Л., Миллер Л. В.* Теория расписаний / Пер. с англ. — М.: Наука, Физмат, 1975. — 360 с.

[33] *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров / Пер. с англ. — М.: Наука, 1973. — 831 с.

[34] *Кошелев В. П.* Эффективный метод моделирования «редких» случайных событий на ЭВМ. //Алгоритмы и организация решения экономических задач, вып.7. — М.: Статистика, 1976. — С. 35–41.

[35] *Крейн М., Лемуан О.* Введение в регенеративный анализ моделей / Пер. с англ. — М.: Наука, 1982. — 104 с.

[36] *Круг Г. К., Сосулин Ю. А., Фатулев В. А.*. Планирование эксперимента в задачах идентификации и экстраполяции. — М.: 1977.

[37] *Крылов А. Н.* Мои воспоминания. — Л.: Судостроение, 1984. 480 с.

[38] *Литвинов В. В., Марьянович Т. П.* Методы построения имитационных систем. — Киев: Наукова думка, 1991. — 117 с.

[39] *Лифшиц А. Л., Мальц Э. А.* Статистическое моделирование систем массового обслуживания. — М.: Сов. радио, 1978. — 247 с.

[40] Основы теории вычислительных систем / Под ред. С. А. Майорова. — М.: Высшая школа, 1978. — 408 с.

[41] Материалы 1-й Всероссийской научно-практической конференции «Опыт практического применения языков и программных систем имитационного моделирования в промышленности и прикладных разработках», тт. 1–2. — СПБ: ЦНИИ технологии судостроения, 2003.

[42] *Мик Б., Хит П., Рашби Н.* Практическое руководство по программированию / Пер. с англ. — М.: Радио и связь, 1986. — 168 с.

[43] *Мину М.* Математическое программирование. Теория и алгоритмы / Пер. с франц. — М.: Наука, 1990. — 488 с.

[44] Пакет прикладных программ МОСТ для расчета стационарных режимов в системах массового обслуживания. — Эстонское НПИ ВТИ. — 1988.

Вып.3. Описание применения. — 36 с.
Вып.5. Руководство программиста. — 92 с.
Вып.6. Теоретические основы, ч.1. — 116 с.
Вып.7. Теоретические основы, ч.2. — 142 с.

[45] *Петров А. А.* Экономика модели. Вычислительный эксперимент. — М.: Наука, 1996. — 250 с.

[46] *Питерсон Дж.* Теория сетей Петри и моделирование систем / Пер. с англ. — М.: Мир, 1984.

[47] *Полляк Ю. Г.* Вероятностное моделирование на электронных вычислительных машинах. — М.: Сов.радио, 1971. — 400 с.

[48] *Полляк Ю. Г.* Оценка малых вероятностей при статистическом моделировании систем. //Изв. АН СССР, Техн. кибернетика. — 1973, №2. — с. 197–202.

[49] *Поляк Д. Г.* Некоторые способы повышения точности статистического моделирования систем массового обслуживания. //Изв. АН СССР. Техн. кибернетика. — 1970, №4. — с. 75–84.

[50] *Поляк Д. Г., Розенталь Г. О.* О повышении точности статистического моделирования систем массового обслуживания. //Автоматика и вычислительная техника. — 1972, №6. — с. 54–58.

[51] *Поттгоф Г.* Теория массового обслуживания / Пер. с нем. — М.: Транспорт, 1979. — 144 с.

[52] *Пугачев В. Н.* Определение характеристик сложных систем методом статистических испытаний с использованием результатов аналитического исследования. //Изв. АН СССР. Техн. кибернетика. —1970, №6. — с. 79–87.

[53] *Романовский В.* Математическая статистика. — Ташкент, 1961–1963.

[54] *Рыжиков Ю. И.* Управление запасами. — М.: Наука, 1969. — 344 с.

[55] *Рыжиков Ю. И.* Машинные методы расчета систем массового обслуживания: Учебн. пособие. — Л.: ВИКИ им. А. Ф. Можайского, 1979. — 177 с.

[56] *Рыжиков Ю. И.* Эффективность и эксплуатация программного обеспечения ЭЦВМ: Учебн. пособие. — МО СССР, 1985. — 263 с.

[57] *Рыжиков Ю. И.* Имитационное моделирование систем массового обслуживания: Учебн. пособие. — Л.: ВИККИ им. А. Ф. Можайского, 1991. — 111 с.

[58] *Рыжиков Ю. И.* Руководство по расчету систем с очередями. СПб: ВИККА им. А. Ф. Можайского, 1995. — 73 с.

[59] *Рыжиков Ю. И.* Программирование на Фортране PowerStation для инженеров: Учебн. пособие. — СПб: КОРОНА принт, 1999. 159 с.

[60] *Рыжиков Ю. И.* Решение научно-технических задач на персональном компьютере: Учебн. пособие. — СПб: КОРОНА принт, 2000. — 272 с.

[61] *Рыжиков Ю. И.* Теория очередей и управление запасами: Учебн. пособие. — СПб: Питер, 2001. — 376 с.

[62] *Салеев В. М.* Библиотека алгоритмов и программ генерации и статистической обработки последовательностей случайных чисел: материалы по программному обеспечению. — Минск: Ин-т математики АН БССР, 1986. — 138 с. М.: Высшая школа, 1998. — 320 с.

[63] *Самарский А. А., Михайлов А. П.* Математическое моделирование: идеи, методы, примеры. — М.: Наука, Физматлит, 1997. — 320 с.

[64] *Советов Б. Я., Яковлев С. А.* Моделирование систем. — М.: Высшая школа, 1998. — 320 с.

[65] *Советов Б. Я., Яковлев С. А.* Моделирование систем: Практикум. — М.: Высшая школа, 1999. — 224 с.

[66] Справочник по теории вероятностей и математической статистике / Под ред. В. С. Королюка. — Киев: Наукова думка, 1978. — 582 с.

[67] *Томашевский В. Н., Жданова Е. Г.* Имитационное моделирование в среде GPSS. — М.: Бестселлер, 2003. — 416 с.

[68] *Тюрин Ю. Н., Макаров А. А.* Анализ данных на компьютере. М.: ИНФРА-М, Финансы и статистика, 1995. — 384 с.

[69] *Фельдман Л. П., Слепцов А. И.* Методы программного моделирования вычислительных систем: Учебн. пособие. — Донецк, Донецкий политехнический ин-т, 1981. — 100 с.

[70] *Харин Ю. С. и др.* Имитационное и статистическое моделирование: Учебн. пособие. — Минск: Белорусский гос. ун-т, 1992. 176 с.

[71] *Хемминг Р. В.* Численные методы / Пер. с англ. — М.: Наука, 1968. — 400 с.

[72] *Холстед М. Х.* Начала науки о программах / Пер. с англ. — М.: Мир, 1981. 128 с.

[73] *Шеннон Р.* Имитационное моделирование систем — искусство и наука / Пер. с англ. — М.: Мир, 1978. — 418 с.

[74] *Шрайбер Т. Дж.* Моделирование на GPSS / Пер. с англ. — М.: Машиностроение, 1980. — 592 с.

[75] *Яковлев Б. И.* Машинная имитация. — М.: Наука, 1975. — 158 с.

[76] *Янбых Г. Ф., Столяров Б. А.* Оптимизация информационно-вычислительных сетей. — М.: Радио и связь, 1987. — 232 с.

[77] *Anderson T. W.* An Introduction to Multivariate Statistical Analysis. — New York: Wiley & Sons, 1958.

[78] *Becker-Jouvinroux, M.* Validité des simulations de files d'attente: Thèse presentée à l'Université Pierre et Marie Curie pour l'obtention du doctorat d'État ès sciences. — Paris, 1976. — 284 pp.

[79] *Bobiller P. A., Kahan B. S., Probst A. R.* Similation with GPSS and GPSS V. — Prentice-Hall, Englewood Cliffs, N.J., 1976. — 495 pp.

[80] *Bratley P., Fox B. L., Schrage L. E.* A Guide to Simulation. — N.Y.: Springer Verlag, 1983. — 383 pp.

[81] *Carson J. S., Law A. M.* Conservation Equations and Variance Reduction in Queuing Simulation. // Opns Res. — 1980. — v. 28, no. 3. — P. 535–546.

[82] Computer Networks and Simulation. — Amsterdam: North-Holland, 1986. — 412 pp.

[83] *Cox D. R.* A Use of Complex Probabilities in the Theory of Stochastic Processes. // Proc. of the Cambridge Phil. Soc. — 1955. v.51. — P. 313.

[84] *Devroye L.* Non-Uniform Random Variate Generation. — N.Y.: Springer Verlag, 1986. — 843 pp.

[85] *Fishman G. S.* Accelerated Accuracy in the Simulation of Markov Chains. // Opns Res. — 1983. — v. 31, No. 3. — P. 466–487.

[86] *Francis N. D.* Simulation of Operating Systems — a Functional Flow-Chart. // Simuletters. — 1981. — v. 12. — P. 86–91.

[87] *Geisler M. A.* The Use of Man-Machine Simulation for Support Planning. // Naval Research Logistics Quarterly. — 1960. — v. 7, No. 4.

[88] *Geisler M. A., Haytorn W. W., Steger W. A.* Simulation and the Logistics Systems Laboratory. // Naval Research Logistics Quarterly. — 1963. — v. 10, No. 1. — P. 23–54.

[89] *Gordon G.* The Application of GPSS V to Discrete System Simulation. — Prentice-Hall, Englewood Cliffs, N. J. — 1975. — 389 pp.

[90] *Hammersley J. M., Handscomb D. C.* Monte Carlo Methods. — N.Y.: Wiley, 1964. — 178 pp.

[91] *Heidelberger P.* Variance Reduction Techniques for the Simulation of Markov Processes. I: Multiple Estimates. // IBM J. of Research and Dev. — 1980. — v. 24. — P. 570–581.

[92] *Iglehart D. L., Shedler G. S.* Regenerative Simulation of Response Times in Networks of Queues. — Berlin: Springer Verlag, 1980. — 204 pp.

[93] *Izydorczyk J.* Some Asymptotic Properties of Controlled Estimators Used in System Simulation. // Modelling Techniques and Tools for Performance Analysis: Proc. of the Internat. Conf. Ed. by D. Potier. — Amsterdam: North-Holland Publ., 1985. — P. 611–622.

[94] *Johnson M. E., Jackson J.* Infinitesimal Perturbation Analysis: a Tool for Simulation. // J. of the Operational Res. Soc. — 1989. — v. 40, no. 3. — P. 243–254.

[95] *Lavenberg S. S., Moeller T. L., Sauer C. H.* Concomitant Control Variables Applied to the Regenerative Simulation of Queuing Systems. // Opns Res. —1979. — v. 27, no. 1. — P. 134–160.

[96] *Lavenberg S. S., Welch P. D.* A Perspective on the Use of Control Variables to Increase the Efficiency of Monte Carlo Simulations. // Mgmt Science. — 1981. — v. 27, no. 3. — P.322–335.

[97] *Lavenberg S. S., Moeller T. L., Welch P. D.* Statistical Results of Control Variables with Application to Queuing Network Simulation. // Opns Res. — 1982. — v. 30, no. 1. — P. 182–202.

[98] *Law A. M., Kelton W. D.*Simulatiom Modeling and Analysis. — N.Y.: McGraw-Hill, 1982.

[99] *Lee A. M.* Applied queuing theory. — London: McMillan, 1966. — 244 pp.

[100] *Nelson B. L.* A Perspective on Variance Reduction in Dynamic Simulation Experiments. // Communications in Statistics, ser. B. 1987. — v. 16, no. 2. — P. 385–426.

[101] *Nelson B. L.* Antithetic-Variable Splitting for Steady-State Simulations. // European J. of Operational Research. — 1988. v. 36. — P. 360–370.

[102] *Ripley D. B.* Uses and Abuses of Statistical Simulation. // Math. Programming. — 1988. — v. 42, no. 1. — P. 53–68.

[103] *Roark L. M., Feldman J. K., Bexfield J.* B1-B Avionics/Automatic Test Equipment: Maintenance Queuing Analysis. // Proc. of the 1984 Winter Simulation Conference. Edited by S. Sheppard, U. Pooch, D. Pegden. — Piscataway (N.J.): North-Holland Publ. — P. 737–744.

[104] *Sharon P. A., Nelson B. L.* Analytic and External Control Variates for Queuing Network Simulation. // J. of the Operational Res. Soc. 1988. — v. 39, no. 6. — P. 595–602.

[105] *Smith G. S.* Multiplicative Pseudo-Random Number Generators with Prime Modulus // J. of ACM. — 1971. V. 18, No 4. — P. 586–593.

[106] *Suri R., Zazanis M.* Perturbation Analysis Gives Strongly Consistent Sensitivity Estimates for the M/G/1 Queue. // Mgmt Science. — 1988. — v. 34. — P. 39–64.

[107] *Tocher K. D.* The Art of Simulation. — London: English Univ. Press, 1963. — 184 pp.

[108] *Walck Ch.* Random Number Generation. // Univ. of Stockholm, Institute of Physics. — Rep. 87–15. — Dec. 1987. — 88 pp.