ВОЕННО-КОСМИЧЕСКАЯ АКАДЕМИЯ
имени А.Ф. Можайского

Ю.И. Рыжиков

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

Курс лекций

Санкт-Петербург
2007

ВОЕННО-КОСМИЧЕСКАЯ АКАДЕМИЯ
имени А.Ф. Можайского

Ю.И. Рыжиков

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

Курс лекций

Санкт-Петербург
2007

004.9
Р939
46367у

Рецензент:
доктор экономических наук, профессор **Д.Н. Верзилин**

Рыжиков Ю.И.
Р939 Имитационное моделирование:
курс лекций / Ю.И. Рыжиков — СПб.: ВКА
им. А.Ф. Можайского, 2007. — 122 с.

В книге описаны теоретические основы и техника программирования имитационных моделей. Приведены примеры работы с инструментальными системами имитационного моделирования.

Набор, верстка и корректура выполнены автором

УДК 004.9

Подписано к печати 23.06.2007 Печ. л. 8.0 Уч.-изд. л. 7,75
Заказ 1745 Цена свободная

Типография ВКА им. А.Ф. Можайского

Оглавление

Глава 1.
Базовые концепции имитационного моделирования

1.1. Моделирование систем

Модель есть материально или теоретически сконструированный объект, который заменяет (представляет) объект исследования в процессе познания, находится в отношении сходства с последним (аналогия, физическое сходство и т. п.) и более удобен для исследования. Изучение модели и выполненные над ней операции позволяют получить информацию о реальном объекте исследования и оптимизировать последний.

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.

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.

При разработке новой системы для начала строится и оптимизируется стандартными методами простейшая модель, учитывающая важнейшие факторы. 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.

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 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 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».

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). 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.

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.

Имитационное моделирование, линейное программирование и регрессионный анализ по диапазону и частоте использования давно и устойчиво занимают три первых места среди всех методов исследования операций [Шрайбер]. Теория, техника и системы автоматизации имитационного моделирования разрабатывались прежде всего для моделирования систем с очередями. Об этом свидетельству-

ют как содержание первой отечественной монографии по моделированию [1], так и первая глава знаменитой «красной книги» признанного классика *GPSS* Т. Дж. Шрайбера — «Вопросы моделирования систем массового обслуживания».

Одной из первых областей применения имитационного моделирования явилось *управление запасами*, что вполне объясняется сложностью вероятностных задач этого вида и их практической важностью. Практически все прикладные аспекты недавно переведенной книги Кельтона и Лоу связаны с задачами управления запасами и теории очередей.

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

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 (вспомним Чернобыль!).

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;

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

- блочная структура;
- эффективность воспроизведения на ЭЦВМ.

Считается [Лоу], 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 обострением 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 процесса разработки и динамики 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 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. *Усреднение* результатов моделирования по времени функционирования модели или числу реализаций процесса позволяет методами математической статистики получить *оценки* искомых характеристик.

1.3. 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*. Для наших целей наибольший интерес представляют чисто дискретные модели, состояния которых меняются скачкообразно при наступлении *событий*. Событие может изменить значение атрибута, создать или уничтожить *сущность*, начать или прекратить *активность*. Любой

процесс разбивается на этапы, каждый из которых соответствует некоторому 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.

«Физическая» длительность имитации определяется уровнем детальности. 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. Соотношение характерных времен практически должно быть не более 100:1, а в упомянутом примере системы коллективного доступа оно 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.

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 ранее с помощью 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 ЦЗС находятся события, развитие которых заблокировано сложившимися в системе на данный момент модельного времени условиями (например, занятостью необходимых ресурсов). Могут использоваться и другие цепи

событий.

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. При вставке очередного события в список, упорядоченный по моментам наступления событий, его целесообразно просматривать с конца или применять половинное деление.

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 обеспечиваются программистом.

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:

- двухэтапный прогон, когда на первом этапе грубо определяются величины, необходимые для формирования критерия остановки на втором этапе;
- рекуррентная обработка результатов моделирования.

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.

1.4. Базовый алгоритм моделирования

На рис. 1.1 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 начала обслуживания. 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), моменты освобождения каналов — 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.

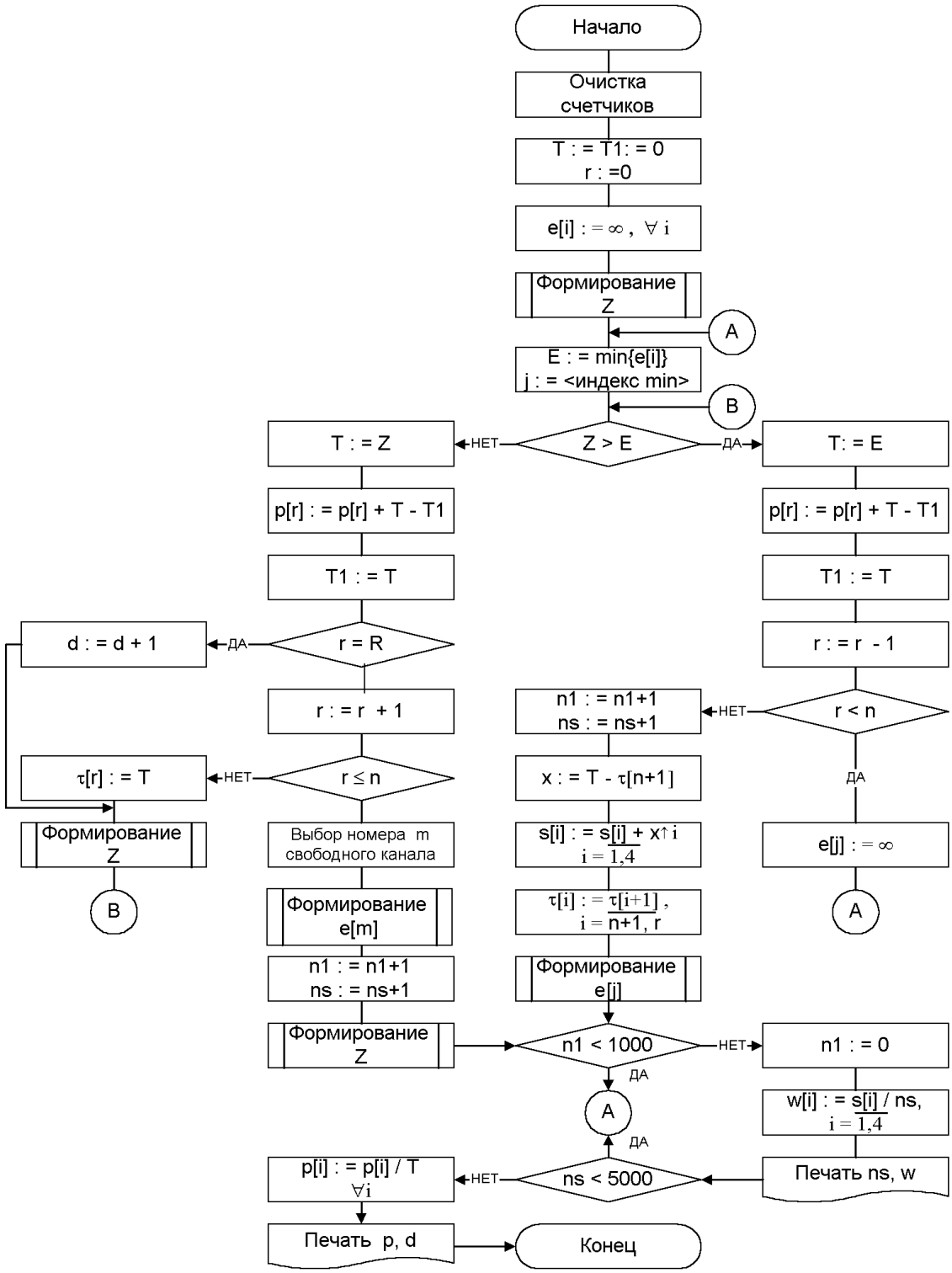

Рис. 1.1. Структурная схема имитационной модели

Приведем словарь этой модели:

`tt`	—	таймер;
`told`	—	момент предыдущего события;
`tlib[1:n]`	—	моменты завершения в каналах текущего обслуживания;
`ttlib`	—	наиболее ранний из них;
$\tau[n+1:R]$	—	моменты прибытия находящихся в очереди заявок;
`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`	—	текущее число заявок в данной серии (ограничено 500);
`nref`	—	количество отказов в приеме на обслуживание;
`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 минимального из {tlib[i]} (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 счетчике n1 $<$ 500 модель возвращается на выбор минимума из {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 полного 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.

1.5. Общая характеристика имитационного моделирования

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.

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).

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.

Перечисленные недостатки имитационного моделирования смягчаются ростом технических характеристик (быстродействия и объема памяти) современных ЭВМ и использованием обсуждаемых ниже методов понижения дисперсии.

1.6. Структуры данных

До сих пор мы рассматривали выбор заявок из очередей в порядке их прибытия, то есть по принципу FCFS (First Come — First Served). При работе по правилу LCFS (Last Come — First Served) ожидающие обслуживания заявки образуют *магазин*, указателем которого служит текущая длина очереди.

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.

Как обычную очередь, так и магазин можно организовать в виде закольцованного массива длины (закольцованность достигается от-

носительной адресацией всех обращений к этому массиву по модулю). В этом случае перемещение информационных элементов заменяется пересчетом соответствующих указателей ($\mathbf{\Gamma}$ и $\mathbf{K}$ — голова и конец очереди, $\mathbf{У}$— указатель магазина), что реализуется значительно экономнее.

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 «по максимуму», работа со списками 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).

1.7. Варианты моделей систем обслуживания

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

1.7.1. Определение 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.

1.7.2. 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.).

1.7.3. Многоресурсное обслуживание

Иногда для начала обслуживания заявки требуется одновременное наличие ресурсов нескольких видов, имеющихся в ограниченном объеме. В подобных случаях возможна ситуация типа известных из теории операционных систем «смертельных объятий» (deadlocks): несколько заявок из очереди частично обеспечили себя, захватив ресурсы разных видов. Они взаимно блокируются, что приводит к параличу процесса обслуживания. Для исключения таких блокировок можно исключить назначение неполных комплектов или ввести приоритеты ресурсов.

1.7.4. 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 текущей очереди и сравнивать его с моментами прибытия находящихся в ней заявок. Для выбора следующей очереди после i-й при $i = n$ принимается $i = 1$.

1.7.5. Статический относительный приоритет

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

1.7.6. Динамический приоритет

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 следующая за ней часть очереди сдвигается вперед.

1.7.7. Многоуровневое квантованное обслуживание

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

1.7.8. 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 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.

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 ее прерывании добавляется единица. При нормальном завершении обслуживания единица добавляется к счетчику, соответствующему итоговой кратности.

1.7.9. Многоканальные приоритетные системы

При моделировании *многоканальных* систем 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 являет-

ся «первый свободный», выявляемый по «запредельному» моменту освобождения.

При нескольких свободных каналах может потребоваться равновероятный выбор из них. Если таких каналов m, то для определения выбираемого канала следует вычислить $N = [m * U] + 1$, где U — случайное число, равномерно распределенное на полуинтервале $[0, 1)$, 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 × 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 C 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.

Обозначим через π младший приоритет текущего обслуживания и сравним его с типом i новой заявки. При $i < \pi$ заявка, находящаяся в n-й позиции списка, прерывается; по номеру канала определяется остаток обслуживания; прерванная заявка помещается в очередь. При $i \geq \pi$ в очередь помещается новая заявка. Для быстрого поиска ме-

ста 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.

Отметим дополнительные особенности моделирования приоритетного обслуживания с прерыванием и возобновлением обслуживания со старой или новой реализациями его случайной длительности (схемы RW и RS соответственно — по Конвею). 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.

1.7.10. Учет временных ограничений

При моделировании системы с *временны́ми ограничениями* добавляется новый тип событий — «исчерпание терпения». Каждую заявку в очереди (магазине) должен сопровождать допустимый момент начала (завершения) обслуживания. Для минимизации числа уходов по нетерпению имеет смысл упорядочить очередь по возрастанию упомянутых моментов. Соответственно для вновь прибывшей заявки придется определять место вставки и сдвигать часть очереди. 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 заявки.

1.7.11. Цели моделирования и счетчики

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.

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.

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 ги-

стограмму 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.

1.8. 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:

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

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

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

3. 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. 1 выбирается узел–преемник.

4. 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.

Отметим принципиальную необходимость в *рекурсивности* процедуры разблокировки: разблокированный узел может вызвать цепь разблокировок других узлов и соответственно повторные обращения к той же процедуре.

В сетях обслуживания часто встречаются процессы типа «Split and Join» — расщепления и слияния заявок. Примером может служить поступление в ремонт сложного изделия, блоки которого после разборки проходят по индивидуальным траекториям 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. Описанному процессу аналогична передача 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.

1.9. 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 аппаратных 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, личных амбиций, хакерского зуда, «виртуальной реальности» и компьютерных вирусов).

1.10. Агентно-ориентированное моделирование

Агенты — это автономные программные сущности, которые находятся в гетерогенной компьютерной среде и (совместно с другими агентами или в одиночку) обеспечивают достижение определенных целей. Предполагается, что они обладают следующими свойствами:

1. Автономность: агенты функционируют без какого-либо внешнего управления, осуществляют самоконтроль своих состояний и действий.

2. Социальное поведение: для достижения своих целей агенты взаимодействуют друг с другом посредством обмена сообщениями.

3. Реактивность: агенты способны воспринимать внешнюю информацию и реагировать на нее.

4. Инициативность: агенты могут выполнять определенные действия не только по запросам окружения, но и согласно своим планам и целям.

Многоагентные системы (МАС) считаются одним из наиболее перспективных подходов к разработке сложных распределенных систем. Примерами проблемных областей для них являются логистика, транспортные и телекоммуникационные сети, мониторинг бизнес-процессов и состояния окружающей среды, ликвидация последствий природных и техногенных катастроф, электронная коммерция и т. д. Известны такие инструментальные среды создания МАС, как *AgentBuilder*, *agentTool* и др. Инструментальные среды, интегрируя отдельные технологические решения, обеспечивают полный цикл разработки прикладных систем: анализ предметной области, проектирование, собственно разработку, верификацию, развертывание и сопровождение.

Основой для разработки таких инструментальных сред как правило служат технологии, разработанные в рамках *объектно-ориентированного подхода*. При этом в качестве формального языка спецификации проекта используется *UML* (`Unified Modeling Language`). На его основе создаются система *Agent UML*, *AML* и язык *AML*.

В методологии *Gaia* рассматриваются два этапа: анализа предметной области и проектирования прикладной системы. Целью *этапа анализа* является достижение понимания системы и ее структуры (выявление и описание задач, решаемых системой, описание ролей элементов и их взаимодействия) — без описания каких-либо деталей разработки. На *этапе проектирования* выполняется переход к моделям более низкого уровня абстракции, которые затрагивают уже детали разработки. В частности, здесь выполняется описание моделей агентов и моделей выполняемых ими задач (сервисов).

Интенсивные исследования в этой области ведутся в Санкт-Петербургском институте информатики РАН. Согласно Трудам СПИИРАН, вып. 3, 2006 г., в общем случае проект МАС реализуется по следующей технологии:

1. На специально разработанном языке *AFW* (`Agent Framework`), являющемся подмножеством *XML*, создается описание МАС.
2. С помощью системы графических редакторов диаграмм выполняется описание ее компонент.
3. Посредством библиотеки классов реализуется инвариантная метамодель агентов («основа агента»).
4. Генератор агентов производит автоматическую генерацию исходного и исполняемого кодов программных агентов на языке C++.
5. Система установки агентов обеспечивает развертывание агентов в сети.

Опишем структуру агентов. Основа агента является готовой компонентой, входящей в состав среды и используемой для генерации всех прикладных агентов. Она реализует набор проблемно независимых функций: запуск и остановка агента, отправка и получение сообщений, запуск и выполнение сервисов, доступ к базе данных агента и внешним компонентам.

Компоненты «Модель поведения агента», «Ментальная модель агента» и «Сервисы» содержат проблемно-ориентированное описание агента на языке *FPW*.

Сервисы описывают в терминах машин состояний функции, которые должны уметь выполнять агенты для исполнения предписанных им ролей. Упомянутые машины определяются множеством состояний, соответствующих сервисным функциям, и переходами между ними. Машины состояний позволяют реализовывать прерывания и режим ожидания.

Ментальная модель описывает данные и знания, которые используются при выполнении сервисов и которыми агенты могут обмениваться.

Модель поведения агента описывает сценарии исполнения сервисов; события, инициирующие запуск последних; действия агентов во внешней среде. Агенты одного класса имеют общую модель поведения, общий набор сервисов и используют одни и те же внешние компоненты. Они различаются только содержанием данных и знаний, хранящихся в ментальной модели.

На этапе анализа предметной области выполняются выявление и описание задач, ролей, распределение задач по ролям, описываются модели взаимодействия ролей. На этапе разработки онтологии предметной области выполняется описание понятий и отношений между ними.

Событие, инициирующее выполнение подзадачи, как правило связано с выполнением коммуникационного акта согласно описываемому протоколом сценарию. Пусть, например, перед исполнителем роли `Manager` стоит задача назначить работу и согласовать время ее выполнения. `Manager` посылает всем агентам, играющим роль `Executor'a`, сообщение с указанием с описанием требований в отношении данной работы. Каждый `Executor`, получивший такое сообщение, анализирует принятые им ранее обязательства и оценивает свои возможности выполнить предложенную работу. На основе этого `Executor` возвращает `Manager'y` либо отказ, либо предложение с описанием своих условий. Далее `Manager` анализирует полученные предложения, выбирает из них наиболее подходящее и посылает соответствующему `Executor'y` подтверждение своего согласия, а остальным участникам переговоров – отказ. Сценарий завершается тем, что выбранный `Executor` извещает `Manager'a` о своем согласии выполнить работу

на оговоренных условиях.

Особый интерес представляют МАС с элементами искусственного интеллекта — *когнитивные системы*. Такие МАС могут решать сложные задачи обработки информации и управления, связанные, в частности, с коллективным поведением роботов. Пример — агенты, разработанные для игровой среды футбола роботов. Агент-футболист имеет трехуровневую организацию поведения (технические навыки, индивидуальное поведение и поведение в команде). Все уровни поведения реализуются с помощью соответствующих баз знаний и средств логического вывода. В частности, предусматривается обучение игроков: агент действует так, чтобы максимизировать суммарное число подкреплений (наград), полученных за взаимодействие со средой в процессе обучения. Разумеется, «робофутбол» является не конечной целью подобных исследований, но полигоном для отработки *общих* подходов к решению *реальных* проблем. Для примера сошлемся на кибернетическое противоборство в компьютерной сети, связанное с атакой типа «отказ по перегрузке». Такая ситуация представляется в виде противоборства команд программных агентов: злоумышленников и агентов защиты.

О важности многоагентного моделирования убедительно свидетельствует большое внимание, уделенное этому подходу в представленном лично Президенту США докладе "Computational Science: Ensuring America's Competitiveness" консультативного комитета по информационным технологиям (июнь 2005 г.). В докладе определяются сущность многоагентного подхода и типы агентов:

«Агентами могут быть индивидуумы (типа потребителей и производителей), социальные группы (семейства, фирмы, общины, правительственные агентства); учреждения (рынки, регулирующие системы); биологические объекты (зерновые культуры, домашний скот, леса); физические объекты (инфраструктура, погода, географические регионы). Таким образом, агенты могут варьироваться от активных (собирающих данные и принимающих решения со способностями к обучению) до пассивных (без познавательной функции). Кроме того, агенты могут быть составлены из других агентов, образуя иерархические конструкции». Далее определяются цели агентных исследований:

1. Эмпирическое понимание. Почему специфическая макрорегулярность развилась и сохранилась, несмотря на отсутствие нисходящего планирования и управления? Примеры таких регулярностей включают торговые сети, рыночные соглашения, деловые циклы, восприятие технологических новшеств. Исследователи ищут причинные объяснения, основанные на повторных взаимодействиях агентов в реалистично представленных мирах.

2. Нормативное понимание. Как использовать агентные модели для открытия хороших экономических решений? Исследователи, преследующие эту цель, пытаются оценить, приведут ли со временем предложенные проекты к социально желательной работе системы.

3. Качественное понимание и построение теории. Как понять полный потенциал экономических систем?

4. Методологическое продвижение. Какие методы и инструменты нужны для строгого изучения экономических систем через управляемые вычислительные эксперименты?

В качестве примера агентного подхода описана модель электроснабжения, в которой как автономные адаптивные агенты используются индустриальные компоненты и владеющие этими компонентами корпорации. Модель предполагала открытый доступ и назначение цен в реальном времени. Она должна была помочь понять действие разрегулированного энергоснабжения. Следовало определить, как программно реализованные интеллектуальные агенты могут использоваться в управлении сложной распределенной системой и обеспечить самовосстановление системы электроснабжения.

1.11. Разработка 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 них ошибки исправляются с наибольшим трудом.

Типичный процесс моделирования проходит следующие фазы:

1. Организация.
2. Создание концептуальной модели.
3. Подготовка данных.
4. Программирование модели.
5. Верификация программы.
6. Планирование прогонов.
7. Машинный эксперимент.
8. Анализ результатов.
9. Интерпретация.
10. Реализация.
11. 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.

1.11.1. Организация разработки

Организация является важным элементом работы вследствие значительной трудоемкости последней. В 1960-х гг. в США трудоемкость «ручной» разработки даже простых моделей оценивалась в 5–6 человекомесяцев (30 тыс. долларов), а сложных — на два порядка дороже. В организацию прежде всего входят:

- установление связей с заказчиком;
- определение задач;

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

Далее по ходу всей работы над проектом 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 и т. п.);
- наблюдение за системой (конкретные объекты наблюдения, единицы измерения, формат представления информации, требуемая точность, гарантии объективности);
- изучение опыта и инструментов решения подобных задач.

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.

Концептуальная модель включает в себя:

- 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.

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.

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:

- 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.

Далее обсуждаются принципиальные проблемы представления исходных данных. Здесь могут приниматься решения по таким вопросам, как:

- стартовый уровень понятия «заявка» (детали, сборки, изделия, коробки);
- типы распределений случайных величин;
- 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 (возможна коррекция после прогонов пробных версий модели);
- 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.

1.11.2. Подготовка исходных данных

Подготовка исходных данных состоит в сборе и обработке результатов наблюдений за моделируемой системой. С точки зрения общей теории управления здесь решается задача *идентификации*. Обычно она ставится как минимизация функционала отклонения траектории модели от траекторий исследуемой системы. При этом тип модели предполагается известным. На практике для ее решения традиционно применяются методы наименьших квадратов и наибольшего правдоподобия.

Обработка в типичном случае заключается в построении функций распределения соответствующих случайных величин или вычислении числовых характеристик (моментов) распределений. Данные могут быть использованы и в необработанном виде (например, результаты трассировки вычислительного процесса на реальной установке — при прогоне ее имитационной модели). Такой («ретроспективный») подход часто используется для верификации моделей путем

сопоставления результатов их работы с ранее наблюденными аналогами.

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

Отчет о концептуальной модели должен быть понятен менеджерам. После рассылки его следует обсудить в кругу исполнителей, заказчиков и консультантов. При этом проверяются:

- полнота учета основных факторов и ограничений, влияющих на работу системы;
- соответствие исходных данных модели реальным (в частности, согласия постулируемых законов распределения с первичными данными).

Принятие положительного решения (с учетом поправок) рассматривается как *валидация* (признание ценности) концепции модели. Одним из оснований валидации может служить репутация разработчиков модели.

Как валидацию *завершенного проекта в целом* можно рассматривать натурные испытания рекомендаций модели (в особо ответственных случаях) и ретроспективную проверку ее прогностических возможностей.

1.11.3. Программирование и отладка модели

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.

Проверка надежности программы модели, т. е. ее соответствия разработанной концепции, называется *верификацией* модели. Проблеме повышения надежности программ посвящена обширная литература. В настоящее время наиболее эффективным средством считается

структурный подход к процессу программирования и жесткая (сертифицированная) технологическая дисциплина с соответствующей программной поддержкой. Здесь мы обсудим особенности разработки и отладки программ *имитационного моделирования*.

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

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.

При отладке необходимы режимы компилятора, позволяющие выявить случаи не описанных переменных и переменных с не установленными до их использования значениями, а также выход индексов за границы массивов.

В процессе отладки подлежат обязательной проверке:

- осмысленность 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 Ко-

ши, *не имеющим моментов*);

- выполнение для модели основных законов предметной области типа законов сохранения;
- правильность преобразования исходных данных в конечные результаты (например, при условиях, приводящих к известному аналитическому решению).

Как дополнительное средство отметим *визуализацию* выходного процесса имитации, часто позволяющую вскрыть дефекты в постановке задачи (например, при имитации транспортных потоков — неучет задержек на ликвидацию последствий дорожно-транспортных происшествий).

1.11.4. Прогоны и эксперименты

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.

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;
- 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.

После сброса статистики разгонного участка для оценки стабильности результатов следует периодически выдавать накопленную с начала стационарного режима стандартную статистику.

1.11.5. Анализ результатов, интерпретация, реализация, аккредитация

Этап *анализа результатов* определяется спецификой модели.

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 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.

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.

В заключение этого 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 книге автора, не готовые ответы, но полезные аналогии.

1.12. Контрольные вопросы

1. Самостоятельно нарисуйте базовую схему имитационной модели. Убедитесь в ее правильности, «поползав» по ветвям алгоритма.
2. Модифицируйте базовую схему имитационной модели применительно к расчету моментов времени *пребывания*.
3. Нарисуйте схему имитационной модели одноканальной системы с абсолютным приоритетом и прежней реализацией длительности обслуживания прерванной заявки.
4. Нарисуйте схему имитационной модели двухканальной системы с относительным приоритетом.
5. Нарисуйте схему имитационной модели одноканальной системы с заявками, нетерпеливыми в очереди. Определите среднюю частоту уходов в единицу времени.
6. Нарисуйте схему имитационной модели одноканальной системы с заявками, нетерпеливыми в системе. Определите среднюю частоту уходов в единицу времени.
7. Нарисуйте схему имитационной модели тандема из двух одноканальных систем с ограниченным буфером перед второй системой.
8. Нарисуйте схему имитационной модели циклической системы с квантованным обслуживанием.
9. Нарисуйте схему имитационной модели многоуровневой системы с квантованным обслуживанием.
10. Укажите особенности моделирования сетей обслуживания.
11. В чем состоит специфика моделирования вычислительных систем?
12. Что такое валидация, верификация и аккредитация модели?

Глава 2.
Генерация случайных чисел

2.1. 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). В компьютерных науках случайность нужна при испытании программ, в играх, для сравнения алгоритмов.

Первичные данные должны быть получены наблюдением за работой реальной системы. 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, $i = 1, 2 \ldots$, равномерно распределенного на полуинтервале $[0, 1)$.

2. Программный переход от U_i к случайному числу X_i, имеющему требуемое распределение $F_X(x)$.

Генераторы оценивают по качеству формируемой последовательности, быстродействию, трудоемкости инициализации, машинной независимости, диапазону применений, простоте и понятности для пользователей. 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 $M/M/1$.

Любой алгоритм, будучи переписан на языке ассемблера, даст значительный выигрыш в быстродействии (для генератора равномерных чисел — вдвое [Салеев]). Скорость генерации можно увеличить, применяя приближенные методы вычисления обратной функции распределения. В программах некоторых современных ДСЧ предусматривается генерация за одно обращение *массива* случайных чисел. Эффективное быстродействие датчиков здесь увеличивается из-за уменьшения числа обращений к подпрограмме.

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

2.2. Генерация равномерно распределенных чисел

2.2.1. 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 с десятичной точкой, фиксированной перед старшим разрядом. При этом 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 $e^{-2\lambda\Delta t}$. При достаточно большом произведении $\lambda\Delta t$ и идеальной работе счетчика разбаланс может быть сделан сколь угодно малым.

Физические ДСЧ формируют истинно случайные числа и исключают затраты процессорного времени на их генерацию. Схемная нестабильность физических датчиков 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. Принято 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».

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 $[0, 1)$, при $r = 2$ — единичного квадрата).

Согласие фактических распределений с ожидаемыми устанавливается, например, по критерию χ^2. Дополнительно проверяется отсутствие корреляции между последовательными числами и отдельными разрядами чисел. Наглядным средством визуального контроля для двумерного случая является равномерность засветки на экране единичного квадрата (координатами каждой точки является пара смежных чисел).

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

2.2.2. Идея построения программных датчиков

Программные датчики чисел $\{U_i\}$ обычно реализуются некоторым закономерным пересчетом целых чисел, нормируемых к интервалу $[0, 1)$ делением на максимально допустимое целое число (модуль датчика) либо умножением на обратную ему величину. Последний вариант работает несколько быстрее.

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

Популярна точка зрения, что при достаточном числе испытаний можно получить все требуемые результаты с *любой* точностью. Однако реальные датчики равномерных псевдослучайных чисел отнюдь не идеальны, и игнорировать это обстоятельство нельзя. Приведем, к примеру, наблюденную зависимость от числа испытаний N погрешности δ расчета числа π методом статистических испытаний через долю точек, попавших во вписанный в квадрат круг (кстати, это поучительный пример статистических испытаний *без имитации*).

Таблица 2.1. Погрешность расчета π методом Монте-Карло

N	δ	N	δ	N	δ
1 тыс.	9.8e-2	50 тыс.	2.8e-3	2 млн.	-1.4e-4
2 тыс.	8.6e-2	100 тыс.	8.3e-3	5 млн.	-1.0e-5
5 тыс.	5.6e-3	200 тыс.	4.3e-3	10 млн.	2.9e-4
10 тыс.	1.8e-2	500 тыс.	4.3e-3	20 млн.	4.2e-5
20 тыс.	2.2e-3	1 млн.	2.0e-3	50 млн.	1.2e-4

Как мы видим, монотонность убывания погрешности по числу испытаний отнюдь не гарантируется. Отметим также, что пик исследований по ДСЧ остался в прошлом — когда работали на гораздо более слабых ЭВМ и число испытаний вынужденно ограничивалось десятками тысяч. «Дальние» зоны работы датчиков исследованы плохо. Поэтому рекомендуется отлаживать модель на задаче с известным решением (здесь могут помочь численные методы теории очередей) и затем подобрать датчики и число испытаний, при которых имитация обнаруживает наилучшее согласие с эталоном. Другая альтернатива — исследование и использование продвинутых схем генерации равномерных псевдослучайных чисел.

2.2.3. Классические конгруентные генераторы

Чаще всего применяют линейные мультипликативные генераторы вида

$$X_k = (aX_{k-1}) \pmod M, \qquad k = 1, 2, \ldots$$

где все операнды — целые и для n-разрядных чисел $M = 2^n$. Конечный продукт $U_k = X_k/M$. Рекомендации по выбору параметров таких генераторов приводятся, например, в книге Д. Кнута.

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

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

Их частным видом являются *генераторы Фибоначчи*. В программе `UNI` принято

$$X_k = X_{k-17} - X_{k-5}.$$

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

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.

2.2.4. 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, номеров 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* не превосходит 30. Алгоритм применим к любым начальным значениям и множителям и (после очевидной модификации) — к любому модулю датчика.

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. Трудоемкость умножения вещественных чисел по отношению к присваиванию целых оценивается в 5 единиц, деление — в 9, а вычисление стандартных функций — в 150. По данным [Лоу], при генерации показательно распределенных чисел 72% времени занимает логарифмирование.

2.3. Метод обратной функции

2.3.1. Идея метода и точные обращения

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

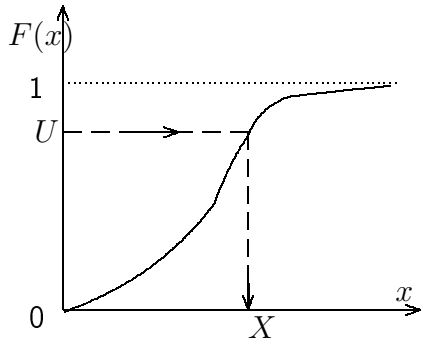

Рис. 2.1. Метод обратной функции

На рис. 2.1 показана его графическая реализация. Здесь U — случайное число, равномерно распределенное на интервале [0,1). Таким образом, из уравнения $F(X) = U$ определяется

$$X = F^{-1}(U). \tag{2.1}$$

Показательное распределение имеет ФР $F(x) = 1 - e^{-\lambda x}$, и (2.1) превращается в $1 - e^{-\lambda X} = U$, откуда $X = -\ln(1-U)/\lambda$. Поскольку $(1-U)$ имеет то же распределение, что и U, удобнее[1] применять формулу

$$X = -\ln U/\lambda. \tag{2.2}$$

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, например, случайное число U в нормализованном виде как $U = m \cdot 2^p$, можно аппроксимировать $\ln U$ произведением $-\ln 2(p - 2.6797 + 4.0391m - 1.3594m^2)$ с погрешностью, не превышающей по абсолютной величине 0.001. Вообще чем бóльшую часть времени моделирования составляет генерация случайных чисел и чем менее точны данные о параметрах

[1] Но с риском обращения к логарифмической функции с нулевым аргументом.

распределений, тем более оправданы наиболее простые и быстродействующие генераторы.

Распределение Релея с функцией распределения

$$F(x) = 1 - e^{-x^2/2\sigma^2} \tag{2.3}$$

приводит к генератору вида

$$X = \sigma\sqrt{-2\ln U}. \tag{2.4}$$

Распределение Вейбулла имеет функцию распределения

$$F(x) = 1 - \exp(-x^k/T). \tag{2.5}$$

Соответственно оказывается, что

$$X = \sqrt[k]{-T\ln(1-U)}. \tag{2.6}$$

Логистическое распределение общего вида имеет ФР

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

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

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

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

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

Обширные таблицы способов генерации различных распределений приведены в [Вадзинский, Лоу]. Метод обратной функции — единственный истинно универсальный метод генерации случайных чисел с требуемым распределением.

Для *численного* обращения применяются методы половинного деления, а также аппроксимации обратных функций распределения. Решение уравнений вида (2.1) для получения каждого нового числа X (а их нужны десятки и сотни тысяч) требует больших затрат машинного времени, в особенности если аналитическую формулу типа (2.2) получить не удается. Поэтому широко применяются приближенные табличные методы. Наиболее употребительный из них состоит в кусочно-линейной аппроксимации обратной функции распределения.

Основные затраты на реализацию линейных аппроксимаций связаны с поиском интервала опорных значений, в который попадает очередной аргумент U. Трудоемкость последовательного просмотра 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.

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;
- при аппроксимации многопараметрических семейств возникают трудности.

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. Последнее обстоятельство оправдывает применение для этих распределений в их стандартной форме более сложных аппроксимаций.

2.3.2. Дискретные распределения

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

$$F(X-1) = \sum_{i<X} p_i < U \leq \sum_{i \leq X} p_i. \tag{2.7}$$

При этом $\Pr(X = i) = F(i) - F(i-1) = p_i$. Метод неудобен для распределений с «толстыми хвостами». 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.

2.4. Частные методы

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

2.4.1. 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.

Эрлангово распределение с параметрами λ и r есть распределение суммы r чисел, получаемых согласно (2.2). Следовательно,

$$X = \sum_{k=1}^{r} \left(-\frac{1}{\lambda} \ln U_k \right) = -\frac{1}{\lambda} \sum_{k=1}^{r} \ln U_k = -\frac{1}{\lambda} \ln \prod_{k=1}^{r} U_k. \qquad (2.8)$$

Последний вариант уменьшает число сравнительно трудоемких операций логарифмирования.

Ниже приведена подпрограмма для моделирования эрлангова распределения длительности обслуживания:

```
 real*8 function flib()
   real*8  mu,x(3),p
   integer i
   data mu /5.0/
   call random_number(x)
   p=1.0
   do i=1,3; p=p*x(i); end do
   flib=-log(p)/mu
end
```

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. 2.3 по выбору дискретных случайных величин), а посредством U_2 формируется согласно (2.2) величина, имеющая показательное распределение с параметром λ_i.

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

Треугольное распределение дается формулой

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

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

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 $n > 6$.

Существует способ получения *пары* стандартных нормально распределенных чисел с помощью пары равномерных. Вспомним, что распределение Релея (2.3) описывает радиальное отклонение точки попадания от центра прицеливания при равной дисперсии нормальных отклонений по осям x и y. «Полярный» алгоритм Марсальи состоит в следующем:

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

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

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 $\xi = \sum_{i=1}^{F} \xi_i^2$, где числа $\{\xi_i\}$ суть стандартизованные нормальные. Аппроксимация (Фишера) этого распределения основана на предельном равенстве $\sqrt{2x} - \sqrt{2F-1} \to N(0,1)$. Нормальные числа генерируются по алгоритму Марсальи.

2.4.2. 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.

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

$$P\{K=k\} = q_k = \int_0^\infty \frac{(\lambda t)^k}{k\,!} e^{-\lambda t}\, dF(t).$$

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

2.5. Контрольные вопросы

1. Сопоставьте физические и программные ДСЧ.
2. Как обеспечить непересекаемость серий псевдослучайных чисел? Запрограммируйте функцию получения псевдослучайного числа с заданным номером.
3. При решении каких задач может оказаться полезной идея ускоренного умножения? Запрограммируйте одну из этих задач.
4. Примените метод обратной функции для генерации чисел с распределением Вейбулла.
5. Как использовать распределение Релея для генерации пары нормально распределенных чисел? Каков КПД этого метода?
6. Как сгенерировать случайное число событий простейшего потока за указанный интервал?

Глава 3. 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 (числом циклов) моделирования и точностью оценок.

3.1. 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:

$$\hat{x} = \frac{1}{N}\sum_{i=1}^{N} X_i. \tag{3.1}$$

Аналогичными формулами выражаются начальные статистические моменты более высокого порядка:

$$\hat{x}_k = \frac{1}{N}\sum_{i=1}^{N} X_i^k, \qquad k = 1, 2, \ldots \tag{3.2}$$

Эмпирический контроль сходимости оценок моментов с ростом числа испытаний позволяет убедиться в *существовании* моментов (у распределения Коши все моменты бесконечны).

Многие современные ЭВМ вместо правильного округления младшего из сохраняемых разрядов *отбрасывают* биты, выходящие за разрядную сетку. При суммировании большого количества чисел одного знака это приводит к заметному дрейфу результатов. Для нейтрализации указанного эффекта можно воспользоваться одним из следующих приемов:

- работать в формате двойного слова;
- 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 знакопеременными, 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

$$\hat{D} = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \hat{x})^2. \tag{3.3}$$

Статистическое среднеквадратическое отклонение

$$\hat{s} = \sqrt{\hat{D}}. \tag{3.4}$$

Формирование в процессе моделирования больших массивов необработанных результатов создает проблему их хранения. В частности, для получения оценки статистической дисперсии согласно (3.3) 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

$$\hat{x}_N = \frac{1}{N}\sum_{i=1}^{N} X_i = \frac{N-1}{N}\hat{x}_{N-1} + \frac{1}{N}X_N. \tag{3.5}$$

Можно показать, что

$$\hat{D}_N \;=\; \frac{N-2}{N-1}\hat{D}_{N-1} + \frac{1}{N}(X_N - \hat{x}_{N-1})^2. \tag{3.6}$$

Теперь ясно, что достаточно накапливать и хранить только *суммы* $\{X_i\}$ и их квадратов. Статистические оценки среднего можно вычислять согласно (3.5), а дисперсии — по формуле (3.6). Применение этих формул дает возможность оперативного завершения счета при достижении заданной точности и надежности оценок, определяемых текущими значениями $\hat{x}_N$ и $\hat{D}_N$.

Оценка корреляционного момента системы двух случайных величин X и Y

$$\hat{K}_{XY} = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \hat{x})(Y_i - \hat{y}), \tag{3.7}$$

а их коэффициента корреляции —

$$\hat{\rho}_{XY} = \hat{K}_{XY}/(\hat{s}_X\hat{s}_Y). \tag{3.8}$$

Оценкой вероятности состояния r служит частота его появления

$$\hat{p}_r = N_r/N, \tag{3.9}$$

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:

$$\hat{p}_r{}' = T_r/T. \tag{3.10}$$

Выборочные средние из-за влияния начальных условий будут, вообще говоря, *смещенными* оценками 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; 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* — следовательно, теоретически при существовании стационарного режима и 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 [Корн], что дисперсия оценки $\hat{\nu}_r$ r-го начального момента

$$D[\hat{\nu}_r] = (\nu_{2r} - \nu_r^2)/N \tag{3.11}$$

(в правую часть входят теоретические значения соответствующих моментов). В частности, дисперсия статистического среднего

$$D[\hat{\nu}_1] = D_X/N. \tag{3.12}$$

Воспользуемся этой формулой для вычисления дисперсии оценки $\hat{v}$ среднего времени пребывания заявки в системе $M/M/1$, которое имеет показательное распределение с параметром $(\mu - \lambda)$. 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, что *усреднение по системному времени дает оценки с меньшей дисперсией, чем усреднение по числу наблюдений.*

3.2. 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 ν; однако выборочная дисперсия $D(n)$ более не является несмещенной. Можно доказать [Лоу], что

$$D(n) = D\Big[1 - \frac{2}{n-1}\sum_{j=1}^{n-1}(1 - j/n)\rho_j\Big].$$

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

$$D[\bar{X}(n)] = \frac{D}{n}\Big[1 + 2\sum_{j=1}^{n-1}(1 - j/n)\rho_j\Big].$$

3.3. 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 (3.12). Следовательно, доверительная вероятность

$$P\{|\hat{w} - w| \le \delta\} = \Phi\left(\frac{\delta\sqrt{N}}{\sigma_W\sqrt{2}}\right), \tag{3.13}$$

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

$$N \geq 2[\Phi^{-1}(P)]^2 \left(\sigma_W/\delta\right)^2 = k(P)(\sigma_W/\delta)^2. \qquad (3.14)$$

Коэффициент $k(P)$ выбирается из таблицы 3.1.

Таблица 3.1. 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 заметно возрастает с повышением доверительной вероятности.

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 (3.5), (3.6). Возможен также двухэтапный эксперимент: на первом этапе грубо определяется оценка $\hat{\sigma}_W$, после чего согласно (3.14) 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.

При оценке вероятностей в формулу (3.14) вместо σ следует подставить $\sqrt{p(1-p)}$. Тогда формула (3.14) примет вид

$$N \geq k(P)\frac{p(1-p)}{\delta^2}. \qquad (3.15)$$

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 (3.15) 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.

3.4. 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. 3.2) соответствующие экспериментальные данные, полученные автором для модели $M/D/2$.

Таблица 3.2. 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. Разобьем период моделирования на *циклы*, состоящие из смежных периодов занятости и простоя. Началом цикла будем считать момент прихода заявки в свободную систему. 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,

$n = \sum_{i=1}^{N} \nu_i$ — общее число обслуженных заявок.

Тогда оценка среднего времени ожидания

$$\hat{w} = \frac{W_1 + W_2 + \ldots W_N}{n} = \frac{(W_1 + W_2 + \ldots W_N)/N}{(\nu_1 + \nu_2 + \ldots \nu_N)/N}. \qquad (3.16)$$

При $N \to \infty$ в силу закона больших чисел числитель и знаменатель стремятся к своим математическим ожиданиям, так что

$$M[\hat{w}] = M[W]/M[\nu]. \qquad (3.17)$$

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 (3.16).

Для процессов с непрерывным временем считают

$$Y_j = \int_{\tau_j}^{\tau_{j+1}} f(X(t))\, dt,$$

где $\{\tau_j\}$ — уже не номера, а сами *точки* регенерации, а $\{\nu_j\}$ из формулы (3.16) суть продолжительности соответствующих циклов.

В книгах [Крейн, Иглхарт] описана процедура построения 100(1-δ)–процентного доверительного интервала для $r = M[f(x)]$ при числе циклов N порядка сотен и более.

Решающими для применения описанной методики являются следующие требования:

- процесс неоднократно возвращается в некоторое фиксированное состояние (подмножество состояний);
- среднее время между возвращениями конечно;

- 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.

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 практически некоррелированными. Этот выигрыш в «чистоте эксперимента» приходится оплачивать: кратность замедления набора статистики 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.

С другой стороны, его несомненными достоинствами являются:

- независимость наблюдений в различных циклах, позволяющая проводить корректную статистическую обработку результатов;
- отсутствие проблемы начальных условий.

3.5. Контрольные вопросы

1. Какими свойствами должны обладать статистические оценки? Дайте формальное определение этих свойств.
2. Приведите формулы для точечных статистических оценок среднего и дисперсии.
3. Получите рекуррентную формулу для оценки среднего.
4. Чем характеризуются точность и надежность статистических оценок? Дайте определение доверительного интервала и доверительной вероятности. Как связаны эти понятия при фиксированном числе испытаний?
5. Как требуемое число испытаний связано с определяемой малой вероятностью и допустимой относительной погрешностью оценки?
6. Дайте характеристику метода регенерирующих процессов.

Глава 4.
Понижение дисперсии

4.1. 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 относительной 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. Еще фон Нейман и Улам в рамках Манхэттенского проекта при моделировании рассеивания нейтронов отказались от принципа «грубой силы» и применяли методы понижения дисперсии (МПД) результатов *единичного наблюдения*. Более развитые формы МПД предложены и описаны достаточно давно — см. [Клейнен, Лоу, Рыжиков], 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.

К МПД-методам относят:

- значимые выборки,
- выбор оценок с наименьшей дисперсией,
- 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 МПД (бутстреп-метод), не привлекающие таких знаний, могут «отметаться с порога» — по аналогии с проектами «вечных двигателей».

4.2. Значимые выборки

Основная идея этого приема состоит в эквивалентном преобразовании задачи путем замены распределения исходной случайной вели-

чины. Пусть мы хотим оценить

$$y = \int\limits_0^\infty h(x) f(x)\, dx, \tag{4.1}$$

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

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} h(X_i). \tag{4.2}$$

Теперь перепишем исходную зависимость в форме

$$y = \int\limits_0^\infty \frac{h(x) f(x)}{g(x)} g(x)\, dx,$$

где $g(x)$ — другая плотность вероятностей, и обозначим

$$\tilde{h}(x) = \frac{h(x) f(x)}{g(x)}. \tag{4.3}$$

Нетрудно видеть, что

$$M[\tilde{h}(x)] = \int\limits_0^\infty \tilde{h}(x) g(x)\, dx = y,$$

так что оценка искомой величины

$$\hat{y}' = \frac{1}{N} \sum_{i=1}^{N} \tilde{h}(\tilde{X}_i) \tag{4.4}$$

при $\{\tilde{X}_i\}$, сформированных в соответствии с плотностью $g(x)$, является несмещенной.

Известно [Клейнен], что дисперсия оценки (4.4) минимальна при выборе в качестве новой плотности

$$g^*(x) = \frac{h(x) f(x)}{\int\limits_0^\infty h(x) f(x)\, dx} = \frac{h(x) f(x)}{y}. \tag{4.5}$$

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 (4.5) оценка (4.4) теоретически имеет *нулевую* дисперсию.

Трудность реализации данного подхода состоит в том, что знаменатель рекомендуемой плотности (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 (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

$$\begin{aligned} c_1 &= (N/D)\Big/\Big(\sum_{i=1}^{2} N_i/\hat{D}_i\Big), \\ c_2 &= 1 - c_1. \end{aligned} \tag{4.6}$$

Усложняются также расчет значений целевой функции [формула (4.3) вместо $h(x)$] и генерация случайных чисел в соответствии с более сложным выражением для плотности.

4.3. 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

$$\hat{p}_k = \sum_i t_i(k)/T, \tag{4.7}$$

чем согласно

$$\hat{p}_k{}' = N(k)/N. \tag{4.8}$$

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.

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.

Выполним сравнение дисперсий времени ожидания в системе $M/M/1$ при прямой имитации и через формулу Литтла. Согласно формуле Полячека—Хинчина, 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

$$w_2 = \frac{\lambda}{1-\rho}\left[\frac{2\,!}{0\,!3\,!}b_3 w_0 + \frac{2}{1\,!2\,!}b_2 w_1\right] = \frac{2\rho}{\mu^2(1-\rho)^2}. \tag{4.9}$$

Следовательно, дисперсия результатов наблюдения при прямой имитации

$$D_W = w_2 - w_1^2 = \frac{\rho(2-\rho)}{\mu^2(1-\rho)^2}. \tag{4.10}$$

Через формулу Литтла находим математическое ожидание *длины очереди*

$$q_1 = \lambda w_1 = \frac{\rho^2}{1-\rho}. \tag{4.11}$$

Математическое ожидание ее квадрата

$$q_2 = \sum_{k=1}^{\infty}(k-1)^2 p_k = \sum_{k=1}^{\infty}(k^2-2k+1)(1-\rho)\rho^k$$

$$= (1-\rho)\Big[\sum_{k=1}^{\infty} k^2\rho^k - 2\sum_{k=1}^{\infty} k\rho^k + \sum_{k=1}^{\infty}\rho^k\Big].$$

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. 4.1. Очевидно, что реальный (и не слишком существенный) выигрыш от подобного пересчета возможен только при умеренной загрузке.

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. Из табл. 4.1 следует, что реальный выигрыш от подобного пересчета возможен только при умеренной загрузке.

Таблица 4.1. 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

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.

Оценка среднего времени пребывания заявки в системе

$$\hat{v} = \hat{w} + \hat{b}, \tag{4.12}$$

где

$\hat{w}$ — оценка среднего времени ожидания обслуживания,

$\hat{b}$ — оценка средней деятельности обслуживания.

Дисперсия нашей оценки

$$D[\hat{v}] = D[\hat{w}] + D[\hat{b}], \tag{4.13}$$

поскольку времена ожидания и обслуживания статистически независимы. Заменим оценку $\hat{b}$ в (4.12) априорно известным математическим ожиданием ее:

$$\bar{v} = \hat{w} + b. \tag{4.14}$$

Дисперсия постоянной величины b равна нулю[1]; следовательно, оценка (4.14) будет иметь меньшую дисперсию, чем (4.12). 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. 3.1).

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.

Итак, для определения средних характеристик СМО посредством имитационного моделирования целесообразно оценить среднее время ожидания $\hat{w}$. Остальные оценки вычисляются по формулам (4.14), (4.9), 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 заключение отметим, что расчет средней длины очереди через среднее время ожидания заявки требует моделирования очереди: запоминания моментов прибытия заявок, их продвижения и т. п. При прямом расчете во всем этом нет необходимости. Теперь ясно, чем оплачивается повышение точности оценок через временные показатели.

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

$$M[T_I] = a_1 = 1/\lambda.$$

4.4. «Противоположные» выборки

Проведем два *зависимых* опыта и возьмем в качестве оценки искомой величины полусумму их результатов

$$\tilde{X} = (X_1 + X_2)/2. \tag{4.15}$$

Дисперсия этой оценки

$$D[\tilde{X}] = (D[X_1] + D[X_2] + 2K_{X_1X_2})/4. \tag{4.16}$$

Если принять $D[X_1] = D[X_2]$, а этого легко добиться соответствующей организацией эксперимента, то

$$D[\tilde{X}] = D_X(1+\rho)/2, \tag{4.17}$$

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 псевдослучайных чисел можно встроить вспомогательную переменную — «кувыркающуюся единицу», знак которой определяет, какое из упомянутых действий должно быть выполнено при текущем 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].

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.

При совместном использовании в одном прогоне $\{U_i\}$ и их дополнений до единицы можно в полтора раза уменьшить число обращений к равномерному датчику. Однако трудоемкое вычисление обратной функции распределения необходимо в обоих случаях, так что выигрыш будет намного меньше. К тому же здесь есть еще одна проблема, обсуждаемая ниже.

Для пары *прогонов* можно использовать дополнение начальной установки до модуля датчика — получаемые числа $\{U_i\}$ (проверено!) будут тождественными.

Известно обобщение метода дополняющих переменных, в котором проводится m параллельных опытов со случайными аргументами и

$$X_j = F^{-1}(U \cdot m/j \pmod 1), \qquad j = \overline{1, m}.$$

К сожалению, последующие нелинейные функциональные преобразования могут «смягчить» запланированную противоположность. В

[2]Рекомендуем проверить на вашей персональной ЭВМ.

общем случае корреляционный момент последовательных интервалов

$$K = \int_0^1 (F^{-1}(u) - \bar{x})(F^{-1}(1-u) - \bar{x})\, du. \qquad (4.18)$$

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 (4.18) 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 2.3 выполним соответствующие расчеты для распределения Вейбулла, полагая для упрощения обозначений масштабный параметр $T = 1$. Теперь (4.18) переходит 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. 4.2.

Таблица 4.2. 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
$1+\rho$	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. Можно предположить, что данный метод будет хорошо работать для любых модальных распределений.

Моделирование дало -0.6445 для показательного закона, -0.9475 для закона Релея и практически -1 — для треугольного закона, что и предсказывалось. Заметим, что преобразование нелинейно для всех перечисленных законов, в том числе для треугольного. Следовательно, сохранению -1 мешает не нелинейность, а *асимметрия* распределения.

У [Лоу] обсуждается также способ преодоления упомянутой нели-

[3]Формула (4.17) предполагает проведение двух опытов.

нейности 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 4.3 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. В этой таблице:

- вариант 1 — результаты стандартного моделирования;
- варианты 2, 3 и 4 — полусуммы с результатами $2\bar{x} - X$, дополняющей начальной установкой и перекоммутацией датчиков соответственно;
- варианты 5 и 6 — результаты «мультиметодов» второго и четвертого порядка.

Обращает на себя внимание очень медленная сходимость «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.

Таблица 4.3. Метод дополняющих ДСЧ

тыс. испыт.	Варианты					
	1	2	3	4	5	6
1	6.8802	6.5509	6.8096	6.0513	7.2137	6.6919
2	7.1234	6.5306	6.6615	7.7375	7.8932	9.6559
5	7.7685	7.0940	6.9470	7.6385	7.9688	8.3496
10	7.8029	7.4005	7.8252	7.4327	7.5717	7.9865
20	8.8405	9.1451	8.7248	8.5933	8.4641	8.2747
50	8.9950	9.2125	8.5911	8.7299	8.8858	8.5966
100	9.3790	9.1427	8.9333	9.2473	9.2571	8.8398
200	9.0235	8.9452	8.7922	9.0156	9.1121	8.7878
500	8.9838	9.0323	8.6549	9.0760	8.8710	8.7661
1000	8.8632	9.0356	8.8701	8.8715	8.8435	8.7463
2000	8.8196	8.9283	8.7768	8.8938	8.8987	8.8131
5000	9.0661	9.0026	8.9039	9.0287	9.0601	8.9424
10000	8.9473	8.9241	8.8485	8.9549	8.9366	8.8382

Усреднение результатов (случаи с дополняющими и перекрестными датчиками, в особенности варианты 2 и 4) дает заметно лучшие результаты. Эффект усреднения вполне окупает удвоение общего числа испытаний, в чем легко убедиться по следующей строке для варианта 1 той же таблицы. «Мультиметоды» дают весьма неустойчивую картину, увеличение m не гарантирует повышение точности.

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

$$D = \frac{1}{4}\Big[\int_0^1 (-\ln u - 1)^2\,du + \int_0^1 (-\ln(1-u) - 1)^2\,du\Big] = \frac{1}{4}[1+1] = 1/2$$

(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. Со-

гласно формуле Полячека—Хинчина среднее время ожидания уменьшится 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. (вопрос о начальных установках нами уже обсуждался);
- 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.

4.5. Контрольные переменные

Здесь мы ограничимся расчетной схемой для одномерного случая. Пусть X — случайная величина с неизвестным математическим ожиданием $M[X] = x$ и $\hat{x}$ — его несмещенная оценка, найденная стандартными методами главы 3. Случайная переменная Y может быть выбрана в качестве *контрольной переменной* для X, если она коррелирована с X и имеет известное математическое ожидание y.

Получим с помощью контрольной переменной несмещенную оценку x с дисперсией меньшей, чем у $\hat{x}$. Введем новую случайную вели-

чину $X(c) = X - c(Y - y)$. Для любой константы c выражение

$$\bar{x}(c) = \hat{x} - M[c(Y - y)] \tag{4.19}$$

также даст несмещенную оценку x, поскольку $M[Y - y] = 0$. Дисперсия контролируемой случайной величины

$$D[X(c)] = D[X - cY] = D[X] - 2cK_{XY} + c^2 D[Y], \tag{4.20}$$

если $c > 0$ и корреляционный момент $K_{XY} > cD[Y]/2$, будет меньше $D[X]$. Приравнивая нулю производную правой части (4.20) по c, получаем оптимальное значение поправочного коэффициента

$$c^* = K_{XY}/D[Y]. \tag{4.21}$$

Минимальная дисперсия $D[X(c)]$ достигается при подстановке c^* и равна

$$\begin{aligned} D^* &= D[X] - 2K_{XY}^2/D[Y] + K_{XY}^2/D[Y] = D[X] - K_{XY}^2/D[Y] \\ &= D[X](1 - K_{XY}^2/(D[X]D[Y])) = D[X](1 - \rho_{XY}^2). \end{aligned} \tag{4.22}$$

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 $(W - \hat{w})(Q - q)$ 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. Делать это можно, 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.

Таблица 4.4. Метод контрольных переменных

Тыс. испыт.	w	q	c*	w*	f
1	6.8802	6.1090	0.5857	8.0464	0.6918
2	7.1234	6.3740	0.7486	8.4155	0.4875
5	7.7685	6.9476	0.8320	8.7273	0.3844
10	7.8029	6.9692	0.8756	8.7931	0.3380
20	8.8405	7.9416	0.9349	8.9886	0.2160
50	8.9950	8.1342	0.9114	8.9638	0.2468
100	9.3790	8.5053	0.9227	9.0050	0.2265
200	9.0235	8.1561	0.9169	8.9723	0.2403
500	8.9838	8.0903	0.9205	8.9927	0.2451
1000	8.8632	7.9816	0.9176	8.9718	0.2517
2000	8.8196	7.9392	0.9169	8.9670	0.2521
5000	9.0661	8.1893	0.9255	8.9834	0.2357
10000	8.9473	8.1378	0.9388	8.9119	0.2338

Результаты применения описанного подхода (табл. 4.4), который можно считать «самым умным», оказались заметно точнее остальных — в особенности при умеренном числе испытаний. Уже при N=10 тыс. испытаний погрешность среднего времени ожидания составила около 2%. Выигрыш f в уменьшении дисперсии асимптотически оказывается четырехкратным.

Поправочный коэффициент c^* с увеличением N очень быстро стабилизируется. Следовательно, он может быть определен по умеренному числу испытаний. Кроме того, с ростом N уменьшается и поправочная разность $\hat{q}-q$ (напомним, что в нашем примере $q = 8.1$). Учет обоих этих обстоятельств позволяет считать, что метод контрольных переменных позволяет ограничиться небольшим N.

Другим вариантом метода контрольных переменных является переход к случайной величине

$$X(b) = X - b(Y - Z), \tag{4.23}$$

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 (4.21) и (4.22), если заменить контрольную переменную Y из (4.19) на $\Delta = Y - X$.

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 4.5. 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, 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 [Лоу] средней из 100 предыдущих задержек.

Внешние контрольные переменные формируются при дополнительном моделировании упрощенного варианта исходной системы.

Выбор в качестве контрольной переменной аналогичной искомой величине и применение при ее формировании тех же случайныех чисел, что в основной модели, гарантируют достаточно высокую корреляцию откликов. Для упрощенного варианта можно (например, численными методами) рассчитать более широкий круг характеристик, чем перечисленные в табл. 4.5. В этих целях применимы известные инварианты ТМО (например, независимость распределения числа заявок в очереди от типа консервативной приоритетной дисциплины). Однако полученный выигрыш должен окупить затраты на параллельное моделирование, тогда как при использовании внутренних контрольных переменных он достается практически «бесплатно».

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.

4.6. 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

$$\tilde{X} = X_1 - X_2. \tag{4.24}$$

Ее дисперсия

$$D[\tilde{X}] = D[\tilde{X}_1] + D[\tilde{X}_2] - 2K_{X_1X_2}$$

в случае близких дисперсий составляющих равна

$$D[\tilde{X}] \approx 2D[X] \cdot (1 - \rho). \tag{4.25}$$

Отсюда вытекает очевидная практическая рекомендация: использовать в параллельных турах одни и те же серии случайных чисел $\{U_i\}$ (разумеется, каждую по своему назначению).

Поскольку данный метод рекомендуется для сравнения *различных* систем, проблема синхронизации серий случайных чисел здесь может осложниться.

4.7. Условная имитация

Условная имитация, как и контрольные переменные, комбинирует имитацию и дополнительные знания о моделируемом процессе, но является гораздо более гибким инструментом понижения дисперсии.

4.7.1. 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.

Применительно к интересующей нас задаче проведем расслоение выборки имитируемых времен ожидания по значениям определяющих это ожидание факторов и запишем условное математическое ожидание $\bar{w}$ через найденные в слоях $\{w_k\}$:

$$\bar{w} = \sum_k p_k w_k,$$

где $\{p_k\}$ — вероятности принадлежности результата к k-му слою.

Применяя к его слагаемым формулу для дисперсии произведения независимых случайных величин, имеем для дисперсии искомой оценки

$$\begin{aligned} D[\bar{w}] &= \sum_k \{D[p_k]D[w_k] + p_k^2 D[w_k] + w_k^2 D[p_k]\} \\ &= \sum_k p_k^2 D[w_k] + \sum_k D[p_k](w_k^2 + D[w_k]). \end{aligned}$$

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 ма-

тематического эксперимента: определить (предпочтительно аналитически) вероятности $\{p_k\}$ и — отдельно для каждого слоя — получить имитацией первые и вторые моменты времени ожидания.

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.

4.7.2. Общий случай

Пусть искомая характеристика СМО является математическим ожиданием функции двух независимых случайных величин X и Y:

$$z = \int_0^\infty \int_0^\infty g(x,y) f_X(x) f_Y(y)\, dx\, dy, \tag{4.26}$$

причем аналитический вид $g(x,y)$ неизвестен. Зафиксируем значение одного из случайных факторов и получим с помощью имитационной

модели *условный* отклик

$$\hat{G}(y) = \int_0^\infty g(x,y) f_X(x)\, dx. \tag{4.27}$$

Тогда конечный результат можно получить численным интегрированием согласно

$$\hat{z} = \int_0^\infty G(y) f_Y(y)\, dy. \tag{4.28}$$

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

$$w = \frac{\lambda b_2}{2(1 - \lambda b_1)}, \tag{4.29}$$

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

$$\bar{w} = \frac{1}{12}(w_1 + 4w_2 + 2w_3 + 4w_4 + w_5)$$

(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.

4.7.3. Определение вероятностей редких событий

Условная имитация является единственным средством получения достаточно точных оценок вероятностей редких событий (например, 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. Попытаемся оценить на примере потенциальный выигрыш в числе испытаний.

Пусть искомая вероятность

$$\pi = \prod_{i=1}^{3} p_i, \tag{4.30}$$

причем составляющие $\{p_i\}$ имеют порядок 0.01. Тогда сама π имеет порядок 10^{-6}, и для ее *прямого* определения с относительной

точностью 10^{-1} требуется $k/(10^{-6} \cdot (10^{-1})^2) = k \cdot 10^8$ испытаний, где коэффициент k — обсуждавшаяся выше функция доверительной вероятности.

При реализации формулы (4.30) относительные погрешности вероятностей $\{p_i\}$ будут складываться, и для расчета каждой из них потребуется $k/(10^{-2} \cdot (10^{-1}/3)^2) = k \cdot 9 \cdot 10^4$ испытаний. Выигрыш по числу испытаний будет в $10^4/(3 \cdot 9) \approx 370$ раз (для сохранения *доверительной вероятности* придется пойти на некоторое его уменьшение).

В целом для успеха условной имитации требуется больше знаний о моделируемом процессе и методологии моделирования, а также умений ими воспользоваться.

4.8. Контрольные вопросы

1. В чем состоит общая идея МПД? Почему они сохраняют актуальность, несмотря на рост быстродействия ЭВМ?
2. Расскажите о выборе оценок с наименьшей дисперсией. Какие факторы влияют на эффективность этого метода?
3. Опишите и сопоставьте варианты метода противоположных (дополняющих) выборок.
4. Опишите идею и расчетную схему метода контрольных переменных. Почему при его применении не требуется большого числа испытаний?
5. Перечислите варианты выбора контрольных переменных.
6. Когда применяются параллельные выборки?
7. Опишите идею условной имитации применительно к расслоенным выборкам.
8. Опишите идею условной имитации в комбинации с численным интегрированием. Какое дополнительное требование предъявляется к моделируемому процессу в этом случае?

9. Как применить условную имитацию к определению вероятностей редких событий? Опишите методику оценки выигрыша в числе испытаний.

Глава 5.
Введение в GPSS

5.1. Общая характеристика

5.1.1. 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). 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.

В книге Дж. Шрайбера рассмотрена история создания *GPSS*. 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 учебные пособия Б.Я. Советова и С.А. Яковлева [12, 13]. Запись в *GPSS* типовых модели теории очередей обсуждается в главе 16 [10].

5.1.2. 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 об ожидании или контролируется время прохождения, и т. п. Некоторые из названных объектов должны быть предварительно описаны специальными командами. Параметры транзакта на траектории могут менять свои значения, оказывать влияние на маршрут и собираемую статистику.

Каждая из названных функций имеет специальное графическое изображение. Латинские буквы означают последовательные операнды блоков. Многие блоки являются парными (сопряженными). Соответственно их графические представления суть взаимные зеркальные отражения. Из этих блоков составляется схема $GPSS$-модели, описывающая траекторию транзактов (рис. 5.1). Разработчик модели должен так подобрать последовательность блоков и команд, чтобы поведение транзактов соответствовало работе реальной или предлагаемой системы. 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) $GPSS$.

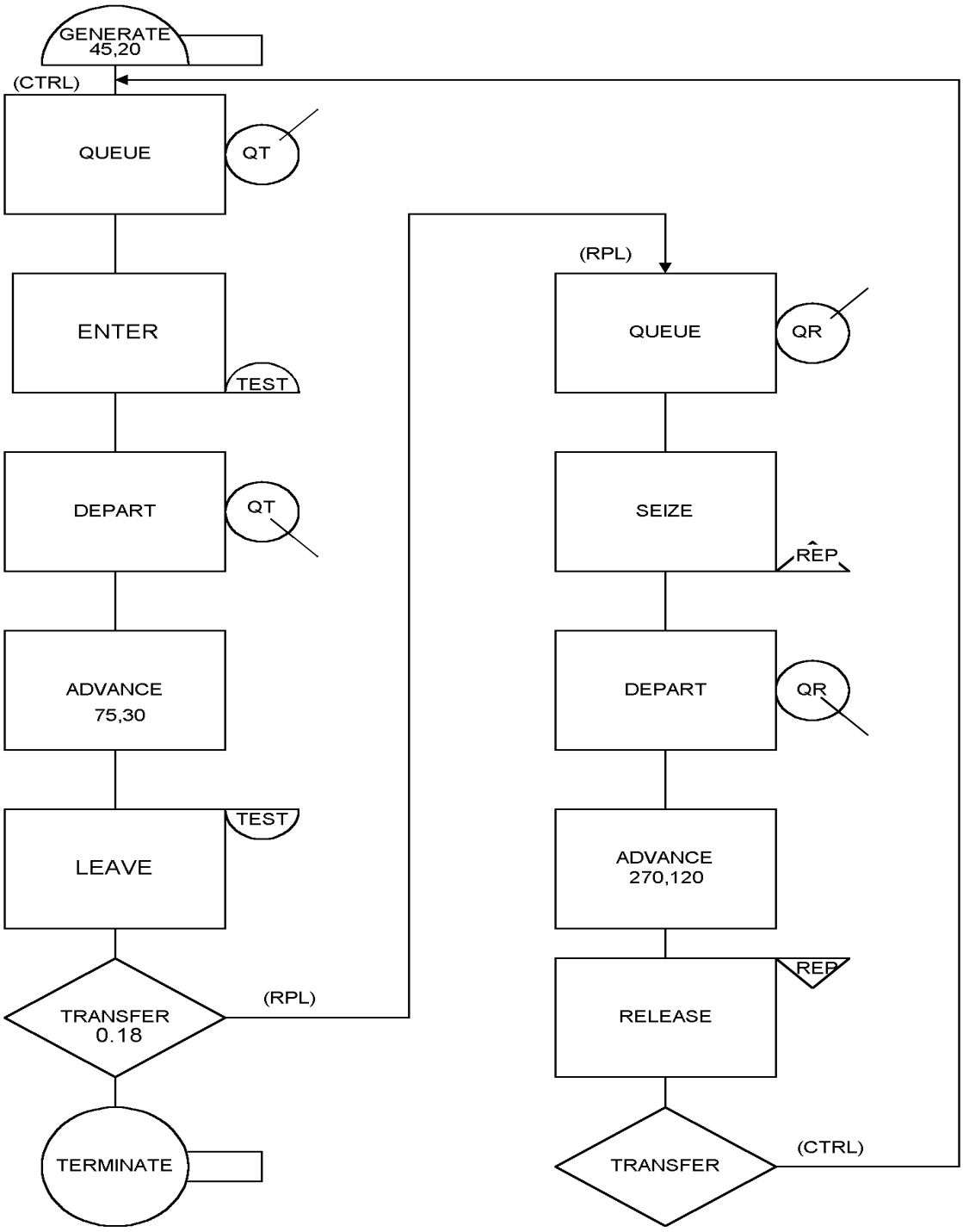

Рис. 5.1. Схема GPSS-модели

5.2. 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, 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). Совместимость с *GPSS/W* программ, ранее составленных на *GPSS/PC*, обеспечивает только коммерческая версия.

Принципиальные особенности *GPSS/W* можно сгруппировать в несколько разделов:

1. Каждая разработка в духе современных технологий программирования рассматривается как *проект* [1]. Проект состоит из объектов четырех типов: исходный модуль, Имитация (продукт компиляции), отчеты и текстовые файлы. Модель может иметь иерархическое строение — включать в себя командой `INCLUDE` ранее разработанные фрагменты. В рамках проекта может существовать множество автоматически нумеруемых версий. Размеры задачи физически доступным объемом оперативной памяти не лимитируются.

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.

В системе реализованы элементы объектного программирования в классическом его понимании: *наследование свойств* (установок) Имитации от исходной модели и *полиморфизм* — автоматическое преобразование операндов выражений к требуемому типу.

2. Интерфейс ориентирован на пользователя. Текстовый полноэкранный редактор позволяет набирать модели 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 тради-

[1] Эта терминология в электронной документации к системе не используется.

ционно 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. Степень загруженности ресурсов показывается цветом, можно наблюдать за перемещением активного транзакта. В окна можно выводить СЧА и накопленную статистику. 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. 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. 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. 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.

5.3. Среда

Меню. Главное окно системы (рис. 5.2) включает в себя Заголовок, Меню, Панель инструментов и Поле клиента для набора текста модели.

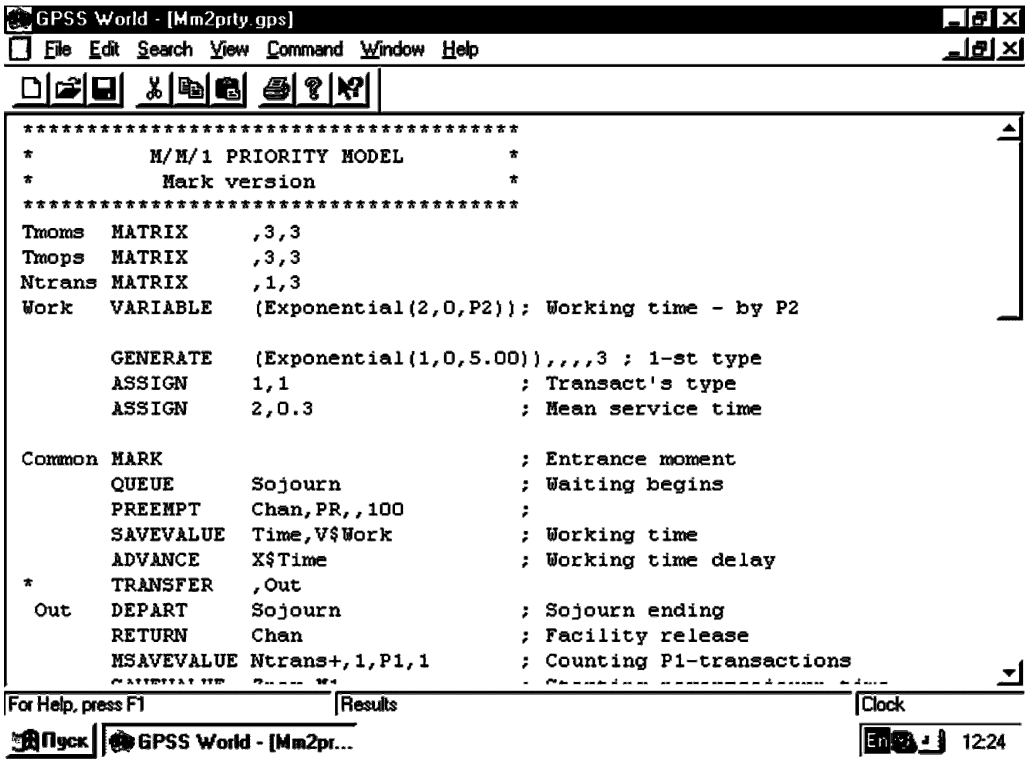

Рис. 5.2. Главное окно *GPSS World*

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. Многие объекты имеют собственные меню, построенные по общему принципу. Обычно организуется диалог, предлагающий кнопки и поля ввода.

Используя меню главного окна, можно создавать *объекты GPSS*: модели, Имитации, отчеты и тексты. Каждый из них можно рассматривать в соответствующих дочерних окнах. Типичный проект начинается с создания или модификации модели.

Все настройки задаются при создании модели через меню `Edit/Settings`. Последующие фазы проекта эти установки только наследу-

ют. Выборочно рассмотрим вкладки упомянутого меню:

Reports: `Standard Report` формирует типовой отчет; `In Windows` обеспечивает вывод результатов в окно вместо файла. `Save Plot Points` определяет объем данных, сохраняемых при построении графика (при недостаточном объеме график будет обрезан слева или справа).

Random Numbers: распределяет потоки случайных чисел.

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.

Окна. Динамические окна могут быть открыты только после создания Имитации (то-есть после компиляции модели) командой

`Window/Simulation Window/<тип_окна>`

Одновременно могут быть открыты несколько окон. Их относительное расположение управляется командами `Cascade` или `Tile`. 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. В дальнейшем нужно выполнять прогоны с закрытыми окнами до запланированного или аварийного останова и открывать динамические окна лишь для режима пошаговой отладки.

5.4. Этапы моделирования

Работа начинается с составления $GPSS$-программы. Посредством команды `INCLUDE "<имя_файла>"` в нее могут вставляться ранее отлаженные фрагменты (например, $PLUS$-процедуры). Вставки из файлов автоматически нумеруются, и диагностика ошибок в них сопровождается указанием номера вставки.

Компиляция организуется по `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`. 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
```

Наибольшие трудности вызывает использование некорректных имен — слишком коротких или совпадающих со стандартными числовыми атрибутами.

При успехе компиляции активируются интерактивные команды и становятся доступными окна имитации. Далее через меню `Command/Start` задается начальное значение счетчика, которое вместе с имеющим ненулевой декремент блоком `TERMINATE` определяет длительность моделирования. О запуске моделирования сообщает текст в статусной строке:

```
Simulation in Progress.
```

За его ходом можно следить по выведенным в статусную строку системным часам.

5.5. 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                    ; При успехе - 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 «отбывает» в нем (блок ADVANCE) 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

FACILITY  ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER  PEND INTER
RETRY
 CHAN      49843  0.896      0.899     1    49844    0
  0     0

QUEUE     MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-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 таб-

лица 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.

Содержание отчета зависит от состава модели и при необходимости корректируется через `Установки модели`. Нестандартные вычисления могут быть запрограммированы с помощью процедур встроенного языка программирования и выведены в файл.

5.6. 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). 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 `Plot` и `Expression`. Окно Таблиц позволяет наблюдать сходимость исследуемых распределений в смысле гистограммы, среднего значения и среднеквадратического отклонения.

5.7. 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. Еще более детальные сведения можно собирать и обрабатывать с помощью сохраняемых значений и $PLUS$-процедур.

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.

Перед проведением статистического анализа следует стереть всю ненужную информацию, то есть результаты пробных прогонов.

5.8. 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.

Типична управляющая 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.

«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.

GPSS/W записывает выполняемые операнды и сообщения о моделировании в *журнал сеанса*, который ведется в кодировке ASCII с накоплением записей в процессе моделирования. В нем отражаются трассировка транзактов, результаты работы команды ANOVA, сообщения об ошибках и т. д.

5.9. Тестирование GPSS World

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. 5.1.

Таблица 5.1. 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 [16] 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 обслуживания 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. Увеличение числа проведенных испытаний в общем приближает результаты к ожидаемым, но даже при 500 тыс. испытаний погрешность составляет около 17%.

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 распределения их равенство приближенно выполняется, однако погрешность самих значений составляет около 2%.

5.10. 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).

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; выражение коэффициента использования устройства в тысячных долях.

4. 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.

5. Однократность прерываний устройств и недопустимость прерываний для памятей, которые могут быть использованы для моделирования многоканальных устройств обслуживания и, следовательно, окажутся подвержены прерываниям.

6. Отсутствуют средства подбора параметров теоретических вероятностных распределений по заданным моментам.

7. Комментарии с кириллическими символами исключают правильную работу Имитации.

8. Невозможно непосредственно определить вектор. Нижняя граница индексов матрицы по любому измерению равна 1, что может нарушить естественность индексации (например, при расчете вероятностей состояний системы). Ненулевая инициализация матрицы требует отдельного оператора `INITIAL` на каждый элемент.

9. 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. Аргументом графиков может быть только время, так что распределение вероятностей состояний системы автоматически построить нельзя. Кстати, для такого графика обязательна логарифмическая шкала.

10. Оставляет желать лучшего электронная документация к системе. 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. В разделе определения табличных функций понятие обратной функции заменено кумулянтой. 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.

Литература

1. *Вадзинский Р.Н.* Справочник по вероятностным распределениям. СПб.: Наука, 2001. — 295 с.

2. *Кельтон В., Лоу А.* Имитационное моделирование /пер. с англ., 3-е изд. — СПб.: Питер, Киев, БХВ, 2004. — 847 с.

3. *Клейнен Дж.* Статистические методы в имитационном моделировании /пер. с англ. — М.: Статистика, 1978. — Вып.1, 221 с.; вып.2, 386 с.

4. *Кнут Д.* Искусство программирования для ЭВМ. Т. 2. Получисленные алгоритмы /пер. с англ. — М.: Мир, 1977. — 724 с.

5. *Конвей Р. В., Максвелл В. Л., Миллер Л. В.* Теория расписаний /пер. с англ. — М.: Наука, Физмат, 1975. — 360 с.

6. *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров /пер. с англ. — М.: Наука, 1973. — 831 с.

7. *Крейн М., Лемуан О.* Введение в регенеративный анализ моделей /пер. с англ. — М.: Наука, 1982. — 104 с.

8. *Петров А. А.* Экономика модели. Вычислительный эксперимент. — М.: Наука, 1996. — 250 с.

9. *Рыжиков Ю. И.* Имитационное моделирование систем массового обслуживания: учебн. пособие. — Л.: ВИККИ им. А.Ф. Можайского, 1991. — 111 с.

10. *Рыжиков Ю. И.* Имитационное моделирование. Теория и технологии. — СПб.: КОРОНА принт, 2004. — 380 с.

11. *Салеев В. М.* Библиотека алгоритмов и программ генерации и статистической обработки последовательностей случай-

ных чисел: материалы по программному обеспечению. — Минск: Ин-т математики АН БССР, 1986. — 138 с.

12. *Советов Б. Я., Яковлев С. А.* Моделирование систем. — М.: Высшая школа, 1998. — 320 с.

13. *Советов Б. Я., Яковлев С. А.* Моделирование систем: практикум. — М.: Высшая школа, 1999. — 224 с.

14. *Томашевский В. Н., Жданова Е. Г.* Имитационное моделирование в среде GPSS. — М.: Бестселлер, 2003. — 416 с.

15. *Шеннон Р.* Имитационное моделирование систем — искусство и наука /пер. с англ. — М.: Мир, 1978. — 418 с.

16. *Шрайбер Т. Дж.* Моделирование на GPSS /пер. с англ. — М.: Машиностроение, 1980. — 592 с.

17. *Bratley P., Fox B. L., Schrage L. E.* A Guide to Simulation. — N.Y.: Springer Verlag, 1983. — 383 pp.

18. *Devroye L.* Non-Uniform Random Variate Generation. — N.Y.: Springer Verlag, 1986. — 843 pp.

19. *Iglehart D. L., Shedler G. S.* Regenerative Simulation of Response Times in Networks of Queues. — Berlin: Springer Verlag, 1980. — 204 pp.