МИНОБРНАУКИ РФ

**ФГБОУ ВПО «УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ЛЕСОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И МОДЕЛИРОВАНИЯ

**Г.Л.Нохрина**

# ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ЭКОНОМИЧЕСКИХ ПРОЦЕССОВ

**Курс лекций**

ЕКАТЕРИНБУРГ 2013 г.

## СОДЕРЖАНИЕ

# ВВЕДЕНИЕ В ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ.

## *Математическое и компьютерное моделирование*

### Классификация видов моделирования

***Модель*** – это материальный или мысленно представляемый объект, который в процессе исследования замещает объект-оригинал, так что его непосредственное: изучение дает новые знания об объекте-оригинале.

Под *моделированием* понимают процесс построения, изучения и применения моделей. Оно является методом познания с помощью объектов-заместителей. Необходимость использования этого метода определяется тем, что многие объекты или проблемы непосредственно исследовать или совсем невозможно, когда объект недосягаем либо реально не существует (будущее состояние экономики), или же это исследование требует много времени и средств.


*Физическое* – используется сама система, либо подобная ей (летательный аппарат в аэродинамической трубе).

*Математическое* – процесс установления соответствия реальной *системе S* математической модели М и исследование этой модели, позволяющее получить характеристики реальной системы.

*Аналитическое* – процессы функционирования элементов записываются в виде математических соотношений (алгебраических, интегральных, дифференциальных, логических и т.д.). Аналитическая модель м.б. исследована методами: а) *аналитическим* (устанавливаются явные зависимости, получаются, в основном, аналитические решения); б) *численным* (получаются приближенные решения); в) *качественным* (в явном виде можно найти некоторые свойства решения).

*Компьютерное* – математическое моделирование формулируется в виде алгоритма (программы для ЭВМ), что позволяет проводить над ней вычислительные эксперименты.

*Численное* – используются методы вычислительной математики (отличается от численного аналитического тем, что возможно задание различных параметров модели).

*Статистическое* – обработка данных о системе (модели) с целью получения статистических характеристик системы.

*Имитационное* – воспроизведение на ЭВМ (имитация) процесса функционирования исследуемой системы, соблюдая логическую и временную последовательность протекания процессов, что позволяет узнать данные о состоянии системы или отдельных ее элементов в определенные моменты времени.

Применение *математического моделирования* позволяет исследовать объекты, реальные эксперименты над которыми затруднены или невозможны (дорого, опасно для здоровья, однократные процессы, невозможные из-за физических или временных ограничений – находятся далеко, еще или уже не существуют и т.п.).

Экономический эффект: затраты в среднем сокращаются в 10-100 раз.

# Математическое моделирование сложных систем

## Понятие сложной системы

*Элемент* s – некоторый объект, обладающий определенными свойствами, внутреннее строение которого для целей исследования не играет роли (самолет: для моделир. полета – не элемент, а для моделир. работы аэропорта –элемент).

*Связь l* между элементами – процесс их взаимодействия, важный для целей исследования.

*Система* S – совокупность элементов со связями и целью функционирования F.

*Сложная система* – состоящая из разнотипных элементов с разнотипными связями.

*Большая система* – состоящая из большого числа однотипных элементов с однотипными связями.

Система:

$$S = \{\{s\},\{l\},F\}$$

*Автоматизированная система* $S_A$ - сложная система с определяющей ролью элементов двух типов: технических средств (прежде всего ЭВМ) $s_T$ и действий человека $s_H$:

$$S_A = \{\{s_T\},\{s_H\},\{s_O\},\{l\},F\}$$

здесь $s_O$ - остальные элементы системы.

*Структура системы* – ее расчленение (декомпозиция) на элементы или группы элементов с указанием связей между ними, неизменное во время функционирования системы.

Практически все системы рассматриваются функционирующими во времени, поэтому определим их динамические характеристики.

*Состояние* – множество характеристик элементов системы, изменяющихся во времени и важных для целей функционирования.

*Процесс (динамика)* – множество значений состояний системы, изменяющихся во времени.

*Цель функционирования* – задача получения желаемого состояния системы. Достижение цели обычно влечет целенаправленное вмешательство в процесс функционирования системы, которое называется *управлением*.

Задачи исследования систем:

1. *анализ* – изучение свойств функционирования системы;
2. *синтез* – выбор структуры и параметров по заданным свойствам системы.

## Понятие математического моделирования

$T = [t_0, t_1]$ - временной интервал моделирования системы S (интервал модельного времени).

Построение модели начинается с определения параметров и переменных, определяющих процесс функционирования системы. *Параметры* системы $\theta_1, \theta_2, \ldots, \theta_m$ - характеристики системы, остающиеся постоянными на всем интервале Т. Если $\theta = (\theta_1,\ldots,\theta_m) \in \Theta \subseteq R^m$, то говорят, что имеется параметрическое семейство систем.

*Переменные* – зависимые и независимые.

Независимые:

*входные воздействия* (в т.ч. управляющие): $u = u(t) = (u_1(t),\ldots,u_{n_1}(t)) \in U \subseteq R^{n_1}$

воздействия внешней среды (контролируемые – неконтролируемые = наблюдаемые – ненаблюдаемые и детерминированные – случайные): $v = v(t) = (v_1(t),\ldots,v_{n_2}(t)) \in V \subseteq R^{n_2}$

*состояния системы*

$x = x(t) = (x_1(t),\ldots,x_{n_3}(t)) \in X \subseteq R^{n_3}$ Отличаются от $\theta$ тем, что характеризуют свойства системы, изменяю-

щиеся во времени. X – *пространство состояний* или *фазовое пространство*. Последовательность nguage: x для $t_1 < t_2 < \ldots < t_N$ называется *фазовой траекторией* системы. А последовательность y – *выходной траекторией*.

Зависимые – *выходные характеристики (сигналы)* $y = y(t) = (y_1(t),\ldots,y_{n_4}(t)) \in Y \subseteq R^{n_4}$

Общая схема ММ функционирования системы:


Множество переменных $u, v, \theta, x, y$ вместе с законами функционирования

$$x(t) = \ldots,$$
$$y(t) = \ldots$$

называется *математической моделью* системы.

Если t непрерывно, то модель называется *непрерывной*, иначе – *дискретной* $(t = i \cdot \Delta,\ i = 1,2,\ldots)$

Если модель не содержит случайных элементов, то она называется *детерминированной*, в противном случае – *вероятностной*.

Если математическое описание модели слишком сложное и частично или полностью неопределена, то в этом случае используются *агрегативные модели*. Сущность агрегативной модели заключается в разбиении системы на конечное число взаимосвязанных частей (подсистем), каждая из которых допускает стандартное математическое описание. Эти подсистемы называются *агрегатами*.

## *Понятие имитационного моделирования*

*Имитационное моделирование* (от англ. simulinion) – это распространенная разновидность аналогового моделирования, реализуемого с помощью набора математических инструментальных средств, специальных имитирующих компьютерных программ и технологий программирования, позволяющих посредством процессов-аналогов провести целенаправленное исследование структуры и функций реального сложного процесса в памяти компьютера в режиме “имитации”, выполнить оптимизацию некоторых его параметров.

*Имитационной моделью* называется специальный программный комплекс, который позволяет имитировать деятельность какого- либо сложного объекта. Он запускает в компьютере параллельные взаимодействующие вычислительные процессы, которые являются по своим временным параметрам (с точностью до масштабов времени и пространства) аналогами исследуемых процессов. В странах, занимающих лидирующее положение в создании новых компьютерных систем и технологий, научное направление (Сопзри1ег Бсіепсе) использует именно такую трактовку имитационного моделирования, а в программах магистерской подготовки по данному направлению имеется соответствующая учебная дисциплина.

Следует отметить, что любое моделирование имеет в своей методологической основе элементы имитации реальности с помощью какой-либо символики (математики) или аналогов. Поэтому иногда в российских вузах имитационным моделированием стали называть целенаправленные серии многовариантных расчетов, выполняемых на компьютере с применением экономико-математических моделей и методов. Однако с точки зрения компьютерных технологий такое моделирование (тое~е1Ипд) – это обычные вычисления, выполняемые с помощью расчетных программ или табличного процессора Ехсе1. Математические расчеты (в том числе табличные) можно производить и без ЭВМ: используя калькулятор, логарифмическую линейку, правила арифметических действий и вспомогательные таблицы. Но имитационное моделирование – это чисто компьютерная работа, которую невозможно выполнить подручными средствами. Поэтому часто для этого вида моделирования используется синоним *компьютерное моделирование.*

Имитационную модель нужно создавать. Для этого необходимо специальное программное обеспечение – *система моделирования* (япш1абоп вув~епз). Специфика такой системы определя-

ется технологией работы, набором языковых средств, сервисных программ и приемов моделирования.

Имитационная модель должна отражать большое число параметров, логику и закономерности поведения моделируемого объекта во времени *(временная динамика)* и в пространстве *(пространственная динамика)*. Моделирование объектов экономики связано с понятием *финансовой динамики* объекта. С точки зрения специалиста (информатика-экономиста, математика-программиста или экономиста-математика), *имитационное моделирование* контролируемого процесса или управляемого объекта – это высокоуровневая информационная технология, которая обеспечивает два вида действий, выполняемых с помощью компьютера:

1. работы по созданию или модификации имитационной модели;
2. эксплуатацию имитационной модели и интерпретацию результатов.

Имитационное (компьютерное) моделирование экономических процессов обычно применяется в двух случаях:

- для управления сложным бизнес-процессом, когда имитационная модель управляемого экономического объекта используется в качестве инструментального средства в контуре адаптивной системы управления, создаваемой на основе информационных (компьютерных) технологий;
- при проведении экспериментов с дискретно-непрерывными моделями сложных экономических объектов для получения и отслеживания их динамики в экстренных ситуациях, связанных с рисками, натурное моделирование которых нежелательно или невозможно.

Можно выделить следующие типовые задачи, решаемые средствами имитационного моделирования при управлении экономическими объектами:

- моделирование процессов логистики для определения временных и стоимостных параметров;
- управление процессом реализации инвестиционного проекта на различных этапах его жизненного цикла с учетом возможных рисков и тактики выделения денежных сумм;
- анализ клиринговых процессов в работе сети кредитных организаций (в том числе применение к процессам взаимозачетов в условиях российской банковской системы);
- прогнозирование финансовых результатов деятельности пред- приятия на конкретный период времени (с анализом динамики сальдо на счетах);
- бизнес-реинжиниринг несостоятельного предприятия (изменение структуры и ресурсов предприятия-банкрота, после чего с помощью имитационной модели можно сделать прогноз основных финансовых результатов и дать рекомендации о целесообразности того или иного варианта реконструкции, инвестиций или кредитования производственной деятельности);
- анализ адаптивных свойств и живучести компьютерной региональной банковской информационной системы (например, частично вышедшая из строя в результате природной катастрофы система электронных расчетов и платежей после катастрофического земле- трясения 1995 г. на центральных островах Японии продемонстрировала высокую живучесть: операции возобновились через несколько дней);
- оценка параметров надежности и задержек в централизованной экономической информационной системе с коллективным доступом (на примере системы продажи авиабилетов с учетом несовершенства физической организации баз данных и отказов оборудования);
- анализ эксплуатационных параметров распределенной многоуровневой ведомственной информационной управляющей системы с учетом неоднородной структуры, пропускной способности каналов связи и несовершенства физической организации распределенной базы данных в региональных центрах;
- моделирование действий курьерской (фельдьегерьской) вертолетной группы в регионе, пострадавшем в результате природной катастрофы или крупной промышленной аварии;
- анализ сетевой модели РБВ.Т (Ргодгаш Еча1иа1юп апс~ В.еч1еи Тесйпн1не) для проектов замены и наладки производственного оборудования с учетом возникновения неисправностей;
- анализ работы автотранспортного предприятия, занимающегося коммерческими перевозками грузов, с учетом специфики товарных и денежных потоков в регионе;
- расчет параметров надежности и задержек обработки информации в банковской информационной системе.

Приведенный перечень является неполным и охватывает те примеры использования имитационных моделей, которые описаны в литературе или применялись авторами на практике. Действительная область применения аппарата имитационного моделирования не имеет видимых ограничений. Например, спасение американских астронавтов при возникновении аварийной ситуации на корабле APOLLO стало возможным только благодаря "проигрыванию" различных вариантов спасения на моделях космического комплекса.

Система ИМ, обеспечивающая создание моделей для решения перечисленных задач, должна обладать следующими свойствами:

- возможностью применения имитационных программ совместно со специальными экономико-математическими моделями и методами, основанными на теории управления;
- инструментальными методами проведения структурного анализа сложного экономического процесса;
- способностью моделирования материальных, денежных и ин- формационных процессов и потоков в рамках единой модели, в общем модельном времени;
- возможностью введения режима постоянного уточнения при получении выходных данных (основных финансовых показателей, временных и пространственных характеристик, параметров рисков и др.) и проведении экстремального эксперимента.

*Историческая справка.* Имитационное моделирование экономических процессов – разновидность экономико-математического моделирования. Однако этот вид моделирования в значительной степе- ни базируется на компьютерных технологиях. Многие моделирующие системы, идеологически разработанные в 1970-1980-х гг., претерпели эволюцию вместе с компьютерной техникой и операционными системами ( эффективно используются в настоящее время на новых компьютерных платформах. Кроме того, в конце 1990-х гг. появились принципиально новые моделирующие системы, концепции которых не могли возникнуть раньше – при использовании ЭВМ и операционных систем 1970-1980-х гг.

## *Технология Имитационного моделирования*

Имитационное моделирование реализуется посредством набора математических инструментальных средств, специальных компьютерных программ и приемов, позволяющих с помощью компьютера провести целенаправленное моделирование в режиме "имитации" структуры и функций сложного процесса и оптимизацию некоторых его параметров. Набор программных средств и приемов моделирования определяет специфику системы моделирования – специального программного обеспечения.

В отличие от других видов и способов математического моделирования с применением ЭВМ имитационное моделирование имеет свою специфику: запуск в компьютере взаимодействующих вычислительных процессов, которые являются по своим временным параметрам – с точностью до масштабов времени и пространства – аналогами исследуемых процессов.

### Этапы имитационного моделирования

Имитационное моделирование как особая информационная технология состоит из следующих основных этапов.

1. *Структурный анализ процессов.*

Проводится формализация структуры сложного реального процесса путем разложения его на подпроцессы, выполняющие определенные функции и имеющие взаимные функциональные связи согласно легенде, разработанной рабочей экспертной группой. Выявленные подпроцессы, в свою очередь, могут разделяться на другие функциональные подпроцессы. Структура общего моделируемого процесса может быть представлена в виде графа, имеющего иерархическую многослойную структуру. В результате появляется формализованное изображение имитационной модели в графическом виде.

Экономические процессы содержат подпроцессы, не имеющие физической основы и протекающие виртуально, так как оперируют с информацией, деньгами, логикой, законами и их обра-

боткой, поэтому структурный анализ является эффективным этапом при моделировании экономических процессов.

На этом этапе описываются *экзогенные переменные*, т.е. те переменные, которые задаются вне модели, т.е. известны заранее. Ещё описываются параметры - это коэффициенты уравнений модели. Часто их не разделяют. *Эндогенные переменные* - это те переменные, кот. опред. в ходе расчетов по модели и не задаются в ней извне.

2. *Формализованное описание модели.*

Графическое изображение модели, ф-ии, выполняемой каждым подпроцессом, условия взаимодействия всех подпроцессов и особенности поведения моделируемого процесса (временная, пространственная, финансовая динамики_ д.б. описаны на спец языке одним из способов:

- Описание вручную на алгоритмическом языке, т.е. написание программы на языке программирования.
- Автоматизир описание с пом компьютерного графического конструктора.

3. *Построение модели.*

- Обычно это трансляция и редактирование связей (сборка модели);
- Режимы интерпритации и компиляция;
- Верификация (калибровка) параметров, работа на тестовых примерах.

4. *Проведение модельного эксперимента.*

Проводится оптимизация определенных параметров реального процесса. Этому должен предшествовать процесс, кот. называется планирование эксперимента.

# МЕТОД МОНТЕ-КАРЛО

**- Способ исследования поведения вероятностных систем экономических, технических и т.п. в условиях, когда неизвестны в полной мере внутренние взаимодействия в этих системах.**

Создателями метода статистических испытаний (метода Монте-Карло) считают американских математиков Д. Неймана и С. Улама. В 1944 году, в связи с работами по созданию атомной бомбы Нейман предложил широко использовать аппарат теории вероятностей для решения прикладных задач с помощью ЭВМ. Данный метод был назван так в честь города в округе Монако, из-за рулетки, простейшего генератора случайных чисел.

Первоначально метод Монте-Карло использовался главным образом для решения задач нейтронной физики, где традиционные численные методы оказались мало пригодными. Далее его влияние распространилось на широкий класс задач статистической физики, очень разных по своему содержанию. К разделам науки, где все в большей мере используется метод Монте-Карло, следует отнести задачи теории массового обслуживания, задачи теории игр и математической экономики, задачи теории передачи сообщений при наличии помех и ряд других.

Метод Монте-Карло (или метод статистических испытаний) можно определить как **метод моделирования случайной величины с целью вычисления характеристик их распределений.** Суть состоит в том, что результат испытаний зависит от некоторой случайной величины, распределенной по заданному закону. Поэтому результат каждого отдельного испытания носит случайный характер. (Как правило, составляется программа для осуществления одного случайного испытания.) Проведя серию испытаний, получают **выборку**. Полученные статистические данные обрабатываются и представляются в виде численных оценок интересующих исследователя величин (характеристик системы).

Т.е. испытание повторяется N раз, причем каждый опыт не зависит от остальных, и результаты всех опытов усредняются. Это значит, что число испытаний должно быть достаточно велико, поэтому метод существенно опирается на возможности компьютера.

Теоретической основой метода Монте-Карло являются предельные теоремы теории вероятностей. Они гарантируют высокое качество статистических оценок при весьма большом числе испытаний. Метод статистических испытаний применим для исследования как стохастических, так и детерминированных систем. Практическая реализация метода Монте-Карло невозможна без использования компьютера.

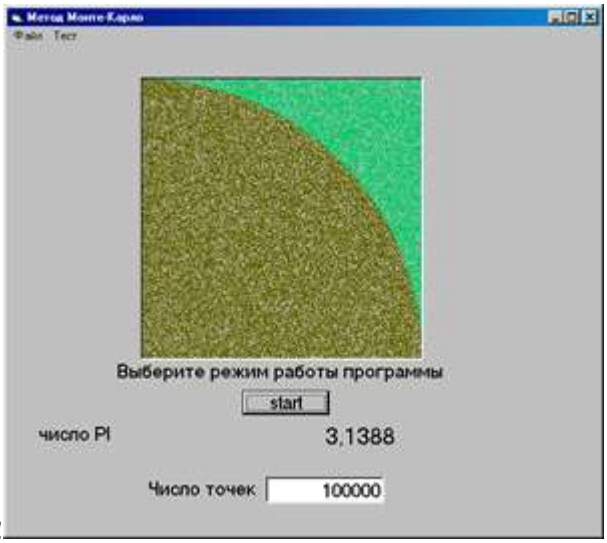
Можно проиллюстрировать метод статистических испытаний на простейшем примере: вычисление числа $\pi$ как отношение площади 1/4 круга к площади всей картинки путем разбрасывания случайным образом точек по всему рисунку. Затем считается отношение попавших в круг точек ко всем точкам, и по этому отношению приблизительно определяется отношение площадей. Увеличением числа вбрасываемых точек можно более точно определить площадь круга, но это так же ведет и к увеличению времени вычислений.

Точность вычислений очень сильно зависит от качества используемого генератора псевдослучайных чисел. Другими словами, точность тем выше, чем более равномерно случайные точки распределяются по единичному квадрату.

$$\pi \approx 4S_{кр}/S_{кв} \qquad (1)$$

Подсчитаем число точек внутри квадрата и внутри четверти круга. Очевидно, что их отношение будет приближенно равно отношению площадей этих фигур , так как попадание капель в различные места чертежа равновероятно. Пусть $N_{кр}$ - число капель в круге,$N_{кв}$ –число капель в квадрате , тогда

$$\pi \approx 4N_{кр}/N_{кв} \qquad (2)$$




**Ошибка!**

| число бросаний | Точное значение | Програмное значение | Погрешность |
|---|---|---|---|
| 10000 | 0.25 | 3,1632 | 0,021607346 |
| 100000 | 0.25 | 3,14436 | 0,002767346 |

Каждому точке поставим в соответствие два случайных числа, характеризующих его положение вдоль осей Ox и Oy (см. рис. 2). Если окажется, что для точки $(x_i,y_i)$ выполняется неравенство $x_t^2+y_t^2>1$, то, значит, она лежит вне круга. Если $x_t^2+y_t^2 \le 1$, то точка лежит внутри круга.

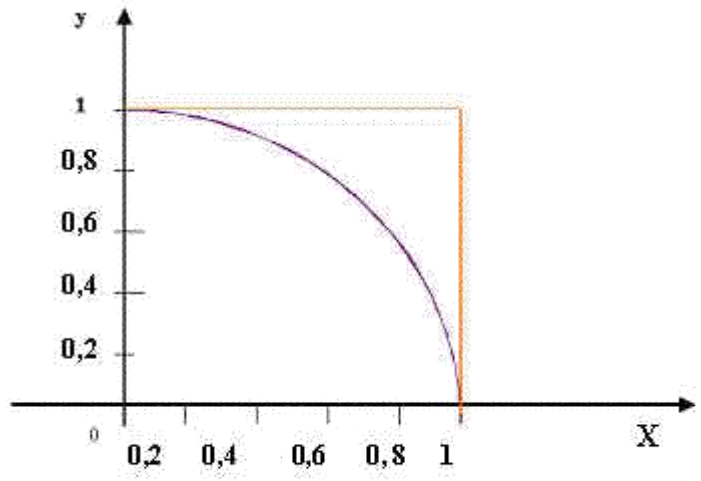



РИС.2

Для подсчета значения π воспользуемся формулой (2). Ошибка вычислений по этому методу, как правило пропорциональна $\sqrt{D/N}$, где $D$ – некоторая постоянная, а $N$ – число испытаний. В этом примере $N = N_{кв.}$. Из этой формулы видно, что для того, чтобы уменьшить ошибку в 10 раз, т.е. получить ещё один верный десятичный знак, нужно увеличить $N$, т.е. объём работы, в 100 раз (см. таблицу с результатами испытаний).

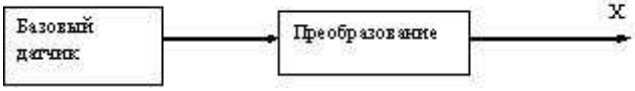
# ИМИТАЦИЯ СЛУЧАЙНЫХ ВЕЛИЧИН И ПРОЦЕССОВ

## *Базовый датчик*

Моделирование случайных элементов в системах является одной из самых главных, базовых задач математического моделирования. Моделирование любой системы или процесса, содержащих случайные компоненты, предполагает использование метода генерирования чисел, которые в определенном смысле являются *случайными*

Моделирование любой системы или процесса, содержащих случайные компоненты, предполагает использование метода генерирования чисел, которые в определенном смысле являются *случайными*

Любая случайная величина или *процесс* X может моделироваться следующим образом:




Базовый датчик выдает независимые равномерно распределенные случайные величины:

1. непрерывные в [0,1);
2. дискретные в $[0,2^p-1)$

Методология генерирования случайных чисел имеет долгую и интересную историю. Самые ранние методы генерирования выполнялись вручную, например: вытягивание жребия, бросание игральных костей, сдача карт или вытаскивание пронумерованных шариков из урны, в которой они предварительно были «тщательно перемешаны». Многие лотереи до сих пор функционируют таким образом. В начале XX века специалисты по статистике также проявили интерес к случайным числам. Были созданы механизированные устройства для более быстрого генерирования случайных чисел: в конце 1930-х годов Кендалл и Бабингтон-Смит использовали быстро вращающийся диск для подготовки таблицы, содержащей 100000 случайных однозначных чисел. Позднее разработаны электрические схемы, основанные на произвольно пульсирующих электронных лампах, которые выдавали до 50 случайных чисел в секунду. Одна из таких схем реализована в электронном индикаторном устройстве для получения случайных чисел (Electronic Random Number Indicator Equipment, ERNIE), использовавшемся Британским почтовым ведомством для выбора победителей в лотерее. Другое электронное устройство применялось компанией Rand Corporation [Rand Corporation, 1955] для генерирования таблицы, содержащей 11000 000 случайных цифр. Было изобретено и множество других схем, таких как произвольный выбор чисел из телефонных книг, результатов переписи населения или использование цифр в числе $\pi$ до 100 000 десятичных разрядов. В последние годы также проявляется интерес к созданию и использованию физических устройств для получения случайных чисел, например, Миятаке [Miyatake et al., 1983] описывает устройство, основанное на подсчете гамма-излучения.

С распространением компьютеров (и моделирования) все более пристальное внимание стало уделяться методам генерирования, или генераторам, случайных чисел, совместимых со способом работы компьютеров. Одна возможность состоит в прямом подключении к компьютеру электронных устройств генерирования случайных чисел, таких как ERNIE. У этого метода есть несколько недостатков, основной заключается в том, что точно воспроизвести ранее сгенерированный поток : чисел невозможно.

Поэтому в исследованиях 1940-1950-х годов ученые обратились к *численным,* или *арифметическим,* способам генерирования «случайных» чисел. Эти методы являются последовательными, каждое новое число определяется одним или несколькими предшествующими числами в соответствии с заданной арифметической формулой. Первый такой арифметический генератор, который был предложен фон Нейманом и Метрополисом в 1940-х годах, известен как *метод срединных квадратов.* Далее приведен пример его использования.

Пример 1. Возведем в квадрат положительное четырехзначное целое число $Z_o$ и получим целое число из восьми цифр. Если понадобится, добавим нули в левую часть, чтобы в числе было ровно восемь цифр. Возьмем *средние четыре* цифры из этого восьмизначного числа в качестве следующего четырехзначного числа $Z_i$. Поместим десятичную запятую слева от числа $Z_{IF}$ чтобы получить первое случайное число с распределением U(0,1), а именно $V_v$ Затем число $Z_2$ представим средними четырьмя цифрами $Z$\, а второе случайное число $U_2$ получим, разместив десятичную запятую слева от цифр числа $Z_2$, и т. д. В табл. 1 собраны несколько первых значений Z.н £/ДЛЯ $Z_o$= 7182 (первые четыре цифры справа от десятичной запятой в числе е).

**Метод срединных квадратов**

| i | Z | | $Z^2$ |
|---|---|---|---|
| 0 | 7182 | – | 51 581 124 |
| 1 | 5811 | 0,5811 | 33 767 721 |
| 2 | 7677 | 0,7677 | 58 936 329 |
| 3 | 9363 | 0,9363 | 87 665 769 |
| 4 | 6657 | 0,6657 | 44 315 649 |
| 5 | 3156 | 0,3156 | 9 960 336 |
| 6 | 9603 | 0,9603 | 92 217 609 |
| 7 | 2176 | 0,2176 | 4 734 976 |
| 8 | 3497 | 0,3497 | 12 229 009 |
| 9 | 2290 | 0,229 | 5 244 100 |
| 10 | 2441 | 0,2441 | 5 958 481 |
| 11 | 9584 | 0,9584 | 91 853 056 |
| 12 | 8530 | 0,853 | 72 760 900 |
| 13 | 7609 | 0,7609 | 57 896 881 |
| 14 | 8968 | 0,8968 | 80 425 024 |
| 15 | 4250 | 0,425 | 18 062 500 |
| 16 | 625 | 0,0625 | 390 625 |
| 17 | 9062 | 0,9062 | 82 119 844 |
| 18 | 1198 | 0,1198 | 1 435 204 |
| 19 | 3520 | 0,352 | 12 390 400 |
| 20 | 3904 | 0,3904 | 15 241 216 |
| 21 | 2412 | 0,2412 | 5 817 744 |
| 22 | 8177 | 0,8177 | 66 863 329 |
| 23 | 8633 | 0,8633 | 74 528 689 |
| 24 | 5286 | 0,5286 | 27 941 796 |
| 25 | 9417 | 0,9417 | 88 679 889 |
| 26 | 6798 | 0,6798 | 46 212 804 |
| 27 | 2128 | 0,2128 | 4 528 384 |
| 28 | 5283 | 0,5283 | 27 910 089 |
| 29 | 9100 | 0,91 | 82 810 000 |
| 30 | 8100 | 0,81 | 65 610 000 |
| 31 | 6100 | 0,61 | 37 210 000 |
| 32 | 2100 | 0,21 | 4 410 000 |
| 33 | 4100 | 0,41 | 16 810 000 |
| 34 | 8100 | 0,81 | 65 610 000 |
| 35 | 6100 | 0,61 | 37 210 000 |

Таблица 1. Метод срединных квадратов

Может показаться, что метод срединных квадратов обеспечивает хорошую перестановку элементов одного числа для получения следующего и такое случайное правило обеспечивает хороший способ генерирования случайных чисел. В действительности это не совсем так. Одна из серьезных проблем состоит в том, что данный метод имеет стойкую тенденцию возвращать значения, которые стремятся к нулю. (Продолжите еще немного табл. 7.1 или назначьте $Z_o$ = 1009, первые четыре цифры из таблицы Rand Corporation.) Таким образом, нельзя рассчитывать на то, что хороший генератор случайных чисел можно создать, производя манипуляции с каким-либо одним числом для получения следующего.

Метод срединных квадратов вовсе не является случайным, то есть непредсказуемым (это наиболее существенный его недостаток). На самом деле, если мы знаем одно число, следующее число является полностью предопределенным, поскольку правило его получения неизменно. Фактически, когда задается $Z_o$, предопределяется вся *последовательность* чисел $Z_t$ и $U_t$. Это замечание касается всех арифметических генераторов (далее в главе мы будем рас-

сматривать только данный тип). Поэтому арифметические генераторы именуются *псевдослучайны*.

Типы базовых датчиков:

1. физические (любой физический шум) – не используются, т.к. характеристики нестабильны и реализацию повторить нельзя;
2. псевдослучайные – строятся на основе *детерминированного* алгоритма, но полученные результаты неотличимы от случайных.

Псевдослучайные базовые датчики строятся по модели $x_{n+1} = f(x_n)$ при заданном $x_0$.

Хороший арифметический генератор случайных чисел должен обладать следующими свойствами.

1. Получаемые числа должны быть равномерно распределены в интервале [0, 1] и не должны иметь корреляции друг с другом, в противном случае результаты моделирования могут оказаться полностью недействительными.
2. Чтобы генератор можно было использовать на практике, он должен обладать быстродействием и не требовать больших затрат памяти.
3. Генератор должен обеспечивать возможность точно воспроизводить заданный поток случайных чисел. Во-первых, это позволяет упростить отладку компьютерной программы и проверить, правильно ли она работает. Во-вторых, что гораздо важнее, вы можете использовать *идентичные* случайные числа при моделировании различных систем и выполнить их более точное сравнение.
4. В генераторе должен быть предусмотрен простой способ получения отдельных потоков случайных чисел. Как вы узнаете в дальнейшем, поток - это просто часть последовательности случайных чисел, производимых генератором, очередной поток начинается в том месте, где заканчивается предыдущий. Мы можем рассматривать различные потоки как отдельные и независимые генераторы (при условии, что весь поток, который, как правило, имеет длину 100 000 или более чисел, не применяется). Таким образом, пользователь может выделить определенный поток для конкретного источника случайности при моделировании

## *Требования к базовым датчикам и их проверка*

### 1. Отрезок апериодичности

*Периодом* T и *длиной отрезка апериодичности* датчика называются наименьшие из величин, удовлетворяющие

$$x_{L+i} = x_{L-T+i},\ i = \overline{0, T-1}$$

$$x_0\ x_1, \ldots, \underbrace{x_{L-T}, \ldots, x_{L-1}}_{T}, \underbrace{x_L, \ldots, x_{L+T-1}}_{T}, \underbrace{\ldots}_{T}, \underbrace{\ldots}_{T}, \ldots$$

совпадают

Чем больше T и L, тем лучше датчик (особенно L).

Как определить их:

1. Берем V – достаточно большое число (обычно $10^7 \div 10^9$
2. Генерируем $x_0\ x_1, \ldots, x_{V-1}$ и проверяем $x_V$, запоминая.
3. Генерируем $x_{V+1}, \ldots, x_{V+V}$ и проверяем $x_{V+i} \stackrel{?}{=} x_V$. Если нет для $i = \overline{1, V-1}$, то $T \geq V$ и $L \geq 2V-1$. Иначе запоминаем $i_1$, для которого $x_{V+i_1} = x_V$, и находим следующий $i_2$, для которого $x_{V+i_2} = x_V$ (если нужно, генерируются дополнительные величины). Тогда $T = i_2 - i_1$
4. Генерируем $x_0\ x_1, \ldots$ и $x_T\ x_{T+1}, \ldots$ и ищем первое совпадение $x_i = x_{T+i}$. Тогда $L = T + i$

## 2. Равномерность

Должно быть $p(x)=1$ для $x \in [0,1)$

Проверка:

1. Берем $N >> 1$ и генерируем $x_0\ x_1,\ldots,x_N$
2. Находим $k = \mathrm{Round}(1 + 3.3 \lg N)$ и разбиваем отрезок [0,1) на k равных частей (длиной $\frac{1}{k}$)
3. Для каждого числа $x_i$ определяем, в какой интервал оно попало:

$r_i = \mathrm{Int}(k \cdot x_i) + 1$

и заполняем массив $\{N_1, N_2, \ldots, N_k\}$: $N_{r_i} := N_{r_i} + 1$

4. По значениям этого массива строим гистограмму (для наглядности).
5. Используем критерий $\chi^2$

$$\chi^2 = \sum_{i=1}^{k} \frac{(N_i - \frac{N}{k})^2}{\frac{N}{k}}.$$

Выбираем уровень значимости $\alpha$ (обычно 5%), а число степеней свободы $f = k - 1$. В таблицах $\chi^2$ находим значение $C_{\alpha,f}$ и проверяем: если $\chi^2 < C_{\alpha,f}$, то «данные эксперимента не противоречат гипотезе о равномерности случайных чисел», иначе – «противоречат».

## 3. Некоррелированность

1. Генерируем $x_0\ x_1,\ldots,x_N\ (N = 10^3 \div 10^4)$
2. Вычисляем $$r_k = \frac{12}{N-k} \sum_{i=1}^{N-k} (x_i - 0.5)(x_{i+k} - 0.5),$$ $k = 1,2,\ldots,10$ (можно и больше).
3. Вычисляем $$t_k = \frac{2k}{\sqrt{1 - r_k^2}} \sqrt{N - k - 2}$$
4. Проверяем для всех k

$|t_k| < g_\alpha$

Если да, то «данные эксперимента не противоречат гипотезе о равномерности случайных чисел», иначе – «противоречат». Здесь $\alpha$ – уровень значимости, а $g_\alpha$ берется из таблиц $g_{5\%} = 1.96$

## 4. Простейшие проверки

Подходят для любой непрерывной случайной величины.

1. Математическое ожидание: $$M = \frac{1}{N} \cdot \sum_{i=0}^{N-1} x_i$$
2. Дисперсия: $$D = \frac{1}{N} \cdot \sum_{i=0}^{N-1} (x_i - M)^2$$

## Некоторые общие замечания по тестированию

Число, разнообразие и диапазон сложности тестов для генераторов случайных чисел в полном смысле слова ставит в тупик. Что еще хуже, не существует (и, наверное, не будет существовать) единого мнения относительного того, какие тесты лучше. Однако, невзирая на любые проверки, даже если они

проводятся в огромном количестве, полностью убедить кого-либо в том, что какой-то из генераторов является «наилучшим», нельзя. Однако можно посоветовать подбирать тесты для генератора согласно замыслу его использования. Это приведет, например, к изучению поведения пар значений $U_i$ (вероятно, с помощью проверки по двухмерному сериальному критерию), если случайные числа действительно используются парами при моделировании. В более широком смысле, этот совет подразумевает, что нужно быть более осторожными в выборе и тестировании генератора случайных чисел, если моделирование, в котором он будет использоваться, является очень дорогостоящим, требует результатов высокой точности или же представляет собой важную составляющую крупных исследований.

## *Модели базовых датчиков*

### Линейные конгруэнтные генераторы

Сегодня очень часто применяются *линейные конгруэнтные генераторы* (ЛКГ), созданные Лемером [Lehmer, 1951]. В них последовательность целых чисел $Z_b$ $Z_2$,... определяется по рекурсивной формуле

$$Z_i = (aZ_{i-1} + c)(mod\ m), \qquad (1)$$

где $m$ (модуль), $a$ (множитель), с (приращение) и $Z_o$ (начальное число или значение) являются неотрицательными целыми числами. Таким образом, согласно формуле (1), для получения $Z_i$ нужно разделить $aZ_{i-1} + c$ на $m$, то есть $Z_i$ будет остатком этого деления. Поэтому $0 \le Z_i \le$ m - 1, а чтобы получить искомые случайные числа $U_i$ (при $i$ = 1,2,...) в интервале [0,1], примем $U_i = Z_i/m$. Мы концентрируем внимание главным образом на $Z_i$, хотя следует иметь в виду и точный характер деления $Z_i$ на m в связи с различиями в способах обработки разными компьютерами компиляторами чисел с плавающей запятой. Кроме того, что целые числа я, а, с и $Z_o$ должны быть неотрицательными, они должны удовлетворять неравенствам $0 < m$, $a < m$, $c < m$ и $Z_o < m$.

Против использования ЛКГ можно высказать следующие соображения. Во-первых:, значения $Z_{it}$ определенные по формуле (1), вовсе не являются случайными; это касается всех генераторов псевдослучайных чисел. Однако, тщательно подбирая эти четыре параметра, мы стараемся вызвать такое поведение величин $Z_i$ при котором соответствующие величины $U_i$ *представляются* как независимые и равномерно распределенные случайные величины с распределением U(0, 1), когда они проверяются с помощью ряда тестов.

Во-вторых, при использовании ЛКГ $\backslash]_x$ могут принимать лишь рациональные ачения 0,1/т, *2/m,..., (m* - 1)/т. В действительности f/j могут принимать только Качения мантисс этих величин в зависимости от определения констант *m, a, с* $iZ_0$, а также от характера деления с плавающей запятой на *m.* Поэтому у нас нет бможности получить значение /7; между, например, 0,1/т и 0,9/т, в то время как *i* возможность *должна* возникать с вероятностью 0,8/тп > 0. Нам предстоит здиться, что для модуля m обычно выбирается очень большое значение, скажем, ' или больше, при котором точки в интервале [0,1], куда могут попадать значе-рия $U_t$, располагаются очень плотно. Для $m > 10^9$ существует по меньшей мере 1 Млрд возможных значений. Но цикличность неизбежна. Согласно формуле (1), как только $Z_i$ получает значение, которое у нее уже было, сгенерируется та же самая последовательность величин, и этот цикл повторяется бесконечно. Длина цикла называется *периодом* генератора. Для ЛКГ значение Z; зависит *только* от предыдущего целого числа $Z_{i-1}$, а поскольку $0 < Z_i < m$ - 1, период генератора не превышает величину *m.* Если период действительно равен *m,* считается, что ЛКГ имеет *полный период.* Разумеется, если генератор имеет полный период, какое бы ни было выбрано начальное число $Z_o$ из последовательности {0,1,..., *m* - 1}, весь цикл будет воспроизведен в некотором порядке. Если же период генератора меньше полного, длина цикла будет зависеть от выбора определенного значения $Z_o$, в этом случае мы будем говорить о периоде *начального значения* для этого генератора.

Поскольку в широкомасштабных проектах имитационного моделирования могут использоваться сотни тысяч случайных чисел, очевидно, что для них требуются ЛКГ с длинными периодами. Более того, лучше всего иметь ЛКГ с полным периодом, поскольку тогда у нас есть уверенность, что каждое целое число в интервале от 0 до m - 1 возникнет в каждом цикле только один раз, а это способствует равномерности распределения значений $U_t$. (Даже ЛКГ с полным периодом на некоторых участках цикла могут иметь неравномерное распределение. Например, если мы сгенерируем только m/2 последовательных значений $Z_i$, могут остаться большие пробелы в последовательности возможных значений 0,1,…,*m* - 1.) Таким образом, полезно знать, как выбирать значения ***m, a*** и ***c*** так, чтобы у соответствующего ЛКГ был

полный период. Следующая теорема, доказанная Халлом и Добеллом [Hull and Dobell, 1962], дает нам определение параметров.

**Теорема 1.** ЛКГ, определенный по формуле (1), имеет полный период тогда и только тогда, когда выполняются три следующих условия:

а) единственное положительное целое число, на которое без остатка делятся как $m$, так и $c$, равно 1 (**c** является взаимно простым по отношению к $m$);

б) если $q$ является простым числом (делится только само на себя и на 1), на которое делится $m$, то $a$ - 1 делится на $q$;

в) если $m$ делится на 4, то $a$ - 1 тоже делится на 4.

Из-за первого условия теоремы 1 ЛКГ работают по-разному с параметром c > 0 *(смешанные* ЛКГ) и с параметром c = 0 *(мультипликативные* ЛКГ).

## Смешанные генераторы

При c > 0 выполнение первого условия теоремы 1 возможно, следовательно, мы можем получить полный период $m$. Сначала выберем значение $m$.

Чтобы получить длинный период и высокую плотность величин $U_i$, в интервале [0,1], величина $m$ должна иметь большое значение. Кроме того, деление на $m$ для получения остатка в формуле (1) — довольно длительная арифметическая операция. Поэтому желательно избежать явного выполнения такого деления. Выбор $m$, который является удачным во всех этих отношениях — $m = 2^b$, где $b$ — число битов (двоичных знаков) в слове задействованного компьютера, которые действительно доступны для хранения данных. В частности, во многих компьютерах и компиляторах используются 32-битовые слова, при этом крайний левый бит является знаковым, следовательно, $b$ = 31. Если $b$ имеет достаточно большое значение, например, $b$ > 31, тогда $m > 2^{31} > 2{,}1*10^9$. Кроме того, выбирая $m = 2^b$ мы *действительно* избегаем явного деления на $m$ для большинства компьютеров, поскольку можно воспользоваться *переполнением целых чисел.* Наибольшее целое число, которое может быть представлено, равно $2^b - 1$, и любая попытка сохранить большее целое число $W$ (имеющее, например, $h > b$ двоичных знаков) завершится потерей левых (наиболее значимых) $h - b$ двоичных знаков целого числа, превысившего допустимый размер. Оставшиеся $b$ двоичных знаков составляют ровно $W(\mathrm{mod}\ 2^b)$.

Как же следует выбирать значения $a$ и $c$, когда $m = 2^b$, чтобы получить хороший смешанный ЛКГ? Опыт показывает, что лучше отказаться от использования смешанного ЛКГ вообще. Более простые и понятные мультипликативные ЛКГ зарекомендовали себя не хуже смешанных и применяются гораздо чаще.

## Мультипликативные генераторы

Мультипликативные ЛКГ выгодно применять, так как для них не требуется определять параметр c, но у них не может быть полного периода, потому что первое условие теоремы 1 для них выполняться не будет (поскольку, например, $m$ является положительным числом и как $m$, так и c = 0 делятся на него без остатка). Однако вы сможете убедиться, что можно получить период $m$ - 1, если осторожно подбирать значения $m$ и a. Мультипликативные ЛКГ появились еще до смешанных и исследовались более интенсивно. Большинство ЛКГ, применяемых сегодня, являются мультипликативными, поскольку факт улучшения эффективности, на которое возлагались надежды в связи с введением смешанных генераторов, окончательно не доказан.

Как и при использовании смешанных ЛКГ, для вычислений по-прежнему подходит выбор $m = 2^b$, таким образом мы избегаем явного деления. Однако можно доказать [Knuth, 1998a, p. 20], что в этом случае период составляет самое большее $2^{b-2}$, то есть лишь четвертая часть целых чисел от 0 до $m$ - 1 может быть получена в качестве значений переменных $Z_i$. (Фактически период составляет $2^{b-2}$, если значение $Z_o$ является нечетным, а параметр $a$ имеет вид $8k + 3$ или $8k + 5$ для некоторых значений $k$ = 0,1,....) Кроме того, нам, как правило, неизвестно, *куда* попадут эти $m$ /4 целых числа, то есть между полученными переменными Z, могут быть недопустимо большие промежутки. К тому же, если для параметра $a$ мы выбираем вид $2^l + j$ (так что умножение $Z_{i-1}$ на $a$ заменяется смещением и прибавлением $j$), можно получить неудовлетворительные статистические свойства. Известен, например, генератор RANDU, который имеет такие значения параметров: $m = 2^{31}$, $a = 2^{16} + 3 = 65\ 539$, c = 0 - и, как оказалось, очень неподходящие

статистические свойства, поэтому следует избегать его применения. Даже отказаться от выбора $a = 2^l + j$, используя $m = 2^b$, - тоже не самый подходящий вариант параметра для мультипликативных ЛКГ, хотя бы потому что период $\boldsymbol{m}/4$ короче и по этой причине возникает вероятность появления пропусков в значениях $Z_i$.

Из-за трудностей, возникавших в связи с выбором $\boldsymbol{m = 2^b}$ в мультипликативных ЛКГ, основное внимание уделялось поиску других способов определения значения $m$. Метод, оказавшийся довольно успешным, был представлен Хатчинсоном [Hutchison, 1966], который приписывает авторство идеи Лемеру. Вместо того чтобы использовать $\boldsymbol{m = 2^b}$, было предложено определять значение $\boldsymbol{m}$ как наибольшее простое число, которое меньше $2^b$. Например, если $b = 31$, то наибольшее простое число, которое меньше $2^{31}$, соответственно будет составлять $2^{31} — 1 = 2\ 147\ 483\ 647$. Теперь для простого числа те можно показать, что период составляет $m - 1$, если $a$ — это *первообразный элемент по модулю* $\boldsymbol{m}$, то есть наименьшее целое число $\boldsymbol{l}$, для которого $a^l - 1$ делится на $\boldsymbol{m}$, составляет $\boldsymbol{l} = \boldsymbol{m} - 1$ [Knuth, 1998a, p. 20]. Если таким образом выбрать значения $\boldsymbol{m}$ и a, то можно получить каждое целое число 1, 2, ..., $n - 1$ один раз в каждом цикле, так что $Z_0$ может быть любым целым числом от 1 до $m - 1$, а в результате все равно будет получен период $m - 1$. Такие генераторы называются *мультипликативными ЛКГ с простым модулем*

## Мультипликативный конгруэнтный метод (метод вычетов)

В основе лежит следующее рекуррентное соотношение:

$$x_i^* = (\beta x_{i-1}^*) \bmod M,$$

$$x_i = \frac{x_i^*}{M},\ i = 1,2,...$$

$\beta$ – множитель, M – модуль, $x_0^* \in \{1,...,M-1\}$ – стартовое значение. Рекомендуемые значения для 64-разрядной сетки:

$$M = 2^{63} = 9\ 223\ 372\ 036\ 854\ 775\ 808$$

$$\beta = 2^{32} + 3 = 4\ 294\ 967\ 299$$

$$x_0^* = \beta = 4\ 294\ 967\ 299$$

Тогда период $$T = \frac{M}{4} = 2\ 305\ 843\ 009\ 213\ 693\ 952$$

Для 32-разрядной:

$$M = 2^{31} = 2147483648\ \ (= MaxLongInt)$$

$$\beta = 2^{16} + 3 = 65539$$

$$x_0^* = \beta = 65539$$

Тогда период $$T = \frac{M}{4} = 536\ 870\ 912$$

Как раз этот датчик случайных чисел не удовлетворяет статистическим требованиям.

## Линейные смешанные формулы.

$$x_i^* = \left(\beta_1 x_{i-1}^* + ... + \beta_p x_{i-p}^* + c\right) \bmod M,$$

$$x_i = \frac{x_i^*}{M},\ M = 1,2,...$$

p – порядок, стартовые значения: $x_{-p+1}^*,...,x_{-1}^*,x_0^*$. Период $T \propto M^p$

Частный случай. Датчик Терпугова.

```
function Rand (var y: Integer): Double;
const b=843314861;
      c=453816693;
      m2=1073741824; {M/2}
begin
{$O-,$R-} {Optimization- & Range Check-}
  y:=y*b+c;
  if y<0 then y:=(y+m2)+m2;
  Result:=Double(y)*0.4656613E-09;
end;
```

$$x_i^* = \left(\gamma(x_{i-1}^*)^2 + \beta x_{i-1}^* + c\right) \bmod M,$$

$$x_i = \frac{x_i^*}{M},\ i = 1,2,\ldots$$

Причем $T_{\max} = M = 2^q, q > 2$ тогда и только тогда, когда $\beta \bmod 4 = (\gamma + 1) \bmod 4$

Существуют и другие методы моделирования базовых датчиков.

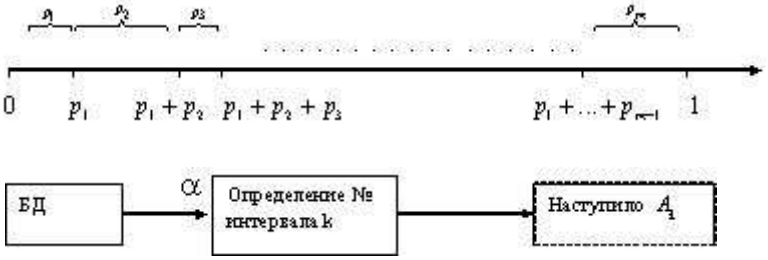
## *Генерация случайных событий*

1. Пусть имеется некоторое случайное событие А, наступающее с вероятностью p(A). Тогда $p\{\alpha < p\} = p$ ($\alpha$ – числа, генерируемые базовым датчиком). Следовательно, генератор 1 случайного события:

$$k = \arg\left\{\sum_{i=0}^{k-1} p_i \le \alpha < \sum_{i=0}^{k} p_i\right\}, \text{где } p_0 = 0$$

2. Полная группа попарно несовместимых событий $A_1, A_2, \ldots, A_m$. Пусть $p(A_i) = p_i$

Идея:



# ГЕНЕРАЦИЯ ДИСКРЕТНЫХ СЛУЧАЙНЫХ ВЕЛИЧИН

При возникновении в процессе моделирования каких-либо случайных факторов следует прибегнуть к выборке, или *генерированию,* случайных величин из распределений вероятностей. Как и в главе 7, выражение «генерирование случайных величин» мы используем для обозначения действий, направленных на получение наблюдений по случайным переменным (или для реализации случайных величин) из искомого распределения. Форма распределения подбирается специально, в результате для сбора данных могут использоваться, например, экспоненциальное распределение, гамма-распределение или распределение Пуассона (см. главу 6). В данной главе мы склонны допустить, что распределение уже было некоторым образом определено (в том числе и значения его параметров), и рассматриваем только вопрос о возможности генерировать случайные величины с этим распределением для выполнения прогона имитационной модели. Например, для моделей систем массового обслуживания (см. раздел 1.4 и главу 2) требуется генерировать время между поступлениями и время обслуживания для обеспечения продвижения модельного времени, а для модели системы управления запасами (из раздела 1.5) нужно генерировать объем спроса в моменты его возникновения.

Как мы убедимся, ознакомившись с этой главой, основной составляющей, необходимой для *каждого* метода генерирования случайных величин из *любого* распределения или случайного процесса, *является* источник независимых и одинаково распределенных случайных величин с распределением U(0, 1). Вот почему очень важно наличие надежного генератора случайных чисел с распределением U(0, 1). В большинстве компьютерных программ и пакетов имитационного моделирования имеются удобные генераторы случайных чисел, но некоторые из них (особенно старые версии) не адекватны уровню современных требований (см. главу 7). Без надлежащего генератора случайных чисел невозможно правильно генерировать случайные величины из любого распределения. Поэтому будем исходить из предположения, что у нас имеется надежный источник случайных чисел

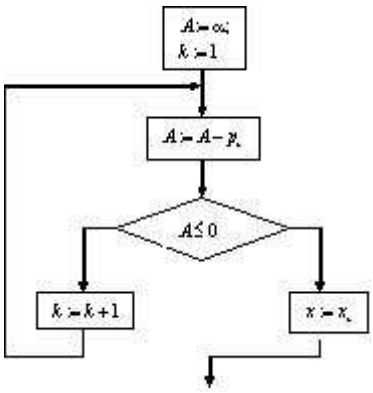
Пусть имеется дискретная случайная величина с рядом распределения.

| x | $x_1$ | $x_2$ | ... | $x_m$ |
|---|---|---|---|---|
| p | $p_1$ | $p_2$ | ... | $p_m$ |

$p_i = p\{x = x_i\}$, $\sum_{i=1}^{m} p_i$

Т.о. задача сводится к генерации полной группы попарно несовместимых событий. Т.е., если наступило $A_k$, $x := x_k$ .

Программная реализация

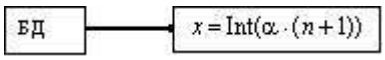



## *Специальные методы генерации некоторых дискретных случайных величин*

### 1. Равномерное распределение

| x | 0 | 1 | ... | n |
|---|---|---|---|---|
| p | $\frac{1}{n+1}$ | $\frac{1}{n+1}$ | ... | $\frac{1}{n+1}$ |

Тогда




Док-во.

$$p\{x=m\}=p\{m\le\alpha(n+1)<m+1\}=p\left\{\frac{m}{n+1}\le\alpha<\frac{m+1}{n+1}\right\}=\text{/* длина интервала */}=$$

$$=\frac{m+1}{n+1}-\frac{m}{n+1}=\frac{1}{n+1}\Rightarrow$$ x удовлетворяет равномерному распределению.

## 2. Геометрическое распределение

x = 0,1,2,... до $\infty$

$$p\{x=m\}=p(1-p)^m,\ 0<p<1$$

$$x=\mathrm{Int}\left(\frac{\ln\alpha}{\ln(1-p)}\right)$$

Док-во.

$$p\{x=m\}=p\left\{m\le\frac{\ln\alpha}{\ln(1-p)}<m+1\right\}=\text{/* } \ln(1-p)<0 \text{ */}=$$

$$=p\{(m+1)\ln(1-p)<\ln\alpha\le m\ln(1-p)\}=p\{(1-p)^{m+1}<\alpha\le(1-p)^m\}=$$

$$=\text{/* длина интервала */}=(1-p)^m-(1-p)^{m+1}=(1-p)^m[1-(1-p)]=p(1-p)^m$$

## 3. Отрицательное биномиальное распределение.

x=0,1,2,... до $\infty$

Параметры: $0<p<1$, и $r=1,2,3,...$

$$p\{x=m\}=C_{m+r-1}^m\cdot p^r(1-p)^m$$

Для $r=1$ это распределение совпадает с геометрическим, поэтому можно представить $x=\xi_1+\xi_2+\ldots+\xi_r$, где $\xi_i,(i=\overline{1,r})$ - независимые случайные величины, распределенные по геометрическому закону. Т.о.

$$x=\sum_{i=1}^{r}Int\left(\frac{\ln\alpha_i}{\ln(1-p)}\right)$$

(базовый датчик должен выдать r чисел для генерации одного x).

## 4. Биномиальное распределение

$x=1,2,....,n$

$p\{x=m\}=C_n^m p^m(1-p)^{n-m}$ (теорема об опытах – вероятность наступления m событий A в n опытах).

Введем $\theta(x)=\begin{cases}1, x\ge 0,\\ 0, x<0.\end{cases}$ (функция Хэвисайда).

Тогда

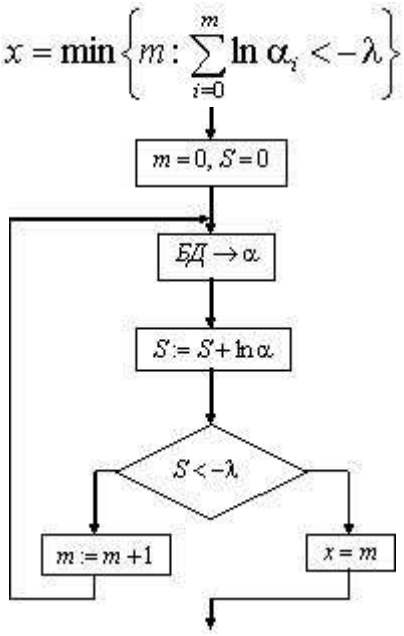
$$x=\sum_{k=1}^{n}\theta(p-\alpha)$$

$(p-\alpha)>0\Leftrightarrow\alpha<p\equiv$ наступило 1 события $\Rightarrow$ сумма дает кол-во событий наступивших в n опытах $\equiv$ биномиальное распределение).

## 5. Пуассоновское распределение

$x = 0,1,\ldots \partial o\ \infty$

$$p\{x = m\} = \frac{\lambda^m}{n!} e^{-\lambda}$$



# ГЕНЕРАЦИЯ НЕПРЕРЫВНЫХ СЛУЧАЙНЫХ ВЕЛИЧИН

Непрерывная случайная величина $\xi$ характеризуется плотностью $p_\xi(x)$ или функцией распределения $F_\xi(x) = \int_{-\infty}^{x} p_\xi(x)du$

## 1. Метод обратной функции

Основная идея: представим $\xi = \varphi(\alpha)$ и попробуем найти $\varphi(\cdot)$.

Допустим, что мы разрешили относительно $\alpha$: $\alpha = \varphi^{-1}(\xi)$. И потребуем, чтобы $\varphi \uparrow$. Тогда

$$F_\xi(x) = p\{\xi < x\} = p\{\varphi(\alpha) < x\} = p\{\alpha < \varphi^{-1}(x)\} = /* \text{ длина интервала } */ = \varphi^{-1}(x) \Rightarrow$$

$$\varphi(x) = F_\xi^{-1}(x) \Rightarrow$$

$$\xi = F_\xi^{-1}(\alpha)$$

Т.к. $\alpha$ равномерно распределена в [0,1), то и $(1-\alpha)$ равномерно распределена там же, следовательно, можно записать и так

$$\xi = F_\xi^{-1}(1-\alpha)$$

Метод обратной функции применяется редко, т.к. обычно найти $F^{-1}$ очень трудно.
Примеры.

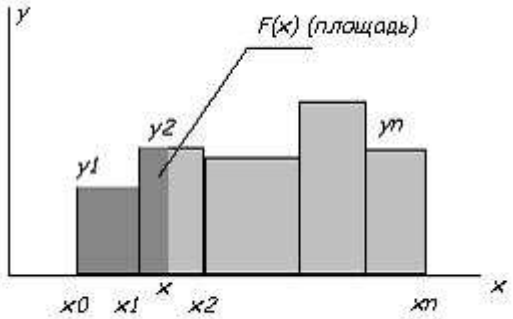
1. Экспоненциальное распределение: $\xi \ge 0,\ p_\xi(x) = \lambda e^{-\lambda x},\ x \ge 0$

$$F_\xi(x) = \int_0^x p_\xi(u)du = \int_0^x \lambda e^{-\lambda u}du = -e^{-\lambda u}\Big|_0^x = 1 - e^{-\lambda x} \Rightarrow$$

$$\alpha = 1 - e^{-\lambda\xi} \Rightarrow e^{-\lambda\xi} = 1-\alpha \Rightarrow -\lambda\xi = \ln(1-\alpha) \Rightarrow$$

$$\xi = -\frac{\ln(1-\alpha)}{\lambda} \Leftrightarrow \xi = -\frac{\ln\alpha}{\lambda}$$

2. Непрерывные случайные величины с заданной гистограммой:



Общая площадь $\sum_{k=1}^{n} y_k(x_k - x_{k-1}) = 1$

$$F(x) = \begin{cases} y_1(x - x_0), \ x_0 \le x \le x_1 \\ y_1(x_1 - x_0) + y_2(x - x_1), \ x_1 \le x \le x_2 \\ \ldots\ldots \end{cases}$$

Функция распределения:

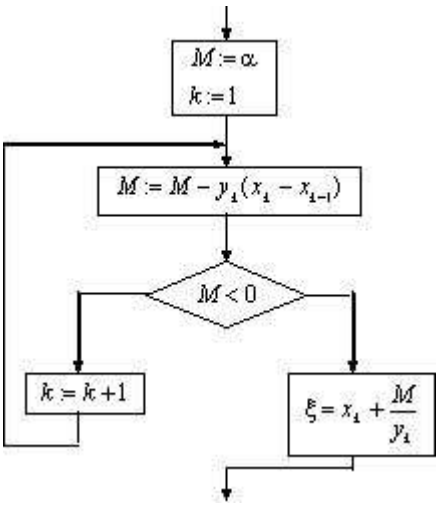
$$F(x) = \sum_{s=1}^{k-1} y_s(x_s - x_{s-1}) + y_k(x - x_k), \ \text{где} \ \ k: \ x_{k-1} \le x \le x_k$$

или

Чтобы найти формулу, решим уравнение $\alpha = F(\xi)$

1. $БД \to \alpha$.Отнимаем от него $y_1(x_1 - x_0)$, затем $y_2(x_2 - x_1)$ и т.д. до тех пор, пока не получим отрицательное значение: $\alpha - y_1(x_1 - x_0) - y_2(x_2 - x_1) - \ldots - y_k(x_k - x_{k-1}) = M < 0$
2. Ясно, что $M = -y_k(x_k - \xi)$. Следовательно,

$$\xi = x_k + \frac{M}{y_k}$$




## 2. Метод суперпозиции

Применим в случае, если $p_\xi(x) = \sum_{i=1}^{n} p_i \cdot p_i(x)$, где $p_i \ge 0$, $\sum_{i=1}^{n} p_i = 1$ и $p_i(x) \ge 0 \int_{-\infty}^{+\infty} p_i(x)dx = 1$.

Тогда моделирование производится следующим образом:

1. Генерируется дискретная случайная величина $\beta$ с рядом $p\{\beta = i\} = p_i, \ i = \overline{1,n}$
2. Генерируется непрерывная случайная величина $\xi$ с плотностью $p_i(x)$

Пример. Гиперэкспоненциальное распределение.

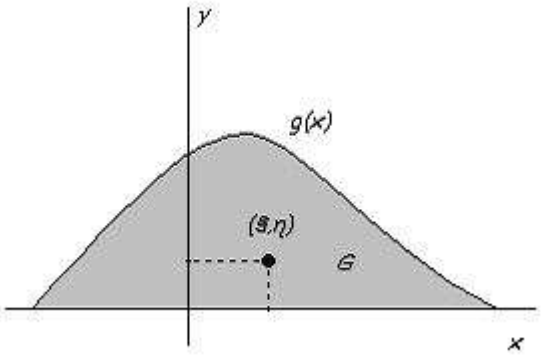
$$p_\xi(x) = \sum_{i=1}^{n} p_i \cdot \lambda_i e^{-\lambda_i x}$$

Моделирование: $\xi = -\frac{\ln \alpha}{\lambda_i}$, где $i$ - смоделирована как дискретная случайная величина $\overline{1,n}$ с рядом $p_i$.

## 3. *Метод исключения*

Пусть некоторая функция $g(x)$ удовлетворяет условиям:

1. $g(x) \geq 0$
2. $\int_{-\infty}^{+\infty} g(x)dx = G < +\infty$




Теорема. Пусть некоторая двумерная случайная величина $(\xi,\eta)$ имеет следующую совместную плотность распределения

$$p_{\xi\eta}(x,y) = \begin{cases} \frac{1}{G}, \text{если } (x,y) \in G \\ 0, \text{если } (x,y) \notin G \end{cases}$$

Тогда СВ $\xi$ имеет плотность распределения $p_\xi(x) = \frac{1}{G} g(x)$

Док-во.

$$F_\xi(x) = p\{\xi < x\} = /^* \text{см. рис.} ^*/ = \int_{-\infty}^{x} d\xi \int_0^{g(\xi)} p_{\xi\eta}(\xi,\eta)d\eta = \frac{1}{G}\int_{-\infty}^{x} d\xi \int_0^{g(\xi)} d\eta =$$

$$= \frac{1}{G}\int_{-\infty}^{x} g(\xi)d\xi \quad \Rightarrow \quad p_\xi(x) = F'_\xi(x) = \frac{1}{G} g(x)$$

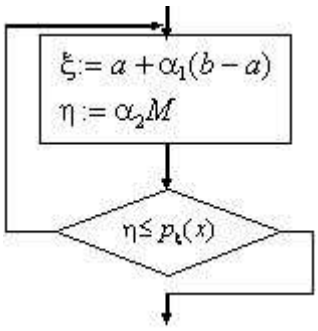
Т.о., если требуется моделировать случайную величину с плотностью $p_\xi(x)$, то принимаем $g(x) \equiv p_\xi(x)$, тогда $G = 1$. Т.е. достаточно генерировать двумерную $(\xi,\eta)$, равномерно распределенную в области $G$ под $p_\xi(x)$, и тогда $\xi$ будет иметь распределение $p_\xi(x)$.

Осталось научиться равномерно попадать под кривую $p_\xi(x)$ (область $G$). Оказывается, это очень просто: достаточно равномерно попадать в некоторую $G_1 : G \subseteq G_1$ и рассматривать только те точки, которые $\in G$ - они будут равномерно распределены в $G$

Например, если $a \leq \xi \leq b$, $\max_x p_\xi(x) = M$, то легче всего взять $G_1 = [a,b][0,M]$ и

$$\xi = a + \alpha_1(b-a),$$
$$\eta = \alpha_2 M.$$

Далее применяем метод исключения, т.е результатом моделирования считаем только те $\xi$, для которых $\eta \le p_\xi(x)$, остальные пропускаем:




## 4. Нормальные случайные величины

Нормальная случайная величина: $\xi = N(a,\sigma^2)$ : $p_\xi(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-a)^2}{2\sigma^2}}$

Стандартная нормальная случайная величина: $\xi = N(0,1)$ $p_\xi = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}}$

Любая нормальная случайная величина: $\xi = \sigma \cdot \zeta + a$, где $\zeta = N(0,1)$

Таким образом достаточно получить датчик стандартной нормальной случайной величины.

Методы:

1. Метод суммирования

   /* ЦПТ: Для независимых случайных величин $\eta_1, \eta_2, \ldots, \eta_n : M\{\eta_i\} = a, D\{\eta_i\} = \sigma^2$ произвольным распределением

   $$\zeta_n = \frac{\eta_1 + \eta_2 + \ldots + \eta_n - n \cdot a}{\sigma\sqrt{n}} \underset{n\to\infty}{\overset{\text{по распред}}{\to}} N(0,1)$$

   .*/

   Пусть $\eta_i = \alpha_i$ ясно, что $M\{\alpha_i\} = \frac{1}{2}$, $D\{\alpha_i\} = \frac{1}{12}$. Тогда $\zeta = \sqrt{\frac{12}{n}}\left(\sum_{i=1}^{n} \alpha_i - \frac{n}{2}\right)$ Если взять $n = 12$, то получим

   $$\zeta = \sum_{i=1}^{12} \alpha_i - 6$$

   Существуют более точные формулы, типа $\zeta_n^* = \zeta_n + \frac{1}{20n}\left(\zeta_n^3 - 3\zeta_n\right)$. В частности, для $n = 12$ :

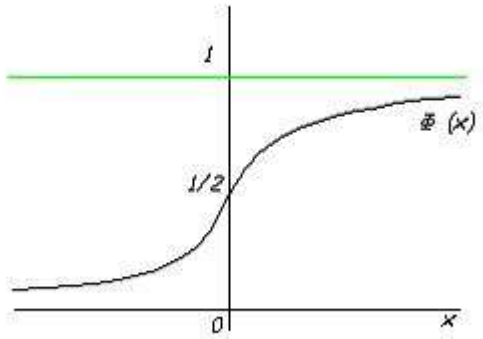
   $$\zeta^* = \zeta + \frac{1}{240}\left(\zeta^3 - 3\zeta\right)$$

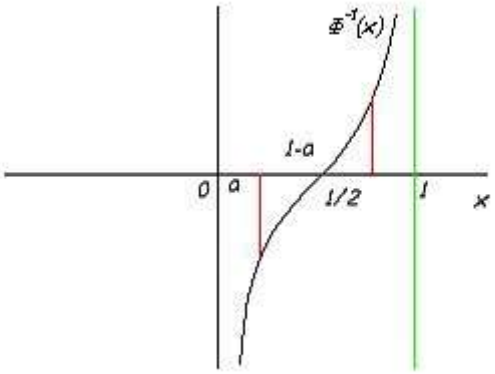
   $$F_\zeta(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du = \Phi(x)$$

2. Метод обратной функции - интеграл вероятностей или функция Лапласа. Свойство: $\Phi(x) = 1 - \Phi(-x)$

3. Метод обратной функции: $\zeta = F_\zeta^{-1}(\alpha)$.

Очевидно, что $\Phi^{-1}(\alpha) = -\Phi^{-1}(1-\alpha)$




$$\zeta = \begin{cases} \Phi^{-1}(\alpha), \text{ если } \dfrac{1}{2} \le \alpha < 1, \\ -\Phi^{-1}(\alpha), \text{ если } 0 < \alpha < \dfrac{1}{2}. \end{cases}$$

Т. о. $\Phi^{-1}$ заменяют аппроксимациями, например:

$$\Phi^{-1}(\alpha) = \frac{2.30753 + 0.27061 \cdot \theta}{1 + 0.99229 \cdot \theta + 0.04481 \cdot \theta^2} - \theta$$

где $\theta = \sqrt{-\ln \alpha}$

(Погрешность=0.003).

# УПРАВЛЕНИЕ МОДЕЛЬНЫМ ВРЕМЕНЕМ

## *Виды представления времени в модели*

Приступая к изучению механизмов управления модельным временем, уместно поговорить о том, какую роль вообще играет время в имитационном моделировании. При знакомстве с имитационным экспериментом мы отмечали, что он представляет собой наблюдение за поведением системы в течение некоторого промежутка времени. Конечно, далеко не во всех статистических испытаниях фактор времени *t* играет ведущую роль, а в некоторых и вообще может не рассматриваться. Вспомните, например,, задачу о вычислении площади круга: полученный результат не зависел от того, сколь долго мы «бомбили» квадрат случайными точками (речь в данном случае не идет о количестве этих точек). Но значительно больше задач, в которых оценка эффективности моделируемой системы напрямую связана с временными характеристиками ее функционирования. К ним относятся упоминавшиеся уже задачи по оценке производительности, некоторые задачи по оценке надежности, качества распределения ресурсов, а также все задачи, связанные с исследованием эффективности процессов обслуживания. Характерной особенностью большинства практических задач является то, что скорость протекания рассматриваемых в них процессов значительно ниже скорости реализации модельного эксперимента. Например, если моделируется работа вычислительного центра в течение недели, вряд ли кому-то придет в голову воспроизводить этот процесс в модели в таком же масштабе времени. С другой стороны, даже те имитационные эксперименты, в которых временные параметры работы системы не учитываются, требуют для своей реализации определенных затрат времени работы компьютера.

В связи с этим при разработке практически любой имитационной модели и планировании проведения модельных экспериментов необходимо соотносить между собой три представления времени:

1. ***реальное*** время, в котором происходит функционирование имитируемой системы;
2. ***модельное*** (или, как его еще называют, системное) время, в масштабе которого организуется работа модели;
3. ***машинное*** время, отражающее затраты времени ЭВМ на проведение имитации.

С помощью механизма ***модельного*** времени решаются следующие задачи:

- отображается переход моделируемой системы из одного состояния в другое;
- производится синхронизация работы компонент модели;
- изменяется масштаб времени «жизни» (функционирования) исследуемой системы;
- производится управление ходом модельного эксперимента;
- моделируется квазипараллельная реализация событий в модели.

Приставка «квази» в данном случае отражает последовательный характер обработки событий (процессов) в ИМ, которые в реальной системе возникают (протекают) одновременно.

Необходимость решения последней задачи связана с тем, что в распоряжении исследователя находится, как правило, однопроцессорная вычислительная система, а модель может содержать значительно большее число одновременно работающих подсистем. Поэтому действительно параллельная (одновременная) реализация всех компонент модели невозможна. Даже если используется так называемая распределенная модель, реализуемая на нескольких узлах вычислительной сети, совсем не обязательно, что число узлов будет совпадать с числом одновременно работающих компонент модели. Немного забегая вперед, следует отметить, что реализация квазипараллельной работы компонент модели является достаточно сложной технической задачей. Некоторые возможные методы ее решения рассматриваются в следующей лекции.

Существуют два метода реализации механизма модельного времени — с постоянным шагом и по особым состояниям.

Выбор метода реализации механизма модельного времени зависит от назначения модели, ее сложности, характера исследуемых процессов, требуемой точности результатов и т. д.

## *Изменение времени с постоянным шагом*

При использовании данного метода отсчет системного времени ведется через фиксированные, выбранные исследователем интервалы времени. События в модели считаются наступившими в момент

окончания этого интервала. Погрешность в измерении временных характеристик системы в этом случае зависит от величины шага моделирования $\Delta t$.

Метод постоянного шага целесообразно использовать в том случае, если:

- события появляются регулярно, их распределение во времени достаточно равно-| мерно;
- число событий велико и моменты их появления близки;
- невозможно заранее определить моменты появления событий.

Данный метод управления модельным временем достаточно просто реализовать в том случае, когда условия появления событий всех типов в модели можно представить как функцию времени.

Пусть, например, событие состоит в том, что летящий самолет пересекает некоторый воздушный рубеж, расстояние до которого равно R. Если самолет движется по прямой с постоянной скоростью V, можно вычислять путь, пройденный самолетом, с интервалом времени $\Delta t$: $S=S+V\text{-}At$. Соответственно, событие считается наступившим, если выполняется условие $S>R$, а момент времени наступления события принимается равным $n\text{-}\Delta t$, где $n$ — номер шага моделирования, на котором условие стало истинным.

В общем виде алгоритм моделирования с постоянным шагом представлен на рис. 1 ($t_M$ — текущее значение модельного времени, $T_M$ — заданный интервал моделирования), а для рассмотренного выше примера с самолетом — на рис. 2. Обратите внимание на то, что в отличие от обобщенного алгоритма, в приведенном примере моделирование завершается не по истечении заданного интервала времени, а при наступлении интересующего нас события. В связи с этим необходимо еще раз подчеркнуть, что при моделировании с постоянным шагом результат моделирования напрямую зависит от величины этого шага. Причем, если шаг будет слишком большим, то результат, скорее всего, будет неверным: момент окончания очередного шага очень редко будет совпадать с реальным моментом пересечения самолетом заданного рубежа. Такая ситуация показана на рис. 3.

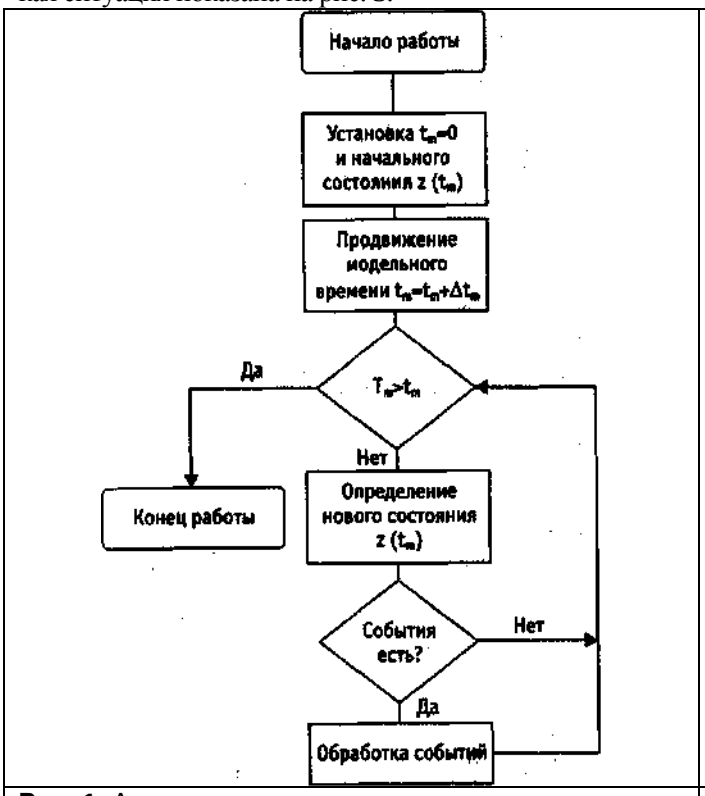



**Рис. 1.** Алгоритм моделирования с постоянным шагом.

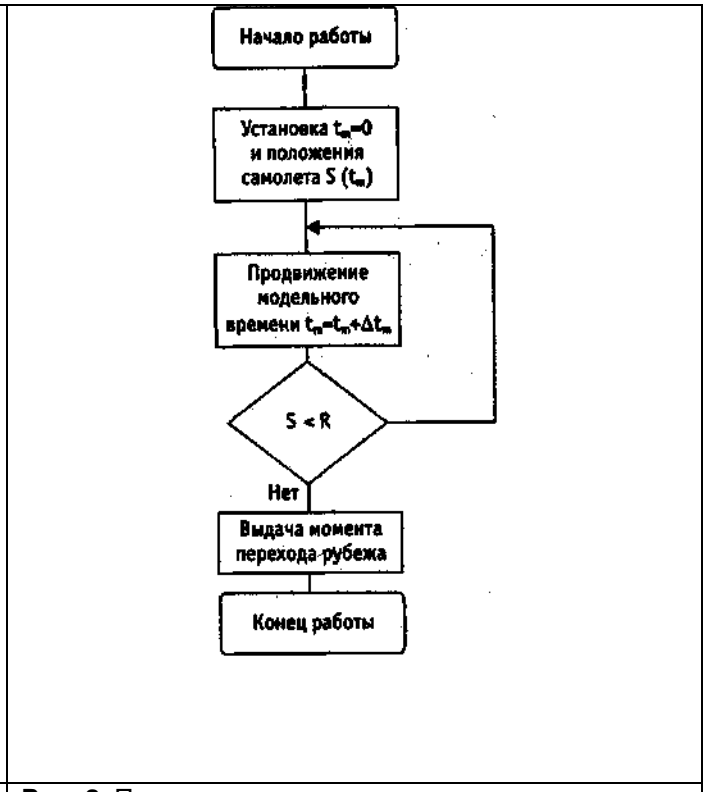



**Рис. 2.** Пример моделирования с постоянным шагом.

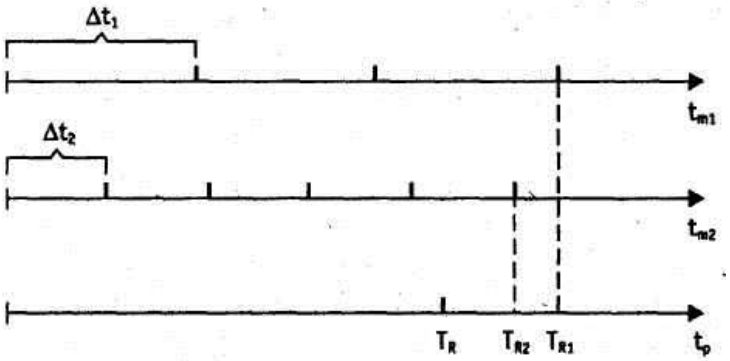



**Рис. 3.** Зависимость результата эксперимента от шага модельного времени

На рисунке использованы следующие обозначения:

- $t_{m1}$ —ось модельного времени при использовании шага $\Delta t_1$;
- $t_{m2}$ —ось модельного времени при использовании шага $\Delta t_2$
- $t_p$ — ось реального времени;
- $T_R$ — реальный момент пересечения самолетом рубежа; ;
- $T_{R1}$, $T_{R2}$ — моменты пересечения рубежа, полученные для соответствующих величин $\Delta t$.

Приведенный пример призван обратить внимание на то, что выбор величины шага моделирования является нелегким и очень важным делом. Универсальной методики решения этой проблемы не существует, но во многих случаях можно использовать один из следующих подходов:

- принимать величину шага равной средней интенсивности возникновения событий различных типов;
- выбирать величину шага равной среднему интервалу между наиболее частым (или наиболее важными) событиями.

## *Изменение времени по особым состояниям*

При моделировании по особым состояниям системное время каждый раз изменяется на величину, строго соответствующую интервалу времени до момента наступления очередного события. В этом случае события обрабатываются в порядке их наступления, а одновременно наступившими считаются только те, которые являются одновременными в действительности.

Для реализации моделирования по особым состояниям требуется разработка специальной процедуры планирования событий (так называемого календаря событий). Если известен закон распределения интервалов между событиями, то такое прогнозирование труда не составляет: достаточно к текущему значению модельного времени добавить величину интервала, полученную с помощью соответствующего датчика.

Пусть, например, за летящим самолетом, фигурировавшем при описании моделирования с постоянным шагом, наблюдает диспетчер. Он вводит информацию о самолёте, причем интервалы между вводом двух соседних сообщений являются случайными величинами, распределенными по нормальному закону с заданными параметрами Иллюстрация к такой ситуации приведена на рис. 4 ($T_c$ — момент ввода очередного сообщения, $\Delta t$—случайный интервал).

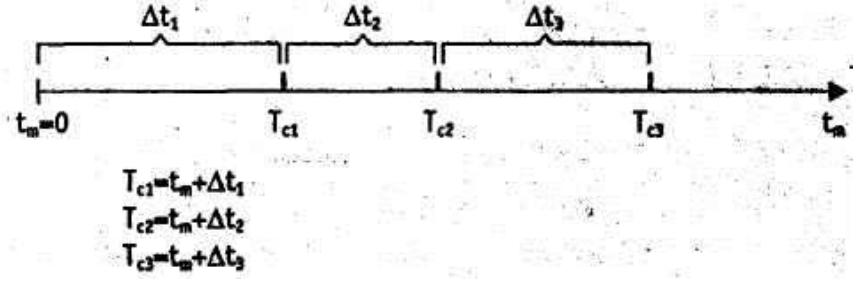



**Рис. 4.** Изменение модельного времени по особым состояниям.

Если же момент наступления события определяется некоторыми логическими условиями, то необходимо сформулировать эти условия и проверять их истинность для каждого последующего шага моделирования. Практика показывает, что сложности в реализации механизма изменения вре-

мени по особым состояниям связаны в первую очередь с корректным описанием таких условий. Трудности еще более возрастают, если в модели фигурируют несколько типов взаимосвязанных событий.

Моделирование по особым состояниям целесообразно использовать, если:

- события распределяются во времени неравномерно или интервалы между ними велики;
- предъявляются повышенные требования к точности определения взаимного положения событий во времени;
- необходимо учитывать наличие одновременных событий.

Дополнительное достоинство метода заключается в том, что он позволяет экономить машинное время, особенно при моделировании систем периодического действия, в которых события длительное время могут не наступать.

Обобщенная схема алгоритма моделирования по особым состояниям представлена на рис. 5 ($t_{соб.i}$ — прогнозируемый момент наступления *i*-го события).

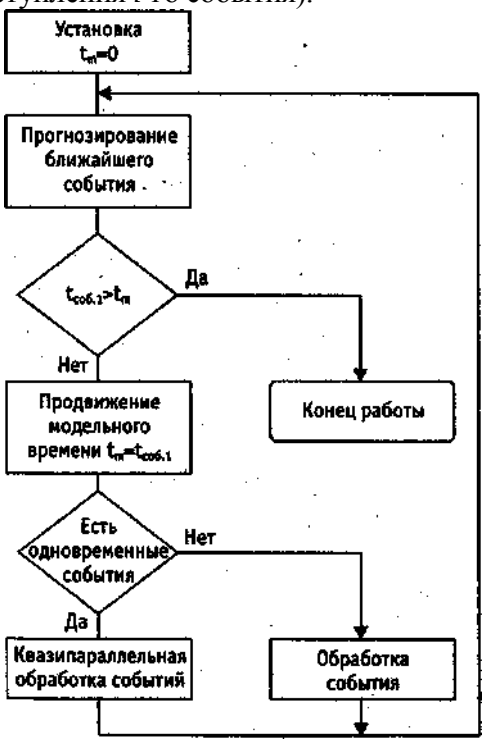



**Рис. 5.** Алгоритм моделирования по особым состояниям

Подведем итоги изложенному в этом разделе.

- Выбор механизма изменения модельного времени определяет технологию реализации имитационной модели.
- На выбор метода моделирования влияет целый ряд факторов, однако определяющим является тип моделируемой системы: для дискретных систем, события в которых распределены во времени неравномерно, более удобным является изменение модельного времени по особым состояниям.
- Если в модели должны быть представлены компоненты реальной системы, работа котЬрых измеряется в разных единицах времени, то они должны быть предварительно приведены к единому масштабу.

# МОДЕЛИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ

Практически любая более или менее сложная система имеет в своем составе компоненты, работающие одновременно, или как принято говорить в технических науках, параллельно. Напомним только один пример, взятый из монолога М. Жванецкого, посвященного работе ликеро-водочного завода. Судя по сюжету монолога, все цеха этого предприятия работали абсолютно параллельно и независимо друг от друга.

Итак, если в составе системы имеются компоненты (подсистемы), выполняющие свои функции одновременно, то можно утверждать, что в такой системе существуют параллельные процессы. Параллельно работающие подсистемы могут взаимодействовать самым различным образом, либо вообще работать независимо друг от друга. Способ взаимодействия подсистем определяет вид параллельных процессов, протекающих в системе. В свою очередь, вид моделируемых процессов влияет на выбор метода их имитации.

## *Виды параллельных процессов*

***Асинхронный параллельный процесс*** — это такой процесс, состояние которого не зависит от состояния другого параллельного процесса (ПП).

Пример асинхронных ПП, протекающих в рамках одной системы, — подготовка и проведение рекламной кампании фирмой и работа сборочного конвейера. Или пример из области вычислительной техники — выполнение вычислений процессором и вывод информации на печать.

***Синхронный ПП*** — это такой процесс, состояние которого зависит от состояния взаимодействующих с ним ПП. Пример синхронного ПП — работа торговой организации и доставка товара со склада (нет товара—нет торговли).

Один и тот же процесс может быть синхронным по отношению к одному из активных ПП и асинхронным по отношению к другому. Так, при работе Вычислительной сети по технологии «клиент-сервер» каждый из узлов сети синхронизирует свою работу с работой сервера, но не зависит от работы других узлов.

***Подчиненный ПП*** создается и управляется другим процессом (более; высокого уровня). Весьма характерным примером таких процессов является ведение боевых действий подчиненными подразделениями.

***Независимый ПП*** — процесс, который не является, подчиненным ни для одного из процессов. Скажем, после запуска неуправляемой зенитной ракеты ее полет можно рассматривать как независимый процесс, одновременно с которым самолет ведет боевые действия другими средствами.

Способ организации параллельных процессов в системе зависит от физической сущности этой системы.

Остановимся несколько подробнее на особенностях реализации параллельных процессов в вычислительных системах (ВС). Это обусловлено следующей причиной. Разработка и использование любой ИМ предполагает ее программную реализацию и исследование с применением ВС. Поэтому для реализации моделей, имитирующих параллельные процессы, в некоторых случаях применимы механизмы, характерные для выполнения параллельных вычислений.

Вместе с тем, реализация параллельных процессов в ВС имеет свои особенности;

- на уровне задач вычислительные процессы могут быть истинно параллельными только в многопроцессорных ВС или вычислительных сетях;
- многие ПП используют одни и те же ресурсы, поэтому даже асинхронные ПП в пределах одной ВС вынуждены согласовывать свои действия при обращении к общим ресурсам;
- в ВС дополнительно используется еще два вида; ПП: родительский и дочерний ПП; особенность их состоит в том, что процесс-родитель не может быть завершен, пока не завершатся все его дочерние процессы.

В силу перечисленных особенностей для организации взаимодействия параллельных процессов в ВС используются три основных подхода:

- на основе «взаимного исключения»;
- на основе синхронизации посредством сигналов;

- на основе обмена информацией (сообщениями).

*«Взаимное исключение»* предполагает запрет доступа к общим ресурсам (общим данным) для всех ПП, кроме одного, на время его работы с этими ресурсами (данными).

*Синхронизация* подразумевает обмен сигналами между двумя или более процессами по установленному протоколу. Такой «сигнал» рассматривается как некоторое событие, вызывающее у получившего его процесса соответствующие действия.

Часто возникает необходимость передавать от одного ПП другому более подробную информацию, чем просто «сигнал-событие». В этом случае процессы согласуют свою работу на основе *обмена сообщениями.*

Перечисленные механизмы реализуются в ВС на двух уровнях – системном и прикладном.

Механизм взаимодействия между ПП на системном уровне определяется еще на этапе разработки ВС и реализуется в основном средствами операционной системы (частично — с использованием аппаратных средств).

На прикладном уровне взаимодействие между ПП реализуется программистом средствами языка, на котором разрабатывается программное обеспечение.

Наибольшими возможностями в этом отношении обладают так называемые языки реального времени и языки моделирования.

Языки реального времени (ЯРВ) — это языки, предназначенные для создания программного обеспечения, работающего в реальном масштабе времени, например для разработки различных автоматизированных систем управления (предприятием, воздушным движением и т. д.). К ним, в частности, относятся: язык Ада, язык Модула и практически единственный отечественный язык реального времени — Эль-76 (использовавшийся в многопроцессорных вычислительных комплексах семейства «Эльбрус»).

## *Методы описания параллельных процессов*

Языки моделирования по сравнению с языками реального времени требуют от разработчика значительно менее высокого уровня подготовки в области программирования, что обусловлено двумя обстоятельствами:

- средства моделирования изначально ориентированы на квазипараллельную обработку параллельных процессов;
- механизмы реализации ПП относятся, как правило, к внутренней организации системы (языка) моделирования и их работа скрыта от программиста.

В практике имитационного моделирования одинаково широко используются как процессно-ориентированные языки (системы) моделирования, например SIMULA, так и языки, ориентированные на обработку транзактов (например, язык GPSS). Для тех и других характерны аналогичные методы реализации квазипараллелизма, основанные на ведении списков событий. В процессно-ориентированных системах используются списки событий следования, а в транзактных системах — списки событий изменения состояний.

Современные языки и системы моделирования, ориентированные на среду многозадачных операционных систем типа Windows, частично используют аналогичные механизмы управления процессами, что делает их применение еще более эффективным. В пакете MATLAB также имеется собственный язык моделирования, и к нему полной мере можно отнести сказанное выше.

## Моделирование на основе транзактов

Рассмотрим общий механизм реализации ПП применительно к моделированию на основе транзактов. В этом случае под событием понимается любое перемещение транзакта по системе, а также изменение его состояния (обслуживается, заблокирован и т. д.)

Событие, связанное с данным транзактом, может храниться в одном из следующих списков:

- *Список текущих событий.* В этом списке находятся события, время наступления которых меньше или равно текущему модельному времени. События с «меньшим» временем связаны с перемещением тех транзактов, которые должны были начать двигаться, но были заблокированы.

- *Список будущих событий.* Этот список содержит события, время наступления которых больше текущего модельного времени, то есть события, которые должны произойти в будущем (условия наступления которых уже определены, например известно, что транзакт будет обслуживаться некоторым устройством 10 единиц времени).
- *Список прерываний.* Данный список содержит события, связанные с возобновлением обработки прерванных транзактов. События из этого списка выбираются том случае, если сняты условия прерывания.

Рассмотрим использование двух первых списков событий в динамике, при моделировании параллельных процессов.

В списке текущих событий транзакты расположены в порядке убывания приоритета соответствующих событий, а при равных приоритетах — в порядке поступления в список.

Каждое событие (транзакт) в списке текущих событий может находиться либо в активном состоянии, либо в состоянии задержки. Если событие активно, то соответствующий транзакт может быть продвинут по системе; если продвижение невозможно (например, из-за занятости устройства), то событие (и транзакт) переводится в состояние задержки.

Как только завершается обработка (продвижение) очередного активного транзакта, просматривается список задержанных транзактов, и ряд из них переводите активное состояние. Процедура повторяется до тех пор, пока в списке текущих событии не будут обработаны все активные события. После этого просматривает список будущих событий. Модельному времени присваивается значение, равное времени наступления ближайшего из этих событий. Данное событие заносится в список текущих событий. Затем просматриваются остальные события списка. Те из них, время которых равно текущему модельному времени, также переписываются в список текущих событий. Просмотр заканчивается, когда в списке остаются события, времена которых больше текущего модельного времени.

В качестве иллюстрации к изложенному рассмотрим небольшой пример. Пусть в систему поступают транзакты трех типов, каждый из которых обслуживается отдельным устройством. Известны законы поступления транзактов в систему и длительность их обслуживания. Таким образом, в системе существуют три параллельных независимых процесса ($P_1$, $P_2$, $P_3$).

Временная диаграмма работы системы при обслуживании одного транзакта каждого типа показана на рис. 7.

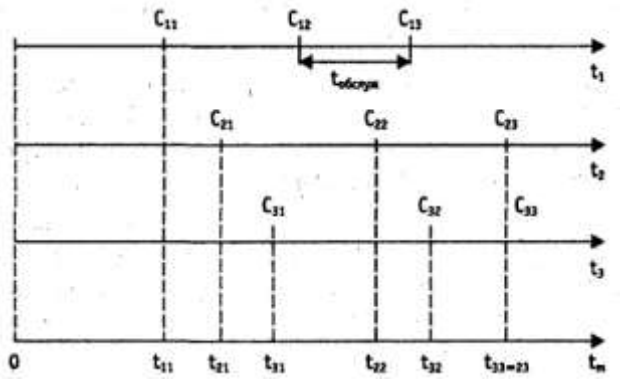



**Рис. 6.** Временная диаграмма параллельных процессов

На рисунке события, относящиеся к процессу $P_1$ обозначены как $C_{1i}$, относящиеся к $P_2$ и к $P_3$ — соответственно как $C_{2i}$ и $C_{3i}$. Время $t_{обслуж}$ соответствует времени обслуживания транзакта.

Для каждого процесса строится своя цепь событий, однако списки событий являются; общими для всей модели. Формирование списков начинается с заполнения списка будущих событий. Как было отмечено выше, в этот список помещаются события, время наступления которых превышает текущее значение модельного времени. Очевидно, что на момент заполнения списка время наступления прогнозируемых событий должно быть известно. На первом шаге $t_m$=0 в список будущих событий заносятся события $C_{11}$, $C_{21}$, $C_{31}$. Затем событие с наименьшим временем наступления — $C_{1Г}$ — переносится в список текущих событий; если одновременных с ним событий нет, то оно обрабатывается и исключается из

списка текущих событий. После этого вновь корректируется список будущих событий и т. д., пока не истечет заданный интервал моделирования.

Динамика изменения списков текущих и будущих событий для рассмотренного примера отражена в приведенной ниже таблице.

| Значения модельного времени | Список будущих событий | Список текущих событий |
|---|---|---|
| **0** | $C_{11}$ $C_{21}$ $C_{31}$ | **0** |
| $\mathbf{T_{11}}$ | $\mathbf{C_{21}}$ $\mathbf{C_{31}}$ $\mathbf{C_{12}}$ | $\mathbf{C_{11}}$ |
| $\mathbf{T_{21}}$ | $\mathbf{C_{31}}$ $\mathbf{C_{12}}$ $\mathbf{C_{22}}$ | $\mathbf{C_{21}}$ |
| $\mathbf{T_{31}}$ | $\mathbf{C_{12}}$ $\mathbf{C_{22}}$ $\mathbf{C_{32}}$ | $\mathbf{C_{31}}$ |
| $\mathbf{T_{12}}$ | $\mathbf{C_{22}}$ $\mathbf{C_{32}}$ $\mathbf{C_{13}}$ | $\mathbf{C_{12}}$ |
| $\mathbf{T_{22}}$ | $\mathbf{C_{32}}$ $\mathbf{C_{13}}$ $\mathbf{C_{23}}$ | $\mathbf{C_{22}}$ |
| $\mathbf{T_{32}}$ | $\mathbf{C_{13}}$ $\mathbf{C_{23}}$ $\mathbf{C_{33}}$ | $\mathbf{C_{32}}$ |
| $\mathbf{T_{13}}$ | $\mathbf{C_{23}}$ $\mathbf{C_{33}}$ | $\mathbf{C_{13}}$ |
| $\mathbf{T_{23}}$ | | $\mathbf{C_{23}}$ $\mathbf{C_{33}}$ |

Знание механизма ведения списков событий просто необходимо разработчику модели; умение проследить в динамике цепь происходящих в модели событий, во-первых, повышает уверенность создателя модели в том, что она работает правильно, и, во-вторых, существенно облегчает процесс отладки и модификации модели.

# СРЕДА MATLAB

## *Общие сведения*

История существования пакета M ATLAB, название которого происходит от словосочетания Matrix Laboratory (Матричная лаборатория) насчитывает уже более двух десятков лет. Можно считать, что развитие MATLAB «шло в ногу» с развитием средств вычислительной техники: от «больших» ЭВМ с маленькими интерактивными возможностями до настольных компьютеров, позволяющих использовать в работе все пять (а иногда и шесть) способов восприятия информации. И, несмотря на достаточно высокую скорость смены поколений вычислительной техники, пакет MATLAB успевал впитывать все наиболее ценное от каждого из них.

В результате к настоящему времени MATLAB представляет собой весьма удачное ; сочетание возможностей математики с последними достижениями в области вычислительной техники.

Одним из основных достоинств пакета MATLAB является то, что для работы пользователю достаточно знать о нем ровно столько, сколько требует решаемая задача. Так, в простейшем случае MATLAB может сыграть роль обыкновенного калькулятора, для использования которого достаточно помнить знаки математических операций. Если же решаемая задача требует создания каких-либо специальных инструментов, MATLAB предоставляет в распоряжение пользователя практически универсальный язык объектно-ориентированного программирования в сочетании с интерактивными средствами отладки создаваемых программ.'

И все-таки в первую очередь MATLAB — это средство математического моделирования, обеспечивающее проведение исследований практически во всех известных областях науки и техники. При этом структура пакета позволяет эффективно сочетать оба основных подхода к созданию модели: аналитический и имитационный.

Именно в сфере математического моделирования MATLAB позволяет наиболее полно использовать все современные достижения компьютерных технологий, в том числе средства визуализации и аудификации (озвучивания) данных, а также возможности обмена данными через Интернет. Кроме того, пользователь имеет возможность создавать средствами MATLAB собственный графический интерфейс, отвечающий как его вкусам, так и требованиям решаемой задачи. Как следует из названия пакета, он ориентирован в первую очередь на обработку массивов данных (матриц и векторов). Это позволило его разработчикам существенно повысить эффективность процедур, работающих с указанными типами данных, по сравнению с языками программирования «общего назначения» (Pascal, C и т. п.).

С точки зрения пользователя, MATLAB представляет собой богатейшую библиотеку функций (в MATLAB 5.3 их около 800), единственная проблема при работе с которой заключается в умении быстро отыскать те из них, которые нужны для решения данной задачи.

Для облегчения поиска библиотека функций разбита на разделы. Те из них, которые носят общий характер и используются чаще, входят в состав ядра MATLAB. Другие функции, относящиеся к конкретной области, включены в состав соответствующих специализированных разделов. Эти разделы называются в MATLAB *Toolboxes* (наборы инструментов). Каждый из них имеет свое собственное название, отражаю? * его предназначение. Полная комплектация пакета MATLAB 5.3 содержит около 30 наборов инструментов. В их число входят как достаточно стандартные для математических пакетов средства (решение дифференциальных и алгебраических уравнений, интегральное исчисление, символьные вычисления и т. д.), так и нетрадиционные, способные претендовать на определенную уникальность в своем роде: средства цифровой обработки изображений, поиска решений на основе нечеткой логики, аппарат построения и анализа нейронных сетей, средства финансового анализа, целый ряд других. Кроме того, имеются средства взаимодействия с популярными офисными продуктами компании Microsoft — MS Word и MS Excel. Для включения в состав рабочей конфигурации пакета того или иного набора инструментов требуется наличие соответствующего лицензионного соглашения.

## *Система визуального моделирования SIMULINK*

Simulink - инструмент для моделирования, анализа, и моделирования физических и математических систем, включая модели с нелинейными элементами и те, которые используют непрерывное и дискретное время.

Особое место среди наборов инструментов занимает система визуального моделирования SIMULINK. В определенном смысле SIMULINK можно рассматривать как самостоятельный продукт фирмы Math Works (который даже в некоторых случаях продается в «именной» упаковке), однако он работает только при наличии ядра MATLAB и использует многие функций, входящие в его состав.

Необходимо отметить, что в MATLAB использована технология ассоциативной обработки файлов, поддерживаемая операционной системой Windows. Она; заключается в том, что каждому типу файлов ставится в соответствие (ассоциируется с ним) определенное приложение, обеспечивающее обработку хранящихся в нем данных. Чтобы активизировать ассоциированное приложение, пользователю достаточно дважды щелкнуть на значке файла кнопкой мыши. Например, при выборе файла с расширением .doc загружается текстовый редактор MS Word.

Для MATLAB характерны файлы нескольких типов, для каждого из которых определен свой допустимый набор операций и реализующие их средства. При работе с SIMULINK в основном используются файлы трех типов:

- *M-файлы* (с расширением *.m*) — файлы, содержащие тексты программ на языке MATLAB; в виде M-файлов реализованы все библиотечные функции MATLAB; по умолчанию M-файлы открываются с помощью собственного редактора/отладчика MATLAB;
- *Mdl-файлы* (с расширением *.mdl*) — файлы моделей SIMULINK; могут быть открыты либо с помощью SIMULINK (в виде графического окна с блок-диаграммой), либо с помощью редактора/отладчика MATLAB;
- *MAT-файлы* (с расширением *.mat*) — файлы, содержащие данные в двоичном коде, доступ к которым возможен либо из командного окна MATLAB, либо с помощью специальных средств SIMULINK/

Система MATLAB не зависит от платформы и может работать под управлением и других операционных систем — UNIX и MacOS. При этом технология моделирования средствами SIMULINK остается неизменной.

Разработка моделей средствами SIMULINK (в дальнейшем S-моделей) основана на технологии drag-and-drop («перетащи и оставь»). В качестве «кирпичиков» для построения S-модели используются модули (или блоки), хранящиеся в библиотеке SIMULINK.

Библиотека SIMULINK хороша тем, что, с одной стороны, обеспечивает пользователю доступ ко всем основным возможностям пакета MATLAB, а с другой – является достаточно самостоятельной его компонентой, в том смысле, что при работе с ней не обязательно иметь навыки в использовании других инструментов, входящих в состав пакета.

Блоки, включаемые в создаваемую модель, могут быть связаны друг с другом как по информации, так и по управлению. Тип связи зависит от типа блока и логики работы модели. Данные, которыми обмениваются блоки, могут быть скалярными величинами, векторами или матрицами произвольной размерности.

## *Начало работы*

Запуск SIMULINK можно произвести одним из трех способов:

- щелкнув на соответствующей кнопке панели инструментов командного окна MATLAB;
- введя команду simulink в активной строке командного окна;
- выбрав команду New ► Model (создать ► модель) в меню File (файл).

Использование первого и второго способов приводит к открытию окна просмотра библиотеки SIMULINK , а при выборе команды New ► Model кроме него открывается еще и пустое окно для создания S-модели.

Окно просмотра разделов содержит панель инструментов, собственно список разделов, реализованный в виде дерева, и два вспомогательных поля, одно из которых используется для вывода комментария к выбранному в списке элементу, а другое — для представления значка этого элемента.

Список разделов библиотеки Simulink представлен в основном окне просмотра в виде дерева. Структура библиотеки Simulink:

- ❖ Основная библиотека;
- ➢ Разделы основной библиотеки;
- ▪ Блоки, входящие в разделы.
- ❖ Расширения основной библиотеки, относящиеся к наборам инструментов MATLAB.

Нижний уровень иерархии образуют собственно блоки SIMULINK, которые и играют роль кирпичиков при построении S-модели. Чтобы вставить блок в S-модель, необходимо нажать кнопку мыши на графической или текстовой метке блока и, не отпуская кнопку мыши, перетащить его в окно блок-диаграммы.

Строка меню окна блока-диаграммы содержит кроме общеизвестных меню ещё два:

Tools – инструменты;

Simulation – моделирование.

## *Демонстрация возможностей.*

Чтобы получить представление о том, что такое модель, разработанная с помощью SIMULINK, можно воспользоваться демонстрационными средствами MATLAB (команда demo в активной строке командного окна).

Пользователь имеет возможность выбрать один из следующих примеров, входящих в раздел Simulink-demo (демонстрационные файлы SIMULINK):

- New in Simulink 3 (новое в Simulink 3) — иллюстрация дополнительных возможностей версии Simulink 3.0, входящей в состав MATLAB 5.3, по сравнению с версиями Simulink 2.x, использовавшимися в предыдущих версиях MATLAB;
- New in Simulink 2 (новое в Simulink 2) — иллюстрация возможностей версий Simulink 2.x по сравнению с версией Simulink 1, входящей в состав MATLAB 4;
- Simple models (простые модели) — примеры простых S-моделей;
- Complex models (сложные модели) — примеры более сложных S-моделей;
- Advanced Products (дополнительные продукты) — вывод справочной информации по использованию RTW (Real Time Workshop — «Мастерская реального времени» — средство генерации кода на языке C из блок-диаграмм, используемое при разработке программного обеспечения систем реального времени).

При выделении одного из разделов в правой части окна MATLAB Demos выводится список входящих в него примеров S-моделей. В свою очередь, название отмеченной в этом списке модели отображается на расположенной ниже кнопке Run (выполнить). Щелчок по этой кнопке приводит к тому, что открывается окно, содержащее блок-диаграмму выбранной модели.

# Библиотека блоков Simulink.



## *Source – блоки-источники.*

1. **Band-Limited White Noise:** Белый шум для непрерывных (s-доменных) систем. Ограничение полосы частот с помощью фиксации нулевого порядка.
2. **Chirp Signal:** Вывод ЛЧМ-импульса (синусоидальная волна, частота которой изменяется линейно со временем).
3. **Clock:** Вывод текущего времени эмуляции.
4. **Constant:** Вывод константы, указанной параметром 'Значение константы'. Если 'Значение константы' - вектор и опция 'Интерпретировать параметры вектора как одномерные', то константа рассматривается как одномерный массив. Иначе - на выходе матрица с размерами - значениями константы.
5. **Digital Clock:** Вывод текущего времени эмуляции с указанной частотой.
6. **From Workspace:** Чтение значений данных, указанных в массиве или структурном формате из рабочей области MATLAB.
   6.1. Формат массива (или матрицы):
      6.1.1. одномерный сигнал:
      var=[ЗначенияВремени ЗначенияДанных]
      6.1.2. Для двумерного сигнала используйте структурный формат
   6.2. структурный формат:
      6.2.1. var.time=[ЗначенияВремени]
      6.2.2. var.signals.values=[ЗначенияДанных]
      6.2.3. var.signals.размеры=[ЗначенияМассива]
   Выберите интерполяцию для интерполяции или экстраполяции для шагов времени, для которых данные не определены.
7. **From File:** Чтение времени и выходных значений из первой матрицы в указанном MAT-файле. Матрица должна содержать значения времени в первой строке. Дополнительные строки соответствуют выходным элементам. Интерполяция между столбцов.
8. **Ground:** Используется для "заземления" входных сигналов. (Предотвращает предупреждения о неподключенных входных портах.) На выходе 0.
9. **In1:** Обеспечить подсистеме или модели входной порт. Для тиггерных подсистем, если указан пункт 'Запирать вход (буфер)', то блок входного порта синтезирует значение входа для предыдущего шага времени. Другие параметры могут быть использованы, чтобы явно указать атрибуты входного сигнала.
10. **Pulse Generator:** Генерировать импульсы с постоянными интервалами, здесь тип импульса определяет используемую методику вычислений.
    10.1. С временным критерием - рекомендуется для использования с решателем с переменным шагом,
    10.2. С критерием отсчета - для решателя с фиксированным шагом в дискретной части модели, использующей решатель с переменным шагом.
11. **Ramp:** Генерация скатового сигнала за указанное время.
12. **Random Number:** Генерация нормально (по Гауссу) распределенного случайного сигнала. Генерация воспроизводима для указанного зерна.
13. **Repeating Sequence:** Генерация повторяющейся последовательности чисел, указанных в таблице пар время-значение. Значения времени должны монотонно возрастать.
14. **Signal Generator:** Генерация различных волновых форм.
15. **Signal Builder:** Sigbuilder GUI
16. **Sine Wave:** Выводит синусоидальную волну, где тип синуса определяет используемую методику вычислений. Параметры двух типов связаны через:
    16.1. Отсчетов за период = 2*pi / (Частота * Время отсчета)
    16.2. Число отсчетов смещения = Фаза * Отсчетов за период / (2*Pi)

Используйте тип синуса С критерием отсчета при появлении численных проблем с многократным повторением (н-р, переполнение абсолютного времени).

17. **Step:** Генерация ступеньки.
18. **Uniform Random Number:** Генерация равномерно распределенного случайного сигнала. Генерация воспроизводима для указанного зерна.

## *Sinks – блоки-получатели.*

1. **Display:** Численное отображение входных значений.
2. **Floating Scope:** simulink/Sinks/Floating Scope
3. **Out1:** Обеспечить выходной порт для подсистемы или модели. Параметры 'Выход при отключении' и 'Начальный выход' применяются только к условно исполняемым подсистемам. Когда условно исполняемая подсистема отключена, выход либо находится в своем последнем значении, либо устанавливается в 'Начальный выход'. Параметр 'Начальный выход' может быть указан как пустая матрица [], в случае чего начальный выход равен выходу блока, подключенного к выходному порту.
4. **Scope:** simulink/Sinks/Scope
5. **Stop Simulation:** Остановить эмуляцию когда на входе не ноль.
6. **Terminator:** Используется для "завершения" выходных сигналов. (Предотвращает предупреждения о неподключенных выходных портах.)
7. **To File:** Записать время и вход в указанный MAT-файл в формате строк. Время - в строке 1.
8. **To Workspace:** Записать вход в указанный массив или структуру в главной рабочей области MATLAB. Данные не будут доступны, пока эмуляция остановлена.
9. **XY Graph:** Осциллограф XY с использованием окна графиков MATLAB. Первый вход используется как временная база. Введите диапазоны построения.

## *Continuous – непрерывные системы.*

1. **Derivative:** Численная производная: du/dt.
2. **Integrator:** Интегрирование входного сигнала в непрерывном времени.
3. **State-Space:** Модель в пространстве состояний:
   $$dx/dt = Ax + Bu$$
   $$y = Cx + Du$$
4. **Transfer Fcn:** Матричное выражение для числителя, векторное выражение для знаменателя. На выходе - ширина равна числу строк в числителе. Коэффициенты - для степеней s по убыванию.
5. **Transport Delay:** Применить указанную задержку к входному сигналу. Лучшая точность достигается, когда задержка больше шага эмуляции.
6. **Variable Transport Delay:** Применить задержку к первому входному сигналу. Второй вход указывает время задержки. Лучшая точность достигается, когда задержка больше шага эмуляции.
7. **Zero-Pole:** Матричное выражение для нулей. Векторное выражение для полюсов и k передачи. На выходе - ширина равна числу столбцов в матрице нулей, или единице, если нули - вектор.

## *Discontinuities – разрывные системы.*

1. **Backlash:** Моделировать зазор, где ширина мертвой зоны указывает величину зазора в системе.
2. **Coulomb & Viscous Friction:** Скачок разрыва в нулевых моделях сухого трения. Вязкое трение моделей с линейным коэффициентом передачи.
   $$y = sign(x) * (Gain * abs(x) + Offset)$$
3. **Dead Zone:** На выходе ноль для входного сигнала в зоне нечувствительности. Сдвиг входных сигналов на значение Начала или Конца за пределами зоны нечувствительности.
4. **Hit Crossing:** Сравнивает входной сигнал со значением перехода через смещение. Если сигнал взрастает выше, опускается ниже или остается на значении смещения, блок выдает 1. Иначе блок выдает 0. Для решателей с переменным шагом, Simulink делает временной шаг перед и после момента перехода.
5. **Quantizer:** Дискретизировать вход на заданном интервале.
6. **Rate Limiter:** Ограничить скорость возрастания и убывания сигнала.
7. **Relay:** Вывод значения 'вкл' или 'выкл', полученного путем сравнения входа с указанными порогами. Состояние реле вкл/выкл не зависит от значения входа между верхним и нижним пределом.
8. **Saturation:** Ограничить входной сигнал верхней и нижней величиной насыщения.

## *Discrete – дискретные системы.*

1. **Discrete Transfer Fcn:** Матричное выражение для числителя, векторное выражение для знаменателя. Выходная ширина равняется числу строк в числителе. Коэффициенты - для степеней z по убыванию.
2. **Discrete Zero-Pole:** Матричное выражение для нулей. Векторное выражение для полюсов и коэффициента передачи. Выходная ширина равняется числу столбцов в матрице нулей или единице, если нули - вектор.
3. **Discrete Filter:** Векторное выражение для числителя и знаменателя. Коэффициенты - для степеней 1/z по возрастанию.

4. **Discrete State-Space:** Модель с дискретным пространством состояний:
   x(n+1) = Ax(n) + Bu(n)
   y(n) = Cx(n) + Du(n)
5. **Discrete-Time Integrator:** Интегрирования входного сигнала в дискретном времени.
6. **First-Order Hold:** Фиксация первого порядка.
7. **Memory:** Применить задержку одного шага интегрирования. На выходе - предыдущее значение входа.
8. **Unit Delay:** Образец и фиксация с задержкой в один образцовый период.
9. **Zero-Order Hold:** Фиксация нулевого порядка.

## *Look-Up Tables – работа с таблицами.*

1. **Direct Look-Up Table (n-D):** Выбор члена таблицы. На входах - индексы таблицы с отсчетом от нуля. Блок может также использоваться для выбора из таблицы столбца или двумерной матрицы. Первый индекс выбора соответствует верхнему (или левому) входному порту.
2. **Interpolation (n-D) using PreLook-Up:** Осуществить n-мерный поиск по таблице интерполяции с использованием заранее рассчитанных индексов и дробей расстояний. n-мерная таблица - дискретное представление функции N переменных. Этот блок должен быть подключен к выходу блока Предварительного Поиска Индексов. Первая размерность соответствует верхнему (или левому) входному порту.
3. **Look-Up Table:** Выполнить одномерную линейную интерполяцию входных значений использую указанную таблицу. За границами таблицы выполняется экстраполяция.
4. **Look-Up Table (2-D):** Выполняет двухмерную линейную интерполяцию входных значений, используя заданную таблицу. Экстраполяция выполняется за пределами таблицы..
5. **Look-Up Table (n-D):** Осуществить n-мерный поиск по таблице интерполяции, включая поиск индексов. Таблица - дискретное представление функции N переменных. Наборы контрольных точек определяют отношение входные значения к позициям в таблице. Первая размерность соответствует верхнему (или левому) входному порту.
6. **PreLook-Up Index Search:** Вычислить относительное положение входного значения в диапазоне чисел (наборе "данных контрольных точек"). Возвращает массив индексов интервала "k" и дробь расстояния "f", которой достигает вход "u" в k-м интервале.

## *Math Operations – математические операторы.*

1. **Abs:** y = |u|
2. **Algebraic Constraint:** Приравнять входной сигнал f(z) к нулю и вывести алгебраическое значение z. Этот блок выводит значение, необходимое для получения нуля на входе. Выход должен воздействовать на вход через некоторую обратную связь. Введите Начальное Предположение выхода, чтобы улучшить эффективность циклического решателя.
3. **Assignment:** Для векторного режима:
   Y = U1
   Y(E) = U2
   Для матричного режима:
   Y=U1
   Y(R,C) = U2 ,где
   U1 = первый входной порт, U2 = второй входной порт, E = элементы, R = строки, а C = столбцы и E, R, и C могут быть указаны в диалоге блока или через внешний входной порт.
4. **Bitwise Logical Operator:** Выполняет побитовую операцию над данными входного порта вида uint8, uint16 или uint32 со значениями Второго операнда. Шестнадцатеричные значения могут быть введены как строки, н-р, 'FF00'.
5. **Combinatorial Logic:** Искать элементы входного вектора (рассматриваемые как булевые значения) в таблице истинности и вывести соответствующую строку параметра 'Таблица истинности'. Входная сторона таблицы истинности - неявная.
6. **Complex to Magnitude-Angle:** Рассчитать модуль и/или аргумент входного сигнала.
7. **Complex to Real-Imag:** Вывести действительную и/или мнимую компоненту входа.
8. **Dot Product:** Внутреннее (точечн.) произведение.

y = sum(conj(u1).*u2). Операнд u1 соответствует верхнему (или левому) входному порту.

9. **Gain:** Поэлементный (y = K.*u) или матричный (y = K*u or y = u*K) k передачи.
10. **Logical Operator:** Логические операторы. Для одиночного входа операторы применяются ко входному вектору. Для множественных входов операторы применяются ко входам.
11. **Magnitude-Angle to Complex:** Создать комплексный выход из модуля и/или аргумента входа.
12. **Math Function:** Математические функции включают логарифмические, экспоненциальные, функции модуля и степени. Если функция имеет более одного аргумента, первый аргумент соответствует верхнему (или левому) входному порту.
13. **Matrix Concatenation:** Выполнить горизонтальное или вертикальное сцепление. Одномерный вектор входных сигналов рассматривается как векторы столбцов, т.е. матрицы [Mx1]. На выходе всегда матрица.
14. **Matrix Gain:** Поэлементный (y = K.*u) или матричный (y = K*u or y = u*K) k передачи.

15. **MinMax:** Вывести минимум или максимум входного сигнала. Для одиночного входа, операторы применяются ко входному вектору. Для множественных входов, операторы применяются ко входам.
16. **Polynomial:** Полиномиальная оценка. Вычисляет P(u) заданный массивом полиномиальных коэффициентов P. P - сортированный по убыванию порядков, в форме, принятой для функции MATLAB polyval.
17. **Product:** Умножить или разделить входы. Выберите поэлементное или матричное произведение и выберите один вариант из следующих

a) * или / для каждого входного порта (н-р, **/*)
b) скаляр указывает число входных портов для перемножения

Скалярное значение '1' для поэлементного произведения приводит к перемножению всех элементов входного вектора.

Если для матричного произведения указано /, считается обратное соответствующему входу.

18. **Real-Imag to Complex:** Создать комплексный выход из действительного и/или мнимого входа.
19. **Relational Operator:** Применить выбранный оператор отношения ко входам и вывести результат. Верхний (или левый) вход соответствует первому операнду.
20. **Reshape:** Изменить размеры вектора или матрицы входного сигнала. На выходе
    a. - одномерный массив (вектор),
    b. - вектор столбца (матрица Mx1),
    c. - вектор строки (матрица 1xN), или
    d. - матрица или вектор с указанными размерами, н-р, [M, N] или [W].
21. **Rounding Function:** Операции округления.
22. **Sign:** На выходе 1 для положительного входа, -1 для отрицательного входа и 0 для нулевого. y = signum(u)
23. **Slider Gain:** simulink/Math Operations/Slider Gain
24. **Sum:** Добавить или вычесть входы. Укажите один из вариантов:

a) содержащий строку + или - для каждого входного порта, | для разделителя между портами (н-р ++|-|++)
b) скаляр >= 1. Значение >1 суммирует все входы; 1 суммирует элементы одного входного вектора

25. **Trigonometric Function:** Тригонометрические и гиперболические функции. Когда функция имеет более одного аргумента, первый аргумент соответствует верхнему (или левому) входному порту.

## *Model Verification – проверка модели*

1. **Assertion:** Утверждение, что входной сигнал не нулевой. Поведение по умолчанию в отсутствии вызова выводит сообщение об ошибке.
2. **Check Discrete Gradient:** Утверждение, что абсолютное значение разницы между сэмплами дискретного сигнала меньше верхней границы.
3. **Check Dynamic Gap:** Утверждение, что входной сигнал 'sig' всегда меньше нижней границы 'min' или выше верхней границы 'max'. Первое входное значение - верхняя грань зазора; второе входное значение - нижняя; третье входное значение - сигнал для тестирования.
4. **Check Dynamic Range:** Утверждение, что один сигнал всегда лежит между двумя другими. Первое входное значение - сигнал верхней грани; второе входное значение - сигнал нижней грани; третье входное значение - сигнал для тестирования.
5. **Check Static Gap:** Утверждение, что входной сигнал меньше чем (или по выбору равен) нижняя граница или больше чем (или по выбору равен) верхней границе.
6. **Check Static Range:** Утверждение, что входной сигнал лежит между статическими нижней и верхней границами или по выбору равен одной из них.
7. **Check Dynamic Lower Bound:** Утверждение, что один сигнал всегда меньше другого. Первое входное значение - сигнал нижней грани. Второе входное значение - сигнал для тестирования.
8. **Check Dynamic Upper Bound:** Утверждение, что один сигнал всегда больше другого. Первое входное значение - сигнал верхней грани. Второе входное значение - сигнал для тестирования.
9. **Check Input Resolution:** Утверждение, что входной сигнал имеет заданное разрешение. Если разрешение скалярное, входной сигнал должен быть множеством разрешений в пределах отклонения 10e-3. Если разрешение - вектор, входной сигнал должен быть равен элементу вектора разрешения.
10. **Check Static Lower Bound:** Утверждение, что входной сигнал больше чем (или по выбору равен) статической нижней границе.
11. **Check Static Upper Bound:** Утверждение, что входной сигнал меньше чем (или по выбору равен) статической верхней границе.

## *Model-Wide Utilities: - широкие возможности обслуживания модели*

1. **DocBlock:** Используйте этот блок, чтобы сохранить длинный текст описания с моделью. Двойное нажатие блока откроет редактор.
2. **Model Info:** Этот блок позволяет показывать вместе с моделью информацию о ее редакции.
3. **Timed-Based Linearization:** Генерировать линейные модели в рабочей области в заданное время.
4. **Trigger-Based Linearization:** Генерировать линейные модели в рабочей области по запуску.

## *Ports & Subsystems – Порты и Подсистемы*

1. **Configurable Subsystem:** Выбрать настройки для блока подсистемы .
2. **Atomic Subsystem:** Шаблон блока подсистемы, содержащий блоки inport и outport.
3. **Enable:** Поместите этот блок в подсистему, чтобы создать действующую подсистему.
4. **Enabled Subsystem:** Шаблон блока подсистемы, содержащий блоки enable port, inport и outport.
5. **Enabled and Triggered Subsystem:** Шаблон блока подсистемы, содержащий блоки enable port, trigger port, inport и outport.
6. **For Iterator Subsystem:** Шаблон блока подсистемы, содержащий итератор for, блоки inport и outport.
7. **Function-Call Generator:** Этот блок реализует операцию итератора. На каждом шаге времени, как указано в поле отсчета времени, этот блок выполняет подсистему(-ы) вызова функции, которой он управляет указанное число раз.

Демультиплексируйте выход блока, чтобы выполнить несколько подсистем вызова функций в заданном порядке. Система, соединенная с первым портом демультиплексора, выполняется первой, со вторым - вторая и т.д..

8. **Function-Call Subsystem:** Шаблон блока подсистемы, содержащий порт вызова функции, блоки inport и outport.
9. **If:** Выражение IF
   Запустить Подсистему действия присоединенную к первому выходному порту
   9.1. Выражение ELSEIF
   Запустить Подсистему действия присоединенную ко второму выходному порту
   9.2. ELSE
   Запустить Подсистему действия присоединенную к последнему выходному порту
   9.3. END

Количество выходных портов Elseif в блоке равно числу разделенных запятыми Elseif выражений введенных в диалоге. If и Elseif выражения могут использовать эти операторы MATLAB :

<, <=, ==, ~=, >, >=, , |, ~, ( ), унарный минус во входных сигналах порта u1, u2, u3, и др.

10. **If Action Subsystem:** Шаблон блока подсистемы содержащий порт действия, блоки входного и выходного портов.
11. **In1:** Обеспечить подсистеме или модели входной порт. Для тиггерных подсистем, если указан пункт 'Запирать вход (буфер)', то блок входного порта синтезирует значение входа для предыдущего шага времени. Другие параметры могут быть использованы, чтобы явно указать атрибуты входного сигнала.
12. **Out1:** Обеспечить выходной порт для подсистемы или модели. Параметры 'Выход при отключении' и 'Начальный выход' применяются только к условно исполняемым подсистемам. Когда условно исполняемая подсистема отключена, выход либо находится в своем последнем значении, либо устанавливается в 'Начальный выход'. Параметр 'Начальный выход' может быть указан как пустая матрица [], в случае чего начальный выход равен выходу блока, подключенного к выходному порту.
13. **Subsystem:** Шаблон блока подсистемы, содержащий блоки inport и outport.
14. **Subsystem Examples:** Это примеры использования различных типов подсистем.
15. **Switch Case:** Выполнить операцию switch-case для входа. Вход должен быть скаляром и его значение будет усечено. Условия case указываются с использованием нотации ячеек MATLAB, где каждый case - элемент-ячейка. Н-р, ввод {1,[2,3]} в качестве условия означает, что порт 1 открыт, когда вход = 1, а порт 2 открыт, когда вход = 2 либо 3. Если указан default case (по умолчанию), то порт 3 открыт для всех остальных входов.
16. **Switch Case Action Subsystem:** Шаблон блока подсистемы содержащий порт действия, блоки входного и выходного портов.
17. **Trigger:** Поместите этот блок в подсистему, чтобы создать переключаемую подсистему.
18. **Triggered Subsystem:** Шаблон блока подсистемы содержащий порт триггера, блоки входного и выходного портов.
19. **While Iterator Subsystem:** Шаблон блока подсистемы содержащий итератор while, блоки входного и выходного портов.

## *Signal Attributes – признаки сигнала*

1. **Data Type Conversion:** Преобразовать входной сигнал к указанному типу данных.
2. **IC:** Начальное условие для сигнала.
3. **Probe:** Исследовать линию на ее ширину, время выборки, комплексный сигнал или размеры.
4. **Rate Transition:** Определить механизм передачи данных между двумя уровнями многоуровневой системы. Стандартная конфигурация предоставляет безопасный и определенный трансфер данных.
5. **Signal Specification:** Указать атрибуты линии сигнала.
6. **Width:** Вывести ширину входного сигнала используя определенный тип выходных данных.

## *Signal Routing – направление сигнала*

1. **Bus Creator:** Этот блок создает сигнал шины используя входные значения.
2. **Bus Selector:** This block accepts input from a Mux or Bus Selector block. The left listbox shows the signals in the input bus. Use the Select button to select the output signals. The right listbox shows the selections. Use the Up, Down, or Remove button to reorder the selections. Check 'Muxed output' to multiplex the output.

3. **Data Store Memory:** Определить область памяти для использования блоками чтения и записи данных. Все блоки чтения и записи данного уровня системы или ниже и имеющие одинаковое имя расположения памяти смогут читать или писать в этот блок.
4. **Data Store Read:** Чтение значений из указанного источника данных.
5. **Data Store Write:** Запись значений в указанный источник данных.
6. **Demux:** Разбить либо (а) векторные сигналы на скаляры или меньшие векторы, или (б) сигналы шины, сгенерированные блоком мультиплексора на составляющие скалярные, векторные или матричные сигналы.

Включите 'Режим выбора шины' для разбиения сигналов шины.

7. **From:** Получить сигналы с блока Перехода с указанным тегом. Если тег задан как 'ограниченный' в блоке Перехода, то для задания видимости тега должен использоваться блок Видимости тега перехода. После 'Обновления диаграммы' значок блока будет показывать имя выбранного тега (локальные теги заключены в кв.скобки [], а имена ограниченных тегов заключены в фигурные скобки {}).
8. **Goto:** Послать сигналы с блока Из, имеющего указанный тег. Если тег задан как 'ограниченный' в блоке Перехода, то для задания видимости тега должен использоваться блок Видимости тега перехода. Значок блока показывает имя выбранного тега (локальные теги заключены в кв.скобки [], а имена ограниченных тегов заключены в фигурные скобки {}).
9. **Goto Tag Visibility:** Используется в сочетании с блоками Переход и Из для задания видимости ограниченных тегов. Например, если этот блок находится в подсистеме (или корневой системе) с названием MYSYS, то тег видим для блоков Из, находящихся в MYSYS или в подсистемах MYSYS.
10. **Manual Switch:** Выход переключается между двумя входами двойным щелчком по блоку.
11. **Merge:** Объединить входные сигналы в одиночный выходной сигнал, начальное значение которого указано в параметре 'Начальный выход'. Если 'Начальный выход' пуст, на выходе блока Слияние - начальный выход одного из его составляющих блоков.
12. **Multiport Switch:** Пройти через входные сигналы соответственно усеченному значению первого ввода. .
13. **Mux:** Мультиплексировать скалярные, векторные или матричные сигналы в шину.
14. **Selector:** Выбрать или переупорядочить выбранные элементы входного вектора или матрицы.
    y=u(элементы), если на входе - вектор.
    y=u(строки,столбцы), если на входе - матрица
    Элементы (E), строки (R), и столбцы (C) могут быть указаны либо в диалоге блока, либо через внешний входной порт.
15. **Switch:** Пройти через ввод 1 когда ввод 2 удовлетворяет выбранному критерию; иначе через ввод 3..

## *User-Defined Functions – Определенные пользователем Функции*

1. **Fcn:** Блок общих выражений. Используйте "u" в качестве имени входной переменной.
2. Пример: sin(u[1] * exp(2.3 * -u[2]))
3. **MATLAB Fcn:** Пройти через входные значения к функции MATLAB для оценки.
4. **S-Function:** Пользовательский блок. Блоки могут быть написаны на M, C, Fortran или Ada и должны согласовываться со стандартами S-функций. t,x,u и flag автоматически передаются в S-функцию Simulink. "Дополнительные" параметры могут быть указаны в поле 'Параметры S-функции'.
5. **S-Function Builder:** simulink/User-Defined Functions/S-Function Builder