Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Н. Ю. Салмина

МОДЕЛИРОВАНИЕ СИСТЕМ

Часть I

Учебное пособие

Томск
«Эль Контент»
2013

УДК 519.866.5(075.8)
ББК 22.18я73
С 164

Рецензенты:
Тарасенко В. Ф., докт. техн. наук, профессор кафедры теоретической кибернетики Томского государственного университета;
Грибанова Е. Б., канд. техн. наук, доцент кафедры автоматизированных систем управления ТУСУРа.

Салмина Н. Ю.
С 164 Моделирование систем : учебное пособие / Н. Ю. Салмина. — Томск : Эль Контент, 2013. — 118 с.

ISBN 978-5-4332-0146-0

В настоящем учебном пособии изложены основные вопросы моделирования систем: классификация моделей, этапы моделирования. Рассматриваются задачи статистического моделирования, основные моменты планирования эксперимента и анализа результатов, а также некоторые вопросы теории массового обслуживания и теории игр. В качестве программного средства моделирования систем рассмотрен язык имитационного моделирования GPSS: синтаксис, особенности применения. Теоретический материал иллюстрируется многочисленными примерами.

Учебное пособие предназначено для студентов, обучающихся по направлению подготовки 231000 «Программная инженерия», и студентов родственных направлений.

УДК 519.866.5(075.8)
ББК 22.18я73

ISBN 978-5-4332-0146-0

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Повышение эффективности реализации процессов управления и планирования в различных отраслях производства, науки, культуры, социальной жизни требует все более широкого применения моделей систем, создаваемых с применением математических методов и средств информационно-вычислительной техники. В настоящее время полное и всестороннее исследование реальных систем невозможно без методов моделирования на ЭВМ. Моделирование на ЭВМ часто называют имитационным моделированием. Именно моделирование является средством, позволяющим без капитальных затрат решать проблемы исследования, построения больших систем, эффективного управления этими системами. Поэтому дисциплина «Моделирование систем» является одной из базовых дисциплин подготовки бакалавров по направлению 231000 — «Программная инженерия».

Основной задачей курса «Моделирование систем» является формирование у студентов профессиональных знаний и практических навыков по разработке и созданию различных моделей сложных систем, с целью их исследования. Курс знакомит студентов с этапами машинного моделирования, вопросами статистического моделирования, планирования эксперимента и двумя классами кибернетических моделей: игровых и моделей массового обслуживания.

Дисциплина «Моделирование систем» состоит из двух частей. В первой части учебного пособия рассматриваются основные понятия моделирования, некоторые вопросы статистического моделирования, планирование эксперимента и язык моделирования GPSS.

Во второй части учебного пособия рассматриваются два класса кибернетических моделей: модели массового обслуживания и игровые модели.

ВВЕДЕНИЕ

Управление в современном мире становится все более трудным делом, поскольку организационная структура общества усложняется. Исследованию подвергаются все более и более сложные системы, в которых изменение одной из характеристик может легко привести к изменениям во всей системе или создать потребность в изменениях в других частях системы. Соответственно возникает необходимость в использовании все более сложных методов научных исследований.

Известны три общих направления в научных исследованиях: экспериментальное и теоретическое исследования и математическое моделирование [1].

Экспериментальные исследования проводятся с реальным объектом в форме натурного эксперимента. В результате получают экспериментальные данные о поведении и свойствах объекта исследования. Известны две стратегии проведения натурного эксперимента: активная и пассивная. В первом случае экспериментатор имеет возможность изменять внешние условия, определяющие состояние объекта, во втором — такой возможности нет. В этом случае говорят о наблюдении за изменениями состояния объекта — исследования (социальные, природные) соответствующих объектов длительны и трудоемки.

Теоретические исследования. Теория имеет дело не с реальным объектом исследования, а с его идеализированным представлением (огрубленным), допускающим применение адекватного математического аппарата для формального описания объекта. Здесь исходной основой является некая совокупность фактов, полученная в результате наблюдений или натурного эксперимента. В результате теоретического осмысления фактов у исследователей возникает ряд гипотез, объясняющих поведение и свойства объекта. При последующей экспериментальной проверке одна из гипотез может принять форму научного закона. Однако этот традиционный путь формирования теоретического знания в случае сложных систем оказывается неприменимым, и роль законов выполняют математические методы. Разрабатывается математический аппарат теории, который обеспечивает возможность проведения исследования объекта аналитическими методами. Но и эти методы в реальных ситуациях часто оказываются неприменимыми. Причины: недостаточная разработанность математического аппарата; чрезмерно большая размерность решаемой задачи; наличие большого числа случайных факторов. Возникает необходимость использования ЭВМ.

Компьютерное моделирование. При использовании экспериментальных и теоретических методов исследования создается (накапливается) исходная информация

об объекте исследования, а *модель* используется как средство интеграции и хранения знаний об объекте исследования. Так, теоретические исследования проводятся с целью построения модели, а натурный эксперимент проводится с целью проверки адекватности модели. Здесь модель играет прикладную роль, она является и результатом исследования, и средством хранения знаний об объекте исследований.

Различают два вида *компьютерных экспериментов*: вычислительный и имитационный. Модель при вычислительном эксперименте строится в виде уравнений (дифференциальных, алгебраических и пр.). При этом приходится использовать численные методы и комплексы эффективных вычислительных алгоритмов. В отличие от вычислительных экспериментов *модели в имитационных экспериментах* из-за отсутствия теории объекта *описывают поведение* реального объекта и *реализуются в виде набора алгоритмов, отображающих ситуации, возникающие в состоянии моделируемого объекта и изменяющиеся по определенным сценариям*. Результатами имитационного эксперимента являются не численные решения каких-либо уравнений модели, а реализации созданных с помощью ЭВМ процессов, имитирующих поведение реального объекта. Термин «реальный» будем использовать в смысле «существующий или способный принять одну из форм существования», т. е. системы, находящиеся только в стадии планирования или разработки, могут моделироваться так же, как и действующие системы.

Необходимо отметить, что при исследовании сложных систем все чаще предпочтение отдается имитационному моделированию или совместному использованию имитационных и математических моделей.

Имитационное моделирование позволяет имитировать поведение системы во времени. Причём плюсом является то, что временем в модели можно управлять: замедлять в случае с быстропротекающими процессами и ускорять для моделирования систем с медленной изменчивостью. Можно имитировать поведение тех объектов, реальные эксперименты с которыми дороги, невозможны или опасны.

Понятие модели, ее функции

Общепринятого определения модели в настоящее время не существует. Приведем некоторые наиболее распространенные определения модели [2–4].

***Модель** является представлением объекта, системы или понятия в некоторой форме, отличной от формы их реального существования.*

***Модель** — это объект-заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала.*

Модель *может определяться как структура для хранения знаний об объекте.*

Математическая модель сложного объекта *представляет собой некоторую знаковую систему, собственные свойства которой настолько близки к свойствам исследуемого объекта, что это позволяет при помощи экспериментов с ней на ЭВМ получить интересующую информацию о поведении или свойствах системы в заданных условиях.*

Имитационная модель *— это компьютерная программа, которая описывает структуру и воспроизводит поведение реальной системы во времени. Имитационная модель позволяет получать подробную статистику о различных аспектах функционирования системы в зависимости от входных данных.*

Модель объекта может быть или точной копией этого объекта или отображать некоторые характерные свойства объекта в абстрактной форме. Модель служит обычно средством, помогающим нам в объяснении, понимании или совершенствовании системы. В настоящее время моделирование становится не только эффективным методом научных исследований сложных объектов, но и мощным инструментом конструирования и проектирования сложных систем. Качество решений задач, получаемых с помощью математического моделирования, определяется степенью адекватности модели реальному объекту (т. е. степенью соответствия результатов моделирования результатам работы реального объекта). Поэтому необходимо помнить, что подобно всем мощным средствам, существенно зависящим от искусства их применения (имитационное моделирование и экспериментирование во многом остаются интуитивными процессами), моделирование способно дать либо очень хорошие, либо очень плохие результаты; либо пролить свет на решение проблемы, либо ввести в заблуждение. Результат моделирования зависит от степени адекватности модели, правильности исходных предпосылок, умения исследователя правильно применять используемые методы, правильно интерпретировать результаты.

В настоящее время существует несколько больших классов моделей. Так как выбор класса зависит от целей исследования и свойств сложной системы, рассмотрим основные функции, выполняемые моделями сложных систем [2].

1. Объяснительная функция модели заключается в способности модели упорядочить нечеткие или противоречивые понятия: выявить взаимозависимости, временные соотношения; помочь интерпретировать данные натурного эксперимента. Уже сама попытка формализовать — помогает в понимании функционирования объекта.
2. Информационная функция заключается в возможности использования модели как средства для накопления и хранения знаний об объекте.

3. Обучающая функция — модель может служить для обучения и тренажа лиц, которые должны уметь справляться с всевозможными случайностями до возникновения реальной критической ситуации (модели космических кораблей, различные тренажеры, деловые игры).
4. Предсказательная функция модели связана с возможностью прогнозировать с заданной точностью по некоторым данным натурных экспериментов поведение и свойства объекта (одна из наиболее важных).
5. Функция постановки и проведения экспериментов для объектов дает возможность характеризовать различные состояния объектов моделирования, таких, где экспериментирование на реальных системах невозможно или нецелесообразно (люди, природа, атомные реакторы).

Зачастую одна модель может выполнять одновременно несколько функций: использоваться для проведения экспериментов и для прогноза; проведения экспериментов и объяснения; для обучения и накопления знаний.

В общем случае модель может служить для достижения двух целей: *описательной* — для объяснения или лучшего понимания объекта; *предписывающей* — для предсказания и/или воспроизведения характеристик объекта, определяющих его поведение. Обычно предписывающие модели являются и описательными, но не наоборот. Предписывающие модели явно предпочтительнее для целей исследования работы системы и предсказания ее будущего поведения, но сложнее в реализации и поэтому не всегда возможны.

Вероятно, в том, что описательная модель не всегда полезна для целей планирования и проектирования, и кроется одна из причин, по которой экономические модели, имеющие (или обнаруживающие) тенденцию к описательности, оказали небольшое воздействие на управление экономическими системами и мало применялись в качестве вспомогательного средства управления на высшем уровне, в то время как модели исследования операций (теория игр) по общему признанию оказали значительное воздействие на эти сферы. В моделях социальных систем наблюдается тенденция к описательности, объяснению явлений (слишком сложный объект моделирования); модели технических систем, предназначенные для разработки новых систем, носят и предписывающий, и описательный характер.

Требования к моделям

Невозможно создать модель полностью адекватную объекту исследования: в этом случае она просто превратится в копию объекта. Поэтому степень адекватности модели зависит от целей моделирования, от имеющихся в наличии ресурсов для создания и эксплуатации модели, а также от того, кто будет пользоваться моделью (т. е. для кого она предназначена). Причем цели моделирования определяют структуру модели, методы, используемые для ее построения. Имеющиеся ресурсы определяют количество априорной (экспериментальной) информации, точность модели и, в некотором смысле, ее универсальность. И наконец, основной определитель качества модели — это пользователь: для того чтобы моделью можно было пользоваться, при ее разработке должны быть тщательно продуманы и потребности, и психология ее конечного потребителя (иначе создание модели будет пустой тратой времени).

На основе вышесказанного определим *ряд существенных черт*, которыми должна обладать хорошая имитационная модель. Модель должна быть [2]:

1) простой и понятной пользователю;
2) целенаправленной, т. е. имеющей четко обозначенные цели и задачи, решаемые системой, с обязательным выделением тех из них, которые должны быть решены с помощью данной модели;
3) целостной, отражающей связь между элементами модели;
4) надежной в смысле гарантии от абсурдных ответов и способной отвечать на вопросы типа «а что если. . .»;
5) удобной в управлении и обращении, т. е. обеспечивающей легкость общения с ней пользователя;
6) полной с точки зрения возможностей решения главных задач;
7) адаптивной, позволяющей легко переходить к другим модификациям или обновлять данные (реальные системы со временем претерпевают изменения);
8) допускающей постепенные изменения в том смысле, что будучи вначале простой, она может во взаимодействии с пользователем становиться все более сложной.

Классификация моделей

Модели вообще, и имитационные модели в частности, можно классифицировать различными способами. К сожалению, ни один из них не является удовлетворительным, хотя каждый служит определенной цели. Укажем некоторые типовые группы моделей, которые могут быть положены в основу системы классификации:

- статические и динамические;
- детерминированные и стохастические;
- дискретные и непрерывные.

Классификация кибернетических моделей

До сих пор мы рассматривали модели, используемые во всех областях исследований — физике, химии и пр. Но нас интересуют модели информационных процессов и систем, предназначенные для обработки информации и управления.

Кибернетика занимается процессами управления в живой и неживой природе, связанными с преобразованием информации. Рассмотрим существующие кибернетические модели (КМ).

Классификация КМ, связанная с математическим аппаратом, выделяет 5 основных классов моделей:

1) массового обслуживания (МО) и надежности;
2) игровые;
3) распознавания образов (РО);

4) графовые;

5) алгебраические.

Каждый класс может быть разделен на подклассы, например алгебраические — на логико-алгебраические, автоматно-лингвистические, модели искусственного интеллекта (ИИ).

- Модели МО — на одноканальные, многоканальные, замкнутые.
- Игровые — на матричные, непрерывные.
- Модели РО — на байесовские, геометрические, модели с дискриминантными функциями.
- Графовые — на потоковые, модели транспортных сетей, модели направленных графов.

Часто модели используются не в чистом классическом виде, а в комбинации, так, теория множеств и математическая логика используются в игровых, моделях РО, графовых; напротив, теория графов используется практически во всех моделях; модели систем ИИ используют теорию РО, графовые, логико-алгебраические и автоматно-лингвистические модели.

При решении каждой конкретной задачи для представления одной и той же системы могут быть использованы различные модели в зависимости от поставленной цели. Поэтому перед началом моделирования необходимо иметь представление о возможно большем круге моделей, чтобы выбрать необходимую.

Этапы моделирования

С развитием вычислительной техники наиболее эффективным методом исследования больших систем стало машинное моделирование. Рассмотрим этапы машинного моделирования реальных систем:

1) определение цели и задач моделирования;

2) определение системы — установление границ, ограничений и измерителей эффективности изучаемой системы;

3) формулирование модели — переход от реальной системы к некоторой логической схеме (абстрагирование);

4) подготовка данных — отбор данных, необходимых для построения модели, и представление их в соответствующей форме;

5) трансляция модели — описание модели на языке, приемлемом для используемой ЭВМ;

6) оценка адекватности — оценка степени уверенности в достаточной корректности выводов о реальной системе, полученных на основании обращения к модели;

7) стратегическое планирование — планирование эксперимента, который должен дать необходимую информацию;

8) тактическое планирование — определение способа проведения каждой серии испытаний, предусмотренных планом эксперимента;

9) экспериментирование — процесс осуществления имитации с целью получения желаемых данных и анализа чувствительности;
10) интерпретация — построение выводов по данным, полученным путем имитации;
11) реализация — практическое использование модели и/или результатов моделирования.

Любое моделирование начинается с постановки задачи моделирования, то есть с ясного изложения целей моделирования. Цели обычно формируются в виде вопросов, на которые надо ответить, либо гипотез, которые надо проверить, либо воздействий, которые надо оценить.

Для определения ограничений задачи моделирования необходимо выявить характеристики системы, подлежащей изучению. Для этого сначала устанавливаются граничные условия, то есть то, что является или не является частью изучаемой системы. Далее определяются существенные параметры и переменные системы, которые характеризуют объект изучения и позволяют установить основные ограничения задачи моделирования.

На этапе формулирования модели разрабатывается моделирующий алгоритм, с помощью которого имитируются явления, составляющие исследуемый процесс. С этим этапом тесно связан этап подготовки данных, в процессе которого определяется, в каком виде будут представлены параметры и переменные системы — в виде констант либо в виде генерируемых последовательностей случайных чисел.

На этапе трансляции модели возникает проблема ее описания на языке, приемлемом для использования компьютера. Здесь использование специальных языков моделирования вместо универсальных существенно экономит время программирования.

Процесс проверки адекватности модели включает в себя несколько шагов последовательных проверок. На первом шаге исследователь должен убедиться, что модель верна в первом приближении. На втором шаге оценки адекватности проверяются исходные предположения. Например, какие параметры и переменные модели можно считать существенными, и охвачены ли моделью все существенные параметры объекта. На третьем шаге проверяются преобразования информации от входа к выходу.

После проверки адекватности модели необходимо спланировать и провести эксперимент с моделью для получения желаемой информации. Анализ данных, полученных в результате эксперимента, позволяет делать выводы о возможностях дальнейшего использования самой модели или результатов моделирования.

Необходимо отметить, что моделирование может быть остановлено на любом этапе (достигнут результат). Кроме того, возможны возвраты к предыдущим этапам. Так, если оценка адекватности модели отрицательна, то может быть возврат на любой из предыдущих этапов моделирования. Если при подготовке данных выясняется, что некоторые данные не могут быть получены или представлены в требуемом виде, то возможно повторное формулирование модели. Этапы стратегического и тактического планирования также тесно связаны между собой, и возможны несколько возвратов с седьмого на шестой этап. После проведения эксперимента осуществляется анализ и интерпретация результатов. Если результаты удовлетворяют исследователя, то на этом процесс моделирования завершается, в противном случае возможен возврат на любой предыдущий этап моделирования.

В предложенном учебном пособии рассмотрены основные общие моменты построения моделей: стохастическое моделирование, планирование эксперимента и анализ результатов. Также в качестве одного из языков моделирования предлагается к рассмотрению язык GPSS.

Во второй части учебного пособия будут рассмотрены два класса кибернетических моделей: модели массового обслуживания и игровые модели.

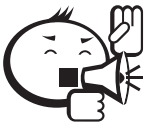
Контрольные вопросы по введению

1. Перечислите основные функции моделей.
2. Перечислите требования к моделям.
3. Перечислите типовые группы моделей, которые могут быть положены в основу классификации.
4. С какого этапа начинается моделирование?
5. Что такое модель?
6. Перечислите основные этапы машинного моделирования.

Соглашения, принятые в книге

Для улучшения восприятия материала в данной книге используются пиктограммы и специальное выделение важной информации.

Эта пиктограмма означает определение или новое понятие.



Эта пиктограмма означает внимание. Здесь выделена важная информация, требующая акцента на ней. Автор здесь может поделиться с читателем опытом, чтобы помочь избежать некоторых ошибок.

Пример

Эта пиктограмма означает пример. В данном блоке автор может привести практический пример для пояснения и разбора основных моментов, отраженных в теоретическом материале.

Контрольные вопросы по главе

Глава 1

ОРГАНИЗАЦИЯ СТАТИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ СИСТЕМ

1.1 Общая характеристика метода

Статистическое моделирование представляет собой метод получения с помощью ЭВМ статистических данных о процессах, происходящих в моделируемой системе. Для получения оценок характеристик моделируемой системы с учетом воздействия внешней среды статистические данные обрабатываются и классифицируются с использованием методов математической статистики.

В некоторых случаях, когда можно построить аналитическую модель случайного процесса (например, систему дифференциальных уравнений для вероятностей состояния или алгебраических уравнений для предельных состояний), статистическое моделирование не является обязательным. Если в системе протекают Марковские процессы, то построение аналитической модели, как правило, не вызывает затруднений. В тех случаях, когда построение аналитической модели является трудно осуществимым, применяется другой метод моделирования — метод статистических испытаний (метод Монте-Карло), который является одним из наиболее широко используемых.

Сущность метода Монте-Карло можно описать следующим образом: производится «розыгрыш» — моделирование случайного явления с помощью некоторой процедуры, дающей случайный результат. Проводя «розыгрыш» много раз, мы получаем статистический материал — множество реализаций случайного явления, который можно обработать обычными методами статистики. Часто этот прием оказывается проще, чем попытки построить аналитическую модель явления и исследовать зависимость между его параметрами на этой модели. Однако заметим, что он оправдан только в том случае, если его использование проще аналитического.

Сущность самого метода статистического моделирования (МСМ) сводится к построению для процесса функционирования исследуемой системы некоторого моделирующего алгоритма с использованием метода Монте-Карло. Данный алго-

ритм имитирует поведение и взаимодействие элементов системы с учетом случайных входных воздействий и воздействий внешней среды.

Рассматриваются две области применения МСМ:

1) для изучения стохастических систем — систем массового обслуживания (СМО), статистических систем РО и т. п.;
2) для решения детерминированных задач (например, вычисление интегралов, обработка больших массивов информации).

При решении детерминированных задач основная идея заключается в замене детерминированной задачи эквивалентной схемой некоторой стохастической системы, выходные характеристики которой совпадали бы с результатами решения детерминированной задачи. Естественно, при замене получается приближенное решение. Погрешность уменьшается с увеличением числа испытаний N.

В результате статистического моделирования получается серия частных значений искомых величин или функций, статистическая обработка которых позволяет получить сведения о поведении реального объекта или процесса в произвольные моменты времени. Допустим, необходимо вычислить математическое ожидание случайных величин x, подчиняющихся некоторому закону распределения $F(x)$. Для этого реализуют датчик случайных чисел, имеющий данное распределение, и определяют оценку математического ожидания:

$$M(x) = \frac{x_1 + x_2 + \ldots + x_N}{N}.$$

Если N достаточно велико, то полученные результаты моделирования приобретают статистическую устойчивость и с достаточной точностью могут быть приняты в качестве оценок искомых характеристик (теоретическая основа — предельная теорема Бернулли, Пуассона, закон больших чисел Чебышева).

В качестве оценки дисперсии используется эмпирическая, или выборочная, дисперсия, определяемая по формуле:

$$S^2 = \frac{1}{N-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{N}\left(\sum_{i-1}^{n} x_i\right)^2\right).$$

Рассмотрим пару примеров применения МСМ: для исследования стохастической системы и для решения детерминированной задачи.

Пример 1.1

Моделирование стохастической системы

С помощью МСМ необходимо найти оценки выходной характеристики стохастической системы S_k, для которой: $x = 1 - e^{-\lambda}$ — входное воздействие; $v = 1 - e^{-\varphi}$ — воздействие внешней среды, где φ и λ — случайные величины с известными функциями распределения. Выходная характеристика распределения описывается выражением $y = \sqrt{x^2 + v^2}$. В качестве оценки математического ожидания может вы-

ступать среднее арифметическое искомой величины: $\overline{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$, где y_i — случайное значение величины y.

Алгоритм решения:

1. Генерация λ_i, φ_i.
2. $x_i = 1 - e^{-\lambda_i}$, $v_i = 1 - e^{-\varphi_i}$.
3. $h_i = x_i^2 + v_i^2$.
4. $y_i = \sqrt{h_i}$.

Точность и достоверность результатов моделирования будет определяться числом реализаций N.

Для получения λ_i, φ_i используются датчики случайных чисел.

Пример 1.2

Решение детерминированной задачи

Найти оценку площади фигуры, ограниченной осями координат, ординатой $\alpha = 1$ и кривой $r = f(\alpha)$; для определенности предполагается, что $0 \leqslant f(\alpha) \leqslant 1$ для всех α, $0 \leqslant \alpha \leqslant 1$.

Аналитическое решение сводится к вычислению интеграла:

$$S = \int_0^1 F(\alpha)\, d\alpha.$$

Искомая величина является площадью заштрихованной фигуры (рис. 1.1) и может быть получена с помощью МСМ: построить адекватную по выходным характеристикам стохастическую систему S_D, оценки характеристик которой будут совпадать с искомыми.

Алгоритм решения:

1. Генерация равномерно распределенных на интервале [0, 1] случайных чисел x_i, x_{i+1}.
2. Вычисление ординаты $r_i = f(x_i)$.
3. Проверка попадания точки (x_i, x_{i+1}) в площадь фигуры:

$$h_i = \begin{cases} 1, & \text{если } x_{i+1} \leqslant f(x_i), \\ 0, & \text{в противном случае.} \end{cases}$$

4. Первые три шага повторяются N раз.
5. Оценка площади фигуры $\overline{S} = \frac{1}{N}\sum_{i=1}^{N} h_i$.

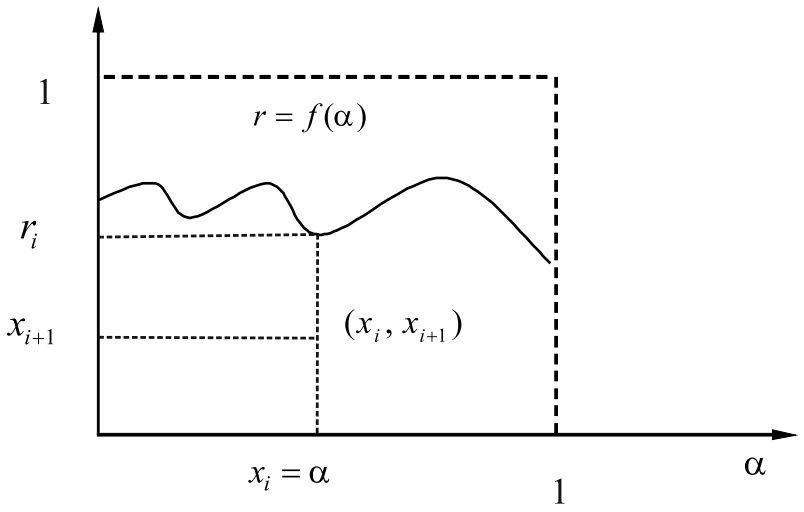


Рис. 1.1 – Графическое представление решения задачи

Таким образом, независимо от природы объекта исследования (статистической или детерминированной) подход для решения является общим. Отметим, что при этом не требуется запоминание всего множества генерируемых случайных чисел. При расчете оценок характеристик исследуемых процессов используются только накопленная сумма исходов и общее число реализаций. Это немаловажное обстоятельство вообще является характерным при реализации имитационных моделей с помощью МСМ.

Одна из основных задач, которую необходимо решить при построении любого моделирующего алгоритма, — это генерация последовательностей случайных чисел (в примерах 1.1 и 1.2 — это λ_i, r_i, x_i, x_{i+1}). Далее мы рассмотрим вопросы, связанные с решением этой задачи.

1.2 Псевдослучайные числа и процедуры их машинной генерации

Успех и точность статистического моделирования зависят в основном от качества генерируемых последовательности случайных чисел. Кроме этого, возможность практического использования машинного моделирования во многом определяется наличием простых и экономичных способов формирования последовательностей случайных чисел. Рассмотрим особенности и возможности получения последовательностей случайных чисел.

Задача получения случайных чисел с помощью программной имитации обычно разбивается на две:

1) получают последовательность случайных чисел, имеющих равномерное распределение на интервале [0, 1]. Может быть выбран и другой базовый процесс, однако при дискретном моделировании базовой является последовательность независимых, равномерно распределенных величин;

2) с помощью различных функциональных преобразований из базовой последовательности получают последовательность случайных чисел с произвольным законом распределения.

Компьютерное представление вещественных чисел всегда ограничено количеством разрядов k, следовательно, максимальное число не совпадающих между собой чисел равно 2^k. Поэтому в машине вместо непрерывной совокупности равномерных случайных чисел можно использовать только дискретную последовательность, множество чисел в которой конечно. Такое распределение называется квазиравномерным. Вероятность каждого значения равна $1/2^k$. При достаточно больших k математическое ожидание и дисперсия квазиравномерной случайной величины близки к математическому ожиданию и дисперсии равномерной случайной величины.

Для получения случайных чисел алгоритмическим путем с помощью специальных программ на ЭВМ разработано большое количество методов. Так как с помощью компьютера невозможно получить идеальную последовательность случайных чисел хотя бы потому, что на ней можно набрать только конечное количество чисел, то последовательности чисел, генерируемых с помощью ЭВМ, называются псевдослучайными. На самом деле повторяемость или периодичность в последовательности псевдослучайных чисел наступает значительно раньше и обусловливается спецификой алгоритма получения случайных чисел.

Сформулируем набор требований, которым должен удовлетворять идеальный генератор. Последовательности псевдослучайных чисел должны:

1) состоять из квазиравномерно распределенных чисел;
2) содержать статистически независимые числа;
3) быть воспроизводимыми;
4) иметь неповторяющиеся числа;
5) получаться с минимальными затратами машинного времени;
6) занимать минимальный объем памяти.

Наибольшее применение на практике для моделирования последовательностей псевдослучайных чисел находят алгоритмы вида $x_{i+1} = \Phi(x_i)$, называемые рекуррентными соотношениями первого порядка, для которых начальное число x_0 либо задается константой, либо генерируется случайным образом (обычно — от таймера).

Равномерное распределение является базовым при имитационном моделировании, наверно, именно поэтому в настоящее время в любой язык программирования и в любые специализированные среды включены датчики случайных чисел с равномерным законом распределения. При разработке имитационной модели можно всегда воспользоваться готовым датчиком, поэтому в данном пособии мы не будем рассматривать методы генерации равномерно распределенных случайных чисел. При желании читатель может изучить эти методы, обратившись к другим источникам.

Можно лишь добавить, что в настоящее время почти все библиотеки стандартных программ для вычисления последовательностей равномерно распределенных случайных чисел основаны на конгруэнтной процедуре.

1.3 Проверка качества последовательностей псевдослучайных чисел

Эффективность статистического моделирования и достоверность получаемых результатов сильно зависит от качества базовых последовательностей псевдослучайных чисел, следовательно необходима проверка на удовлетворение предъявленных требований — иначе даже при абсолютно правильном алгоритме моделирования, результаты могут быть недостоверны.

Генераторы случайных чисел должны проходить предварительно тестирование, которое представляет собой комплекс проверок по различным статистическим критериям. Основными являются тесты на равномерность, стохастичность и независимость.

Рассмотрим наиболее часто используемые методы проведения таких проверок [4].

Проверка равномерности может быть выполнена двумя способами:

1) по гистограмме;

2) с использованием косвенных признаков.

При проверке *по гистограмме* выдвигается гипотеза о равномерности чисел в интервале (0, 1). Интервал разбивается на m равных частей. Тогда при генерации каждое случайное число с вероятностью $p_j = 1/m$, $j = \overline{1, m}$ попадает в один из подинтервалов. Общее число опытов N равно $N = \sum_{j=1}^{m} N_j$, где N_j — количество попаданий случайных чисел x_i в j-й подинтервал. Относительная частота попадания N_j/N. При достаточно больших N экспериментальная гистограмма приблизится к теоретической прямой $1/m$ (рис. 1.2).

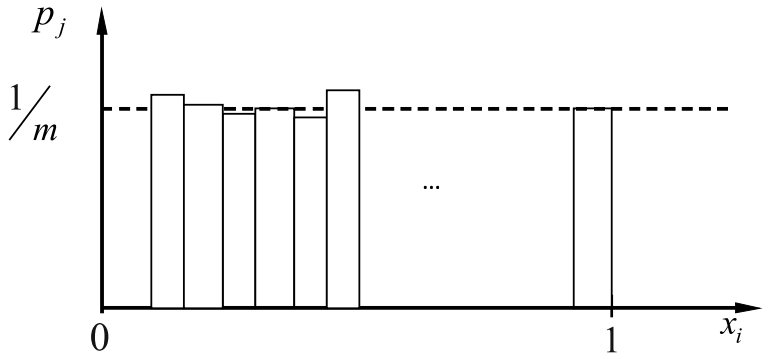


Рис. 1.2 – Проверка равномерности по гистограмме

Оценка степени приближения может быть проведена с помощью критериев согласия. Обычно принимается $m = 20 \div 50$, $N = (10^2 \div 10^3)m$.

В качестве косвенных признаков при проверке равномерности может быть рассмотрен пример с отношением площадей. Генерируемая последовательность чисел разбивается на две, из которых формируются массивы точек с координатами (x_{2i-1}, x_{2i}). Затем проверяется попадание точки в площадь четверть круга (рис. 1.3).

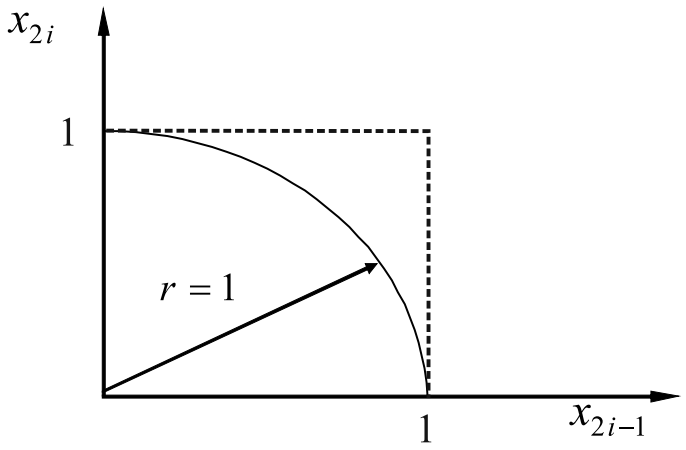


Рис. 1.3 – Косвенная проверка равномерности

Теоретическая вероятность попадания в круг

$$P = \frac{S_{1/4\,\text{круга}}}{S_{\text{квадрата}}} = \frac{\frac{\pi r^2}{4}}{1} = \frac{\pi}{4}.$$

На практике частота попадания равна $\frac{k}{N/2}$, где k — количество выполненных условий $x_{2i-1}^2 + x_{2i}^2 \leqslant 1$. Величина $\frac{2k}{N}$ должна стремиться к $\frac{\pi}{4}$ с ростом N.

Проверка стохастичности наиболее часто проводится методами комбинаций и серий.

Метод комбинаций — определение закона распределения (появления) числа единиц (нулей) в n-разрядном двоичном числе x. На практике длину последовательности N берут достаточно большой и проверяют все n разрядов или только l старших разрядов числа x_i. Теоретически закон появления j единиц в l разрядах двоичного числа x_i описывается исходя из независимости отдельных разрядов биномиальным законом распределения:

$$p(j, l) = C_l^j p^j(1)\left[1 - p(1)\right]^{l-j} = C_l^j p^l(1),$$

где $p(1) = p(0) = 0,5$, $C_l^j = \frac{l!}{j!(l-j)!}$.

При фиксированном N ожидаемое число появлений случайных чисел x_i с j единицами в l разрядах будет равно $n_j = N \cdot C_l^j p^l(1)$. После нахождения теоретических и экспериментальных n_j при различных значениях $l \leqslant n$ гипотеза о стохастичности проверяется с использованием критериев согласия.

Метод серий — последовательность разбивается на элементы первого и второго рода (a и b):

$$x_i = \begin{cases} a, & \text{если } x_i < p; \\ b, & \text{в противном случае,} \end{cases}$$

где $0 < p < 1$.

Серия — любой отрезок последовательности $\{x_i\}$, состоящий из следующих друг за другом элементов одного рода. Число элементов — длина серии. Последовательность при разбиении на серии:

$$\ldots aabbbbaaabaaabbab\ldots.$$

Так как все числа a и b независимы и принадлежат последовательности $\{x_i\}$, равномерно распределенной на (0, 1), то теоретическая вероятность появления серии длиной j в последовательности длиной l в N опытах определяется формулой Бернулли:

$$P(j, l) = C_l^j p^j (1-p)^{l-j}, \quad j = \overline{0, l},$$

где $l = \overline{1, n}$.

При экспериментальной проверке оцениваются частоты появления серий длиной j. Теоретическая и экспериментальная зависимости $P(j, l)$ проверяются на сходимость по известным критериям согласия (при различных j и l).

Проверка независимости проводится на основе вычисления корреляционного момента.

Случайные величины ξ и η называются независимыми, если закон распределения каждой из них не зависит от того, какое значение приняла другая. Тогда независимость элементов последовательности $\{x_i\}$ может быть проверена путем введения для рассмотрения последовательности $\{y_i\} = \{x_{i+\tau}\}$, где τ — величина сдвига последовательности $\{x_i\}$.

Между последовательностями $\{x_i\}$ и $\{y_i\}$ вычисляется коэффициент корреляции по формуле: $\overline{p}_{\xi\eta}(\tau) = \dfrac{k_{\xi\eta}}{(\sigma_x \sigma_y)}$, где σ_x, σ_y — среднеквадратичные отклонения величин ξ и η.

Для определения оценок коэффициента корреляции на ЭВМ удобно пользоваться формулой:

$$p_{\xi\eta}(\tau) = \frac{\left(\dfrac{1}{N-\tau}\sum_{i=1}^{N-\tau} x_i x_{i+\tau} - \dfrac{1}{(N-\tau)^2}\sum_{i=1}^{N-\tau} x_i \sum_{i=1}^{N-\tau} x_{i+\tau}\right)}{\sqrt{D[x_i]\cdot D[x_{i+\tau}]}},$$

где $D[x_i] = \dfrac{1}{N-\tau}\sum_{i=1}^{N-\tau} x_i^2 - \dfrac{1}{(N-\tau)^2}\left(\sum_{i=1}^{N-\tau} x_i\right)^2$, $D[x_{i+\tau}] = \dfrac{1}{N-\tau}\sum_{i=1}^{N-\tau} x_{i+\tau}^2 - \dfrac{1}{(N-\tau)^2}\left(\sum_{i=1}^{N-\tau} x_{i+\tau}\right)^2$.

Заметим, что при использовании приведенных формул нет необходимости в хранении всей последовательности генерируемых случайных чисел. В процессе генерации рассчитываются и хранятся суммы $\sum_i x_i$, $\sum_i x_{i+\tau}$, $\sum_i x_i x_{i+\tau}$, $\sum_i x_i^2$, $\sum_i x_{i+\tau}^2$, а затем на основе сумм рассчитывается коэффициент корреляции.

При любом $\tau \neq 0$ для достаточно больших N с доверительной вероятностью β должно быть справедливо $|\overline{\rho}_{\xi\eta}(\tau)| \leqslant \beta\sqrt{\dfrac{1}{N}}$.

Если найденное значение коэффициента корреляции находится в указанных пределах, то с вероятностью β можно утверждать, что полученная последовательность $\{x_i\}$ удовлетворяет корреляционной независимости.

Важными характеристиками качества генератора случайных чисел является *длина периода R* и *длина отрезка апериодичности L*.

Длина отрезка апериодичности L — это наибольшее целое число, такое, что при $0 \leqslant j \leqslant i \leqslant L$ событие $P\{x_i = x_j\}$ не имеет места, т. е. все числа x_i в пределах отрезка L не повторяются. Соответственно, увеличение последовательности больше L не даст новых статистических результатов.

Длина периода R показывает периодичность повторяемости чисел в последовательности, то есть это минимальное целое число, для которого выполняется:

$$x_i = x_{i+R}.$$

Экспериментальное определение R и L: с начальным x_0 генерируется v чисел x_i $\left(\text{обычно } v = (1 \div 5) \cdot 10^6\right)$ и фиксируется число x_v (рис. 1.4).

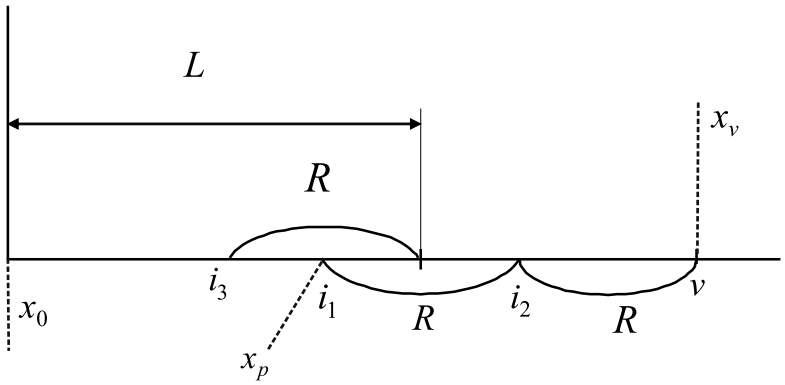


Рис. 1.4 – Определение длины отрезка апериодичности

Затем программа запускается повторно с начальным x_0, и при генерации очередного числа проверяется истинность $P'\{x_i = x_v\}$. Если это событие истинно при $i = i_1$ и $i = i_2$ $(i_1 < i_2 < v)$, то вычисляется длина периода последовательности $R = i_2 - i_1$ (т. е. находится сначала i_1 при $x_{i_1} = x_v$, а затем i_2 при $x_{i_2} = x_v$). Проводится запуск программы генерации с начальными числами x_0 и x_p. При этом фиксируется минимальный номер $i = i_3$, при котором истинно событие $P''\{x_i = x_{p+i}\}$ и $L = i_3 + R$. Если P' оказывается истинным лишь для $i = v$, то $L > v$.

Рассмотрим основные *методы улучшения качества последовательности случайных чисел*.

Один из наиболее употребительных методов заключается в увеличении порядка рекуррентности: вместо $x_{i+1} = \Phi(x_i)$ используются *рекуррентные формулы r*-го порядка $x_{i+1} = \Phi(x_i, x_{i-1}, \ldots, x_{i-r+1})$, где начальные значения $x_0, x_1, \ldots, x_{r-1}$ заданы. Здесь длина L при $r > 1$ гораздо больше, чем при $r = 1$. Однако возрастает и сложность метода, связанная с увеличением затрат машинного времени на вычисления.

Для увеличения L можно воспользоваться *методом возмущений*.

Здесь, для расчета случайной величины, в основном используется формула $\Phi(x_i)$ и только когда i кратно M, последовательность «возмущается» — используется формула $\Psi(x_i)$. Целое число M называется периодом возмущения.

$$x_{i+1} = \begin{cases} \Phi(x_i), & \text{если } (i \bmod M) \neq 0, \\ \Psi(x_i), & \text{если } (i \bmod M) = 0. \end{cases}$$

Все рассмотренные *методы проверки последовательности чисел являются необходимыми* при постановке имитационных экспериментов, но об их достаточности можно говорить лишь при рассмотрении задачи моделирования конкретной системы.

1.4 Моделирование случайных воздействий

Перейдем к рассмотрению вопросов преобразования базовых последовательностей случайных чисел $\{x_i\}$ в последовательности $\{y_i\}$ для имитации воздействия на моделируемую систему. Эти задачи очень важны, т. к. на практике существенное количество операций (временных ресурсов) расходуется на действия со случайными числами. Таким образом, наличие эффективных методов, алгоритмов и программ формирования таких последовательностей во многом определяет возможность практического использования машинной имитации.

Рассмотрим простейший случайный объект — *случайное событие A*, наступающее с вероятностью p.

Здесь процедура моделирования состоит в следующем: генерируется случайная величина x_i, равномерно распределенная на интервале [0, 1], и сравнивается с вероятностью p. Если условие $x_i \leqslant p$ выполняется, то исходом испытания является наступление события A, в противном случае событие A не наступает.

Следующим рассмотрим *группу событий*: известно, что может наступить одно из событий $A_1, A_2, \ldots, A_k$. Заданы вероятности наступления событий $p_1, p_2, \ldots, p_k$. Так как в любой момент времени какое-то из событий наступает обязательно, два события одновременно наступить не могут, то выполняется условие $\sum_{i=1}^{k} p_i = 1$.

Здесь процедура моделирования состоит в сравнении равномерно распределенной случайной величины x_i со значениями $l_r = \sum_{i=1}^{r} p_i$ для соответствующего A_r. Событие A_m наступает, если выполняется условие: $l_{m-1} < x_i \leqslant l_m$. Другими словами, проверяется попадание случайной величины x_i в интервал, ширина которого равна вероятности наступления соответствующего события A_m.

Пример 1.3

Самолет, производящий полет над территорией противника, после стрельбы по нему может оказаться в одном из следующих состояний:

- A_1 — невредим;
- A_2 — поврежден, продолжает полет;
- A_3 — совершил вынужденную посадку;
- A_4 — сбит.

Вероятности соответствующих состояний равны:

$$P(A_1) = 0.4;\ P(A_2) = 0.1;\ P(A_3) = 0.15;\ P(A_4) = 0.35.$$

Генератор событий моделируется следующим образом:

1) генерация равномерно распределенного на [0, 1] случайного числа x_i;
2) если $0 \leqslant x_i \leqslant 0.4$, то наступает событие A_1, иначе —

   если $0.4 < x_i \leqslant 0.5$, то наступает событие A_2, иначе —

   если $0.5 < x_i \leqslant 0.65$, то наступает событие A_3, иначе —

   если $0.65 < x_i \leqslant 1$, то наступает событие A_4.

. . . . . .

Заметим, что для группы событий вероятность наступления отдельного события зависит от ширины рассматриваемого интервала и не зависит от его расположения на отрезке [0, 1]. Так, если в примере 1.3 рассматривать наступление события A_1 как попадание x_i в интервал [0.6, 1], то вероятность его наступления по-прежнему остается равной 0.4 и результаты моделирования не изменятся.

Если искомый вариант зависит от двух *независимых событий* A и B, то в результате возможны исходы AB, $\overline{A}B$, $A\overline{B}$, $\overline{AB}$ с вероятностями $p_A p_B$, $(1-p_A)p_B$, $p_A(1-p_B)$, $(1-p_A)(1-p_B)$.

Для моделирования независимых событий существуют два возможных варианта:

1) последовательная проверка условия $x_i \leqslant p$ для каждого события в отдельности;
2) моделирование ситуации как группы событий $(A_1 = AB,\ A_2 = \overline{A}B\ldots)$.

Для первого варианта требуется сгенерировать два числа x_i и выполнить два сравнения; для второго варианта требуется одно число x_i, но больше сравнений. Первый вариант является более предпочтительным: удобство алгоритма, экономия количества операций, ячеек памяти и т. д.

Рассмотрим два *зависимых события*: пусть события A и B могут наступить с вероятностями соответственно p_A и p_B. При этом наступление события B зависит от наступления события A с вероятностью $p(B/A)$. Один из вариантов моделирования можно построить по следующему алгоритму:

1) генерируются два равномерно распределенных случайных числа x_1 и x_2;
2) x_1 используется для проверки наступления события A по условию $x_1 \leqslant p_A$;
3) если событие A наступило, то наступление события B проверяется по условию $x_2 \leqslant p(B/A)$;
4) если событие A не наступило, то вычисляется вероятность $p\left(B/\overline{A}\right) = \left(p_B - p_A \cdot p(B/A)\right)/(1-p_A)$ и проверяется выполнение условия $x_2 \leqslant p\left(B/\overline{A}\right)$.

Для формирования возможных значений *случайных величин* необходимо получить последовательность с заданным законом распределения.

Дискретная случайная величина η принимает значения $y_1 \leqslant y_2 \leqslant \ldots \leqslant y_i \leqslant \ldots$ с соответствующими вероятностями $p_1, p_2, \ldots, p_i, \ldots$.

Для рассматриваемой величины интегральная функция распределения имеет вид

$$F_\eta(y) = p(\eta \leqslant y) = \sum_{j=1}^{m} p_j, \text{ если } y_m \leqslant y \leqslant y_{m+1},$$

и

$$F_\eta(y) = 0, \text{ если } y < y_1.$$

Для получения η можно использовать метод обратной функции. Если ξ — равномерно распределенная случайная величина на интервале [0, 1], то искомая случайная величина может быть выражена $\eta = F_\eta^{-1}(\xi)$.

Здесь алгоритм генерации аналогичен моделированию группы событий:

- если $x_1 \leqslant p_1$, то $\eta = y_1$, иначе —
- если $x_2 \leqslant p_1 + p_2$, то $\eta = y_2$, иначе —
- ...,

где x_i — равномерно распределенное случайное число.

Непрерывная случайная величина задается интегральной функцией распределения

$$F_\eta(y) = p(\eta \leqslant y) = \int_{-\infty}^{y} f_\eta(y)\, dy.$$

Для генерации непрерывной случайной величины можно воспользоваться, как и для дискретной случайной величины, методом обратной функции (другое название метода — *метод нелинейных преобразований*)

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

Функция преобразует равномерно распределенную случайную величину ξ в случайную величину η с требуемой плотностью распределения $f_\eta(y)$. Другими словами, необходимо разрешить относительно y_i уравнение

$$\int_{-\infty}^{y_i} f_\eta(y)dy = x_i.$$

Пример 1.4

Пусть необходимо сгенерировать последовательность чисел с экспоненциальным законом распределения

$$f_\eta(y) = \lambda e^{-\lambda y}, y > 0 \Rightarrow \lambda \int_0^{y_i} e^{-\lambda y} dy = x_i,$$

где x_i — равномерно распределенное случайное число.

Возьмем первообразную и выразим искомую величину через равномерно распределенную случайную величину:

$$1 - e^{-\lambda y_i} = x_i \Rightarrow y_i = -\ln(1 - x_i)/\lambda.$$

Учитывая, что случайная величина $\xi_1 = 1 - \xi$ также имеет равномерное распределение в [0, 1], можно записать $y_i = -\ln(x_i)/\lambda$.

Метод нелинейных преобразований является единственным точным методом генерации случайных величин с заданным законом распределения, хотя и имеет ограниченную сферу применения по следующим причинам:

1) для многих законов распределения интеграл не берется, т. е. приходится прибегать к численным методам решения, что увеличивает затраты машинного времени;

2) если интеграл берется, получаются формулы, содержащие логарифм, корень и т. д., что резко увеличивает затраты машинного времени за счет большого количества выполняемых операций.

По причине указанных недостатков метода нелинейных преобразований на практике используются и другие способы моделирования непрерывных случайных величин (все они являются приближенными):

а) универсальные — для любых законов распределения;

б) неуниверсальные — для конкретных законов распределения.

Наиболее простым из *универсальных способов* является *метод Неймана* (режекции). Он заключается в генерации двух равномерно распределенных случайных чисел x_1 и x_2 и их преобразовании: $x_1' = a + (b - a) \cdot x_1$, $x_2' = W \cdot x_2$. Здесь a и b — левая и правая границы интервала задания $f_\eta(y)$, а $W = \max\limits_{[a,\,b]} f_\eta(y)$. Если $x_2' \leqslant f_\eta(x_1')$, то x_1' — искомое значение случайной величины η.

Следующим универсальным приближенным методом генерации случайных величин является *метод кусочной аппроксимации*.

Требуется получить последовательность $\{y_i\}$ с функцией плотности распределения $f_\eta(y)$, возможные значения которой лежат в интервале (a, b). Представим $f_\eta(y)$ в виде кусочно-постоянной функции — разобьем (a, b) на интервалы, на которых функцию плотности распределения будем аппроксимировать равномерным законом. Тогда случайная величина может быть получена по формуле $\eta = a_k + \eta_k^*$, где a_k — абсцисса левой границы k-го интервала, η_k^* — случайная величина, значения которой располагаются равномерно внутри k-го интервала, т. е. считается равномерно распределенной.

Чтобы аппроксимировать $f_\eta(y)$ наиболее удобным для практических целей способом, целесообразно разбить (a, b) на m интервалов так, чтобы вероятность попадания η в любой интервал была постоянной (рис. 1.5 — площади прямоугольников должны быть одинаковыми), т. е. не зависела от k, тогда для вычисления a_k используют формулу:

$$\int\limits_{a_k}^{a_{k+1}} f_\eta(y)dy = \frac{1}{m}. \tag{1.1}$$

Зная значение левой границы $a_0 = a$, можно рассчитать значение следующей границы a_1. Подставив полученное значение в нижнюю границу интеграла, рассчитываем a_2 и т. д.

Алгоритм генерации:

1) генерируется случайное число x_i на интервале [0, 1];

2) с помощью сгенерированного числа случайным образом выбирается интервал $[a_k, a_{k+1}]$;

3) генерируется x_{i+1} и масштабируется с целью приведения к интервалу $[a_k, a_{k+1}]$, т. е. $(a_{k+1} - a_k) \cdot x_{i+1}$;

4) $y_j = a_k + (a_{k+1} - a_k) \cdot x_{i+1}$ — вычисляется y_j с требуемым законом распределения.

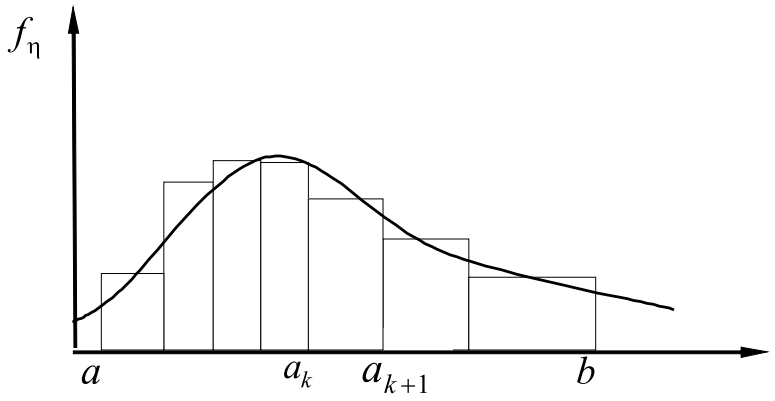


Рис. 1.5 – Метод кусочной аппроксимации

Рассмотрим подробнее второй шаг алгоритма: целесообразно для этого построить таблицу, содержащую номера интервалов k, их границ и коэффициентов масштабирования, определенные из (1.1).

Сгенерировав x_i, мы можем определить номер интервала. Так как вероятность выбора интервалов одинакова и равна $\frac{1}{m}$, то, используя метод генерации группы событий, получаем:

- если $0 \leqslant x_i \leqslant \frac{1}{m}$, то выбираем 1-й интервал;
- если $\frac{1}{m} < x_i \leqslant \frac{2}{m}$, то выбираем 2-й интервал и т. д.

Достоинство данного метода состоит в небольшом количестве операций, т. к. границы интервалов мы рассчитываем один раз до начала генерации, а сам алгоритм генерации содержит только операции сравнения, сложения и вычитания. Точность метода всегда можно повысить за счет увеличения количества интервалов m. На практике обычно выбирают m в пределах от 16 до 20.

Неуниверсальные способы строятся на основе предельных теорем теории вероятностей и предназначены для получения последовательностей случайных чисел с конкретным законом распределения. Рассмотрим некоторые из этих способов.

Нормальный закон распределения

Требуется получить нормальное распределение с математическим ожиданием a и среднеквадратичным отклонением σ.

Будем формировать y_i в виде сумм последовательностей $\{x_i\}$. По центральной предельной теореме: если независимые одинаково распределенные случайные величины имеют каждая математическое ожидание a_1 и среднеквадратичное отклонение σ_1, то сумма $\sum_{i=1}^{N} x_i$ асимптотически нормальна с математическим ожиданием $a = N \cdot a_1$ и среднеквадратическим отклонением $\sigma = \sigma_1\sqrt{N}$. Сумма имеет распределение, близкое к нормальному, уже при небольшом N (может быть $8 \div 12$ или даже $4 \div 5$ — в простейших случаях).

Генерируем нормально распределенную величину $V = \sum_{i=1}^{N} x_i$, где x_i — равномерно распределенная случайная величина на [0, 1] с математическим ожиданием, равным 0.5, поэтому математическое ожидание $m_V = N \cdot 0,5$.

$D_{x_i} = \frac{(\beta - \alpha)^2}{12} = \frac{1}{12}$ — дисперсия для равномерно распределенных величин на [0, 1], отсюда $D_V = \sum_i D_{x_i} = \frac{N}{12}$, а $\sigma_V = \sqrt{D_V} = \sqrt{\frac{N}{12}}$.

$z = \frac{V - m_V}{\sigma_V} = \frac{V - N \cdot 0,5}{\sqrt{N/12}}$ — нормально распределенная величина с нулевым математическим ожиданием и единичной дисперсией.

Тогда искомая нормально распределенная величина с заданными σ_y и m_y будет определяться по формуле $y = \sigma_y z + m_y$.

Пуассоновское распределение

Случайная величина с пуассоновским распределением — это целочисленная величина, которая характеризует количество наступивших событий. Вероятность наступления m событий равна:

$P(m, a) = \frac{a^m}{m!} e^{-a}$, где a — математическое ожидание величины.

Для получения пуассоновских чисел с заданным математическим ожиданием можно использовать следующий метод. Образуется произведение равномерно распределенных последовательных случайных чисел x_i. Количество сомножителей k выбирается таким, чтобы удовлетворялось неравенство $\prod_{i=1}^{k} x_i < e^{-a}$.

Число $k - 1$ представляет собой случайную величину η, принадлежащую совокупности, распределенной по закону Пуассона с математическим ожиданием a. Если неравенству удовлетворяет первое из равномерно распределенных чисел, то $\eta = 0$.

В таблице 1.1 приведен ряд неуниверсальных методов для генерации случайных величин с указанными законами распределения, основанных на использовании нормально распределенных величин. Здесь z_i — нормально распределенная величина с математическим ожиданием, равным 0, и среднеквадратическим отклонением, равным 1, а v — число степеней свободы. При этом для каждого закона указано, какими должны быть математическое ожидание и дисперсия случайной величины.

Таблица 1.1 – Неуниверсальные методы для генерации случайных величин

Закон распределения	Формула	M	Дисперсия
Хи-квадрат	$u = \sum_{i=1}^{v} z_i^2$	v	$2v$
Стьюдента	$T = \frac{z}{\sqrt{u/v}}$ или $T = \frac{z}{\sqrt{\sum_{i=1}^{v} z_i^2/v}}$	0	$\frac{v}{v-2}$
F-распределение	$W = \frac{u_1/v_1}{u_2/v_2}$ или $W = \frac{\sum_{i=1}^{v_1} z_i^2/v_1}{\sum_{i=1}^{v_2} z_i^2/v_2}$	$\frac{v_2}{v_2-2}$	$\frac{2v_2^2(v_1+v_2-2)}{v_1(v_2-2)^2(v_2-4)}$

Рассмотрим пример построения имитационного алгоритма работы стохастической системы.

Пример 1.5

Начальник пожарной охраны обнаружил, что число пожаров за сутки следует распределению Пуассона со средним значением 4 пожара в сутки.

Изучив данные по прежним пожарам, он нашел, что в 70% случаев для тушения потребовалась только одна пожарная машина, а время, необходимое для ликвидации пожара, имеет нормальное распределение с m = 2.5 часа и δ = 0.5 часа. В остальных 30% случаев нужны были 2 пожарные машины, а время для ликвидации этих пожаров распределено нормально с m = 3.5 часа и δ = 1 час.

Предполагая, что необходимые пожарные машины всегда находятся под рукой, необходимо определить, сколько часов в среднем они бывают нужны каждые сутки (построить алгоритм).

При построении алгоритма считаем, что генераторы случайных чисел x_i, равномерно распределенных на интервале [0, 1], существуют.

Сначала построим алгоритм генератора случайных чисел для моделирования числа пожаров в сутки ***gen1***. Для генерации случайных чисел с Пуассоновским распределением воспользуемся неуниверсальным методом генерации:

1) $p := 1$; $k := 0$;
2) генерируем равномерно распределенное случайное число x_i;
3) $p := p \cdot x_i$; $k := k + 1$;
4) если $p > \exp(-4)$, то переход к п. 2;
5) выход: $k - 1$.

Теперь построим генератор нормально распределенных случайных чисел ***gen2*** для моделирования времени, необходимого для ликвидации пожара с входными

параметрами m и δ. Для этого также воспользуемся неуниверсальным методом генерации:

1) генерируем 12 равномерно распределенных случайных чисел x_i;
2) суммируем $y = \sum_{i=1}^{12} x_i$;
3) вычисляем $N := \delta \cdot (y - 6) + m$;
4) выход: N.

Для определения оценки среднего времени промоделируем работу системы в течение 100 дней. Общий алгоритм работы системы будет выглядеть следующим образом:

1) $N = 0$ {счетчик дней};
2) $T = 0$ {общее время тушения пожаров};
3) $N = N + 1$;
4) $K = gen1$ {определяем количество пожаров сегодня};
5) $i = 0$;
6) $i = i + 1$;
7) генерируем x_i {для определения типа пожара};
8) если $x_i \leqslant 0.7$, то $t = gen2(2.5, 0.5)$ {требовалась 1 машина}
   иначе $t = gen2(3.5, 1) \cdot 2$ {требовались 2 машины};
9) $T = T + t$ {добавляем сгенерированное время тушения пожара};
10) если $t < k$, перейти к п. 6 {цикл по количеству пожаров сегодня};
11) если $N < 100$, перейти к п. 3 {моделируем в течение 100 дней};
12) $T = \dfrac{T}{100}$ {оценка среднего времени тушения пожаров в день}.

. . . . . .

1.5 Идентификация закона распределения

Еще одна проблема, возникающая при исследовании и моделировании стохастических систем, — как проверить совместимость экспериментальных данных с некоторым теоретическим распределением. Иначе говоря, возникает вопрос: соответствует ли частота наблюдаемых выборочных значений той частоте, с которой они должны бы появляться при некотором вероятностном распределении, отвечающем определенному теоретическому закону. Если частота близка к теоретической, то в дальнейшем можно строить модель событий на основе теоретического распределения, используя любой метод генерации случайных величин из рассмотренных выше.

Если собрать достаточное количество данных натурного эксперимента, то можно построить и проанализировать гистограмму. Визуальное сравнение гистограммы и функции плотности распределения позволяет лишь предположить, к какому тео-

ретическому распределению нужно «подогнать» экспериментальное. После того как аналитик подобрал одно или несколько теоретических распределений, близких к экспериментальным данным, ему следует определить параметры распределения, с тем чтобы подвергнуть их проверке по статистическим критериям. Если предполагаемое распределение является функцией двух параметров, последние обычно удается оценить на основе выборочного среднего и выборочной дисперсии.

Когда экспериментальные данные разбиты на группы (гистограмма), среднее и дисперсию можно вычислить по формулам:

$$\overline{X} = \frac{\sum_{i=1}^{k} M_i F_i}{n}, \quad S^2 = \frac{\sum_{i=1}^{k} M_i^2 F_i - n\overline{X}^2}{n-1},$$

где n — полный объем выборки, $n = \sum_{i=1}^{k} F_i$; k — число групп (интервалов выборки); M_i — средняя точка i-го интервала или (для дискретных данных) значение i-й группы; F_i — частота появления i-го интервала.

После получения оценок параметров распределения проверяются гипотезы принадлежности экспериментальных данных тому или иному закону распределения. Для проверки используют критерии согласия, два из которых мы рассмотрим на примере.

Оценка по критерию согласия «хи-квадрат»

Для статистической оценки гипотезы о том, что совокупность эмпирических или выборочных данных незначительно отличается от той, которую можно ожидать при некотором теоретическом распределении, можно использовать критерий χ^2, который определяется формулой:

$$\chi^2 = \sum^{k} \frac{(f_0 - f_e)^2}{f_e},$$

где f_0 — наблюдаемая частота для каждого интервала; f_e — ожидаемая частота для каждого интервала; $\sum$ — предсказанная теоретическим распределением сумма по всем группам или интервалам.

Если $\chi^2 = 0$, то наблюдаемые и теоретические значения частот точно совпадают. Чем больше χ^2, тем больше расхождение между наблюдаемыми и ожидаемыми значениями. При $\chi^2 > 0$ статистика сравнивается с табличным значением (табулирована для различных чисел степеней свободы и различных уровней доверительной вероятности $1 - \alpha$).

Высказывается нулевая гипотеза H_0 о том, что между наблюдаемым и ожидаемым распределениями нет значительных расхождений. Если $\chi^2_{\text{расч}} > \chi^2_{\text{табл}}$, то гипотеза H_0 отвергается.

Ограничения при использовании критерия:

1) необходимо использовать абсолютные значения частот (не относительные);
2) значения наблюдений для каждого интервала должны быть больше или равны 5;
3) число степеней свободы $\nu = k - 1 - m$, где k — число интервалов или групп, m — число параметров, определяемых опытным путем для вычисления ожидаемых значений частот.

Пример 1.6

Пусть мы проверяем, является ли частота телефонных запросов Пуассоновским распределением.

Для определения расчетного значения χ^2 построим таблицу (табл. 1.2):

Таблица 1.2 – Проверка гипотезы о принадлежности выборки Пуассоновскому распределению по критерию χ^2

Число запросов	Число одночасовых интервалов с числом запросов f_0	Относительная частота	Теоретическая вероятность $P(n)$	Ожидаемая частота $f_e = P(n)x$	$\frac{(f_0 - f_e)}{f_e}$
0	315	0.619	0.571	291	1.98
1	142	0.279	0.319	162	2.47
2	40	0.078	0.089	45	0.56
3	9	0.018	0.017	9	
4	2} 12	0.004	0.003	1} 11	} 0.09
5	1	0.002	0.001	1	
Сумма	509	1.000	1.000	509	5.1

Здесь $\bar{x} = 0.5147$. $s^2 = 0.6007$. Так как для Пуассоновского закона математическое ожидание и дисперсия равны, т. е. должно выполняться $\bar{x} = s^2$, то можно уточнить оценку параметра: $\lambda = \dfrac{\bar{x} + s^2}{2} = 0.5577$.

Теоретическая вероятность наступления n событий определяется по формуле $P\{x = n\} = \dfrac{\lambda^n e^{-\lambda}}{n!}$.

Для доверительной вероятности 0.95 и числа степеней свободы $4 - 1 - 1 = 2$ имеем $\chi^2_{\text{табл}} = 5.99$. Значит, $\chi^2_{\text{расч}} < \chi^2_{\text{табл}}$ и, следовательно, гипотеза принимается.

Критерий Колмогорова—Смирнова

Еще один используемый критерий, который применяется, когда проверяемое распределение непрерывно и известны среднее и дисперсия совокупности, это критерий Колмогорова—Смирнова. Здесь проверка осуществляется путем задания интегральной функции, следующей из теоретического распределения, и ее сравнения с интегральной функцией распределения эмпирических данных.

Сравнение основывается на выбранной группе, в которой экспериментальное распределение имеет наибольшее абсолютное отклонение от теоретического. Эта абсолютная разность сопоставляется с критическим значением $D_{\text{крит}} = \frac{\lambda}{\sqrt{N}}$, где λ — табличное значение функции Колмогорова при заданной доверительной вероятности.

Пример 1.7

Возьмем данные из примера 1.5 и рассчитаем абсолютную разность по всем интервалам (табл. 1.3).

Таблица 1.3 – Проверка гипотезы о принадлежности выборки Пуассоновскому распределению по критерию Колмогорова—Смирнова

	I	II	III	IV	V	VI
Число запросов	f_0	Наблюдаемая вероятность	Теоретическая $P(n)$	Интегральная вероятность II	Интегральная вероятность III	Абсолютная разность IV–V
0	315	0.619	0.571	0.619	0.571	0.048
1	142	0.279	0.319	0.838	0.890	0.008
2	40	0.078	0.089	0.976	0.979	0.003
3	9	0.018	0.017	0.994	0.996	0.002
4	2	0.004	0.003	0.998	0.999	0.001
5	1	0.002	0.001	1.000	1.000	0

При $\alpha = 0.05$ и $n = 509$ $D_{\text{крит}} = \frac{1.36}{\sqrt{509}} = 0.0603$. Максимальная абсолютная разность 0.0048 меньше $D_{\text{крит}}$, следовательно, гипотеза принимается.

При малых выборках критерий χ^2 не применим, здесь предпочтительнее критерий Колмогорова—Смирнова. Напротив, если объем выборки велик, предпочтителен критерий χ^2 (при $n \geqslant 100$ данный критерий является достаточно мощным). Чем больше число интервалов, тем точнее выводы. Для критерия Колмогорова—Смирнова можно относить каждое наблюдение к отдельной группе — тогда он работает эффективнее.

Контрольные вопросы по главе 1

1. Какая последовательность называется базовой при стохастическом моделировании?
2. Какие проверки проводятся при определении качества генератора случайных чисел?
3. Почему при машинном моделировании генерируемые последовательности чисел называются псевдослучайными?
4. В чем преимущества и недостатки различных способов генерации последовательностей псевдослучайных чисел?
5. Какие методы генерации случайных величин с заданным законом распределения Вы знаете?
6. Каковы их недостатки?

Глава 2

ЯЗЫК МОДЕЛИРОВАНИЯ СИСТЕМ GPSS

2.1 Программные средства имитационного моделирования

Успех или неудача любого исследования системы на программно реализуемой модели прежде всего зависит от правильности моделирующего алгоритма, а также от эффективности программы. Поэтому при машинной реализации модели большое значение имеет вопрос правильного выбора языка моделирования.

Каждый язык моделирования должен отражать определенную структуру понятий для описания широкого класса явлений. При выборе языка необходимо учитывать большую роль уровня его проблемной ориентации. Высокий уровень проблемной ориентации языка моделирования значительно упрощает программирование моделей и анализ результатов экспериментов.

Определим основные моменты, характеризующие качество языка моделирования:

- удобство описания процесса функционирования системы;
- удобство ввода исходных данных моделирования;
- удобство варьирования структуры, алгоритмов и параметров модели;
- реализуемость статистического моделирования;
- эффективность анализа и вывода результатов моделирования;
- простота отладки и контроля работы моделирующей программы;
- доступность восприятия и использования языка.

Языки общего назначения (ЯОН), или алгоритмические языки (такие как Паскаль, Си и пр.), предназначены для формального представления и решения задач любого типа. В отличии от ЯОН, любой язык моделирования представляет собой процедурно-ориентированный язык, обладающий специфическими чертами. Основные языки моделирования разрабатывались в качестве программного обес-

печения имитационного подхода к изучению процесса функционирования определенного класса систем.

Целесообразность использования языков имитационного моделирования вытекает из трех причин:

1) удобство программирования модели системы;
2) концептуальная направленность языков на класс систем, необходимая на этапе построения модели системы и выбора общего направления исследований в планируемом машинном эксперименте. Практика моделирования систем показывает, что именно использование языков имитационного моделирования во многом определило успех имитации как метода экспериментального исследования сложных реальных объектов;
3) языки моделирования позволяют описывать системы в терминах, разработанных на базе основных понятий имитации, что образует связь между отдельными разработками в единых способах описания.

При создании системы моделирования необходимо решить вопрос о синхронизации процессов в модели: для одновременно протекающих событий должна быть реализована псевдопараллельная организация имитируемых процессов. Это является основной задачей монитора моделирования, который выполняет следующие функции: управление процессами (согласование системного и машинного времени) и управление ресурсами (выбор и распределение в модели средств моделируемой системы).

При разработке моделей систем возникает целый ряд специфических трудностей. В связи с этим в языках имитационного моделирования должен быть предусмотрен набор таких программных средств и понятий, которые не встречаются в универсальных языках программирования. К таким понятиям относятся:

1. *Совмещение*. Параллельно протекающие в реальных системах процессы представляются с помощью последовательно работающих компьютеров, следовательно, возникает необходимость во введении понятия системного времени.
2. *Размер*. Большие по объему модели требуют динамического распределения памяти, когда компоненты модели появляются в оперативной памяти или покидают ее в зависимости от текущего состояния. Отсюда возникает необходимость обеспечения блочной конструкции модели.
3. *Изменения*. Пространственная конфигурация динамических систем претерпевает изменения во времени, поэтому в языках имитационного моделирования предусматривают обработку списков, отражающих изменения состояний процесса функционирования системы.
4. *Взаимосвязанность*. При моделировании процесса функционирования системы необходимо учитывать сложные условия свершения событий, поэтому в языки моделирования включают логические возможности и понятия теории множеств.
5. *Стохастичность*. Для моделирования случайных событий и процессов в язык моделирования включают процедуры генерации случайных чисел, на основе которых можно получать стохастические воздействия на систему.

6. *Анализ*. Для анализа результатов моделирования необходимы статистические характеристики процесса функционирования модели, поэтому в языках моделирования предусмотрены различные способы статистической обработки и анализа результатов.

В настоящее время существует несколько десятков развитых языков имитационного моделирования. Язык GPSS, рассматриваемый в данном учебном пособии, является одним из наиболее распространенных языков моделирования, используемых как для исследований систем, так и для учебных целей в различных университетах мира.

2.2 Общие сведения о языке GPSS

Язык моделирования GPSS (General Purpose System Simulation) разработан фирмой IBM в США и с 1962 года входил в стандартное математическое обеспечение машин серии IBM 360/370. Язык GPSS получил наиболее широкое распространение по сравнению с другими языками моделирования. Он включен в учебные курсы ВУЗов по моделированию систем у нас в стране и изучается в аналогичных курсах во многих колледжах и университетах США и других стран. В данном учебном пособии рассматривается одна из версий языка GPSS.

Язык GPSS ориентирован на решение задач статистического моделирования на ЭВМ процессов с дискретными событиями. Такими процессами описывается, прежде всего, функционирование систем массового обслуживания произвольной структуры и сложности: систем обработки данных, систем транспорта и связи, технологических процессов, предприятий торговли, а также функционирование вычислительных систем и разного рода автоматизированных систем.

Язык основан на схеме *транзактов* (сообщений).

*Под **транзактом** понимается формальный объект, который «путешествует» по системе (перемещается от блока к блоку), встречая на пути всевозможные задержки, вызванные занятостью тех или иных единиц оборудования.*

Транзакты имеют прямую аналогию с заявками в системах массового обслуживания. В качестве транзакта может выступать программа обработки информации, телефонный вызов, покупатель в магазине, отказ системы при исследовании надежности и т. д. Каждый транзакт обладает совокупностью параметров, которые называются атрибутами транзакта. В процессе имитации атрибуты могут меняться в соответствии с логикой работы исследуемой системы.

Язык GPSS — язык интерпретируемого типа, он связан с пошаговым выполнением операторов, называемых блоками. Совокупности блоков описывают функционирование самой моделируемой системы либо содержат информацию о порядке моделирования (о продвижении транзактов). Каждое продвижение транзакта (сообщения) является событием в модели.

Комплекс программ, планирующий выполнение событий, реализующий функционирование блоков моделей, регистрирующий статистическую информацию о прохождении транзактов, называется ***симулятором*** [4].

Симулятор регистрирует время наступления каждого из известных на данный момент событий и выполняет их с нарастающей временной последовательностью. Симулятор обеспечивает отсчет модельного времени в принятых единицах, называемых *абсолютным условным временем*. С каждым сообщением связано *относительное условное время*, отсчет которого начинается при входе сообщения в моделируемую систему и заканчивается при выходе сообщения из системы. *Основными функциями* управляющих операторов/блоков языка являются:

1) создание и уничтожение транзактов,
2) изменение их атрибутов,
3) задержка транзактов,
4) изменение маршрутов транзактов в системе.

Основные группы объектов языка:

1) объекты, имитирующие единицы оборудования системы (устройство, память и логические переключатели),
2) статистические объекты (очередь, таблица),
3) вычислительные объекты (ячейка, арифметическая и логические переменные),
4) списки,
5) прочие объекты.

Дадим описание некоторых объектов.

Устройство *имитирует единицу оборудования, которое может одновременно обрабатывать только один транзакт. Устройство аналогично обслуживающему прибору в системах массового обслуживания. Оно служит для моделирования таких средств обработки элементов потоков, как станки, рабочие, каналы связи и т. п. На устройствах можно реализовать самые различные дисциплины обслуживания транзактов, включающие учет требуемого времени обслуживания, значения приоритетов, возможности прерывания и т. д.*

Память *(накопитель) имитирует единицу оборудования, в которой может обрабатываться (храниться) несколько транзактов одновременно. Память позволяет легко моделировать средства обработки с ограниченной емкостью (стоянки автотранспорта, портовые причалы, складские помещения, конвейеры и т. п.).*

Очередь *— объект, связанный со сбором статистики о задержках, возникающих на пути прохождения транзакта. Чаще всего очередь помещают перед устройством либо памятью. Следует учитывать, что естественно образующиеся в процессе моделирования очереди транзактов обрабатываются симулятором автоматически, а описываемый объект языка служит лишь для обеспечения вывода на печать соответствующих статистических данных.*

Таблица *обеспечивает накопление в процессе моделирования статистики о каком-либо заданном случайном параметре модели. По окончании прогона модели эта статистика автоматически обрабатывается и выводится на печать, в частности, в виде таблицы относительных частот попадания значений случайного параметра (аргумента таблицы) в указанные частотные интервалы. Печатаются также среднее значение и среднее квадратичное отклонение аргумента.*

Ячейки *используются для записи, накопления и хранения численных значений различных входных и выходных параметров моделируемой системы. Эти значения могут быть использованы для организации счетчиков числа проходящих транзактов, для вывода значений варьируемых параметров модели, для временного хранения значений* ***стандартных числовых атрибутов*** *(СЧА). Значения ячеек всегда выводятся на печать.*

Арифметическая переменная *позволяет выполнить заданную последовательность арифметических операций над любыми СЧА модели для вычисления значения зависимого от них параметра.*

Любая программа на GPSS связана с созданием транзактов, проведением их через последовательность блоков и уничтожением транзактов. При этом создание или

генерация транзактов основывается на знании закономерностей информационных потоков, циркулирующих в моделируемой системе, а путь прохождения транзакта через блоки определяется спецификой работы оборудования исследуемой системы.

Вложить в рамки формальной схемы GPSS конкретное смысловое содержание, определяемое исследуемой системой — задача непростая: для этого необходимо знать как формализмы языка, так и логику работы моделируемой системы. Тем не менее программирование на GPSS существенно облегчает пользователю процесс моделирования, сокращая и время чистого программирования (по сравнению с универсальными алгоритмическими языками), и время отладки программы.

2.3 Синтаксис языка

Алфавит. Алфавит языка GPSS состоит из латинских букв от *A* до *Z*, цифр от 0 до 9 и следующих специальных символов: $, #, *, +, –, /, (,), ', точка, запятая, пробел.

Идентификаторы. Они состоят из алфавитно-цифровых символов, причем первый символ должен быть буквой. Идентификаторы используются для формирования имен объектов и блоков. При формировании собственного имени необходимо помнить, что оно не должно совпадать с ключевым словом языка, поэтому рекомендуется использовать имена с количеством букв в начале имени не менее трех. Исключение составляют ячейки и атрибуты транзактов, которые могут обозначаться не только идентификаторами, но и просто числами.

Блоки/операторы. Каждый блок языка записывается в отдельной строке и имеет следующую структуру:

[метка] операция [операнды] [комментарии].

Каждое поле отделяется друг от друга пробелами, обязательным является только поле операции, остальные поля могут отсутствовать.

Метка является именем-идентификатором блока. Поле операндов может содержать от 1 до 7 подполей: *A*, *B*, *C*, *D*, *E*, *F*, *G*, содержимое которых отделяется друг от друга запятой. Для пропуска одного из подполей поля операндов ставится просто запятая, например: *A*,,*C*.

Комментарии, кроме поля комментариев, могут быть заданы отдельной строкой: любая строка, начинающаяся с символа «*» или «;», тоже будет комментарием. Поле комментария должно начинаться с символа «;».

Стандартные числовые атрибуты (СЧА). В процессе моделирования язык GPSS автоматически регистрирует и корректирует определенную информацию различных объектов, используемых в модели. Доступ к этой информации осуществляется с помощью СЧА, которые однозначно определяют статус объектов модели. СЧА меняются в процессе имитации, изменить их может как симулятор, так и пользователь. Для указания конкретного объекта, по которому необходимо получить требуемую информацию, за именем СЧА должно следовать числовое или символьное имя этого объекта. Если используется символьное имя, то между СЧА и именем объекта ставится знак $.

В таблице 2.1 приведены некоторые СЧА основных объектов языка. Здесь каждый СЧА обозначается либо <имя СЧА> i, либо

<имя СЧА> $ <имя объекта>,

где i обозначает номер объекта.

Таблица 2.1 – СЧА основных объектов языка GPSS

Объект	СЧА	Назначение
Блок	N$<имя блока>	Число транзактов, вошедших в блок с указанным именем
	W$<имя блока>	Число транзактов, находящихся в указанном блоке
Генераторы случайных чисел	Rni	Случайное число в диапазоне 0–999. При использовании СЧА в качестве аргумента функции представляются действительными числами в диапазоне 0.–0.999999
Транзакт	Pi	Значение i-го параметра
	PR	Значение приоритета
Память	S$<имя памяти>	Текущее содержимое памяти
	R$<имя памяти>	Свободный объем памяти
	SA$<имя памяти>	Среднее число занятых единиц памяти
	SC$<имя памяти>	Число транзактов, вошедших в память с начала моделирования
	SE$<имя памяти>	Память пуста? Если да — 1, нет — 0
	SF$<имя памяти>	Память полна? Если да — 1, нет — 0
	SM$<имя памяти>	Максимальное число занятых единиц памяти
	ST$<имя памяти>	Среднее время нахождения транзакта в памяти
Очередь	Q$<имя очереди>	Текущая длина очереди
	QA$<имя очереди>	Средняя длина очереди
	QC$<имя очереди>	Число транзактов, вошедших в очередь с начала моделирования
	QM$<имя очереди>	Максимальная длина очереди
	QX$<имя очереди>	Среднее время нахождения транзакта в очереди без нулевых входов
	QZ$<имя очереди>	Количество нулевых входов
	QT$<имя очереди>	Среднее время нахождения транзакта в очереди
Устройство	F$<имя устройства>	Состояние устройства: занято — 1, свободно — 0
	FC$<имя устройства>	Число транзактов, вошедших в устройство с начала моделирования
	FT$<имя устройства>	Среднее время занятия транзактом устройства
Переменные	V$<имя переменной>	Значение арифметической переменной
Ячейки	X$<имя ячейки> или Xi	Значение ячейки
Функции	FN$<имя функции>	Значение функции

Мнемокоды. В некоторых блоках языка требуется указывать состояние объектов, для этого используются специальные коды. Перечень мнемокодов языка приведен в таблице 2.2.

Таблица 2.2 – Мнемокоды состояния объектов языка

Состояние объекта	**Мнемокод**
Память: пуста	SE
не пуста	SNE
заполнена	SF
не заполнена	SNF
Устройство: свободно	NU
занято	U
Логический переключатель:	
включен	LS
выключен	LR

2.4 Блоки языка GPSS

2.4.1 Создание и уничтожение транзактов

Генерирование транзактов — GENERATE. Этот блок генерирует поток сообщений — транзактов, поступающих в систему. Программа составляется с учетом того, что в этот блок не могут входить транзакты. В простых программах это обычно первый блок, временные интервалы между поступающими в систему транзактами определяются содержимым поля операндов. Подполя:

- A — среднее время между поступлениями транзактов в систему (по умолчанию равно 1);
- B — модификатор времени;
- C — начальная задержка (время появления первого транзакта);
- D — общее число транзактов, которое должно быть сгенерировано этим блоком (по умолчанию — неограниченное число транзактов);
- E — приоритет транзакта, может принимать значения от 0 до 127. Приоритет возрастает в соответствии с номером (по умолчанию равен 0).

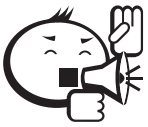
В поле B может быть модификатор двух типов: модификатор-интервал и модификатор-функция. Если задан модификатор-интервал (просто число), то для каждого временного интервала поступления транзактов длительность определяется как значение случайной величины, равномерно распределенной на интервале $[A - B, A + B]$.

Значение параметров A и B могут задаваться как константами, так и любым СЧА, за исключением СЧА параметра транзакта (эта величина в момент генерации транзакта еще не определена).

Например, блок GENERATE 10,5 будет генерировать транзакты через интервалы времени, длительность каждого из которых выбирается случайно в пределах от

5 до 15. Каждое из этих значений будет выбираться с одинаковой вероятностью. Таким образом, блок генерирует случайный поток транзактов, в котором время между транзактами равномерно распределено в диапазоне $A \pm B$ и имеет среднее значение A.

При использовании модификатора-функции интервал времени между транзактами определяется произведением содержимого полей A и B. Функция определяется специальными блоками языка, которые будут рассмотрены ниже.



В программе может быть несколько блоков GENERATE. Все эти блоки работают параллельно и начинают генерировать транзакты одновременно с момента начала моделирования.

Необходимо помнить, что смысл единицы времени в языке GPSS (секунда, минута, час, день и т. д.) закладывает пользователь, поэтому при написании программы необходимо все операнды, связанные со временем, привести к единому масштабу.

Примечания:

- время не может быть отрицательной величиной;
- в обязательном порядке должно быть задано либо поле A, либо поле D.

Блок уничтожения транзактов — TERMINATE. Обычно для простых программ это последний блок программы. Транзакты, попадающие в этот блок, уничтожаются и больше не участвуют в процессе моделирования. Никаких других действий этот блок не выполняет, если единственный возможный операнд A в блоке не задан. Если же операнд A задан, то его значение вычитается из содержимого блока транзактов. Операнд A может принимать только положительное целочисленное значение.

Первоначальная величина счетчика устанавливается специальным управляющим блоком START и пишется в поле A этого блока. Когда в результате входа очередного транзакта в блок TERMINATE значение счетчика становится нулевым или отрицательным, симулятор прекращает моделирование и передает управление программе вывода, которая распечатывает накопленные симулятором данные о модели.

Например:

```
TERMINATE 1
START 100
```

через программу модели пропускается 100 транзактов.

В программе должен быть хотя бы один блок TERMINATE с заданным операндом A.

Если в программе несколько блоков TERMINATE, то обычно операнд A задается только в одном блоке; чаще всего — в блоке, относящемся к имитатору интервала времени моделирования (таймеру).

```
GENERATE 480
TERMINATE 1
START 1
```

Таймер взаимодействует только с блоком START и никак не связан с содержательной стороной остальных фрагментов модели. Таймер служит для задания времени моделирования.

2.4.2 Задержка транзактов в блоках

Блок ADVANCE предназначен для задержки транзактов на определенные интервалы модельного времени.

Обязательный операнд *A* задает время задержки транзакта в блоке ADVANCE. Необязательный операнд *B* является модификатором-функцией или модификатором-интервалом. Значение операнда *B* используется здесь для модификации значения операнда *A* так же, как и в блоке GENERATE.

Любой транзакт входит в блок ADVANCE беспрепятственно. В нем транзакт задерживается на период модельного времени, величина которого определяется операндами *A* и *B*. После этого транзакт направляется к следующему блоку.

Например, в блоке

```
ADVANCE 10
```

транзакт будет задержан на 10 единиц модельного времени.

В блоке

```
ADVANCE 10,P1
```

транзакт будет задерживаться на случайное время, выбранное из диапазона 10± значение первого параметра транзакта (следует помнить, что значение первого параметра при этом не должно превышать 10, т. к. время не может быть отрицательным).

Рассмотрим суммарную задержку в блоках

```
ADVANCE 10,10
ADVANCE 10,10
ADVANCE 10,10
ADVANCE 10,10
ADVANCE 10,10
ADVANCE 10,10
```

Задержка в каждом из них имеет равномерное распределение вероятностей на интервале (0, 20). Следовательно, ее среднее значение составляет $M = 20\cdot(1/2) = 10$; дисперсия $D = 20\cdot(1/2)$. Поэтому сумма шести таких задержек имеет среднее значение $6\cdot M = 60$ и среднее квадратическое отклонение $\sqrt{6}\cdot D \approx 14$. По центральной предельной теореме теории вероятностей заключаем, что закон распределения суммарной задержки приблизительно нормальный. Поэтому ни в коем случае нельзя заменять эти пять блоков на один

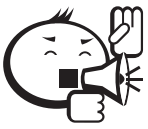
```
ADVANCE 50,50,
```

т. к. этот блок будет определять задержку как равномерно распределенную величину.

2.4.3 Работа с устройствами

Блок SEIZE — занять устройство. При входе транзакта в блок SEIZE выполняется операция занятия устройства, имя которого задается операндом *A* блока SEIZE. Занятие устройства транзактом выполняется следующим образом. Когда транзакт направляется из какого-нибудь блока в блок SEIZE, симулятор проверяет, свободно ли соответствующее устройство. Если оно не свободно, транзакт не может войти в этот блок. Он остается в предыдущем блоке до тех пор, пока устройство не освободится. Если же устройство свободно, то транзакт передвигается в блок SEIZE, занимает устройство и в тот же момент времени направляется к следующему за SEIZE блоку.

Блок RELEASE — освободить устройство. При входе транзакта в блок RELEASE происходит освобождение устройства, имя которого задается операндом *A*.



При составлении моделей пользователь должен соблюдать правило: освободить устройство может только тот транзакт, который его занимает. Если транзакт попытается освободить устройство, занятое другим транзактом, симулятор прервет выполнение модели и выдаст сообщение об ошибке.

В момент освобождения устройства должен быть решен вопрос о том, какой из задержанных транзактов (перед блоком SEIZE) имеет право первым занять устройство. Этот вопрос решается следующим образом: когда транзакты задерживаются перед блоком SEIZE, они регистрируются симулятором в списке, где упорядочиваются по приоритетам: любой транзакт с более высоким приоритетом ставится впереди транзакта, имеющего более низкий приоритет. Если у двух транзактов одинаковые приоритеты, то они упорядочиваются между собой по времени прихода: впереди ставится транзакт, который раньше обратился к устройству. В момент освобождения устройства его занимает тот из задержанных транзактов, который находится в списке первым. Транзакт может занимать любое число устройств. Освобождать занятые устройства транзакт может в любом порядке.

Пример 2.1

Посетители приходят в кассу кинотеатра через 20±10 с, знакомятся в течение 15±15 с обстановкой и занимают очередь. Каждый посетитель приобретает у кассира билеты в течение 20±5 с в зависимости от числа билетов. Построить модель работы кассы кинотеатра в течение четырех часов.

```
GENERATE    20,10    ; приход посетителей
ADVANCE     15,15    ; знакомство с обстановкой
SEIZE       KASS     ; обращение к кассиру
ADVANCE     20,5     ; покупка билета
RELEASE     KASS     ; освобождение кассира
TERMINATE            ; уход посетителя
```

```
GENERATE    1440   ; таймер
TERMINATE   1
START       1
```

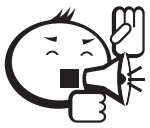
В результате выполнения модели на печать автоматически выводится информация о наличии транзактов в каждом блоке на момент завершения моделирования, а также информация обо всех устройствах, к которым производилось обращение в модели. Формат выводимых данных приведен в приложении.

2.4.4 Сбор статистических данных с помощью очередей

Некоторые виды статистических данных накапливаются симулятором автоматически. Другие виды данных могут быть получены с помощью специальных блоков. При необходимости сбора данных по задержке транзактов перед блоками занятия устройства или памяти используются блоки QUEUE и DEPART.

Блок QUEUE — поставить в очередь. При входе транзакта в этот блок он ставится в очередь, имя которой задается операндом *A*. В начальный момент времени, когда очередь пуста, ее длина равна нулю. В момент входа транзакта в блок QUEUE ее длина увеличивается на величину, указанную в поле *B*. Если операнд *B* пуст, то длина очереди увеличивается на единицу.

Блок DEPART — вывести из очереди. При входе транзакта в блок DEPART длина очереди, имя которой задается операндом *A*, уменьшается на величину, указанную в операнде *B*. При использовании пустого поля *B* в блоках QUEUE и DEPART длина очереди в каждый момент времени соответствует текущему числу транзактов в этой очереди. Транзакты могут проходить любое число блоков QUEUE и DEPART с произвольными значениями полей *A* и *B*, чередующихся в любом порядке.



Необходимо помнить, что данные блоки не влияют на реальное образование очередей транзактов, а служат только для сбора статистических данных. Поэтому пользователь должен следить за правильным расположением этих блоков, чтобы не получать отрицательные длины образуемых очередей. Симулятор только подсчитывает статистику по очередям и не считает за ошибку отрицательные длины очередей.

Пример 2.2

Изменим модель, построенную в примере 2.1 таким образом, чтобы получить информацию об очереди, образующейся перед кассой.

```
GENERATE    20,10
ADVANCE     15,15
QUEUE       OCH    ; включение в очередь
```

```
SEIZE       KASS   ; обращение к кассиру
DEPART      OCH    ; выход из очереди
ADVANCE     20,5
RELEASE     KASS
TERMINATE
GENERATE    1440   ; таймер
TERMINATE   1
START       1
```

В этой модели момент включения каждого транзакта в очередь OCH совпадает с моментом его обращения к блоку SEIZE , т. к. блок QUEUE выполняется в модельном времени мгновенно. Каждый транзакт находится в очереди до тех пор, пока не займет устройство KASS. Момент занятия устройства совпадает с моментом выхода транзакта из очереди. В данном случае очередь OCH имеет естественную интерпретацию как очередь посетителей к кассиру, а длина очереди интерпретируется как число посетителей в очереди.

. . . . . .

При наличии в модели очередей симулятор выдает статистику по очередям. Формат выводимых данных приведен в приложении.

2.4.5 Функции

При использовании в блоках GENERATE и ADVANCE поля *B* в качестве модификатора функции, саму функцию необходимо описать специальным блоком языка FUNCTION.

В поле метки данного блока стоит имя функции (поле метки в данном случае является обязательным). В операнде *A* блока FUNCTION указывается аргумент функции, а в операнде *B* — тип функций и количество пар аргументов и значений.

Аргумент функции задается с помощью СЧА. Чаще всего в качестве аргумента используются датчики случайных чисел RN1, RN2, RN200.

Существуют следующие типы функций: *C*, *D*, *E*, *L*, *M*. Функции типа *C* — непрерывны (аргумент *X* и значение *Y* функции задаются типами integer и real), типа *D* — дискретны (*Y* кроме «integer» и «real» может принимать значение «имя»). Например, *C* 12 означает, что функция непрерывна и для ее описания будет использоваться 12 пар аргументов-функций. Тип *E* — дискретная функция со значением *Y*, заданным с помощью СЧА. Типы *L* и *M* — функции со списком, здесь для *L* значение функции интерпретируется как номер элемента списка, для *M* аргумент функции является элементом списка.

При описании любой из функций с помощью языка GPSS происходит интерполяция. Для дискретных функций — это кусочно-постоянная интерполяция, для непрерывных — линейная интерполяция. Координаты функции, задаваемые парами, являются узлами интерполяции.

За блоком описания функции FUNCTION всегда следует блок задания функции, в котором задаются координаты и значения функции. Каждая пара чисел координата-значение отделяется друг от друга слэджем, пробелы недопустимы. В паре аргумент отделяется от значения функции запятой.

Необходимо помнить, что аргумент функции может принимать только неотрицательные значения. Поле комментариев в данном блоке не используется.

Например, функция, график которой показан на рис. 2.1, *а*, описывается на языке GPSS следующим образом:

```
FUNC1 FUNCTION RN1, D3
.4,26.0/.8,40.8/1,6.0
```

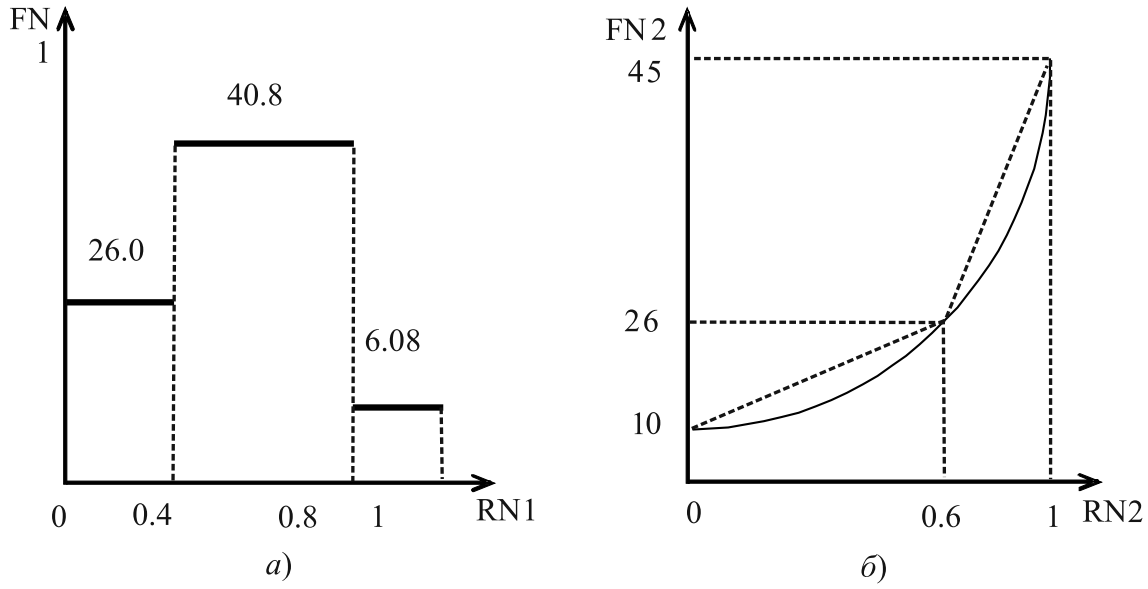


Рис. 2.1 – Графики дискретной (*а*) и непрерывной (*б*) функций

Непрерывная функция, показанная на рис. 2.1, *б*:

```
FUNC2 FUNCTION RN2, C3
0,10/.6,26/1,45
```

Блоки описания и задания функции располагаются в начале программы, до первого блока GENERATE. Координаты точек функции записываются как числа с фиксированной точкой либо именем (для значения функции типа *D*).

В языке существует 3 датчика равномерно распределенных случайных чисел, которые обозначаются RN1, RN2, RN200. Эти датчики выдают равновероятные целочисленные значения из диапазона 0–999. Если датчик используется в качестве аргумента функции, то он выдает вещественные числа в диапазоне 0–1.

Для генерации случайных величин, распределенных по экспоненциальному закону, можно использовать встроенное вероятностное распределение, которое описывается выражением

```
EXPONENTIAL(A,B,C)
```

и при обращении к нему заключается в скобки.

Здесь поле *A* определяет номер датчика случайных чисел, поле *B* — сдвиг среднего и поле *C* — сжатие функции. В результате значение среднего определяется суммой полей *B* и *C*, а значение дисперсии равно квадрату поля *C*.

Пример 2.3

В аэропорту производится регистрация пассажиров перед посадкой в самолет. На регистрацию подходят отдельные пассажиры через каждые 20±10 с либо туристические группы через каждые 60±20 сек. При этом туристические группы обслуживаются вне очереди. Время обслуживания подчинено экспоненциальному закону и равно в среднем для отдельных пассажиров — 15 сек, для туристических групп — 25 сек. Промоделировать работу отдела регистрации, изучив статистику по очереди за 2 ч.

```
EXP FUNCTION RN1,C6
0,0/.1,.1/.2,.2/.5,.69/.8,1.6/1,8
GENERATE    20,10          ; приход отдельных пассажиров
QUEUE       LIN            ; включение в очередь
SEIZE       REG
DEPART      LIN            ; выход из очереди
ADVANCE     15,FN$EXP      ; регистрация пассажира
RELEASE     REG
TERMINATE                  ; уход пассажира
GENERATE    60,20,,,1      ; приход туристической группы
QUEUE       LIN
SEIZE       REG
DEPART      LIN
ADVANCE     25,FN$EXP      ; регистрация группы
RELEASE     REG
TERMINATE                  ; уход группы
; таймер
GENERATE    720
TERMINATE   1
START       1
```

Здесь отдельные пассажиры и туристические группы встают в одну и ту же очередь и обслуживаются одним регистратором. Внеочередность обслуживания групп в модели обеспечивается заданием приоритета для транзактов, имитирующих туристические группы.

В программе три самостоятельных сегмента, каждый из которых начинается блоком GENERATE и заканчивается блоком TERMINATE. Они могут быть поставлены в программе в любом порядке. При этом процесс моделирования останется неизменным: все блоки GENERATE работают параллельно.

В примере экспоненциальная функция распределения описана первыми двумя строками программы. Можно пользоваться встроенной функцией, тогда описание функции можно опустить, а первый блок ADVANCE, например, будет выглядеть следующим образом:

```
ADVANCE (EXPONENTIAL(1,0,15))    ; регистрация пассажира
```

2.4.6 Изменение маршрутов сообщений

Блок TRANSFER позволяет осуществлять безусловные, статистические и условные переходы. Тип перехода определяется в операнде *A*, направление перехода — в операндах *B*, *C* и *D*.

В режиме *безусловного перехода* операнд *A* в блоке пуст. Все транзакты переходят к блоку, указанному в поле *B*. Например:

```
TRANSFER ,NEXT
```

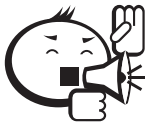
Если блок, к которому направляется транзакт, в текущий момент системного времени не может его принять (например, блок SEIZE), то транзакт остается в блоке TRANSFER и повторяет попытку перехода при каждом пересчете системного времени симулятором.

Если в поле *A* блока TRANSFER записана десятичная дробь, начинающаяся точкой, то блок работает в режиме *статистического перехода*. Здесь десятичная дробь определяет вероятность перехода транзакта к блоку, имя которого указывается в поле *C*. При этом поле *B* пустое. С вероятностью (1−<A>) транзакт переходит к блоку, следующему за блоком TRANSFER.

Если оба блока заняты, то транзакт остается в блоке TRANSFER и повторяет попытку перехода к выбранному ранее блоку при каждом изменении системного времени.

С помощью этого блока можно промоделировать, например, выбор покупателями в магазине одного из двух отделов, если известно, что половина покупателей направляется в 1-й отдел, а вторая половина — во 2-й отдел:

```
TRANSFER     .5,,OTD2
OTD1 SEIZE   PROD1
...
OTD2 SEIZE   PROD2
```



При моделировании статистических переходов необходимо обращать внимание на следующий момент: если требуется разбить поток заявок более чем на два, необходимо внимательно рассматривать вероятности статистических переходов в модели. В этом случае, если вы используете несколько блоков статистического перехода TRANSFER, для всех блоков, кроме первого, необходимо пересчитывать вероятности переходов с учетом того, что часть заявок уже была перенаправлена к другому блоку.

Например, известно, что поток покупателей разбивается на три потока (три отдела в магазине) следующим образом: 50% покупателей направляется в 1-й отдел, 20% — во второй и 30% — в третий. Тогда перенаправление транзактов будет происходить следующим образом:

```
TRANSFER     .5,,OTD1
TRANSFER     .4,,OTD2   ; 20% от оставшихся 50% — это 40%
TRANSFER     ,OTD3      ; все остальные — в 3-й отдел
```

Ситуация значительно усложняется, если проценты распределения заявок равны, например 17, 47 и 36 (теперь надо будет рассчитать долю 47 от оставшихся 100 – 17 = 83, а это 0.566265…). Чтобы упростить моделирование такой ситуации, можно построить дискретную функцию типа *D*, значением которой будут имена блоков, к которым должны направляться потоки заявок. Такая функция строится по принципу моделирования группы событий (см. главу 1).

Для распределения заявок на 17, 47 и 36 процентов можно построить следующую функцию:

```
PEREX FUNCTION RN1,D3
.17,OTD1/.64,OTD2/1,OTD3
```

а затем обращаться к функции в блоке безусловного перехода:

```
TRANSFER ,FN$PEREX
```

Условный переход. Режим условного перехода определяется мнемокодом, заданным в поле *A*. Рассмотрим различные режимы.

Если в поле *A* определено значение BOTH, то транзакт первоначально направляется к блоку, имя которого определено в поле *B*. Если переход невозможен (например, занято устройство), то делается попытка перейти к блоку, чье имя определено в поле *C*. Если оба блока заняты, то транзакт остается в блоке TRANSFER и повторяет попытку перехода при каждом изменении системного времени.

Если в поле *A* определено значение ALL, то поля *B* и *C* содержат имена блоков, поле *D* содержит целое число. Транзакт последовательно пытается войти в блоки, отстоящие друг от друга на расстояние *D*, начиная с блока *B* и заканчивая блоком *C* до первой успешной попытки. Если ни один из блоков не может принять транзакт, то он остается в блоке TRANSFER и повторяет попытку перехода при каждом изменении системного времени. Здесь значение поля *D* должно задаваться таким образом, чтобы выполнялось условие $№C = №B + M \cdot D$, где M — любое целое число. Если поле *D* не задано, то транзакт пытается последовательно войти в каждый блок между *B* и *C*.

Если в поле *A* определено значение PICK, то поля *B* и *C* содержат имена блоков, а транзакт направляется в любой блок между блоками *B* и *C*, выбранный случайным образом.

Блок GATE позволяет изменять путь транзакта в зависимости от состояния моделируемого оборудования. Блок имеет следующую структуру:

```
GATE O A,B
```

В поле *O* задается проверяемое состояние оборудования в виде мнемокода. В поле *A* задается имя проверяемой единицы оборудования, в поле *B* — имя блока, к которому направляется транзакт, если проверяемое условие ложно.

Данный блок может работать в двух режимах: в режиме отказа и в режиме условного перехода.

Режим отказа: транзакт задерживается в блоке GATE до тех пор, пока не выполнится условие состояния проверяемого объекта. Как только это произойдет, транзакт направляется к следующему за GATE блоку. В этом режиме поле *B* опускается.

Режим условного перехода: если проверяемый объект не находится в требуемом состоянии, транзакт направляется к блоку, указанному в поле *B*. В противном случае транзакт направляется к следующему за GATE блоку. Например, в блоке

GATE SF STR

транзакт будет задержан до тех пор, пока память с именем STR не будет полной.

Блок TEST изменяет маршрут транзакта в зависимости от выполнения разнообразных логических условий, определенных на множестве СЧА. Блок имеет следующую структуру:

TEST O A,B,C

В поле *O* указывается мнемоника отношения: «L» — «<», «LE» — «⩽», «E» — «=», «NE» — «≠», «G» — «>», «GE» — «⩾».

В полях *A* и *B* указываются левое и правое значения условия, соответственно. В поле *C* указывается имя блока, к которому направляется транзакт, если проверяемое условие ложно. Если проверяемое условие истинно, то транзакт переходит к следующему за TEST блоку.

Например, при входе транзакта в оператор

TEST G Q$OCH,5,OTD1

проверяется длина очереди OCH. Если длина очереди больше пяти, то транзакт направляется к следующему за TEST блоку, иначе транзакт переходит к блоку с именем OTD1.

Пример 2.4

В магазине находится два отдела: продовольственный и промтоварный. Около 30-ти процентов приходящих в магазин покупателей направляются в промтоварный отдел, остальные — в продовольственный. Причем если очередь в промтоварном отделе больше двух человек, а в продовольственном — больше пяти, то покупатели уходят из магазина, не дожидаясь обслуживания. Время прихода и обслуживания покупателей распределено экспоненциально. Среднее значение времени прихода равно соответственно 20 сек, времени обслуживания в продовольственном отделе — 30 сек и в промтоварном — 35 сек.

Модель, имитирующая работу магазина за 8 ч:

```
GENERATE (EXPONENTIAL(1,0,20))   ; приход покупателей
TRANSFER .3,,PROM                ; выбор покупателем отдела
; работа продовольственного отдела
PROD TEST LE Q$LIN1,5,BYBY       ; если очередь больше 5-ти чел. —
; уход покупателя
QUEUE LIN1                       ; поставить в очередь
; в продовольственный отдел
SEIZE PROD1                      ; занять продавца
DEPART LIN1                      ; покинуть очередь
; в продовольственный отдел
ADVANCE (EXPONENTIAL(1,0,30))    ; обслуживание покупателя
RELEASE PROD1                    ; освободить продавца
TERMINATE                        ; уход покупателя
; работа промтоварного отдела
PROM TEST LE Q$LIN1,2,BYBY
QUEUE LIN2
```

```
SEIZE PROD2
DEPART LIN2
ADVANCE (EXPONENTIAL(1,0,35))
RELEASE PROD2
BYBY TERMINATE
; таймер
GENERATE 2880
TERMINATE 1
START 1
```

2.4.7 Работа с памятью

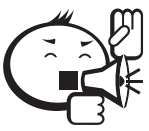
Память — особый объект языка, который призван имитировать разного рода накопители, используемые в исследуемых системах, в которых может одновременно находиться несколько транзактов. Для каждой применяемой памяти пользователь должен указать ее емкость — объём памяти, определяющий максимальное количество транзактов, которые могут одновременно находиться в ней. Для указания емкости используется *оператор описания памяти* STORAGE. Как любой оператор описания языка этот блок помещается до первого блока GENERATE. Поле метки содержит имя памяти, а операнд *A* указывает емкость памяти. Например, для описания памяти емкостью 10 единиц используется блок

```
STR STORAGE 10
```

Блок ENTER — занять память. В поле *A* блока указывается имя памяти, в которую помещается транзакт, в поле *B* — число единиц памяти, занимаемых транзактом при входе. Когда транзакт входит в блок ENTER, определяется число свободных единиц памяти. Если значение операнда *B* не превышает числа свободных единиц памяти, то число занятых единиц увеличивается на значение операнда *B*. В этом случае транзакт входит в блок ENTER без задержки. Если же значение операнда *B* превышает число свободных единиц памяти, то транзакт задерживается перед входом в блок ENTER. Задержанные при обращении к памяти транзакты упорядочиваются по приоритету.

Если поле *B* в блоке ENTER пустое, то число занимаемых единиц памяти принимается равным единице.

Пусть транзакт «*x*» задержан перед входом в блок ENTER. Если для транзакта «*y*», приходящего после «*x*», свободной емкости памяти достаточно, то «*y*» войдет в блок без задержки.



Примечание: если вы используете блок ENTER, память должна быть обязательно описана ранее командой STORAGE.

Блок LEAVE — освободить память. В поле *A* блока указывается имя освобождаемой памяти, в поле *B* — число освобождаемых единиц. В случае пустого поля *B* число освобождаемых единиц памяти принимается равным единице. При входе транзакта в блок LEAVE количество занятых единиц памяти, указанной в поле *A*,

уменьшается на значение операнда *B*. Перед входом в блок транзакты не задерживаются. Транзакт не должен освобождать большее число единиц памяти, чем их всего занято. Если же транзакт пытается это сделать, то симулятор выдает на печать сообщение об ошибке и прекращает выполнение модели.

В тот момент модельного времени, когда транзакт освобождает память, симулятор просматривает список задержанных у памяти транзактов, если они есть. Для каждого очередного транзакта проверяется, может ли он теперь быть обслужен памятью. Если такая возможность есть, то симулятор перемещает этот транзакт в блок ENTER, и в результате число занятых единиц памяти соответствующим образом увеличивается.

Транзакт не обязан освобождать такое же число единиц памяти, какое занимал. Он может также освобождать память, которую не занимал. Транзакт имеет право занимать и освобождать любое количество памятей, при этом операции занятия и освобождения могут чередоваться в произвольном порядке.

Пример 2.5

Автомобили подъезжают к бензозаправочной станции в среднем каждые 4±2 мин. На станции есть две бензоколонки, каждая из которых используется в среднем 5±1 мин. Автостоянка при станции рассчитана на 4 автомобиля. Если подъехавший автомобиль застает обе бензоколонки занятыми, то он встает в очередь на автостоянку. Если же все места и на автостоянке заняты, то автомобиль проезжает мимо. Промоделировать работу станции за 12 часов.

```
STO STORAGE 4                ; места под автостоянку
COL STORAGE 2                ; бензоколонки
; описание работы бензоколонки
GENERATE 4,2                 ; приезд автомобиля
GATE SNF STO,BYBY            ; если места заняты — проезжает
ENTER STO                    ; занять место на автостоянке
ENTER COL                    ; занять бензоколонку
LEAVE STO                    ; освободить автостоянку
ADVANCE 5,1                  ; заправиться
LEAVE COL                    ; освободить бензоколонку
BYBY TERMINATE               ; покинуть станцию
; таймер
GENERATE 720
TERMINATE 1
START 1
```

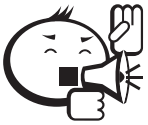
2.4.8 Вычислительные объекты языка

Арифметические переменные. Для того чтобы использовать в программе переменную, необходимо сначала ее описать оператором описания VARIABLE либо FVARIABLE. В поле метки оператора записывается имя переменной, в операн-

де *A* — арифметическое выражение, составляемое из СЧА, знаков арифметических операций и круглых скобок. Используются следующие арифметические операции: +, –, # (умножение), /, @ (взять остаток от деления), \ (целое от деления). Приоритет операций стандартный. Деление на ноль не считается ошибкой, и результатом такого деления является ноль. Остаток от деления на ноль также считается равным нулю.

При использовании переменной в программе указывается СЧА переменной: V$<имя переменной>, например,

ADVANCE V$VAR1 — задержать транзакт на время, заданное переменной VAR1.



Примечание: переменная является единственным объектом языка, по которому по окончанию моделирования в отчете не выдается никакой информации. Поэтому, если вам необходимо по окончанию моделирования проанализировать значение переменной, то можно присвоить ее значение ячейке, значение которой выводится в результирующем отчете.

Ячейки служат для хранения некоторых постоянных и/или изменяющихся значений данных программы. В отличие от большинства объектов языка ячейка может обозначаться как именем, так и числом. Для работы с ячейками используется блок SAVEVALUE. В поле *A* этого блока указывается номер/имя ячейки, сохраняющей значение, и вид изменения этого значения («+» — накопление, «–» — уменьшение). В поле *B* содержится либо СЧА, либо число, которое добавляется либо вычитается, либо заменяет содержимое ячейки.

Например, оператор

SAVEVALUE 10+,1

означает, что при поступлении транзакта в блок, к содержимому 10-й ячейки прибавляется единица. Или оператор

SAVEVALUE FRT,V$VAR1

означает, что при поступлении транзакта в блок, в ячейку с именем FRT записывается значение переменной VAR1.

При необходимости обращения к ячейке указывается СЧА ячейки: X$<имя ячейки> (если ячейка задана именем) или XN (если ячейка задана номером N).

Чаще всего на базе ячеек организуются разного рода счетчики. Перед началом имитации содержимое всех используемых в программе ячеек устанавливается в 0. Если же требуется задать значение какой-либо из ячеек до начала моделирования, то для этого используется оператор INITIAL, в поле *A* которого задается СЧА ячейки, в поле *B* — присваиваемое значение. Например,

INITIAL X$UCH1,10 — присвоить ячейке с именем UCH1 значение 10. Этот оператор должен помещаться до первого блока GENERATE.

Необходимо помнить, что в поле *B* ячейки не могут стоять арифметические выражения. При необходимости используйте в этом поле СЧА переменной, которая описывает требуемое выражение.

Матрицы служат для хранения некоторых постоянных и/или изменяющихся значений данных программы в виде массивов. Для того чтобы использовать в программе матрицу, необходимо сначала ее описать оператором описания MATRIX. В поле метки оператора записывается имя матрицы. Поле *A* в операторе не используется, поля *B* и *C* содержат числовые значения, определяющие количество строк и количество столбцов в матрице, соответственно. Начальные значения всех элементов матрицы равны нулю. Если необходимо присвоить всем элементам матрицы одинаковые значения, отличные от нуля, используется оператор INITIAL, в поле *A* которого задается имя матрицы, в поле *B* — присваиваемое значение.

Для изменения значения отдельного элемента матрицы используется *блок MSAVEVALUE*. В поле *A* блока указывается имя матрицы, после которого может быть указан «+» или «–». Поля *B* и *C* служат для выбора конкретного элемента матрицы и содержат номер строки и номер столбца, соответственно. В поле *D* указывается значение, которое должно быть добавлено, вычтено или присвоено элементу матрицы. Если после имени матрицы не стоит никакого знака, то значение *D* присваивается элементу. Если после имени матрицы стоит знак «+» или «–», то указанное значение добавляется или вычитается из текущего значения элемента соответственно.

При необходимости обращения к элементу матрицы указывается СЧА элемента: MX$<имя матрицы>(I, J) (если матрица задана именем) или XN(I, J) (если матрица задана номером N). Здесь I означает номер строки, J — номер столбца.

Пример 2.6

Модель работы двухтактного буферного запоминающего устройства.

Идея двухтактного буфера: совместить во времени процессы сбора и записи информации в буферное запоминающее устройство (БЗУ) ограниченного объема и перезаписи информации в долговременное запоминающее устройство (ДЗУ), объем которого неограничен.

Техническое осуществление — с использованием канала прямого доступа и программного канала. Рассмотрим упрощенную модель системы двухтактного буферирования. Экспериментальная информация в процессе сбора записывается словами в буфер. При заполнении приемного БЗУ до определенного уровня генерируется запрос к управляющему устройству (УУ) на передачу информации в ДЗУ (включается система прерываний). Между моментом возникновения запроса и переключением ключей (рис. 2.2) проходит определенное время, по истечении которого приемное БЗУ становится передающим (например, подключается канал прямого доступа и информация пишется на диск), а для сбора информации в оперативной памяти выделяется новая область, которая становится приемным БЗУ.

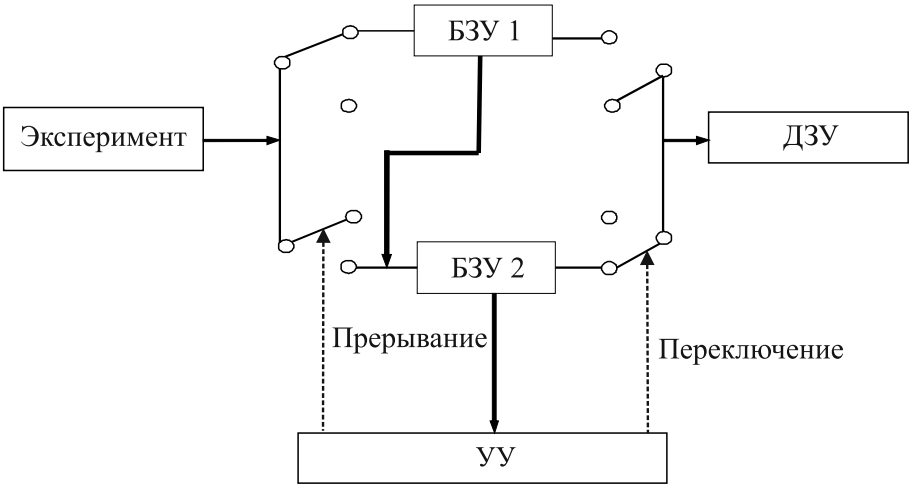


Рис. 2.2 – Схема двухтактного БЗУ

При исследовании такой системы на модели могут ставиться различные вопросы: каков должен быть уровень заполнения БЗУ, при котором в системе генерируется запрос на переключение ключей; как связан этот уровень с интенсивностью потока экспериментальных данных и вероятностью переполнения приемного БЗУ за время между запросом и переключением ключей и т. п.

Рассмотрим упрощенное описание модели на языке GPSS. Допустим, объем БЗУ — 1024 слова. Разрядность слова — 8 единиц. Тогда память БЗУ1/БЗУ2:

```
VAR2 VARIABLE 1024#8
STR1 STORAGE V$VAR2
```

Допустим, запрос на прерывание возникает при заполнении БЗУ до 1020 слов, тогда уровень заполнения:

```
VAR3 VARIABLE 1020#8
```

Еще одна переменная — описание среднего времени задержки между запросом и переключением БЗУ:

```
VAR4 VARIABLE 10/4
```

Пусть время между приходом слов в БЗУ распределено экспоненциально и равно в среднем 10 мин., тогда функционирование этой системы будет выглядеть следующим образом:

```
GENERATE (EXPONENTIAL(1,0,10))
GATE SNF STR1,OTKAZ                  ; если память заполнена — отказ
ENTER STR1,8
TEST E S$STR1,V$VAR3,ENDM
ADVANCE (EXPONENTIAL(1,0,V$VAR4))    ; переключение БЗУ
LEAVE STR1,S$STR1
TERMINATE 1
OTKAZ SAVEVALUE 1+,1                 ; счетчик слов, потерянных
                                     ; из-за переполнения БЗУ
ENDM TERMINATE
```

Здесь транзакт попадает в блок ADVANCE только тогда, когда БЗУ заполнена до необходимого уровня. Блок LEAVE полностью очищает память STR1, что равносильно переключению на новое БЗУ.

Пример 2.7

Производство деталей включает длительный процесс сборки, заканчивающийся коротким периодом обжига в печи. Поскольку содержание печи обходится дорого, несколько сборщиков используют одну печь, в которой одновременно можно обжигать только одну деталь. Сборщик не может начать новую сборку, пока не вытащит из печи предыдущую деталь.

Время сборки детали равно 30±5 минут. Время обжига равно 8±2 минуты. Зарплата сборщика составляет 60 рублей в час, стоимость эксплуатации печи — 3200 рублей за 8 часов. Цена материала, идущего на изготовление одного изделия равна, 100 рублей, стоимость готового изделия — 380 рублей. Тогда прибыль от изготовления детали составит 380 – 100 = 280 рублей.

Необходимо определить оптимальное количество сборщиков исходя из максимизации прибыли за неделю (возьмем 8-часовой рабочий день без выходных $8 \cdot 7 = 56$ часов).

Для удобства исследования число сборщиков в модели зададим переменной SBOR. Меняя значение переменной, мы сможем подобрать требуемое число рабочих.

Затраты на содержание печи и зарплату сборщиков опишем в переменной ZATR. Доходы от производства будем подсчитывать с помощью ячейки IZD. А результирующую прибыль определим в переменной PRIB.

В качестве транзактов в модели будем рассматривать самих сборщиков, подразумевая, что в каждый момент времени один сборщик производит одну деталь. Устройство PECH вводим для обозначения печи.

```
SBOR VARIABLE N                          ; при прогоне модели вместо N
                                         ; ставим конкретное целое число
ZATR VARIABLE 3200#7+V$SBOR#60#56
PRIB VARIABLE X$IZD–V$ZATR
GENERATE ,,,V$SBOR
PROD ADVANCE 30,5                        ; процесс сборки изделия
SEIZE PECH
ADVANCE 8,2
RELEASE PECH
SAVEVALUE IZD+,280
TRANSFER ,PROD
GENERATE (56#60)
SAVEVALUE REZULT,V$PRIB
TERMINATE 1
START 1
```

В таблице 2.3 приведены результаты прогона модели для разного количества работников.

Таблица 2.3 – Прибыль предприятия при разном количестве работников

№	Прибыль	Комментарий
1	–10360	1 работник — убыточное предприятие
2	840	
3	12320	
4	22400	
5	30240	Максимальная прибыль — оптимальное количество работников
6	29960	

. . . . . .

2.4.9 Приоритеты

Каждый транзакт может иметь свой приоритет — от 0 до 127. Чем больше номер, тем больше приоритет. Предпочтение в системе отдается транзактам с большим приоритетом, ранее поступившим.

Для изменения приоритета транзакта в процессе его путешествия по системе используется блок PRIORITY. Поле этого блока определяет значение присваиваемого приоритета. Например, при прохождении через блок PRIORITY 3 транзакту будет присвоен приоритет 3.

2.4.10 Изменение параметров транзакта

Каждый транзакт может иметь до 100 параметров (атрибутов). *Значения параметрам присваиваются* с помощью *блока ASSIGN*. В поле *A* этого блока указывается номер/имя параметра и вид его изменения, в поле *B* определяется записываемое в параметр значение, в поле *C* задается при необходимости модификатор значения *B* в виде имени функции, значение которой умножается на *B*.

Приписывая к номеру параметра в поле *A* символ + или –, можно обеспечить не запись значения поля *B* в параметр, а добавление или вычитание этого значения из значения параметра. В поле *B* значение может быть задано как целым числом, так и СЧА. Например:

```
ASSIGN 1,10              ; занести 10 в P1
ASSIGN 2+,V$VAR1,EXP     ; добавить в P2 значение
                         ; V$VAR1*FN$EXP
ASSIGN TRE –,S$STR       ; вычесть из P$TRE значение текущего
                         ; содержимого памяти
```

Используя блок ASSIGN, можно организовывать циклы в программе. Например, если необходимо прогнать транзакт 10 раз через блок ADVANCE, это можно осуществить следующим образом:

```
ASSIGN 1,10          ; занести 10 в P1 транзакта
PROD ADVANCE 52
ASSIGN 1–,1          ; вычесть 1 из P1
TEST E P1,0,PROD     ; продолжать цикл, пока счетчик не
                     ; обнулится
```

Пример 2.8

В магазине электротоваров работают два консультанта. Посетители заходят в магазин в среднем каждые 2 минуты. Покупатели осматривают товар в среднем в течение 10 минут, после чего примерно 70% из них обращаются к консультанту за помощью (время на консультацию составляет в среднем 5 минут). После осмотра товара и получения консультации примерно 40% посетителей уходят без покупки. Остальные покупатели с выбранным товаром направляются к кассе. Кассир обслуживает клиентов в среднем 3 минуты.

Стоимость покупки распределена равномерно на интервале от 500 до 20000 рублей. Известно, что примерно у 3-х процентов покупателей есть карточка со скидкой в 10%, а у 12-ти процентов покупателей есть карточки со скидкой в 5%.

Необходимо оценить прибыль магазина за 10-часовой рабочий день.

Наличие карточки со скидкой у покупателя мы опишем функцией SKIDKA, и будем записывать ее значение в 1-й атрибут транзакта.

Стоимость производимой покупки опишем с помощью функции POKUP, прибыль магазина опишем переменной SUM.

Консультанты в модели будут представлены памятью CONS, кассир — устройством KAS.

Модель работы магазина:

```
CONS STORAGE 2
POKUP FUNCTION RN1,C2
0,500/1,20000
SKIDKA FUNCTION RN1,D3
.03,0.10/.15,0.05/1,0
SUM VARIABLE FN$POKUP#(1–P1)
    GENERATE (EXPONENTIAL(1,0,2))
    ASSIGN 1,FN$SKIDKA
    ADVANCE (EXPONENTIAL(1,0,10))
    TRANSFER .3,,MET1
    ENTER CONS
    ADVANCE (EXPONENTIAL(1,0,5))
    LEAVE CONS
MET1 TRANSFER .4,,UXOD
    SEIZE KAS
    ADVANCE (EXPONENTIAL(1,0,3))
    SAVEVALUE PRIB+,V$SUM
    REKEASE KAS
UXOD TERMINATE
    GENERATE 600
    TERMINATE 1
    START 1
```

2.4.11 Косвенная адресация

Косвенная адресация используется в сочетании с любым СЧА, кроме текущего времени C1, случайного числа RNi и времени пребывания транзакта в системе M1.

Структура записи косвенной адресации СЧА∗<целое число/имя>, где <целое число/имя> — это номер/имя параметра, в котором указан номер или имя соответствующего СЧА, например: SEIZE FN∗1.

Здесь при поступлении транзакта определяется сначала имя функции (по первому параметру транзакта), а затем — значение этой функции. По этому значению и определяется устройство, занимаемое транзактом. Получается, что разные транзакты, попадая в данный блок, могут занимать различные устройства в зависимости от значения собственных параметров и значения выбранных функций.

В блоке GENERATE косвенная адресация допускается только в поле *A*.

2.4.12 Списки

Списки определяют внутреннюю организацию GPSS. В языке существует 5 типов списков. Следующие четыре типа ведет симулятор (пользователь не имеет к ним доступа):

- списки текущих событий: содержат транзакты, у которых время очередной передвижки в программе модели меньше системного времени; активные транзакты находятся в состоянии задержки, например по причине занятости устройства;
- списки будущих событий: содержат транзакты, у которых время очередной передвижки больше системного времени (например, все транзакты в блоках ADVANCE);
- списки прерываний: содержат транзакты, обслуживание которых на определенных устройствах было прервано другими транзактами;
- списки синхронизируемых и задержанных транзактов.

Пятый тип списков — это списки пользователя, которые можно создавать и использовать в процессе моделирования. Управление списками осуществляется с помощью двух следующих блоков.

Блок LINK выполняет включение транзакта в список пользователя. Поле *A* блока содержит номер или имя списка, куда включается транзакт. Транзакт, попадая в блок LINK, помещается в список, определенный полем *A*, и временно исключается из процесса (становится пассивным).

Дисциплина постановки в список определяется полем *B*, которое может принимать одно из следующих значений:

- FIFO (первым пришел, первым вышел) — вновь поступивший транзакт помещается в конец списка;
- LIFO (последним пришел, первым вышел) — вновь пришедший транзакт помещается в начало списка.

В любом другом случае значение поля *B* вычисляется, и транзакт помещается в список в соответствии с вычисленным значением (за транзактом, у которого соответствующее значение больше, и перед транзактом, у которого соответствующее

значение меньше). Например, транзакты могут быть упорядочены в соответствии со значениями их параметров. Если используется СЧА PR, то транзакты упорядочиваются в списке по приоритетам.

Блок UNLINK осуществляет извлечение транзакта из списка. Поле *A* блока содержит номер или имя списка, из которого извлекается транзакт. Поле *B* содержит имя блока, к которому направляется извлеченный транзакт. В поле *C* указывается количество транзактов, извлекаемых из списка (по умолчанию извлекаются все транзакты).

При попадании некоторого транзакта в блок UNLINK пассивный транзакт, находящийся в списке пользователя, извлекается оттуда и направляется к блоку, имя которого указано в поле *B*, основной же транзакт, вызвавший это извлечение, продолжает свое обычное движение по программе.

Со списками связаны следующие СЧА:

- CA$<имя списка> — среднее число транзактов в списке;
- CC$<имя списка> — общее число транзактов, прошедших через указанный список;
- CH$<имя списка> — текущее число транзактов в списке;
- CM$<имя списка> — максимальное число транзактов в списке;
- CT$<имя списка> — среднее время, проведенное транзактом в списке.

2.4.13 Статистические таблицы

Объект типа таблицы представляет собой эквивалент понятия «гистограмма». Гистограммы применяются для статистического анализа такой случайной величины, функция распределения которой неизвестна, но зато имеется достаточно большое число независимых реализаций этой величины.

Для того, чтобы таблицы можно было использовать в модели, они должны быть описаны.

Для описания таблицы используется *блок TABLE*. В поле метки этого блока задается имя таблицы, в поле *A* — аргумент таблицы в виде СЧА. Здесь аргументом таблицы является исследуемая случайная величина. В поле *B* указывается верхняя граница первого частотного интервала, в поле *C* — ширина интервалов, а в поле *D* — их число, включающее оба полубесконечных интервала.

Исключением может быть время, проводимое транзактом в очереди. Если необходимо исследовать это время, то используется *блок QTABLE*, в поле *A* которого указывается имя очереди, время нахождения в которой нас интересует. Остальные поля данного блока описываются аналогично блоку TABLE. Например, если нас интересует гистограмма времени, проводимого одним транзактом в очереди LIN, то мы можем описать таблицу следующим блоком:

TBL QTABLE LIN,10,20,5

Графическое представление гистограммы приведено на рисунке 2.3.

Если таблица описана, то транзакты могут *фиксировать* в ней *информацию* с помощью *блока TABULATE*. В поле *A* этого блока указывается имя таблицы, в которой накапливается информация. При входе транзакта в блок TABULATE вычисляется значение аргумента указанной таблицы и определяется, в какой из

интервалов таблицы это значение попадает. После этого счетчик соответствующей интервальной частоты увеличивается на 1.

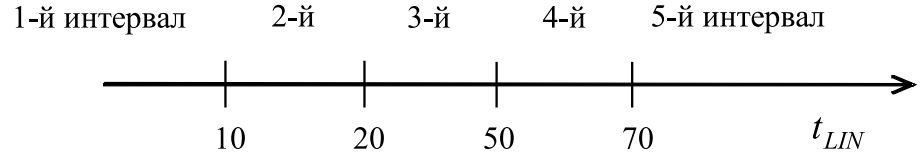


Рис. 2.3 – Графическое представление гистограммы

Примечание: если используется блок QTABLE, то блок TABULATE не нужен, вычисление искомой характеристики в данном случае происходит автоматически.

В результате моделирования на печать по каждой таблице выдается информация в виде, показанном в приложении. Для каждой таблицы автоматически осуществляются оценки среднего и среднеквадратического отклонений.

Пример 2.9

Модель вычислительной системы с несколькими абонентскими пунктами (АП).

Пусть ЭВМ обслуживают 5 АП. Программа управления каналом связи опрашивает АП в соответствии со списком опроса. Если у опрашиваемого АП имеется сообщение для передачи, оно посылается в ЭВМ. После завершения передачи данных канал освобождается, и как только выходное сообщение от ЭВМ готово, оно занимает канал связи для передачи, т. е. выходного сообщения (от ЭВМ). Выходное сообщение имеет приоритет перед входным (от АП).

Исходные данные:

1) опрос циклический;
2) время, затрачиваемое на опрос одного АП — 100 мс;
3) интервалы времени между опросами — 10 мс;
4) передача информации происходит со скоростью 300 сотен символов/сек;
5) сообщение может содержать от 6 до 60 сотен символов;
6) интервалы времени между возникающими сообщениями от АП распределены экспоненциально со средним значением 500 мс;
7) время обработки сообщения от ЭВМ — 500 мс.

Определим вычислительные объекты:

- функция OPR имитирует список опроса: аргумент — параметр P1, при условии, что в него будет записываться номер опрашиваемого АП;
- функция NAP определяет случайным образом номер АП, на котором возникло сообщение для ЭВМ;
- функция SMV определяет число символов во входном сообщении;
- переменная TZAN определяет время занятия канала связи, равное длине сообщения, разделенное на скорость передачи (время переведено в мс):

$$30000\frac{\text{символов}}{\text{с}} = \frac{30000}{1000}\cdot\frac{\text{символов}}{\text{мс}}.$$

Программа:

```
OPR FUNCTION P1,D5
1,2/2,3/3,4/4,5/5,1
NAP FUNCTION RN1,D5
.2, 1/.4, 2/.6, 3/.8, 4/1, 5
SMV FUNCTION RN2,C2
.0,6/1.0,61
TZAN VARIABLE FN$SMV*10/3
GENERATE (EXPONENTIAL(1,0,500))   ; возникновение сообщения от АП
ASSIGN 1,FN$NAP                   ; определение номера АП
LINK P1,FIFO
; опрос абонентских пунктов
GENERATE ,,,1                     ; циклический опрос от ЭВМ
ASSIGN 1,1
POLL ASSIGN 1,FN$OPR              ; определяется очередной АП
SEIZE CAN                         ; занять канал под опрос
ADVANCE 100
TEST NE CH*1,0,PROD               ; если нет сообщения —
                                  ; продолжать опрос
UNLINK P1,XMIT,1                  ; передача сообщения в ЭВМ
PROD RELEASE CAN                  ; освободить канал от опроса
ADVANCE 10                        ; задержка между опросами
TRANSFER ,POLL                    ; продолжение опроса
; передача и обработка сообщения
XMIT SEIZE CAN                    ; передача к ЭВМ
ADVANCE V$TZAN
RELEASE CAN
ADVANCE 500                       ; обработка сообщения
PRIORITY 1
SEIZE CAN                         ; передача от ЭВМ
ADVANCE V$TZAN
RELEASE CAN
TERMINATE
; таймер
GENERATE 10000
TERMINATE 1
START 1
```

В программе при возникновении сообщения от АП в список с соответствующим номером заносится единица. Блок TEST проверяет, пуст ли список, соответствующий опрашиваемому АП. Если да — то продолжается опрос, если нет — то блок UNLINK выделяет из списка сообщение и передает управление на блок XMIT. Транзакт, вызвавший извлечение из списка сообщения, по-прежнему направляется к блоку PROD и продолжает цикл опроса. Таким образом, в модели

опроса единственный транзакт циркулирует «по кругу», имитируя циклический опрос АП.

2.4.14 Логические переключатели

Логические переключатели могут находиться в двух положениях: «включен» и «выключен». Перед началом выполнения программы все переключатели устанавливаются в положение «выключен».

Для работы с логическими переключателями используется *блок LOGIC*. При поступлении транзакта в блок состояние логического переключателя, номер или имя которого указан в поле *A*, меняется в соответствии с мнемоникой:

- LOGIC R 1 — логический переключатель с номером 1 устанавливается в состояние «выключен»;
- LOGIC S 1 — логический переключатель с номером 1 устанавливается в состояние «включен»;
- LOGIC I 1 — состояние логического переключателя с номером 1 инвертируется.

Состояние логического переключателя может быть проверено в любой части модели с помощью блока GATE или с помощью СЧА LS$<имя логического переключателя>, который принимает значение 1, если логический переключатель «включен», и 0 — в противном случае.

Пример 2.10

Паспортный стол работает с 9 до 18 часов с часовым перерывом на обед с 13 до 14 часов. Посетители приходят в среднем каждые 5 минут, причем все сначала направляются к начальнику паспортного стола, который работает с каждым посетителем в среднем 4 минуты. После начальника примерно 5% посетителей покидают отделение (получен отказ либо вопрос решен), а остальные направляются в отдел прописки, в котором работают три паспортистки. Время приема посетителя в отделе прописки равно в среднем 12 минутам (с каждым посетителем). Время прихода и время обслуживания в системе распределено экспоненциально.

Те посетители, кто стоял в очереди и не успел обслужиться до обеда, обслуживаются после перерыва в первую очередь. Будем считать, что во время обеденного перерыва никто не приходит. Примерно за полчаса до окончания рабочего дня просят не занимать очередь к начальнику паспортного стола, и подошедшие в это время посетители не обслуживаются.

Проверить, успеют ли все посетители, стоящие в очереди, обслужиться до конца рабочего дня. Протабулировать время нахождения посетителей в очередях к начальнику паспортного стола и в отдел прописки.

В переменной RAZN подсчитывается количество посетителей, которые встали в очередь, но не успели обслужиться до конца рабочего дня.

```
PROP STORAGE 3
NAB1 TABLE OCH_NACH,10,10,10
TAB2 QTABLE OCH_PROP,10,10,10
RAZN VARIABLE N$VXOD–N$UXOD
; работа начальника паспортного стола
GENERATE (EXPONENTIAL(1,0,5))
GATE LR TIME,BYE                ; если рабочий день закончился —
                                ; уход
VXOD QUEUE OCH_NACH
GATE LR OBED                    ; ожидание окончания обеда
SEIZE NACH
DEPART OCH_NACH
ADVANCE (EXPONENTIAL(1,0,4))
RELEASE NACH
TRANSFER .05,,UXOD
; работа отдела прописки
QUEUE OCH_PROP
GATE LR OBED                    ; ожидание окончания обеда
ENTER PROP
DEPART OCH_PROP
ADVANCE (EXPONENTIAL(1,0,12))
LEAVE PROP
UXOD TERMINATE
BYE TERMINATE
; таймер
GENERATE 240,,,1                ; начало рабочего дня
LOGIC S OBED                    ; начало обеда
ADVANCE 60
LOGIC R OBED                    ; окончание обеда
ADVANCE 210
LOGIC S TIME                    ; за 30 минут до конца рабочего дня
ADVANCE 30
SAVEVALUE NO_OBSL,V$RAZN        ; подсчет посетителей, которые
                                ; встали в очередь, но не успели
                                ; обслужиться до конца рабочего дня
TERMINATE 1
START 1
```

2.4.15 Синхронизация транзактов

Любые элементы в системах прямо или опосредованно связаны, взаимодействуют. Зависимость между процессами, протекающими в разных частях системы, нередко выражается в форме синхронизации, то есть в форме взаимного согласования этих процессов по времени.

Блок SPLIT предназначен для *моделирования одновременного начала нескольких процессов*. В момент входа транзакта в блок SPLIT создается несколько копий этого транзакта. Число копий задается в поле *A*. Все копии переходят в блок, определенный в поле *B*. Исходный (порождающий) транзакт переходит к блоку, следующему за SPLIT. Если поле *C* блока SPLIT пустое, то все копии идентичны породившему их транзакту. Например, при входе транзакта в блок

```
SPLIT 4,NEXT
```

порождается четыре транзакта, идентичных вошедшему, и передается в блок, в поле метки которого записано NEXT. Породивший их транзакт передается в блок, записанный после блока SPLIT. Всего из этого блока SPLIT выходит пять транзактов.

Если поле *C* непустое, то его значение интерпретируется как номер или имя параметра транзакта. Пусть N — значение этого параметра в момент входа транзакта в блок SPLIT. Тогда в момент выхода из SPLIT данный параметр у исходного транзакта будет иметь значение $N + 1$, а у копий транзактов соответственно $N + 2$, $N + 3, \ldots, N + K$, где K — общее число вышедших из блока SPLIT транзактов. Например, если транзакт, имеющий нуль в десятом параметре, войдет в блок

```
SPLIT 2,ABCD,10,
```

то параметр P10 у этого транзакта приобретет значение 1, а у копий — соответственно 2 и 3.

Транзакты — копии могут двигаться в модели независимо друг от друга. Копии могут проходить блоки SPLIT и порождать новые копии.

Множество, состоящее из исходного транзакта и всех его копий, называется семейством транзактов. Копия члена семейства является членом того же семейства. Любой транзакт — член только одного семейства.

Блок ASSEMBLE — одновременное завершение нескольких процессов. Блок собирает заданное в поле *A* число транзактов одного семейства и превращает их в один транзакт. Первый из транзактов какого-либо семейства, вошедший в блок, задерживается до тех пор, пока в этом блоке не накопится заданное число транзактов того же семейства. После этого первый транзакт выходит из блока ASSEMBLE, а остальные транзакты этого семейства уничтожаются.

В одном блоке ASSEMBLE могут одновременно проходить сборку транзакты, принадлежащие к разным семействам. Например, если в блок

```
ASSEMBLE 4
```

поступают транзакты разных семейств, то транзакты каждого семейства собираются по четыре и каждая четверка превращается в один транзакт.

Блок GATHER работает аналогично блоку ASSEMBLE с тем отличием, что транзакты, попав в блок GATHER, не уничтожаются, а только задерживаются и после того, как в блоке накапливается заданное число транзактов, они *все* переходят к следующему блоку.

Блок MATCH предназначен для *синхронизации процессов*. В программе всегда должно быть два поименованных блока MATCH. В поле *A* блока указывается имя парного блока MATCH. Когда транзакт попадает в блок, определяется второй блок MATCH и проверяется, находится ли в нем транзакт этого же семейства. Если да, то транзакты выходят из обоих блоков одновременно. Если в парном блоке нет транзакта, то в первом блоке транзакт задерживается до тех пор, пока во второй блок не поступит транзакт этого же семейства. Таким образом, использование блоков MATCH позволяет синхронизировать передвижение транзактов по модели.

Пример 2.11

Промоделировать сборку изделий рабочими A, B и C. Изделия в разобранном виде поступают каждые 300±100 мин. Каждое из них разделяется между рабочими A и B, которые параллельно готовят свою часть изделия к сборке. Подготовка состоит из двух фаз, причем после первой фазы производится сверка с одновременным участием обоих рабочих, а затем A и B независимо выполняют вторую фазу работы. На первой фазе рабочий A тратит на работу 100±20 мин, рабочий B — 80±20 мин; на второй фазе рабочий A тратит 50±5 минут, рабочий B — 80±20 минут. После окончания работы рабочими A и B рабочий C выполняет сборку изделия за 50±5 мин, причем он может начинать сборку только тогда, когда оба первых рабочих закончат свою работу.

Модель сборки изделий:

```
GENERATE 300,100      ; поступление изделий
SPLIT 1,MANB          ; разделение изделий
SEIZE RABA            ; занять рабочего A
ADVANCE 100,20        ; 1-я фаза
FAZ1A MATCH FAZ1B     ; ждать, если B не закончил 1-ю фазу
ADVANCE 50,5          ; 2-я фаза
RELEASE RABA
TRANSFER ,MANC
MANB SEIZE RABB       ; занять рабочего B
ADVANCE 80,20
FAZ1B MATCH FAZ1A     ; ждать, если A не закончил 1-ю фазу
ADVANCE 80,20
RELEASE RABB
MANC ASSEMBLE 2       ; ждать обе части изделия
SEIZE RABC            ; занять рабочего C
ADVANCE 50, 5
RELEASE RABC
TERMINATE 1           ; завершение сборки
START 1000
```

2.4.16 Прерывание работы устройства

Блок PREEMPT — захватить устройство. Транзакт, попадающий в блок PREEMPT, захватывает устройство, имя которого указано в поле *A* блока. Если при захвате устройства оно свободно, то транзакт просто занимает устройство, в этом случае блок PREEMPT работает аналогично блоку SEIZE. Если при входе транзакта в блок PREEMPT устройство занято другим транзактом, то в этом случае транзакт входит в блок PREEMPT, а устройство прерывает обслуживание занимающего его транзакта и переключается на обслуживание транзакта, вошедшего в блок PREEMPT. При этом из состояния «занято» устройство переходит в состояние «захвачено».

Когда транзакт, захватывающий устройство, освободит его, устройство возобновит прерванное обслуживание другого транзакта и перейдет в состояние «занято».

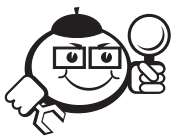
Если прерываемый транзакт в момент прерывания находится в блоке ADVANCE, то, начиная с момента прерывания, отсчет времени пребывания транзакта в этом блоке прекращается до тех пор, пока не будет восстановлено обслуживание транзакта. Таким образом, в момент восстановления прерванного обслуживания транзакта время, оставшееся этому транзакту до выхода из блока ADVANCE, такое же, каким оно считалось непосредственно в момент прерывания. Такое прерывание обслуживания называется прерыванием с последующим дообслуживанием.

Все транзакты, задержанные при обращении к устройству, упорядочиваются по приоритету. Кроме поля *A*, в блоке PREEMPT могут быть заданы операнды *B*, *C*, *D* и *E*. Операнд *B* записывается в виде обозначения PR, задающего приоритетный режим работы блока. В этом режиме транзакт захватывает устройство, если оно свободно или обслуживает менее приоритетный транзакт. Прерывание обслуживания менее приоритетного транзакта происходит с последующим дообслуживанием.

Для определения последующего движения прерванных транзактов используются другие операнды. В поле *C* может быть указана метка какого-либо блока, на который будет передан прерванный транзакт. При этом прерванный транзакт продолжает претендовать на данное устройство. В поле *D* блока может быть задан номер параметра транзакта. Тогда, если прерванный транзакт находится в блоке ADVANCE, то вычисляется остаток времени обслуживания (время дообслуживания) и полученное значение помещается в параметр, заданный в поле *D*. Прерванный транзакт при этом будет послан в блок, указанный в поле *C*. Прерванный транзакт продолжает претендовать на данное устройство. Если в поле *E* блока записано обозначение RE, то прерванный транзакт больше не будет претендовать на данное устройство.

Необходимо помнить, что если поле *E* не задано, а поле *C* указано, то прерванный транзакт не может быть уничтожен до тех пор, пока он явно не освободит устройство (он должен пройти либо блок RELEASE, либо блок RETURN). Чтобы не забывать явно освободить устройство, обычно поля *C* и *E* применяют одновременно.

Блок RETURN — освободить устройство. Этот блок используется в паре с блоком PREEMPT. Если транзакт захватил устройство посредством блока PREEMPT, то освободить его он может только в блоке RETURN. Имя освобождаемого устройства задается в поле *A* блока.



Пример 2.12

Детали поступают в цех обработки в среднем каждые 5±2 минуты. Мастер обрабатывает детали в среднем 4±1 минуту. Каждые 30±5 минут приходит срочный заказ на обработку деталей второго типа, для обработки которых мастеру требуется еще 10±3 минуты. Детали второго типа имеют безусловный приоритет. Если

в момент прихода детали второго типа мастер обрабатывает деталь первого типа, то он прерывает свою работу, текущую деталь передает для доделки своему ученику, а сам принимается за обработку вновь поступившей детали второго типа. Ученику требуется в два раза больше времени на обработку (или доработку) детали. Если у мастера скапливается очередь из деталей больше двух, то вновь приходящие детали первого типа также передаются на обработку ученику. Детали второго типа обрабатываются только мастером. Если в момент прерывания обработки детали первого типа мастеру не хватило меньше 30 секунд, то считается, что деталь обработана.

Промоделировать работу мастера и ученика в течение 4-х часов. Проверить, не будет ли скапливаться очередь у ученика. Определить загрузку мастера и ученика.

Для моделирования процесса обработки деталей учеником введем переменную VAR (чтобы увеличить время обработки в два раза).

Функция OBSLU используется для определения времени обработки деталей первого типа. Данная функция описывает равномерный закон распределения на интервале от 3 до 5.

```
VAR VARIABLE P1#2
OBSLU FUNCTION RN1,C2
0,3/1,5
; работа мастера с деталями первого типа
GENERATE 5,2                        ; поступление деталей
ASSIGN 1,FN$OBSLU                   ; определение времени обработки
TEST LE Q$MAS,2,UCH                 ; если скопилась очередь – деталь
                                    ; направляется к ученику
QUEUE MAS
SEIZE MAST
DEPART MAS
ADVANCE P1
RELEASE MAST
TERMINATE
; работа мастера с деталями второго типа
GENERATE 30,5,,,1
QUEUE MAS
PREEMPT MAST,PR,UCH,1,RE            ; прерванная деталь 1-го типа
                                    ; направляется к ученику
DEPART MAS
ADVANCE 10,3
RETURN MAST
TERMINATE
; работа ученика
UCH TEST G P1,0.5,UXOD              ; если время дообработки меньше
                                    ; 30 секунд, то деталь
                                    ; не обрабатывается
SEIZE UCHEN
ADVANCE V$VAR
RELEASE UCHEN
```

```
UXOD TERMINATE
; таймер
GENERATE 240
TERMINATE 1
START 1
```

. . . . . .

2.4.17 Организация циклов

Для организации циклов используется *блок LOOP*. Поле *A* этого блока содержит имя или номер параметра, который выполняет функцию счетчика циклов. Каждый раз при поступлении транзакта в блок LOOP из указанного параметра вычитается единица, и полученная разность снова записывается в данный параметр. Как только значение параметра становится равным нулю, транзакт направляется в блок, следующий за блоком LOOP. Если значение параметра остается положительным, то транзакт направляется к блоку, указанному в поле *B*.

Например, если необходимо, чтобы через блок ADVANCE все транзакты проходили по 10 раз, то этот процесс можно промоделировать следующим образом.

```
ASSIGN 1,10
POVT ADVANCE 34,12
LOOP 1,POVT
```

2.4.18 Системное время

СЧА, связанные с системным временем, используются для проверки временных соотношений пребывания транзактов в системе.

В СЧА C1 и AC1 хранится текущее значение системного времени. СЧА C1 содержит значение относительного системного времени (с момента последнего блока RESET). СЧА AC1 содержит значение абсолютного системного времени (с момента последнего блока CLEAR). Данные СЧА доступны пользователю в любой точке программы.

Каждый транзакт при генерации снабжается отметкой времени. Время пребывания транзакта в модели содержится в СЧА M1 или MP и отсчитывается от момента рождения:

M1 = AC1 – <дата рождения>.

СЧА M1 возвращает время пребывания транзакта в модели, СЧА MPi или MP$<имя параметра> возвращает значение, равное абсолютному системному времени минус значение соответствующего параметра транзакта.

«Дату рождения», зафиксированную блоком GENERATE, можно изменить в любом месте программы, используя *блок MARK* с пустым полем *A*. Блок MARK с пустым полем *A* изменяет «дату рождения» на текущее системное время. Если в блоке MARK используется поле *A*, то в этом поле содержится номер или имя параметра транзакта. В этом случае транзакт сохраняет «дату рождения», а в указанном параметре записывается текущее значение AC1. Работа блока

```
MARK 10
```

эквивалентна работе блока

```
ASSIGN 10,C1
```

Например:

```
. . .
MARK
. . .
MARK 10
. . .
TEST E C1,50,BGN1
SR1 TEST GE M1,MP10,BGN2
```

Здесь первый блок TEST проверяет условие, связанное с текущим значением системного времени (C1 = 50 ?). Второй блок TEST сравнивает M1 (время от попадания транзакта в блок MARK до попадания его в блок SR1) с MP10 (время от попадания транзакта в блок MARK 10 до попадания его в блок SR1).

Пример 2.13

В специализированный магазин промышленных товаров покупатели приходят в среднем каждые 15 минут. В магазине работает единственный продавец, который обслуживает покупателей в среднем 8 минут. Время прихода и время обслуживания подчиняется экспоненциальному закону. Известно, что примерно 20% покупателей в течение месяца приходят в магазин повторно.

В магазине действует накопительная система: если в течение месяца покупатель набирает товара на сумму, превышающую 800 рублей (по чекам), то он получает накопительную карту, по которой ему предоставляются скидки на покупки. Причем процент скидок зависит от общей накопленной на карте суммы. Кроме того, в конце месяца, покупателю, набравшему товара на максимальную сумму, в качестве премии начисляется на карту дополнительная сумма, равная сумме на карте, что позволяет ему в следующем месяце сразу получать максимальную скидку на стоимость товара.

Промоделировать работу в течение месяца и определить размер премии «лучшего» покупателя.

Для моделирования суммы покупок определим случайную функцию CONT, считая, что стоимость покупки может варьироваться в размере от 5 до 500 рублей.

Модель работы магазина будет выглядеть следующим образом:

```
CONT FUNCTION RN1,C2
0,5/1,500
; приход посетителей в магазин
GENERATE (EXPONENTIAL(1,0,15)),,,1
VXOD ASSIGN PRICE+,FN$CONT          ; общая сумма покупок
                                    ; посетителя магазина
QUEUE OCH
SEIZE PROD
DEPART OCH
ADVANCE (EXPONENTIAL(1,0,8))
TEST G P$PRICE,800,NEXT             ; если общая сумма превышает 800,
```

```
JOIN CART     ; посетителю выдается карта
RELEASE PROD
TRANSFER .2,POVT
TEST G C1,14400    ; задерживаем транзакты группы, чтобы в конце месяца
                   ; определить лучшего покупателя
TERMINATE
NEXT RELEASE PROD
TRANSFER .2,,POVT
TERMINATE
; перед повторным посещением ставим задержку
POVT ADVANCE (EXPONENTIAL(1,0,600))
TRANSFER ,VXOD
; определение лучшего покупателя в конце месяца
GENERATE 14400
SCAN MAX CART,PRICE,,PRICE,SUM
ASSIGN SUM+,P$SUM
SAVEVALUE 20,P$SUM
TERMINATE 1
START 1
```

. . . . . .

2.4.19 Управляющие блоки

Блок START воспринимается как команда симулятору начать выполнение прочитанной части модели. В этом блоке в поле *A* задается начальное значение счетчика транзактов. Здесь также может быть использовано поле *B* в значении NP, что означает — не выводить статистику по окончании моделирования. Если задан блок

```
START 1,NP,
```

то подавляется вывод стандартного отчета — всей информации об устройствах, памятях, очередях, таблицах и ячейках.

Содержимое счетчика транзактов уменьшается при входе транзактов в блок TERMINATE. Когда значение счетчика становится равным нулю или отрицательным, производится выдача статистики и заканчивается процесс моделирования.

Блок RESET предназначается для стирания в заданный момент времени статистики о предыстории процесса. Достигнутое состояние объектов при этом сохраняется.

Применение блока RESET позволяет уменьшить затраты машинного времени на сбор статистики о стационарном (в смысле вероятностных характеристик) процессе в тех случаях, когда предшествующий ему переходный процесс вносит заметные искажения в накапливаемую статистику.

Обычно блок RESET помещается в модели после блока START, а после RESET располагается следующий блок START. После прочтения блока RESET засылается нулевое содержимое в счетчики числа входов в блоки, коэффициенты использования устройств и памятей, а также обнуляются все накопленные статистики. При этом сохраняются текущие состояния и значения устройств, памятей, очередей, ячеек и датчиков случайных чисел.

Блок CLEAR переводит всю модель — всю статистику и все объекты — в исходное состояние. Исключением является лишь датчик случайных чисел — он не возвращается к начальному значению. Применение блока CLEAR позволяет осуществить независимые реализации моделируемого случайного процесса.

Пример 2.14

Моделирование работы заправочной станции

Заправочная станция открыта с 7 часов до 19 часов. Машины, поступившие после 19 часов, не обслуживаются. Тем не менее все машины, попавшие в очередь до 19 часов, должны быть обслужены. Машины останавливаются на обслуживание лишь в том случае, если число ожидающих обслуживания автомашин меньше или равно числу обслуживаемых машин (т. е. не более одной машины в очереди на колонку). Провести моделирование для 1, 2 и 3 колонок в течение дня с целью определения такого их числа, при котором достигается максимальная прибыль.

С одной обслуженной машины получают доход 10 рублей. Расходы на содержание одной колонки составляют 700 рублей. Предусмотреть в модели, что в случае возникновения временного узла между событиями завершения обслуживания машины и прибытием другой машины завершение обслуживания было бы обработано в первую очередь: это эквивалентно предположению о том, что водитель прибывшей машины видит возможность завершения обслуживания, т. е. может предпочесть остаться. Если есть временной узел между событиями поступления автомашины на станцию и закрытием заправочной станции в конце дня, прибывшая машина должна попасть до закрытия.

Пусть прибытие машин описывается простейшим потоком со средним временем между приходом машин, равным 1 минуте. Время обслуживания равно 1±0.5 минут.

```
STR STORAGE 1      ; работает 1 колонка
PRB VARIABLE N$DONE-(R$STR+S$STR)#70
GENERATE (EXPONENTIAL(1,0,1)),,,,1      ; приход автомашины
GATE LR LOCK       ; если рабочий день еще не закончен,
                   ; то машина поступает на обслуживание
TEST LE Q$LIN,S$STR,BYE
BEG QUEUE LIN
ENTER STR
DEPART LIN
PRIORITY 2
ADVANCE 1,0.5
DONE LEAVE STR
BYE TERMINATE
GENERATE 720
LOGIC S LOCK                ; рабочий день закончился,
TEST E N$BEG,N$DONE         ; если все машины обслужены
SAVEVALUE 1,V$PRB           ; подсчет прибыли
```

```
TERMINATE 1
START 1
CLEAR
STR STORAGE 2      ; работают 2 колонки
START 1
CLEAR
STR STORAGE 3      ; работают 3 колонки
START 1
```

Переменная PRB используется для расчета прибыли заправочной станции. Временные узлы между событиями обрабатываются через приоритеты.

Примечание: в результате прогона модели будут получены три стандартных отчета — по одному на каждый блок START. Каждый отчет размещается в отдельном файле выдачи.

2.5 Внутренняя организация GPSS

Система GPSS в целом как программный продукт состоит из ряда модулей, из которых только модуль управления (симулятор) находится постоянно в ОЗУ и осуществляет процесс имитации. Динамика функционирования симулятора основана фактически на схеме событий, при этом событием считается любое изменение состояния моделируемой системы. Основной функцией симулятора является поддержание правильного хода часов системного времени и выяснение возможностей продвижения транзактов в программе модели. Симулятор оперирует с рядом информационных структур, основными из которых являются: список будущих событий (FEC), список текущих событий (CEC), список прерываний, список задержанных транзактов и другие списки.

Работа симулятора разделяется на три основные фазы:

1) изменение значения системного времени;
2) просмотр списка текущих событий;
3) движение сообщений.

Фаза «Изменение значения системного времени» (рис. 2.4) выполняется симулятором всегда, когда на текущий момент системного времени ни одно из активных сообщений, находящихся в CEC, не может быть продвинуто в программе модели и, кроме того, состояние системы не может быть изменено.

Выбирая первое сообщение, симулятор присваивает системному времени STIME время очередной передвижки этого сообщения TEV(H) в программе модели и перемещает его в CEC. Подобная процедура осуществляется для всех событий в FEC, время наступления которых равно TEV(H), т. е. текущему значению системного времени. При этом после просмотра в FEC останутся события, время наступления которых больше STIME, т. е. события, наступающие в будущем.

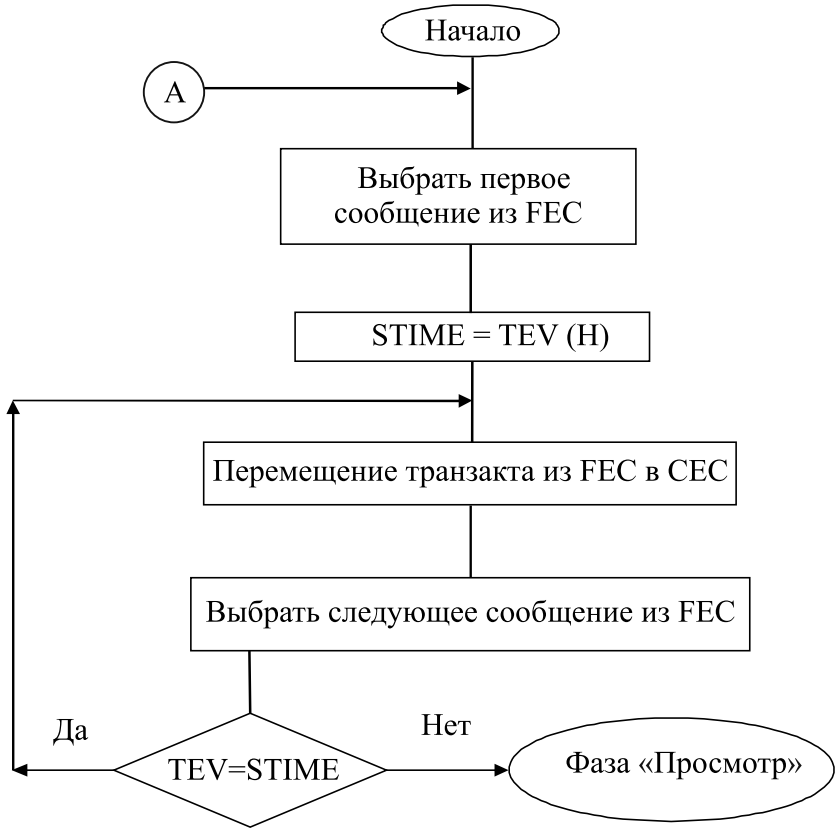


Рис. 2.4 – Фаза «Изменение значения системного времени»

Фаза «Просмотр списка текущих событий» (рис. 2.5). Установив флаг изменения состояния системы в ноль, симулятор в зависимости от значения индикатора просмотра сообщения (транзакта) — 0 или 1 — решает вопрос: передать сообщение на третью фазу или нет. На *фазу «Движение сообщений»* (рис. 2.6) передаются только активные сообщения, индикатор просмотра которых равен нулю. Пассивные сообщения находятся в состоянии задержки, например по причине занятости имитируемого оборудования. Такие сообщения не попадут на третью фазу до тех пор, пока соответствующее оборудование не будет освобождено, т. е. пока не изменится состояние системы.

На *фазе «Движение сообщений»* активные сообщения симулятор пытается продвинуть как можно дальше по программе модели. Если при этой передвижке меняется состояние системы, все пассивные сообщения, находящиеся в CEC и задержанные по той или иной причине, получают статус активных. Их индикаторы просмотра устанавливаются в «0». Если же при передвижке сообщений явно задана задержка, то сообщение перемещается в FEC. Таким образом, на третьей фазе происходит передвижка активных сообщений, изменение состояния системы, пересмотр индикаторов сообщений и планирование будущих событий (перемещение в FEC).

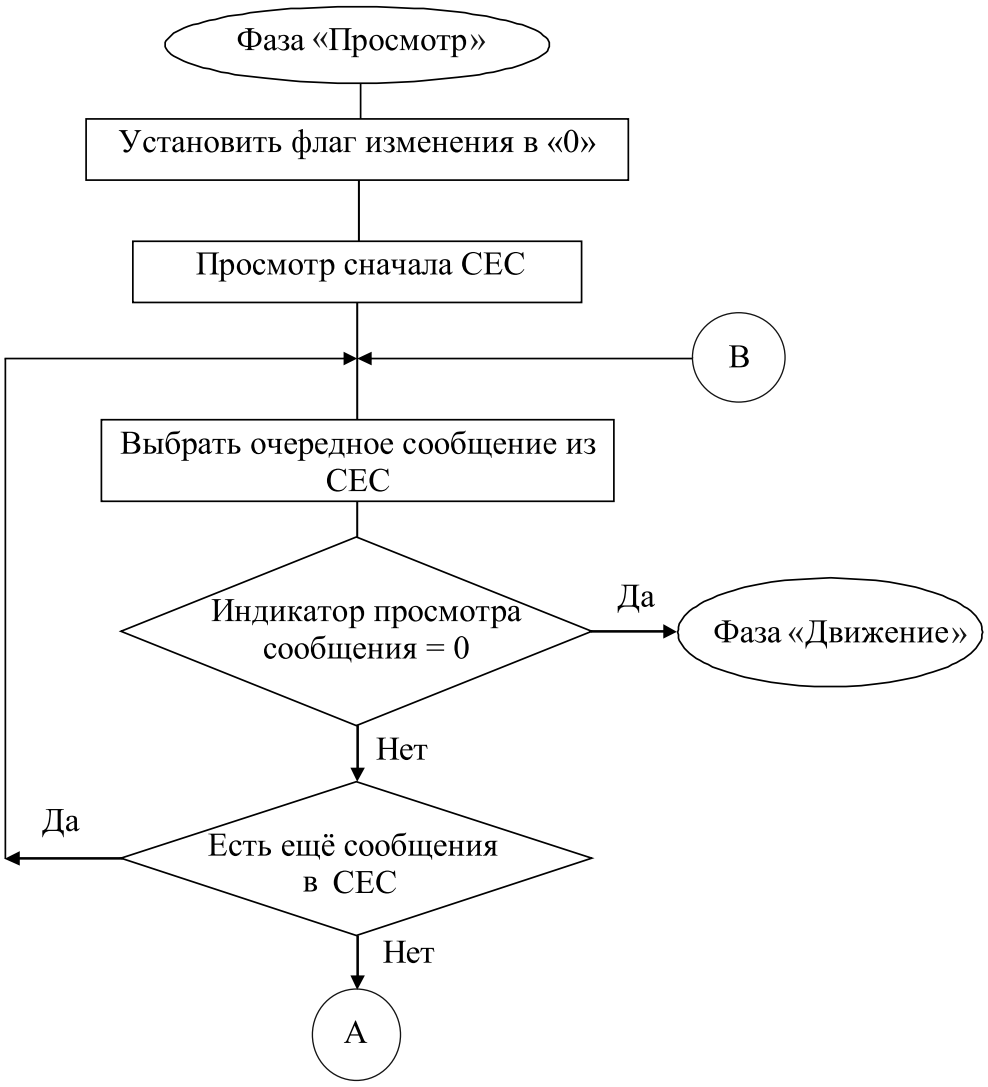


Рис. 2.5 – Фаза «Просмотр списка текущих событий»

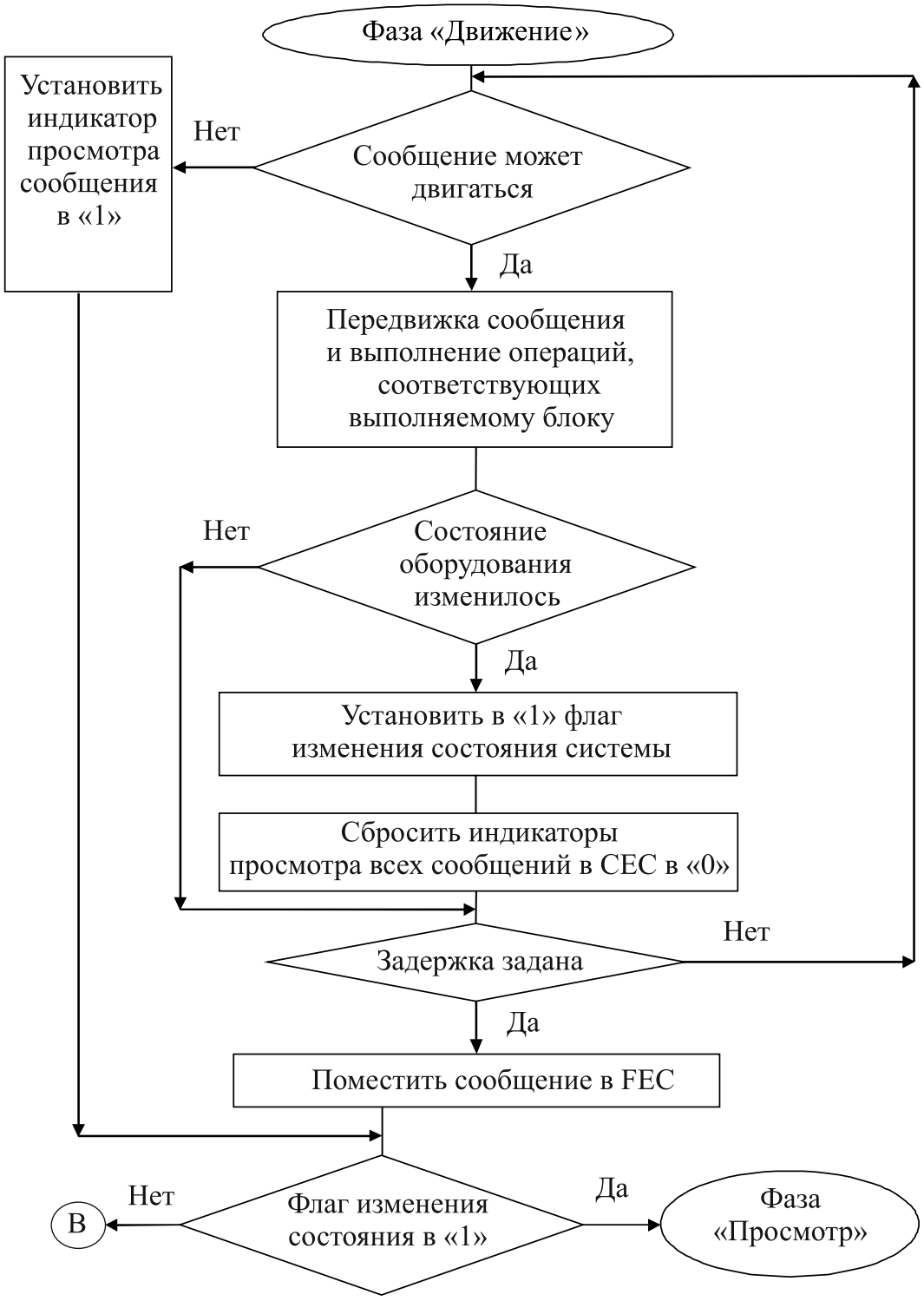


Рис. 2.6 – Фаза «Движение сообщений»

Контрольные вопросы по главе 2

1. Что такое транзакт?
2. Чем отличается работа устройства и работа памяти?
3. Перед какими блоками образуются физические очереди транзактов?
4. Для чего служат блоки работы с очередями?
5. С помощью каких объектов языка можно организовать счетчики транзактов?
6. На что влияет приоритет транзакта?
7. Как можно организовать обслуживание транзактов по приоритету?
8. С помощью каких блоков осуществляется синхронизация транзактов?

Глава 3

ПЛАНИРОВАНИЕ МАШИННЫХ ЭКСПЕРИМЕНТОВ

3.1 Методы планирования экспериментов

Машинный эксперимент с моделью проводится с целью получения информации о характеристиках процесса функционирования исследуемого объекта. Эта информация может быть необходима как для анализа характеристик, так и для их оптимизации при заданных ограничениях, т. е. для синтеза структуры, алгоритмов и параметров моделируемой системы.

В связи с этим основной задачей планирования машинных экспериментов является получение необходимой информации об исследуемой системе при ограничениях на ресурсы (затраты машинного времени, памяти и т. п.). Отсюда возникают частные задачи: уменьшение затрат машинного времени на моделирование; увеличение точности и достоверности результатов моделирования и т. д.

Эффективность машинных экспериментов с моделями существенно зависит от выбора плана эксперимента, т. к. именно план определяет объем и порядок проведения вычислений, способы накопления и статистической обработки результатов моделирования. Как обработать априорную информацию, сколько и каких опытов провести, как обработать результаты — вопросы планирования эксперимента, при этом одновременно решается и задача минимизации машинных ресурсов (минимальное количество опытов) на реализацию процесса моделирования.

К настоящему времени в ряде естественных наук сложилась теория планирования экспериментов, в которой разработаны достаточно мощные методы. Перечислим преимущества и недостатки машинных экспериментов, которые имеют наиболее важное значение при планировании экспериментов.

Преимущества машинных экспериментов:

1) простота повторения условий эксперимента на ЭВМ с моделью: возможность сравнивать две альтернативы при одинаковых повторных условиях (использование одинаковых последовательностей случайных чисел);

2) возможность управления экспериментом с моделью, включая его прерывание и возобновление, что может быть нереализуемым с реальным объектом; можно прервать эксперимент для анализа результатов и решения о дальнейшем ходе исследований;
3) легкость варьирования условий проведения эксперимента (воздействий внешней среды).

Недостатки машинных экспериментов:

1) наличие корреляции между последовательностью точек в процессе моделирования, т. е. результаты одних наблюдений зависят от результатов одного или нескольких предыдущих, и поэтому в них содержится меньше информации, чем в независимых наблюдениях;
2) трудности, связанные с определением интервала моделирования (0, T) — от начального периода до момента стабилизации работы.

Тип эксперимента выбирается в зависимости от конкретных целей эксперимента для анализа его результатов. Наиболее распространены следующие *типы экспериментов*:

1) сравнение средних и дисперсий различных альтернатив (обычно однофакторные эксперименты);
2) определение важности учета или значимости влияния переменных и ограничений, наложенных на эти переменные (основные методы истолкования результатов этих экспериментов — это дисперсионный и регрессионный анализы);
3) отыскание оптимальных значений на некотором множестве возможных значений переменных (обычно предполагается использование последовательных или поисковых методов построения экспериментов).

Рассмотрим основные понятия теории планирования эксперимента. Исследуемый объект (над которым проводится эксперимент) будем представлять в виде модели «черного ящика» с входами x_i и выходами y_i. Цель эксперимента — изучение влияния переменных x_i на y_i. Входы x_i называются *факторами* (независимые, экзогенные переменные); выходы y_i — *реакцией/откликом* (параметром оптимизации, целевой функцией, эндогенными переменными).

Фактор может принимать одно из нескольких значений, которые называются *уровнями факторов*. Фиксированный набор уровней факторов определяет одно из возможных состояний системы и представляет собой условия проведения одного из возможных *опытов* одного эксперимента — точки в факторном пространстве. Факторное пространство — это координатное пространство, на осях которого откладывают значения исследуемых факторов. Если перебрать все возможные наборы состояний системы, то мы получим полное множество состояний — число возможных опытов.

Математическая модель объекта — это функциональная зависимость, связывающая отклик y с факторами x_i:

$$y = \varphi(x_1, x_2, \ldots, x_k). \tag{3.1}$$

Данная зависимость называется функцией отклика, а ее геометрический образ — поверхностью отклика.

При проведении эксперимента с моделью для оценки некоторых характеристик экспериментатор стремится создать такие условия, которые способствуют *выявлению влияния факторов* на исследуемую характеристику. Для этого необходимо [2]:

1) отобрать факторы $\{x_i\}$, влияющие на искомую характеристику;
2) описать функциональную зависимость $y = \varphi(x_1, x_2, \ldots, x_k)$;
3) установить диапазон изменения $x_{i\,\min} \div x_{i\,\max}$;
4) определить координаты точек факторного пространства $\{x_1, x_2, \ldots\}$, в которых следует проводить эксперимент;
5) оценить необходимое число реализаций и их порядок в эксперименте.

В общем случае, когда исследование модели ведется при неполном знании механизма изучаемых явлений, аналитическое выражение функции (3.1) неизвестно. Наибольшее в этом случае применение нашли модели в виде полиномов

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i<j} \beta_{ij} x_i x_j + \sum_{i=1}^{k} \beta_{ij} x_i^2 + \ldots$$

с теоретическими коэффициентами регрессии β_0, β_i, β_{ij}, β_{ii}, …. Функция отклика может иметь и более сложную зависимость от факторов. Некоторые из них удается привести к линейному виду. Такими моделями являются мультипликативная регрессионная, экспоненциальная и др. Если выбрана модель планирования, т. е. выбран вид функции (3.1) и записано уравнение, то остается спланировать и провести эксперимент для оценки числовых значений коэффициентов этого уравнения.

План эксперимента, позволяющий вычислить коэффициенты линейного уравнения регрессии, называют *планом первого порядка*. План эксперимента, позволяющий вычислить коэффициенты полного уравнения регрессии k-й степени, называется *планом k-го порядка*.

Одна из основных целей в теории эксперимента — это *оптимальное использование факторного пространства*. Проиллюстрируем идею на простом примере — задаче о взвешивании трех объектов A, B, C. Традиционно эксперимент проводится по схеме, приведенной в таблице 3.1 [7, 8].

Таблица 3.1 – План однофакторного эксперимента

№ опыта	***A***	***B***	***C***	**Результат взвешивания**
1	−1	−1	−1	y_0
2	+1	−1	−1	y_1
3	−1	+1	−1	y_2
4	−1	−1	+1	y_3

где «+1» — объект положен на весы; «−1» — объект отсутствует на весах.

Сначала проводится «холостое» взвешивание — определяется нулевая точка весов, затем по очереди взвешиваются все объекты. Это пример *однофакторного* эксперимента — здесь изучается поведение каждого фактора в отдельности. Вес определяется по результатам двух опытов: $A = y_1 - y_0$; $B = y_2 - y_0$; $C = y_3 - y_0$.

Дисперсия результатов взвешивания:

$$\sigma^2\{A\} = \sigma^2\{y_1 - y_0\} = 2\sigma^2\{y\},$$

где $\sigma^2\{y\}$ — дисперсия ошибки взвешивания.

В таблице 3.2 рассмотрена другая схема проведения эксперимента.

Таблица 3.2 – План многофакторного эксперимента

№ опыта	*A*	*B*	*C*	Результат взвешивания
1	+1	−1	−1	y_1
2	−1	+1	−1	y_2
3	−1	−1	+1	y_3
4	+1	+1	+1	y_4

Здесь веса объектов определяются по результатам всех четырех опытов:

$$A = \frac{y_1 - y_2 - y_3 + y_4}{2}; \quad B = \frac{-y_1 + y_2 - y_3 + y_4}{2}; \quad C = \frac{-y_1 - y_2 + y_3 + y_4}{2}.$$

Вес объекта A не искажается весом объектов B и C, т. к. вес каждого из них входит в формулу дважды и с разными знаками.

Дисперсия результатов взвешивания

$$\sigma^2\{A\} = \sigma^2\left\{\frac{y_1 - y_2 - y_3 + y_4}{2}\right\} = \frac{4\sigma^2\{y\}}{4} = \sigma^2\{y\}.$$

При новой схеме взвешивания дисперсия вдвое меньше, хотя в каждом случае выполнялось по четыре опыта.

За счет чего увеличилась точность? В первом случае вес определялся по результатам двух опытов, во втором случае — по результатам всех четырех.

Вторая схема эксперимента — *многофакторная*: здесь оперируют всеми факторами так, чтобы каждый вес вычислять по результатам всех опытов, проведенных в данной серии экспериментов.

3.2 Планы первого порядка

Построение плана первого порядка начинается с выбора области эксперимента, выбора основного уровня и выбора интервалов изменения факторов.

При выборе *области эксперимента* необходимо оценить границы областей определения факторов. При этом учитываются ограничения трех типов:

1) принципиальные ограничения (невозможные значения, например температура — абсолютный нуль);
2) технико-экономические (стоимость сырья, дефицит компонентов, время);
3) конкретные условия (определяются аппаратурой в наличии, технологией и т. п.).

Выбор основного уровня осуществляется в зависимости от поставленной задачи. Если определяется оптимальный режим, то в качестве основного уровня выбираются наилучшие условия из априорной информации. Построение плана эксперимента сводится к выбору экспериментальных точек, симметричных относительно основного (нулевого) уровня. При определении функциональной зависимости между откликом и факторами можно выбирать в качестве начального уровня центр области планирования эксперимента.

Выбор интервалов изменения: для получения линейной модели достаточно каждый фактор варьировать на двух уровнях. Интервал варьирования — это расстояние на координатной оси между основным и верхним/нижним уровнем.

Для упрощения записи условий эксперимента и обработки экспериментальных данных масштабы по осям выбираются так, чтобы верхний уровень соответствовал +1, нижний — –1, а основной — нулю. Для факторов с непрерывной областью определения масштабирование производится по следующей формуле:

$$x_i = \frac{\widetilde{x}_i - \widetilde{x}_{i0}}{I_i},$$

где x_i — кодированное значение фактора; $\widetilde{x}_i$ — натуральное значение фактора; $\widetilde{x}_{i0}$ — натуральное значение основного уровня; I_i — интервал изменения фактора $I_i = \frac{x_{\max i} - x_{\min i}}{2}$; i — номер фактора.

Геометрически масштабирование факторов означает перенос начала координат факторного пространства в точку, соответствующую основному уровню, и изменение масштаба осей факторов так, что единице измерения в новых координатах соответствует величина интервала варьирования (рис. 3.1).

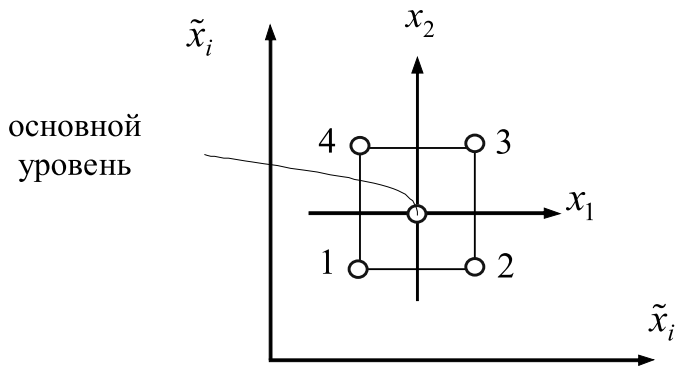


Рис. 3.1 – Геометрическое представление перехода к кодированным переменным при двух факторах

На интервал изменения накладываются ограничения: он не может быть меньше ошибки, с которой фиксируется уровень фактора, и не может быть настолько большим, чтобы уровни выходили за область определения. В остальном выбор интервала варьирования, как и основного уровня, основан на интуитивных решениях.

Полный факторный эксперимент (ПФЭ)

Полным факторным экспериментом *называется план эксперимента, в котором рассматриваются все сочетания уровней всех факторов.*

Если обозначить число факторов — k, а число уровней каждого фактора — m, то мы будем иметь полный факторный эксперимент типа m^k. Для планов первого порядка, в которых достаточно варьировать каждый фактор по двум уровням, полный факторный эксперимент потребует 2^k опытов.

Рассмотрим эксперимент типа 2^k для двух факторов ($k = 2$). Условия эксперимента можно записать в виде таблицы-матрицы планирования эксперимента 2^2, где строки соответствуют опытам, а столбцы — значениям факторов (табл. 3.3). Здесь x_0 — фиктивная переменная (вводится для удобств математических выкладок при оценке свободного члена b_0).

Таблица 3.3 – План эксперимента 2^2

№ опыта	x_0	x_1	x_2	y
1	+1	−1	−1	y_1
2	+1	+1	−1	y_2
3	+1	−1	+1	y_3
4	+1	+1	+1	y_4

Такая запись матрицы планирования весьма громоздка. Можно для удобства вводить условные буквенные обозначения факторов: $x_1 - a$, $x_2 - b$.... Для каждой строки матрицы планирования выписываются буквы только для факторов, находящихся на верхних уровнях (+1). Опыт, где все факторы равны −1, обозначим (I). Тогда наша матрица будет иметь вид (I); a; b; ab.

С ростом числа факторов возникает необходимость в некотором приеме построения матриц, для того чтобы случайно не продублировать какой-либо опыт и не пропустить другой. Рассмотрим два варианта построения плана эксперимента большой размерности [7].

Первый прием заключается в переходе от плана меньшей размерности к большей. В случае, если мы записываем план эксперимента в виде условных буквенных обозначений факторов, можно записать исходный план для одного уровня нового фактора, а затем повторить его для другого уровня: (I); a; b; ab; c; ac; bc; abc.

Второй прием основан на правиле чередования знаков. В первом столбце знаки меняются поочередно, во втором — через два, в третьем — через четыре, в четвертом — через восемь и т. д. по степени двойки. Пример использования такого приема для построения плана ПФЭ для трех факторов приведен в табл. 3.4.

Таблица 3.4 – План эксперимента 2^3

№ опыта	x_0	x_1	x_2	x_3	y
1	+	–	–	–	y_1
2	+	+	–	–	y_2
3	+	–	+	–	y_3
4	+	+	+	–	y_4
5	+	–	–	+	y_5
6	+	+	–	+	y_6
7	+	–	+	+	y_7
8	+	+	+	+	y_8

Здесь и далее вместо «+1» будем проставлять в плане просто «+», а вместо «–1» — просто «–».

После проведения эксперимента значения коэффициентов регрессионного уравнения первого порядка рассчитываются по следующим формулам:

$$\forall i = \overline{0, k}, \quad b_i = \sum_j x_{ji} \cdot \bar{y}_j / N,$$

где i — номер фактора, j — номер опыта, k — количество факторов, N — количество опытов в эксперименте, x_{ji} — отмасштабированное значение i-го фактора в j-м опыте, $\bar{y}_j$ — среднее значение отклика в j-м опыте (по количеству повторений эксперимента).

Рассмотрим основные *свойства ПФЭ типа* 2^k:

а) *симметричность* относительно центра эксперимента: алгебраическая сумма элементов вектор-столбца каждого фактора равна 0: $\sum_{j=1}^{N} x_{ij} = 0$;

б) *нормированность*: сумма квадратов элементов каждого столбца равна числу опытов: $\sum_{j=1}^{N} x_{ij}^2 = N$ (т. к. значения факторов по модулю равны единице);

в) *ортогональность*: сумма почленных произведений любых двух вектор-столбцов равна нулю: $\sum_{j=1}^{N} x_{ji} x_{jm} = 0$, $i \neq m$. Данное свойство влечет независимость соответствующих коэффициентов регрессии;

г) *ротатабельность*: точки в матрице планирования подбираются так, что точность предсказания значений параметра оптимизации (отклика) одинакова на равных расстояниях от центра эксперимента и не зависит от направления. Из данного свойства следует равномерность распределения дисперсии предсказанных значений параметра (отклика).

Заметим, что свойства ортогональности и ротатабельности одновременно выполняются только для планов первого порядка. Для планов более высокого порядка добиться одновременного выполнения этих двух свойств не удается. Обычно исследователь сам выбирает, какое из двух свойств ему более важно для исследований, и, исходя из этого, делает нужные преобразования переменных.

Дробный факторный эксперимент (ДФЭ)

Количество испытаний в ПФЭ значительно превосходит число определяемых коэффициентов линейной модели плана эксперимента, причем тем более, чем больше факторов: 2 фактора — 4 опыта, 3 неизвестных; 3 фактора — 8 опытов, 4 неизвестных; 4 фактора — 16 опытов, 5 неизвестных и т. д.

Таким образом, ПФЭ обладает большой избыточностью (в случае, если факторы независимы и не определяются коэффициенты эффектов взаимодействия x_1x_2, x_1x_3, $x_1x_2x_3$,...). Возникает вопрос: как построить план, позволяющий определить коэффициенты линейного уровня, который содержал бы меньшее число опытов, чем ПФЭ, но сохранял бы все свойства ПФЭ.

Пусть мы хотим построить ортогональный план для четырехфакторной модели. Полный факторный эксперимент содержит 16 опытов (точек). Для получения линейной зависимости требуется 5 точек, а для выполнения свойств ортогональности — 6. Есть ли ортогональный план, для которого $6 \leqslant m < 16$? Такой план существует. Это план для случая $k = 3$. В табл. 3.5 представлен ПФЭ 2^3, в который включены все возможные сочетания произведений факторов. В предположении о независимости факторов можно исключить из плана все столбцы, кроме линейных, так как при этом условии все коэффициенты при эффектах взаимодействия должны быть равны нулю. Тогда для четвертой переменной можно воспользоваться любым из столбцов, характеризующим эффекты взаимодействия, например столбцом $x_1x_2x_3$.

Таблица 3.5 – План дробного факторного эксперимента

№	x_0	x_1	x_2	x_3	$x_4 = x_1x_2x_3$	x_1x_2	x_1x_3	x_2x_3
1	+	+	+	+	+	+	+	+
2	+	–	+	+	–	–	–	+
3	+	+	–	+	–	–	+	–
4	+	–	–	+	+	+	–	–
5	+	+	+	–	–	+	–	–
6	+	–	+	–	+	–	+	–
7	+	+	–	–	+	–	–	+
8	+	–	–	–	–	+	+	+

Полученный план содержит половину опытов ПФЭ и носит название полуреплики. Используются также четверть-реплики, 1/8 реплики и т. д. При введенных допущениях о незначимости эффектов взаимодействия, ПФЭ 2^3 можно использовать также для построения модели при $k = 5$. В этом случае один из столбцов эффектов взаимодействия может быть приравнен к линейному фактору x_4, а другой — к линейному фактору x_5. В результате получим матрицу с шестью столбцами и восемью строками вместо $2^5 = 32$ строк, которые потребовались бы при ПФЭ.

Планы ДФЭ, построенные с помощью описанной процедуры, называются *дробными репликами*. Дробные реплики символически записывают следующим образом: 2^{k-d}, где k — общее число факторов; d — число линейных эффектов, приравненных к эффектам взаимодействия; $k - d$ — число факторов в плане ПФЭ, к которому приравнивается дробная реплика. Если выполнить арифметические действия 2^{k-d},

то получим количество опытов в ДФЭ. Целесообразность применения дробных реплик возрастает с ростом количества факторов.

Процедура выбора дробных реплик неформальная. Можно лишь порекомендовать выбирать генерирующее соотношение таким образом, чтобы линейные факторы были смешаны со взаимодействиями возможно более высокого порядка. Так, в полуреплике 2^{4-1} лучше выбирать в качестве генерирующего соотношения $x_4 = x_1x_2x_3$, а не $x_4 = x_1x_2$.

Планы ДФЭ, так же как и планы ПФЭ, обладают свойствами симметричности, нормированности, ортогональности и ротатабельности. Оценки коэффициентов регрессии, найденные с помощью ДФЭ, некоррелированы и имеют одинаковые дисперсии, так же как и в ПФЭ.

3.3 Планирование второго порядка

Обычно исследователь начинает изучение объекта или процесса с моделей максимально низкого порядка, т. е. с первого. Необходимость перехода к уравнениям второго порядка возникает либо когда есть априорная информация о существенной нелинейности модели (тогда исследование начинают с построения квадратичного уравнения регрессии), либо когда построенная линейная модель оказалась неадекватной (и тогда исследование продолжают, переходя к квадратичной модели). Во втором случае желательно использовать максимум информации, извлеченной из линейного плана.

В случае квадратичной модели уже недостаточно варьировать факторы только на двух уровнях. Так, полное уравнение второго порядка для двух факторов

$$y = b_0 + b_1x_1 + b_2x_2 + b_{12}x_1x_2 + b_{11}x_1^2 + b_{22}x_2^2$$

содержит 6 неизвестных коэффициентов регрессии, а варьирование на двух уровнях дает только $2^2 = 4$ точки. В то же время полный факторный эксперимент 3^k явно избыточен. Уже для трех факторов ПФЭ требует $3^3 = 27$ опытов, в то время как модель содержит 10 неизвестных коэффициентов регрессии. Для четырех факторов ПФЭ требует 81 опыт и содержит 15 неизвестных.

Существует ряд подходов к построению планов второго порядка, более экономных, чем ПФЭ, и в то же время отвечающих требованиям оптимальности. Заметим, что в отличие от линейных планов в планах второго порядка не удается одновременно обеспечить сочетание всех свойств (ортогональности, ротатабельности и др.).

Один из подходов к построению планов второго порядка основывается на максимально возможном использовании информации, полученной при линейном планировании. Здесь планы второго порядка получают путем добавления $2k$ точек (где k — число факторов), называемых «звездными» точками, и точки в центре плана к плану ПФЭ 2^k или ДФЭ 2^{k-d}. Предполагается, что в центральной точке может ставиться несколько опытов, число которых n_0 зависит от свойств плана.

Тогда общее число опытов для плана второго порядка

$$N = N_0 + 2k + n_0,$$

где $N_0 = 2^{k-d}$, $0 \leqslant d < k$.

«Звездные» точки выбираются на осях факторного пространства, по две на каждой из осей, на равных расстояниях от центра плана (рис. 3.2). Расстояние α от начала координат до «звездной» точки называется «звездным» плечом. Запишем план второго порядка для $k = 2$ и $n_0 = 1$ (табл. 3.6).

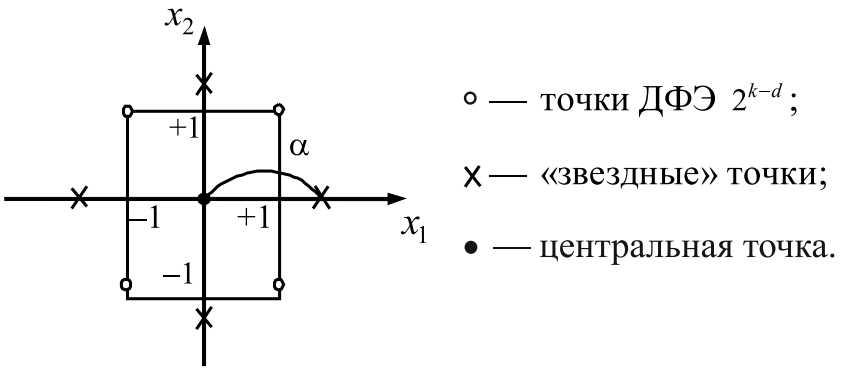


Рис. 3.2 – Графическое представление плана второго порядка для $k = 2$

Таблица 3.6 – План второго порядка для $k = 2$

№	x_0	x_1	x_2	x_1x_2	x_1^2	x_2^2
1	+	–	–	+	+	+
2	+	+	–	–	+	+
3	+	–	+	–	+	+
4	+	+	+	+	+	+
5	+	–α	0	0	α^2	0
6	+	α	0	0	α^2	0
7	+	0	–α	0	0	α^2
8		0	α	0	0	α^2
9	+	0	0	0	0	0

Здесь каждый из факторов варьируется на пяти уровнях: $-\alpha$, -1, 0, 1, $+\alpha$. В рассматриваемых планах, которые называются *центральными композиционными планами* (ЦКП), расположение точек на осях не нарушает ортогональности столбцов первого порядка и эффектов взаимодействия. Это дает возможность получать независимыми соответствующие коэффициенты уравнения регрессии. Но здесь ортогональность плана нарушается для столбцов x_0 и x_i^2:

$$\sum_{k=1}^{N} x_{0k}x_{ik}^2 \neq 0; \quad \sum_{k=1}^{N} x_{ik}^2 x_{jk}^2 \neq 0,$$

так как x_{0k} всегда равно единице, а неотрицательные величины x_{ik}^2, x_{jk}^2 не могут быть все равны нулю.

Чтобы получить ортогональное планирование второго порядка, нужно произвести некоторое преобразование квадратичных переменных и оптимальным образом выбрать величину «звездного» плеча [8].

Таким образом, ЦКП позволяют применять последовательное планирование, при котором сначала строится ДФЭ 2^{k-l}, а затем, если результаты линейного приближения не устраивают исследователя, ДФЭ достраивается до ЦКП и проводится более полное исследование поверхности отклика.

3.4 Поиск оптимальной области

Решение различных задач управления, проектирования и планирования в той или иной мере связано с оптимизацией, нахождением наилучших в определенном смысле различных параметров (факторов). Задача оптимизации может быть решена путем исследования математической зависимости, описывающей область факторного пространства в широком интервале изменения факторов. Однако такие зависимости часто очень сложны. Обычно исследователи идут другим путем: сначала находят оптимальную область, а затем описывают ее уравнением второго или третьего порядка. В настоящее время существует много различных методов поиска оптимальной области, которые можно разбить на две группы: детерминированные и стохастические.

В *детерминированных* методах движение к оптимуму осуществляется на основе информации, получаемой от пробных движений, совершаемых в определенной последовательности.

В *стохастических* методах пробные движения производятся случайным образом, а движение к оптимуму является вполне определенным следствием реакций на случайные пробы.

В практике ПФЭ наиболее широко применяются детерминированные методы поиска. Рассмотрим некоторые из них.

Метод Гаусса—Зейделя (метод покоординатного подъема)

Здесь поочередно фиксируются все факторы, кроме одного, вдоль которого осуществляется движение из начальной точки (используется схема однофакторного эксперимента). Начальная точка выбирается либо случайным образом, либо на основе анализа априорной информации. Таким образом, исследователь изучает поведение каждого фактора в отдельности. Ставится серия опытов при различных значениях одного фактора и, двигаясь параллельно одной из осей факторного пространства, находится наилучшее для рассматриваемого разреза поверхности отклика значение параметра оптимизации. Затем исследованный фактор фиксируется в найденном наилучшем значении, и ставится серия опытов для следующего фактора. Последовательное прохождение всех осей факторного пространства составляет первый цикл исследования. Затем процедуру повторяют до получения оптимума или до попадания в некоторую точку, любое движение из которой «ухудшает» значение параметра оптимизации.

Распространенным недостатком при использовании данного метода является прекращение работы после выполнения первого цикла. Данный метод требует большого количества опытов, но в то же время прост в реализации.

Метод крутого восхождения (метод Бокса-Уилсона)

Сначала выбирается начальная точка, в которой ставится небольшая серия опытов (ДФЭ) для локального описания небольшого участка поверхности отклика полиномом первого порядка. Далее движение осуществляется по поверхности отклика в направлении градиента линейного приближения. Это движение сопровождается *одновременным изменением всех факторов*. Если одного линейного приближения оказывается недостаточно, то ставится новая небольшая серия опытов и находится новое направление для движения по поверхности отклика. Такой шаговый процесс продолжается до тех пор, пока исследователь не попадет в «почти стационарную область», где линейное приближение уже недостаточно. Здесь ставится большая серия опытов, и поверхность отклика описывается уже полиномом второго или третьего порядка. При таком подходе достигается высокая концентрация опытов в той части поверхности отклика, которая преимущественно и интересует исследователя.

Известно, что движение в направлении градиента — это движение по кратчайшему пути. Если поверхность отклика описана локально линейным уравнением, то частные производные равны коэффициентам регрессии. Для движения здесь независимые переменные необходимо изменять пропорционально величине соответствующих коэффициентов регрессии с учетом их знака. Величина шага должна быть пропорциональна произведению b_i на интервал изменения i-го фактора.

На рис. 3.3 приведено сравнение двух рассмотренных методов.

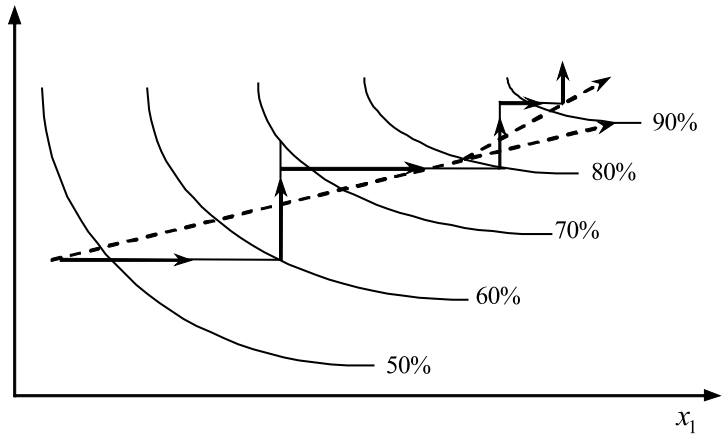


Рис. 3.3 – Графическое представление поисковых методов

Пунктирной линией показано движение методом крутого восхождения, а сплошной — методом покоординатного подъема. Из рисунка видно, что при использовании метода крутого восхождения путь, который необходимо пройти экспериментатору, значительно сокращается. С возрастанием числа факторов эффект от применения метода крутого восхождения значительно повышается.

Практическое применение методов связано с принятием решений после построения линейной модели.

Принятие решений после построения линейной модели

Выбор решения зависит от числа факторов, дробности плана, цели исследования и т. д. (возможно несколько десятков тысяч различных решений). Мы рассмотрим только некоторые «типичные» решения [7].

Случай адекватной линейной модели

Возможны три варианта:

1. Все коэффициенты регрессии значимы.
2. Часть коэффициентов значима, часть — нет.
3. Все коэффициенты незначимы.

Оптимум может быть далеко, близко или неизвестно.

В первом варианте если область оптимума близка, то возможны три решения: окончание исследования, переход к планам второго порядка и движение по градиенту.

При неопределенной или далекой области оптимума выполняется движение по градиенту.

Во втором варианте выбираются решения, реализация которых приводит к получению значимых коэффициентов:

1) изменение интервалов варьирования;
2) перенос центра плана;
3) отсеивание незначимых факторов;
4) увеличение числа параллельных опытов;
5) достройка плана (например, переход к ПФЭ).

Третий вариант чаще всего происходит из-за большой ошибки эксперимента или из-за узких интервалов варьирования. Поэтому здесь возможные решения — это увеличение точности эксперимента за счет улучшения методики исследования или увеличения интервалов варьирования. Если область оптимума близка, то возможно планирование второго порядка.

Случай неадекватной линейной модели

Признаки, по которым можно установить неадекватность:

а) значимость хотя бы одного эффекта взаимодействия;

б) значимость суммы коэффициентов регрессии при квадратичных членах $\sum_i \beta_{ii}$. Оценкой этой суммы служит разность между b_0 и значением y в центре плана y_0. Если разность превосходит ошибку опыта, то гипотеза о незначимости $\sum_i \beta_{ii}$ не может быть принята.

Если область оптимума близка, то либо исследование заканчивается, либо реализуется план второго порядка.

3.5 Стратегическое планирование

Рассмотрим основные проблемы, которые возникают при планировании эксперимента, и способы их решения. В общем случае планирование эксперимента

можно разделить на два этапа: стратегическое и тактическое планирование. Заметим, что такое разделение достаточно условно, но при этом оно помогает выделить основные шаги проведения машинных экспериментов.

Стратегическое планирование *(первая составляющая планирования машинных экспериментов с моделями систем) ставит целью решение задачи получения необходимой информации о системе с помощью модели, реализованной на ЭВМ, с учетом ограничения на ресурсы.*

Тактическое планирование *(вторая составляющая) — это определение способа проведения каждой серии испытаний модели, предусмотренных планом эксперимента.*

Этапы стратегического планирования:

1) построение структурной модели — осуществляется исходя из того, что должно быть сделано;
2) построение функциональной модели — производится исходя из того, что может быть сделано.

Структурная модель плана эксперимента характеризуется числом факторов и числом уровней для каждого фактора. Число элементов эксперимента

$$N_c = q_1 q_2 \ldots q_k,$$

где k — число факторов эксперимента; q_i — число уровней i-го фактора.

Вид и число k выбирается исходя из цели машинного эксперимента, структуры системы и свойств откликов. Отбираются наиболее существенные факторы: опыт показывает, что 20% факторов определяют 80% свойств системы. Минимальное число уровней по каждому фактору равно двум. Выбирается q_i, достаточное для достижения цели. Число уровней чаще всего зависит от предполагаемой функциональной зависимости между откликом и факторами. Анализ результатов упрощается, если уровни равноотстоят друг от друга и если $q_i = q_j$ для всех i, j (пример — ПФЭ первого порядка).

Функциональная модель определяет количество необходимых элементов структурной модели $N_ф$. Говорят, что модель полная, если $N_ф = N_с$ (ПФЭ), неполная, если $N_ф < N_с$ (ДФЭ).

Для нахождения компромиссного решения можно варьировать количеством факторов k, числом уровней q, количеством повторений эксперимента p, при этом учитываются затраты времени на прогон модели τ и стоимость машинного времени c. Полное число прогонов, необходимое при симметрично повторяемом эксперименте, определяется по формуле $N = pq^k$. Если τ — затраты времени на один прогон модели, то общее время, требуемое для проведения эксперимента, определяется как $T = \tau \cdot N$. При затратах c в единицу времени общие затраты составят $C = c \cdot T$.

Пример 3.1

Необходимо спланировать эксперимент при трех факторах (k = 3), каждый из которых имеет три уровня, требуется 15 повторений эксперимента с затратами τ = = 120 сек/на один прогон при стоимости c = 100 руб. в час. В день на моделирование выделяется 60 минут, т. е. требуется $K = \dfrac{N \cdot \tau}{3600}$ дней.

Такой эксперимент потребует $N = 15 \cdot 3^3 = 405$ прогонов, $T = 120/3600 \cdot 405 =$ = 13.5 часов машинного времени, то есть примерно $T_к$ = 13 дней и $C = 100 \cdot 13.5 =$ = 1350 руб. для оплаты машинного времени. Если уменьшить число факторов и установить k = 2, то потребуется $N = 15 \cdot 3^2 = 135$ прогонов, 4.5 часа машинного времени, или $T_к$ = 4.5 дня, и 450 рублей. В результате планирования мы получили, что использование двух факторов вместо трех влечет сокращение затрат примерно на 67%.

Таким образом, использование при стратегическом планировании машинных экспериментов с разработкой структурных и функциональных моделей плана позволяет решить вопрос о практической реализуемости модели на ЭВМ исходя из допустимых затрат ресурсов на моделирование систем.

3.6 Тактическое планирование

Определим основные *задачи тактического планирования*:

1) определение начальных условий и их влияния на достижение установившегося результата при моделировании;
2) обеспечение точности и достоверности результатов моделирования;
3) выбор правил автоматической остановки имитационного эксперимента с моделями.

Первая задача возникает из-за искусственного характера функционирования модели: модель, в отличие от реальной системы, работает эпизодически. Поэтому, когда начинается прогон модели, требуется определенное время для достижения условий равновесия, которые соответствуют условиям работы реальной системы. Решение этой проблемы может быть реализовано двумя способами: либо исключается из рассмотрения начальное время моделирования $(0, T)$, либо начальные условия выбираются так, чтобы сократить время достижения установившегося режима.

Вторая задача тактического планирования может быть сформулирована двумя способами:

1) оценка точности и достоверности при заданном методе реализации модели, при заданном объеме выборки;
2) оценка необходимого числа реализаций при заданных точности и достоверности результатов.

Противоречие между желанием достичь более точных оценок и эффективностью связано с затратами ресурсов. Соответственно, если ограничения на ресурсы существенны, то точность оценивается для заданного объема выборки. Если более важным для процесса исследования является точность моделирования, то определяется объем выборки, необходимый для обеспечения заданной точности результатов.

Степень точности определяется дисперсией случайного фактора. Желаемую степень точности можно задавать в различных формах:

а) в виде доли стандартного отклонения;

б) в процентах от величины среднего значения;

в) в абсолютных величинах.

Достижимая степень точности зависит от природы режимов и величины различий между режимами.

Размер выборки может определяться по одному из двух путей:

1) априорно, т. е. независимо от работы модели (основано на знании модели): многие методы анализа используют предположение о независимости и нормальности распределения откликов модели (основано на центральной предельной теореме);

2) в процессе работы модели и на основе полученных с помощью модели результатов.

Рассмотрим некоторые методы определения объема выборки.

1. *Оценивание среднего значения совокупности*.

Пусть мы хотим построить такую оценку $\overline{X}$ истинного среднего значения совокупности, что

$$P\left\{\mu - d \leqslant \overline{X} \leqslant \mu + d\right\} = 1 - \alpha,$$

где $\overline{X}$ — выборочное среднее, $(1 - \alpha)$ — вероятность того, что интервал $\mu \pm d$ содержит $\overline{X}$. Задача состоит в определении необходимого для этого условия объема выборки. В предположении нормальности распределения выборочных значений совокупности можно использовать для этого следующую формулу:

$$n = \left(\sigma Z_{\alpha/2}\right)^2 / d^2,$$

где $Z_{\alpha/2}$ — двусторонняя стандартная нормальная статистика; d — допустимая разность между оценкой и истинным значением параметра (заданная точность); σ — величина изменчивости совокупности, σ необходимо либо знать, либо выявить в результате эксперимента.

Пример 3.2

Предположим, что мы хотим оценить среднесуточный выход продукции химического завода так, чтобы оценка с вероятностью 0,95 лежала в пределах $\mu \pm \sigma/4$.

Желаемая степень точности здесь задана в виде доли стандартного отклонения: $d = \sigma/4$, табличное значение $Z_{\alpha/2} = 1.96$. Получаем требуемый объем выборки $n = (1,96\sigma)^2/(\sigma/4)^2 = 61$.

Если это возможно, следует определить дисперсию выхода с помощью пробного эксперимента и получить оценку дисперсии S^2, а затем вычислить полное число необходимых наблюдений. Тогда формула для определения объема выборки будет выглядеть следующим образом:

$$n = t^2 S^2 / d^2,$$

где t — табличное значение критерия Стьюдента для заданного доверительного интервала и числа степеней свободы начальной выборки $(n' - 1)$; d — половина ширины доверительного интервала.

2. *Оценивание дисперсии совокупности*.

Построение доверительных интервалов часто требует оценивания среднеквадратического отклонения совокупности как меры дисперсии выборочных значений. Задачу оценивания дисперсии совокупности можно поставить как задачу отыскания оценки S^2 такой, что

$$P\left\{(1-d)\sigma^2 \leqslant S^2 \leqslant (1+d)\sigma^2\right\} = 1 - \alpha,$$

где $0 \leqslant d \leqslant 1$ — число, характеризующее степень близости оценки S^2 к истинной дисперсии σ^2.

Объем выборки, необходимый для обеспечения выбранных условий, будет определяться по формуле

$$n = 1 + \frac{2\left(Z_{\alpha/2}\right)^2}{d^2}.$$

3. *Сравнение двух распределений*.

Часто возникает задача проверки близости распределения отклика модели к некоторому другому распределению. Нас интересует объем выборки, при котором достигается желаемая точность сравнения распределений. Под желаемой точностью здесь будем понимать максимальную разность сравниваемых распределений во всех точках (d). При заданном уровне значимости получим следующее уравнение:

$$n = \left(\frac{\lambda}{d}\right)^2,$$

где λ — табличное значение функции Колмогорова.

Если требуется решить обратную задачу — определить достигаемую точность при заданном объеме выборки, то из всех приведенных формул можно вывести зависимость d от N путем обратных преобразований. Например, в задаче сравнения двух распределений, получаем:

$$d = \frac{\lambda}{\sqrt{n}}.$$

Еще одним методом увеличения точности является метод, состоящий в задании доверительных интервалов для выходных величин и остановки прогона модели по достижении заданного доверительного интервала (*третья задача* тактического планирования). Однако при введении в модель правил остановки и операций вычисления доверительных интервалов увеличивается машинное время, необходимое

для получения одной выборочной точки. Кроме того, существует риск преждевременной остановки и получения оценок на основе слишком малой выборки.

Существует два основных способа включения в модель правил автоматической остановки:

1) проводить прогон модели в два этапа. Сначала сделать прогон, позволяющий получить выборку объемом n, затем использовать полученную выборку для оценивания необходимого объема выборки n^*. Если $n^* < n$, то следует закончить прогон, иначе продолжать до получения недостающих $(n^* - n)$ выборочных значений;
2) использовать последовательную выборку для определения минимальной величины n, по которой можно с достаточной точностью определять среднеквадратическое отклонение S, затем сравнивать величину $\frac{S \cdot t_{(1-\alpha, n-1)}}{\sqrt{n}}$ с заданным уровнем d и останавливать прогон в тот момент, когда эта величина становится меньше d.

3.7 Анализ результатов моделирования

3.7.1 Проверка адекватности системы

Первый вопрос, который нас интересует после проведения эксперимента и вычисления коэффициентов модели, — это проверка адекватности. С этой целью вычисляем *остаточную дисперсию*, или дисперсию *адекватности*:

$$S^2_{\text{ад}} = \frac{1}{N - k - 1} \sum_{i=1}^{N} (y_i - \widehat{y}_i)^2,$$

где N — количество опытов в эксперименте; k — количество факторов; y_i — экспериментальные значения отклика; $\widehat{y}_i$ — величина отклика, предсказанная уравнением регрессии.

Для проверки гипотезы об адекватности модели пользуются F-критерием Фишера, для этого вычисляется соотношение:

$$F = \frac{S^2_{\text{ад}}}{S^2\{y\}},$$

где $S^2\{y\}$ — это выборочная дисперсия, которая вычисляется по формуле:

$$S^2\{y\} = \frac{1}{N(n-1)} \sum_{j=1}^{N} \sum_{i=1}^{n} (y_{ji} - \bar{y}_j)^2,$$

где n — количество повторений одного опыта; y_{ji} — экспериментальное значение отклика в j-ом опыте при i-ом наблюдении;

$$\bar{y}_j = \frac{1}{n} \sum_{i=1}^{n} y_{ji}.$$

Если рассчитанное значение F-критерия не превышает табличного, то с соответствующей доверительной вероятностью модель можно считать адекватной.

Рассмотренный способ расчета дисперсии адекватности применим в случае, если опыты в матрице планирования не дублируются, а информация о выборочной дисперсии извлекается из параллельных опытов в нулевой точке или из предварительных экспериментов. В общем случае

$$S^2_{\text{ад}} = \frac{1}{N-k-1}\sum_{i=1}^{N} n_i(\bar{y}_i - \widehat{y}_i)^2,$$

где n_i — число повторений i-го опыта; $\bar{y}_i$ — среднее арифметическое отклика из n_i параллельных опытов.

Здесь смысл введения n_i в формулу заключается в следующем: различию между экспериментальным и расчетным значением придается тем больший вес, чем больше число повторных опытов.

Адекватность линейного уравнения можно проверить и другим путем. Очевидно, что коэффициент b_0, определенный по результатам полного или дробного факторного эксперимента, всегда является оценкой:

$$b_0 \to \beta_0 + \sum_{i=1}^{k} \beta_{ii}.$$

С другой стороны, величина b_0 является оценкой результата опыта на основном уровне. Поэтому, если выполнить опыт на основном уровне, т. е. получить y_0 и найти разницу $(b_0 - y_0)$, то эта величина является оценкой суммы квадратичных членов в уравнении регрессии. Если разность $(b_0 - y_0)$ велика, линейным уравнением пользоваться нельзя, если мала — возможность использования линейного уравнения не исключена. Значимость различия можно оценить по критерию Стьюдента

$$t_{\text{расч}} = \frac{|b_0 - y_0|\sqrt{N}}{S\{y\}},$$

где $S\{y\} = \sqrt{S^2\{y\}}$ — выборочное среднеквадратическое отклонение.

Гипотеза об адекватности уравнения принимается в случае, когда $t_{\text{расч}} \leqslant t_{\text{табл}}\left(\alpha; f = N(n-1)\right)$.

3.7.2 Проверка значимости коэффициентов

Проверка значимости коэффициентов регрессии осуществляется двумя равноценными способами: проверкой по t-критерию Стьюдента или построением доверительного интервала.

Сначала находятся оценки дисперсии коэффициентов регрессии $S^2\{b_i\} = \frac{S^2\{y\}}{N}$, т. е. дисперсии всех коэффициентов равны друг другу, так как они зависят только от ошибки опыта и числа опытов. Затем вычисляется t_i по уравнению

$$t_i = \frac{|b_i|\sqrt{N}}{\sqrt{S^2\{y\}}} = \frac{|b_i|}{\sqrt{S^2\{b_i\}}}.$$

Расчетное значение критерия сравнивается с табличным значением. Если $t_i \leqslant t_{\text{табл}}\left(\alpha; f = N(n-1)\right)$, то соответствующий коэффициент регрессии незначим. Факторы, имеющие большие значения t_i, оказывают более существенное влияние на процесс.

Проверку значимости коэффициентов регрессии можно осуществлять и построением доверительного интервала. В случае ортогонального планирования первого порядка доверительный интервал Δb_i равен:

$$\Delta b_i = \pm t_{\text{табл}}\left(\alpha, f = N(n-1)\right) \cdot S\{b_i\} = \pm t_{\text{табл}} \frac{S\{y\}}{\sqrt{N}}.$$

Коэффициент значим, если его абсолютная величина больше доверительного интервала.

Если некоторые коэффициенты регрессии признаны незначимыми, то соответствующие факторы могут быть выведены из состава уравнения.

3.7.3 Обоснованность модели

Оценить качество модели означает оценить уровень нашей уверенности в том, что выводы, сделанные с помощью модели, применимы и к реальной системе. Можно показать, что при достаточно большом числе повторений эксперимента (величине выборки) результаты моделирования можно сделать сколь угодно точными.

Проблема состоит в том, что эти точные результаты точны только в том случае, когда идентичны процессы, происходящие в модели и в реальной системе.

Правильность теории построения моделей может быть проверена только на практике. Так как при построении любой модели используются упрощения и абстракции реальной системы, то модель не является абсолютно точной в смысле однозначного соответствия ее реальной системе. Следовательно, существуют различные степени корректности и точности модели.

Возможные недостатки модели:

- модель может содержать несущественные переменные;
- модель может не содержать существенных переменных;
- одна или несколько существенных переменных могут быть оценены или представлены неточно;
- структура модели может быть ошибочной (т. е. зависимость, связывающая отклик с управляемыми и неуправляемыми переменными, может быть неверной).

Задачу обоснованности модели решают в несколько этапов.

На *первом этапе* устанавливается, в какой мере каждый из факторов влияет на параметр оптимизации. Величина коэффициента регрессии — количественная мера этого влияния. О характере влияния факторов говорят знаки коэффициентов. Знак «плюс» свидетельствует о том, что с увеличением значения фактора растет величина параметра оптимизации, а при знаке «минус» — убывает.

Далее выясняется, как расположить совокупность факторов в ряд по силе их влияния на параметр оптимизации. Факторы, коэффициенты которых не значимы,

не интерпретируются, и о них можно сказать только, что при данных интервалах изменения и ошибке воспроизводимости они не оказывают существенного влияния на отклик.

На *втором этапе* проверяется правильность априорных представлений о механизме процесса. Например, если ожидается, что с ростом температуры должно происходить увеличение отклика, а коэффициент регрессии «минус», то возникает противоречие. Здесь возможны две причины: либо в эксперименте допущена ошибка, либо неверны априорные представления.

Когда имеется большая априорная информация, позволяющая выдвигать гипотезы о механизме явлений, можно перейти к *третьему этапу*: проверке гипотез о механизме явлений и выдвижении новых гипотез.

Статистические испытания, используемые для проверки принятых при построении модели допущений, а также выводов, которые делаются на основе результатов эксперимента, решают, главным образом, две задачи:

А. Проверка гипотез:

1) оценка параметров совокупности в предположении о виде функции распределения вероятностей (на основе критериев F, t и Z);
2) оценка параметров совокупности независимо от вида функции распределения (посредством непараметрических критериев);
3) установление закона распределения для совокупности, из которой получена выборка (используются критерии согласия χ^2 и Колмогорова—Смирнова);
4) определение тесноты связи между двумя или более переменными (с использованием корреляционного анализа).

Б. Оценивание:

1) вычисление точечных и интервальных оценок параметров совокупности (выборочные $\overline{X}$, S^2);
2) определение количественных уравнений, связывающих две или более переменные (на основе регрессионного анализа).

Перечислим основные статистические методы, используемые для оценивания и проверки гипотез:

1) проверка средних значений,
2) анализ дисперсий и ковариаций,
3) проверка по критериям согласия,
4) регрессионный и корреляционный анализ.

Рассмотрим более подробно первый из перечисленных методов.

Проверка средних значений

С помощью данной проверки решается вопрос о том, принадлежат ли две различные выборки одной данной совокупности. Выбор критерия определяется конкретной ситуацией и зависит от решения следующих вопросов:

1) имеют ли обе совокупности одинаковую дисперсию σ^2;

2) известны ли дисперсии совокупностей, или они оцениваются;
3) проводится ли сравнение среднего значения выборки с известным средним значением второй совокупности;
4) имеют ли обе выборки одинаковую величину;
5) каковы размеры выборок.

Рассмотрим задачу проверки гипотезы H_0 о том, что две совокупности имеют одинаковую дисперсию σ^2. Вычислим выборочные дисперсии S_1^2 и S_2^2. Пусть $S_1^2 > S_2^2$, тогда расчетное $F = S_1^2/S_2^2$ сравниваем с табличным F-распределением для заданного числа степеней свободы $(N_1 - 1, N_2 - 1)$. Если $F_{\text{расч}} < F_{\text{табл}}$, то гипотеза о равенстве дисперсий принимается.

С учетом всех известных данных конкретный критерий выбирается по схеме, приведенной на рис. 3.4.

Пример 3.3

При исследовании работы химического завода получены данные измерений некоторой характеристики функционирования одной из подсистем в течение пяти недель. Модель этой подсистемы использовалась ранее для предсказания той же характеристики в течение семи недель. Результаты (тонны в неделю):

Для подсистемы: 22; 22.5; 22.5; 24; 23.5.

Для модели: 24.5; 19.5; 25.5; 20; 18; 21.5; 21.5.

При $\alpha = 0.05$ требуется проверить, есть ли значимая разница между средними значениями характеристик.

Определим для каждой выборки значения выборочного среднего и выборочной дисперсии:

$$N_1 = 5, \quad \bar{x}_1 = 22.9, \quad S_1^2 = 0.68.$$
$$N_2 = 7, \quad \bar{x}_2 = 21.5, \quad S_2^2 = 7.25.$$

Проверим гипотезу о равенстве дисперсий: $F = \dfrac{S_2^2}{S_1^2} = 10.66$, $F_{\text{табл}} = 6.16 \Rightarrow F > F_{\text{табл}} \Rightarrow \sigma_1^2 \neq \sigma_2^2$.

Выбираем по таблице (см. рис. 3.4) критерий для сравнения средних: теоретические дисперсия и математическое ожидание неизвестны, выборочные дисперсии двух выборок не равны, объемы выборок меньше 30 и не равны между собой.

$$t'_{\text{выч}} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\dfrac{S_1^2}{N_1 - 1} + \dfrac{S_2^2}{N_2 - 1}}} = 1.2,$$

$$\omega_1 = \frac{S_1^2}{N_1}, \quad \omega_2 = \frac{S_2^2}{N_2},$$
$$t_1(\text{табл}, N_1 - 1) = 2.78, \quad t_2(\text{табл}, N_2 - 1) = 2.45.$$
$$t_{\text{крит}} = \frac{\omega_1 t_1 + \omega_2 t_2}{\omega_1 + \omega_2} = 2.49.$$

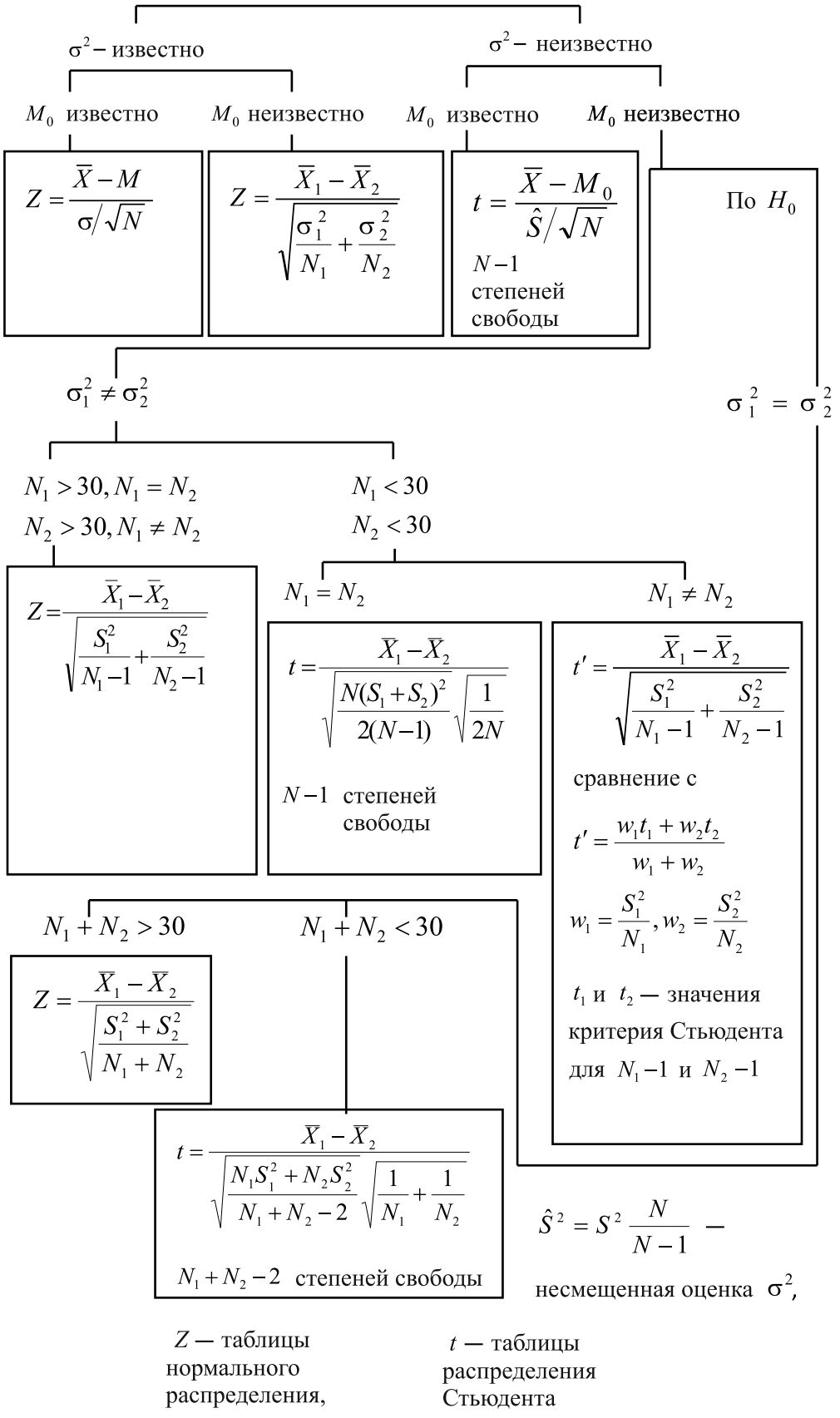


Рис. 3.4 – Схема выбора критерия проверки гипотезы о равенстве средних

Мы получили $t'_{\text{выч}} < t_{\text{крит}}$, следовательно, гипотеза о равенстве средних $\bar{x}_1 = \bar{x}_2$ принимается.

Однако различие дисперсий заставляет пересмотреть модель, т. к. большая дисперсия может привести к нежелательным последствиям при включении модели в более общую модель.

В заключение перечислим основные требования к обоснованности модели. Наибольшая обоснованность модели достигается:

1) использованием здравого смысла и логики;
2) максимальным использованием опыта тех, кто хорошо знает исследуемую систему;
3) эмпирической проверкой с помощью соответственных статистических испытаний всех используемых допущений, гипотез и т. п.;
4) максимальным вниманием к всевозможным деталям с проверкой и перепроверкой каждого шага процесса построения модели;
5) применением на стадии доводки модели контрольных данных для проверки того, что модель работает должным образом;
6) сравнением соотношений входа и выхода модели и реальной моделируемой системы (если возможно) с использованием статистических критериев;
7) проведением, когда это возможно, полевых испытаний модели и ее подсистем;
8) проведением анализа чувствительности выхода модели по отношению к изменениям входных параметров и т. п.;
9) тщательным сравнением результатов, предсказываемых моделью и получаемых в процессе работы реальной системы, которая подвергается исследованию.

Контрольные вопросы по главе 3

1. Что такое фактор, уровень, отклик?
2. Каково преимущество многофакторного эксперимента перед однофакторным?
3. Какое количество опытов требуется для ПФЭ первого порядка?
4. Что такое ДФЭ?
5. В каких случаях строятся планы второго порядка?
6. Перечислите вопросы тактического планирования.
7. Как проверяется адекватность модели?
8. Какие методы поиска оптимальной области Вы знаете?

ЛИТЕРАТУРА

[1] Полищук Ю. М. Имитационно-лингвистическое моделирование систем с природными компонентами / Ю. М. Полищук. — Новосибирск : Наука, 1992.

[2] Шеннон Р. Имитационные моделирование систем — искусство или наука / Р. Шеннон. — М. : Мир, 1981.

[3] Кузин Л. Т. Основы кибернетики : в 2 т. / Л. Т. Кузин. — М. : Энергия, 1980.

[4] Моделирование систем : учебник для вузов / Б. Я. Советов [и др.]. — М. : Высш. шк., 2005. — 342 с.

[5] Нургужин М. Р. Компьютерное моделирование систем / М. Р. Нургужин, В. В. Яворский. — Караганда : Изд-во КарГТУ, 2006. — 200 с.

[6] Решетникова Г. Н. Моделирование систем : учеб. пособие / Г. Н. Решетникова. — 2-е изд., перераб. и доп. — Томск : ТУСУР, 2007. — 440 с.

[7] Шрайбер Г. Дж. Моделирование на GPSS / Г. Дж. Шрайбер. — М. : Высш. шк., 1980.

[8] Кориков А. М. Математические методы планирования эксперимента / А. М. Кориков. — Томск : Изд-во Томск. гос. ун-та, 1973.

[9] Решетников М. Т. Планирование эксперимента и статистическая обработка данных / М. Т. Решетников. — Томск : Томск. гос. ун-т систем управления и радиоэлектроники, 2000.

Приложение А

ОТЧЕТ

Приведем пример стандартного отчета для задачи, рассмотренной в примере 2.10.

GPSS World Simulation Report — pasport.3.1

Friday, January 21, 2005 13:51:41

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	540.000	26	1	1

NAME	VALUE
BYE	17.000
NACH	10008.000
NO_OBSL	10009.000
OBED	10007.000
OCH_NACH	10002.000
OCH_PROP	10004.000
PROP	10000.000
RAZN	10005.000
TAB1	10001.000
TAB2	10003.000
TIME	10006.000
UXOD	16.000
VXOD	3.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	88	0	0
	2	GATE	88	0	0
VXOD	3	QUEUE	82	0	0
	4	GATE	82	0	0
	5	SEIZE	82	0	0
	6	DEPART	82	0	0
	7	ADVANCE	82	0	0

	8	RELEASE	82	0	0
	9	TRANSFER	82	0	0
	10	QUEUE	78	0	0
	11	GATE	78	0	0
	12	ENTER	78	0	0
	13	DEPART	78	0	0
	14	ADVANCE	78	0	0
	15	LEAVE	78	0	0
UXOD	16	TERMINATE	82	0	0
BYE	17	TERMINATE	6	0	0
	18	GENERATE	1	0	0
	19	LOGIC	1	0	0
	20	ADVANCE	1	0	0
	21	LOGIC	1	0	0
	22	ADVANCE	1	0	0
	23	LOGIC	1	0	0
	24	ADVANCE	1	0	0
	25	SAVEVALUE	1	0	0
	26	TERMINATE	1	0	0

FACILITY	ENTRIES	UTIL.	AVE.TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
NACH	82	0.542	3.570	1	0	0	0	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(−0)	RETRY
OCH_NACH	13	0	82	31	2.367	15.589	25.064	0
OCH_PROP	3	0	78	53	0.431	2.982	9.305	0

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
PROP	3	3	0	3	78	1	1.643	0.548	0	0

TABLE	MEAN	STD.DEV	RANGE	RETRY	FREQUENCY	CUM.%
TAB1	15.589	21.814		0		
			_ – 10.000		52	63.41
			10.000 – 20.000		5	69.51
			20.000 – 30.000		9	80.49
			30.000 – 40.000		1	81.71
			40.000 – 50.000		4	86.59
			50.000 – 60.000		3	90.24
			60.000 – 70.000		7	98.78
			70.000 – 80.000		1	100.00
TAB2	2.982	9.252		0		
			_ – 10.000		72	92.31
			10.000 – 20.000		2	94.87
			20.000 – 30.000		1	96.15
			30.000 – 40.000		1	97.44
			40.000 – 50.000		1	98.72
			50.000 – 60.000		1	100.00

LOGICSWITCH	VALUE	RETRY
TIME	1	0
OBED	0	0

SAVEVALUE	RETRY	VALUE
NO_OBSL	0	0

FEC	XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
	90	0	540.784	90	0	1		

Ниже приведена смысловая интерпретация выдаваемых в отчете результатов.

Заголовок.

GPSS World Simulation Report – pasport.3.1

Friday, January 21, 2005 13:51:41

В заголовок включена информация об имени файла, из которого получен отчет, а также информация о времени и дате прогона модели.

Общая информация.

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	540.000	26	1	1

- START TIME. Абсолютное системное время на начало рассматриваемого периода. START TIME устанавливается равным абсолютному системному времени, определенному командами RESET или CLEAR.
- END TIME. Абсолютное системное время на момент окончания моделирования.
- BLOCKS. Количество блоков в программе, исключая блоки описания.
- FACILITIES. Количество объектов «устройство» в программе.
- STORAGES. Количество объектов «память» в программе.

Имена.

NAME	VALUE
BYE	17.000
NACH	10008.000
NO_OBSL	10009.000
OBED	10007.000
OCH_NACH	10002.000
OCH_PROP	10004.000
PROP	10000.000
RAZN	10005.000
TAB1	10001.000
TAB2	10003.000
TIME	10006.000
UXOD	16.000
VXOD	3.000

- NAME. Определенные пользователем имена, используемые в программе.
- VALUE. Числовое значение, присвоенное имени. Система присваивает значения именам, начиная с 10000. Исключение составляют имена блоков, им присваивается числовое значение в соответствии с порядковым номером в программе.

Блоки.

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	88	0	0
	2	GATE	88	0	0
VXOD	3	QUEUE	82	0	0
	4	GATE	82	0	0
	5	SEIZE	82	0	0
	6	DEPART	82	0	0
	7	ADVANCE	82	0	0
	8	RELEASE	82	0	0
	9	TRANSFER	82	0	0
	10	QUEUE	78	0	0
	11	GATE	78	0	0
	12	ENTER	78	0	0
	13	DEPART	78	0	0
	14	ADVANCE	78	0	0
	15	LEAVE	78	0	0
UXOD	16	TERMINATE	82	0	0
BYE	17	TERMINATE	6	0	0
	18	GENERATE	1	0	0
	19	LOGIC	1	0	0
	20	ADVANCE	1	0	0
	21	LOGIC	1	0	0
	22	ADVANCE	1	0	0
	23	LOGIC	1	0	0
	24	ADVANCE	1	0	0
	25	SAVEVALUE	1	0	0
	26	TERMINATE	1	0	0

- LABEL. Имя блока, которое ему присвоено.
- LOC. Порядковый номер блока в программе.
- BLOCK TYPE. Имя блока-оператора в GPSS.
- ENTRY COUNT. Количество транзактов, вошедших в данный блок с момента последнего RESET или CLEAR или с момента начала моделирования.
- CURRENT COUNT. Количество транзактов, находящихся в блоке на момент окончания моделирования.
- RETRY. Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния объекта данного блока.

Устройства.

FACILITY	ENTRIES	UTIL.	AVE.TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
NACH	82	0.542	3.570	1	0	0	0	0	0

- FACILITY. Имя или номер объекта «устройство».
- ENTRIES. Количество раз, которое устройство было занято, с момента последнего RESET или CLEAR или с момента последнего запуска модели.

- UTIL. Средняя загрузка устройства за последний измеряемый период времени (доля системного времени, которое устройство было занято, от общего времени моделирования). Измеряемый период времени отсчитывается от начала моделирования или с момента последнего использования команды RESET или CLEAR.
- AVE. TIME. Среднее время нахождения одного транзакта в устройстве.
- AVAIL. Состояние доступности устройства на конец моделирования. 1 означает, что устройство доступно, 0 — не доступно.
- OWNER. Номер транзакта, который занимает устройство. 0 означает, что устройство свободно.
- PEND. Количество транзактов, ожидающих в очереди, чтобы занять устройство через блок PREEMPT.
- INTER. Количество транзактов, претендующих на устройство после прерывания.
- RETRY. Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния данного устройства.
- DELAY. Количество транзактов, ожидающих в очереди, чтобы занять устройство (включает транзакты, которые пытаются занять устройства через блоки SEIZE и PREEMPT).

Очереди.

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(−0)	RETRY
OCH_NACH	13	0	82	31	2.367	15.589	25.064	0
OCH_PROP	3	0	78	53	0.431	2.982	9.305	0

- QUEUE. Имя объекта «очередь».
- MAX. Максимальная длина очереди в течение рассматриваемого периода моделирования. Рассматриваемый период считается с момента начала моделирования или с момента последнего оператора RESET или CLEAR.
- CONT. Длина очереди на момент окончания моделирования.
- ENTRY. Общее количество входов за рассматриваемый период.
- ENTRY(0). Количество «нулевых» входов. Общее количество транзактов, находящихся в очереди 0 единиц времени.
- AVE.CONT. Средняя длина очереди за рассматриваемый период.
- AVE.TIME. Среднее время нахождения одного транзакта в очереди.
- AVE.(-0). Среднее время нахождения одного транзакта в очереди за исключением «нулевых» входов.
- RETRY. Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния данной очереди.

Память.

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
PROP	3	3	0	3	78	1	1.643	0.548	0	0

- STORAGE. Имя объекта «память».
- CAP. Емкость памяти, определенная блоком STORAGE.
- REM. Количество свободных ячеек памяти на момент окончания моделирования.
- MIN. Минимальное количество занятых ячеек памяти в течение рассматриваемого периода.
- MAX. максимальное количество занятых ячеек памяти в течение рассматриваемого периода.
- ENTRIES. Общее количество входов за рассматриваемый период.
- AVL. Состояние доступности памяти на конец моделирования. 1 означает, что память доступна, 0 — не доступна.
- AVE.C. Среднее количество занятых ячеек памяти в течение рассматриваемого периода.
- UTIL. Средняя загрузка памяти за последний измеряемый период времени (доля системного времени, которое память была занята, от общего времени моделирования).
- RETRY. Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния данной памяти.
- DELAY. Количество транзактов, ожидающих в очереди, чтобы занять память через блок ENTER.

Таблицы.

TABLE	MEAN	STD.DEV	RANGE	RETRY	FREQUENCY	CUM.%
TAB1	15.589	21.814		0		
			_ – 10.000		52	63.41
			10.000 – 20.000		5	69.51
			20.000 – 30.000		9	80.49
			30.000 – 40.000		1	81.71
			40.000 – 50.000		4	86.59
			50.000 – 60.000		3	90.24
			60.000 – 70.000		7	98.78
			70.000 – 80.000		1	100.00
TAB2	2.982	9.252		0		
			_ – 10.000		72	92.31
			10.000 – 20.000		2	94.87
			20.000 – 30.000		1	96.15
			30.000 – 40.000		1	97.44
			40.000 – 50.000		1	98.72
			50.000 – 60.000		1	100.00

- TABLE. Имя объекта «таблица».
- MEAN. Среднее арифметическое табулируемой величины.
- STD.DEV. Выборочное стандартное отклонение табулируемой величины, рассчитанное по формуле: $s.d. = \sqrt{\frac{\sum x^2 - 1/N\,(\sum x)^2}{N-1}}$.
- RANGE. Границы интервалов, по которым рассчитывается частота попадания табулируемой величины. Интервалы, частота попадания в которые равна 0, не выводятся.
- RETRY. Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния данной таблицы.
- FREQUENCY. Частота попадания в интервал.
- CUM.% Интегральная частота попадания, выраженная в процентах.

Логические переключатели.

LOGICSWITCH	VALUE	RETRY
TIME	1	0
OBED	0	0

- LOGICSWITCH. Имя или номер логического переключателя.
- VALUE. Значение логического переключателя на момент окончания моделирования. 1 означает «включен» или «истина», 0 означает «выключен» или «ложь».
- RETRY. Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния данного логического переключателя.

Ячейки.

SAVEVALUE	RETRY	VALUE
NO_OBSL	0	0

- SAVEVALUE. Имя или номер ячейки.
- VALUE. Значение ячейки на момент окончания моделирования.
- RETRY. Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния данной ячейки.

Список будущих событий.

FEC	XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
90	0		540.784	90	0	1		

- XN. Номер транзакта в списке будущих событий.
- PRI. Текущий приоритет транзакта.
- BDT. Время в абсолютном системном измерении, когда транзакт должен покинуть список будущих событий.
- ASSEM. Номер транзакта в общем списке транзактов.
- CURRENT. Номер блока, в котором находится транзакт на момент создания отчета.

- NEXT. Номер блока, куда будет направлен транзакт после выхода из списка будущих событий.
- PARAMETER. Номера или имена параметров транзакта.
- VALUE. Значение параметра.

ГЛОССАРИЙ

Арифметическая переменная — вычислительный объект языка GPSS, позволяющий выполнить заданную последовательность арифметических операций над любыми стандартными числовыми атрибутами модели для вычисления значения зависимого от них параметра.

Атрибут транзакта — параметр транзакта, характеризующий какие-то свойства транзакта. В процессе имитации значения параметра могут быть изменены. Каждый транзакт обладает совокупностью до 100 таких параметров.

Имитационная модель — это компьютерная программа, которая описывает структуру и воспроизводит поведение реальной системы во времени.

Логический переключатель — объект языка моделирования GPSS, имитирующий единицу оборудования, которая может находится только в двух положениях: «включен» и «выключен». Перед началом выполнения программы все переключатели устанавливаются в положение «выключен».

Матрица — вычислительный объект языка GPSS, который служит для хранения некоторых постоянных и/или изменяющихся значений данных программы в виде массивов.

Метод кусочной аппроксимации — метод генерации случайных величин с заданным законом распределения, основанный на аппроксимации заданной функции плотности распределения равномерным законом. Является универсальным, приближенным.

Метод нелинейных преобразований — метод генерации случайных величин с заданным законом распределения, основанный на получении обратной функции, преобразующей равномерно распределенную случайную величину в искомую. Является универсальным, точным.

Метод комбинаций — определение закона распределения (появления) числа единиц (нулей) в *n*-разрядном двоичном числе.

Модель — объект-заместитель, который в определенных условиях может заменять объект-оригинал, обеспечивая изучение некоторых свойств и характеристик оригинала.

Опыт — фиксированный набор уровней факторов в плане эксперимента, который определяет одно из возможных состояний исследуемой системы.

Отклик — наименование выходной (зависимой, эндогенной) переменной в теории планирования эксперимента.

Очередь — статистический объект языка моделирования GPSS, связанный со сбором статистики о задержках, возникающих на пути прохождения транзакта.

Память (накопитель) — объект языка моделирования GPSS, имитирующий единицу оборудования, в которой может обрабатываться (храниться) несколько транзактов одновременно. Память позволяет легко моделировать средства обработки с ограниченной емкостью (стоянки автотранспорта, портовые причалы, складские помещения, конвейеры и т. п.).

План второго порядка — план эксперимента, позволяющий вычислить коэффициенты уравнения полинома второго порядка.

План первого порядка — план эксперимента, позволяющий вычислить коэффициенты линейного уравнения регрессии.

План эксперимента — набор опытов, который определяет объем и порядок проведения вычислений, способы накопления и статистической обработки результатов моделирования.

Полный факторный эксперимент — план эксперимента, в котором рассматриваются все сочетания уровней всех факторов.

Семейство транзактов — множество транзактов, состоящее из исходного транзакта и всех его копий. Копия члена семейства является членом того же семейства. Любой транзакт — член только одного семейства.

Симулятор — комплекс программ, планирующий выполнение событий, реализующий функционирование блоков моделей, регистрирующий статистическую информацию о прохождении транзактов. Основной функцией симулятора является поддержание правильного хода часов системного времени и выяснение возможностей продвижения транзактов в программе модели.

Стандартный числовой атрибут (СЧА) — атрибут объекта языка GPSS, однозначно определяющий его статус. СЧА меняется в процессе имитации, изменить его значение может как симулятор, так и пользователь. Для указания конкретного объекта, по которому необходимо получить требуемую информацию, за именем СЧА должно следовать числовое или символьное имя этого объекта.

Таблица — статистический объект языка моделирования GPSS, обеспечивающий накопление в процессе моделирования статистики о каком-либо заданном случайном параметре модели. По окончании прогона модели эта статистика автоматически обрабатывается и выводится на печать, в частности, в виде таблицы относительных частот попадания значений случайного параметра (аргумента таблицы) в указанные частотные интервалы.

Транзакт — формальный объект, который «путешествует» по системе, встречая на пути всевозможные задержки, вызванные занятостью тех или иных единиц оборудования. В качестве транзакта может выступать программа обработки информации, телефонный вызов, покупатель в магазине, отказ системы при исследовании надежности и т. д.

Уровень — одно из возможных значений фактора в эксперименте.

Устройство — объект языка моделирования GPSS, имитирующее единицу оборудования, которое может одновременно обрабатывать только один транзакт. Оно служит для моделирования таких средств обработки элементов потоков, как станки, рабочие, каналы связи и т. п.

Фактор — название входной, независимой (экзогенной) переменной в теории планировании эксперимента.

Язык GPSS — язык имитационного моделирования, ориентированный на решение задач статистического моделирования на ЭВМ процессов с дискретными событиями.

Язык моделирования представляет собой процедурно-ориентированный язык, обладающий специфическими чертами. Основные языки моделирования разрабатывались в качестве программного обеспечения имитационного подхода к изучению процесса функционирования определенного класса систем.

Ячейка — вычислительный объект языка GPSS, используемый для записи, накопления и хранения численных значений различных входных и выходных параметров моделируемой системы. Эти значения могут быть использованы для организации счетчиков числа проходящих транзактов, для вывода значений варьируемых параметров модели, для временного хранения значений стандартных числовых атрибутов.

Учебное издание

Салмина Нина Юрьевна

МОДЕЛИРОВАНИЕ СИСТЕМ

Учебное пособие

Корректор Осипова Е. А.
Компьютерная верстка Насынова Н. Е.

Подписано в печать 09.12.13. Формат 60x84/8.
Усл. печ. л. 13,95. Тираж 100 экз. Заказ

Издано в ООО «Эль Контент»
634029, г. Томск, ул. Кузнецова д. 11 оф. 17
Отпечатано в Томском государственном университете
систем управления и радиоэлектроники.
634050, г. Томск, пр. Ленина, 40
Тел. (3822) 533018.