# 6) Принципы имитационного моделирования

**Тема 3. ПРИНЦИПЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ**

**3.1. Понятие статистического эксперимента**

Имитационное моделирование представляет собой наблюдение поведения модели системы под влиянием входных воздействий. При этом часть из них (а может быть и все) носят случайный характер. В результате такого наблюдения исследователь получает набор экспериментальных данных, на основе которых могут быть оценены характеристики системы.

Очевидно, что аналитические модели для проведения имитационного эксперимента не годятся, и здесь нужна специальная «имитационная» модель, которая должна отвечать следующим основным требованиям:

· Отражать логику функционирования исследуемой системы во времени;

· Обеспечить возможность проведения статистического эксперимента.

Одним из основных понятий имитационного моделирования является понятие статистического эксперимента.

В его основе лежит метод статистических испытаний (метод Монте-Карло). Суть метода заключается в том, что результат испытания ставится в зависимость от значения некоторой случайной величины (СВ), распределенной по заданному закону. Результат каждого конкретного испытания носит случайный характер.

Проведя серию испытаний получают множество частных значений наблюдаемой характеристики (то есть выборку). Поученные статистические данные обрабатываются и представляются в виде численных оценок интересующих исследователя параметров.

Отметим, что метод статистических испытаний применим для исследования как стохастических, так и детерминированных систем.

Важной особенностью метода является то, что его применение практически невозможно без использования компьютерной техники.

Имитационное моделирование не ограничивается разработкой модели и написанием соответствующей программы, а требует подготовки и проведения статистического эксперимента. В cвязи с этим результаты имитационного моделирования следует рассматривать как экспериментальные данные, требующие специальной обработки и анализа. Для любого модельного эксперимента необходимо ответить на следующие вопросы:

1. Какова должна быть продолжительность эксперимента для достижения стационарных условий?
2. Как получить статистически независимые наблюдения?
3. Сколько наблюдений необходимо для обеспечения требуемой точности?

**3.2. Область применения и классификация имитационных моделей**

**Имитационная модель** (ИМ) — это формальное (то есть выполненное на некотором формальном языке) описание логики функционирования исследуемой системы и взаимодействия отдельных ее элементов во времени, учитывающее наиболее существенные причинно-следственные связи, присущие системе, и обеспечивающее проведение статистических экспериментов.

Необходимо отметить два важных обстоятельства:

1. взаимосвязь между отдельными элементами системы, описанными в модели, а также между некоторыми величинами (параметрами) может быть представлена в виде аналитических зависимостей (например, при моделировании полета управляемой ракеты отработка поступающих на борт команд может быть описана на уровне логики, а возникающие перегрузки рассчитываются аналитически);
2. модель можно считать реализуемой и имеющей практическую ценность только в том случае, если в ней отражены лишь те свойства реальной системы, которые влияют на значение выбранного показателя эффективности.

Как было отмечено выше, для ИМ практически отсутствуют ограничения на область их применения (по типу моделируемой системы), и речь может идти только о целесообразности использования ИМ в данной предметной области и об объеме трудозатрат на ее разработку.

Поскольку основой имитационного моделирования является метод статистических испытаний, наибольший эффект от его применения достигается при исследовании сложных систем, на функционирование которых существенное влияние оказывают случайные факторы.

Применение имитационного моделирования целесообразно также в следующих случаях:

1. если не существует законченной постановки задачи на исследование и идет процесс познания объекта моделирования;
2. если характер протекающих в системе процессов не позволяет описать эти процессы в аналитической форме;
3. если необходимо наблюдать за поведением системы (или отдельных ее компонентов) в течение определенного периода, в том числе с изменением скорости протекания процессов;
4. при изучении новых ситуаций в системе либо при оценке функционирования ее в новых условиях;
5. если исследуемая система является элементом более сложной системы, другие элементы которой имеют реальное воплощение;
6. когда необходимо исследовать поведение системы при введении в нее новых компонентов;
7. при подготовке специалистов и освоении новой техники (в качестве тренажеров).

Но имитационные модели имеют целый ряд недостатков. Первый, и весьма существенный, заключается в том, что разработка ИМ, как правило, требует больших затрат времени и сил. Кроме того, любая имитационная модель сложной системы значительно менее «объективна», чем аналитическая модель, поскольку она прежде всего отражает субъективные представления разработчика о моделируемой системе. Причем бывает достаточно сложно как опровергнуть, так и обосновать адекватность созданной ИМ, особенно если речь идет о проектируемой системе. Кроме того, результаты имитационного моделирования, как и при любом численном методе, всегда носят частный характер. Для получения обоснованных выводов необходимо проведение серии модельных экспериментов, а обработка результатов требует применения специальных статистических процедур.

Указанные недостатки можно преодолеть следующим образом.

Во-первых, современное состояние вычислительной техники и ее программного обеспечения позволило создать пакеты моделирования, использование которых существенно сокращает трудозатраты на создание моделей, статистический анализ и визуализацию полученных результатов.

Во-вторых, «объективность» создаваемой модели может быть обеспечена в том случае, когда для каждого варианта постановки задачи исследования выбирается соответствующая схема построения модели.

В этом отношении знание существующих схем построения имитационных моделей является весьма полезным.

Наиболее важный признак — **способ представления в модели динамики**

**(движения) системы.** Она может быть описана посредством событий, работ (активностей), процессов и транзактов.

Другой важный признак — **способ изменения модельного времени.** По этому признаку различают моделирование с постоянным шагом и моделирование по особым состояниям.

Все эти понятия являются основополагающими в теории имитационного моделирования.

В зависимости от этапа и назначения проводимых исследований применяется один из трех наиболее распространенных видов имитационных экспериментов:

1. исследование относительного влияния различных факторов на значения выходных характеристик системы;
2. нахождение аналитической зависимости между интересующими исследователя выходными характеристиками и факторами;
3. отыскание оптимальных значений параметров системы (так называемый «экстремальный эксперимент»).

Вид эксперимента влияет не только на выбор схемы ее формализации, но также на построение плана эксперимента и выбор метода обработки его результатов.

С точки зрения организации взаимодействия исследователя с моделью в ходе эксперимента ИМ делятся на автоматические и диалоговые.

**Автоматическими** называются ИМ, взаимодействие пользователя с которыми сводится только к вводу исходной информации и управлению началом и окончанием работы моделей.

**Диалоговыми** называются ИМ, позволяющие исследователю активно управлять ходом моделирования.

**3.3. Описание поведения системы**

Описание динамики системы, или, проще говоря, ее поведения, составляет основу любой имитационной модели. В качестве исходных посылок для решения этой задачи используются результаты, полученные на этапе разработки концептуальной модели системы. К ним относятся:

* определение принадлежности моделируемой системы одному из известных классов;
* описание рабочей нагрузки системы;
* выбор уровня детализации представления системы в модели и ее декомпозиция.

Все последующие действия исследователя по созданию модели могут быть отнесены к этапу ее формализации, который в общем случае предполагает:

* выбор метода отображения динамики системы (на основе событий, процессов или транзактов);
* формальное (математическое) описание случайных факторов, подлежащих учету в модели;
* выбор механизма изменения и масштаба модельного времени.

**Работа (активность)** — это единичное действие системы по обработке (преобразованию) входных данных. В зависимости от природы моделируемой системы под входными данными могут пониматься информационные данные или какие-либо материальные ресурсы. Каждая из работ характеризуется временем выполнения и потребляемыми ресурсами.

Под **процессом** понимают логически связанный набор работ. Некоторые процессы могут рассматриваться, в свою очередь, как работы в процессе более высокого уровня. Процесс характеризуется совокупностью статических и динамических характеристик.

**К статическим** характеристикам процесса относятся:

* длительность;
* результат;
* потребляемые ресурсы;
* условия запуска (активизации);
* условия останова (прерывания).

В общем случае статические характеристики процесса не изменяются в ходе его реализации, однако, при необходимости любая из них может быть представлена в модели как случайная величина, распределенная по заданному закону.

**Динамической характеристикой** процесса является его состояние (активен или находится в состоянии ожидания).

Моделирование в терминах процессов производится в тех случаях, когда система оценивается по каким-либо временным показателям, либо с точки зрения потребляемых ресурсов.

Например, при оценке производительности вычислительной сети обработка заданий может быть представлена в модели как совокупность соответствующих процессов, использующих ресурсы сети (оперативную память, пространство на жестких дисках, процессорное время, принтеры и т. д.).

В том случае, если модель строится с целью изучения причинно-следственных связей, присущих системе, динамику системы целесообразно описывать в терминах событий.

**Событие** представляет собой мгновенное изменение некоторого элемента системы или состояния системы в целом.

Событие характеризуется:

* условиями (или законом) возникновения;
* типом, который определяет порядок обработки (дисциплину обслуживания) данного события;
* нулевой длительностью.

Обычно события подразделяют на две категории:

**события следования,** которые управляют инициализацией процессов (или

отдельных работ внутри процесса);

**события изменения состояний** (элементов системы или системы в целом).

Как было отмечено, механизм событий используется в качестве основы построения моделей, предназначенных для исследования причинно-следственных связей в системах при отсутствии временных ограничений. К таким задачам можно отнести, например, некоторые задачи по оценке надежности.

Еще один способ имитационного моделирования систем основан на использовании понятия транзакта.

**Транзакт** — это некоторое сообщение (заявка, на обслуживание), которое поступает извне на вход системы и подлежит обработке. В некоторых случаях, например, при моделировании автоматизированных систем управления, более удобно проследить функционирование системы именно относительно алгоритма обработки транзакта. В рамках одной ИМ могут рассматриваться транзакты нескольких типов. Каждый транзакт характеризуется соответствующим алгоритмом обработки и необходимыми для его реализации ресурсами системы. Учитывая это, прохождение транзакта по системе можно в некоторых случаях рассматривать как последовательную активизацию процессов, реализующих его обработку («обслуживание заявки»).

В связи с упоминанием термина «обслуживание заявки» уместно вспомнить о существовании теории массового обслуживания. При разработке и исследовании имитационных моделей на основе транзактов целесообразно использовать методику и показатели, применяемые при анализе систем массового обслуживания.

**(С) БГУИР**

# 7) Управление модельным временем

**Тема 5. УПРАВЛЕНИЕ МОДЕЛЬНЫМ ВРЕМЕНЕМ**

**5.1. Механизм управления модельным временем**

Приступая к изучению механизмов управления модельным временем, уместно поговорить о роли времени в имитационном моделировании. Ранее было отмечено, что имитационное моделирование представляет собой наблюдение за поведением системы в течение некоторого промежутка времени. Конечно, далеко не во всех статистических испытаниях фактор времени играет ведущую роль, а в некоторых и вообще может не рассматриваться. Но значительно больше таких задач, в которых оценка эффективности моделируемой системы напрямую связана с временными характеристиками ее функционирования. К ним относятся задачи по оценке производительности, некоторые задачи по оценке надежности, качества распределения ресурсов, а также все задачи, связанные с исследованием эффективности процессов обслуживания. Характерной особенностью большинства практических задач является то, что скорость протекания рассматриваемых в них процессов значительно ниже скорости реализации модельного эксперимента. Например, если моделируется работа авторемонтной мастерской в течение недели, вряд ли кому-то придет в голову воспроизводить этот процесс в модели в таком же масштабе времени. А в ряде задач требуется именно реализация реального масштаба времени.

При разработке практически любой имитационной модели и планировании проведения модельных экспериментов необходимо соотносить между собой три представления времени:

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

С помощью механизма модельного времени решаются следующие задачи:

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

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

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

Ранее были названы два метода реализации механизма модельного времени — с постоянным шагом и по особым состояниям.

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

При использовании метода **постоянного шага** отсчет системного времени ведется через фиксированные, выбранные исследователем интервалы времени. События в модели считаются наступившими в момент окончания этого интервала. Погрешность в измерении временных характеристик системы в этом случае зависит от величины шага моделирования Δt.

Метод постоянного шага предпочтительнее, если:

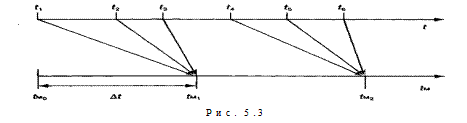
* события появляются регулярно, их распределение во времени достаточно равномерно;
* число событий велико и моменты их появления близки;
* невозможно заранее определить моменты появления событий.

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

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

Выбор величины шага моделирования является нелегким и очень важным делом. Универсальной методики решения этой проблемы не существует, но во многих случаях можно использовать один из следующих подходов:

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



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

Метод моделирования по особым состояниям сложнее в реализации, так как для него требуется разработка специальной процедуры планирования событий (так называемого календаря событий).

Моделирование по особым состояниям целесообразно использовать, если:

* события распределяются во времени неравномерно или интервалы между ними велики;
* предъявляются повышенные требования к точности определения взаимного положения событий во времени;
* необходимо реализовать квазипараллельную обработку одновременных событий.

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

Таким образом:

· Выбор механизма изменения модельного времени определяет и технологию реализации имитационной модели.

· На выбор метода моделирования влияет целый ряд факторов, однако определяющим является тип моделирующей системы: для дискретных систем, события в которых распределены во времени неравномерно, более удобным является изменение модельного времени по особым состояниям.

Если в модели должны быть представлены компоненты реальной системы, работа которых измеряется в разных единицах времени, то они должны быть предварительно приведены к единому масштабу.

**(С) БГУИР**

# 8) Моделирование параллельных процессов

**Тема 5. МОДЕЛИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ**

**6.1. Моделирование параллельных процессов**

Практически любая более или менее сложная система имеет в своем составе компоненты, работающие одновременно, или, как принято говорить на языке техники, параллельно. Параллельно работающие подсистемы могут взаимодействовать самым различным образом, либо вообще работать независимо друг от друга. Способ взаимодействия подсистем определяет вид параллельных процессов, протекающих в системе. В свою очередь, вид моделируемых процессов влияет на выбор метода их имитации.

**6.1.1.Виды параллельных процессов в сложных системах**

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

Пример асинхронных ПП из области вычислительной техники: выполнение вычислений процессором и вывод информации на печать.

**Синхронный ПП** — такой процесс, состояние которого зависит от состояния взаимодействующих с ним ПП.

Пример синхронного ПП: работа торговой организации и доставка товара со склада (нет товара — нет торговли).

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

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

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

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

Остановимся несколько подробнее на особенностях реализации параллельных процессов в вычислительных системах (ВС). Это обусловлено следующей причиной.

Разработка и использование любой ИМ предполагает ее программную реализацию и исследование с применением ВС. Поэтому для реализации моделей, имитирующих параллельные процессы, в некоторых случаях применимы механизмы, используемые при выполнении параллельных вычислений.

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

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

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

* на основе «взаимного исключения»;
* на основе синхронизации посредством сигналов;
* на основе обмена информацией (сообщениями).

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

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

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

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

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

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

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

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

**6.1.2. Методы описания параллельных процессов в системах и языках моделирования**

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

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

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

Современные языки и системы моделирования, ориентированные на использование в среде многозадачных операционных систем типа Windows, частично используют их механизмы управления процессами, что делает их применение еще более эффективным. В пакете MATLAB также имеется собственный язык моделирования, и к нему в полной мере можно отнести сказанное выше. Тем не менее во многих случаях оказывается полезным знание общего механизма реализации ПП в языках моделирования.

Рассмотрим его применительно к моделированию на основе тракзактов.

В этом случае под событием понимается любое перемещение транзакта по системе, а также изменение его состояния (обслуживается, заблокирован и т. д.).

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

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

**Список будущих событий.** Этот список содержит события, время наступления которых больше текущего модельного времени, то есть события, которые должны произойти в будущем (условия наступления которых уже определены — например, известно, что транзакт будет обслуживаться некоторым устройством 10 единиц времени).

**Список прерываний.** Данный список содержит события, связанные с возобновлением обработки прерванных транзактов. События из этого списка выбираются в том случае, если сняты условия прерывания.

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

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

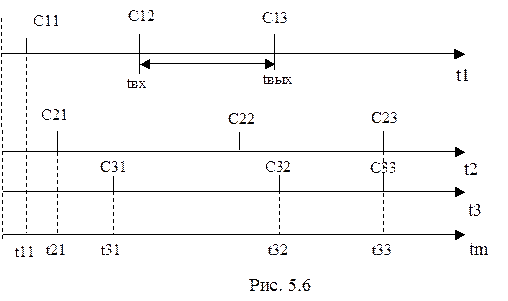
Как только завершается обработка (продвижение) очередного активного транзакта, просматривается список задержанных транзактов, и ряд из них переводится в активное состояние. Процедура повторяется до тех пор, пока в списке текущих событий не будут обработаны все активные события. После этого просматривается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

В качестве иллюстрации к изложенному рассмотрим небольшой пример.

*►*Пусть в систему поступают транзакты трех типов, каждый из которых обслуживается отдельным устройством. Известны законы поступления транзактов в систему и длительность их обслуживания. Таким образом, в системе существуют три параллельных независимых процесса (PI, P2, РЗ).

Временная диаграмма работы системы при обслуживании одного транзакта каждого типа показана на рис.2.7.

На рисунке события, относящиеся к процессу Р1, обозначены как С1i, относящиеся к Р2 и к РЗ — соответственно как С2i. и СЗi. Моменты времени tвх и tвых соответствуют началу и окончанию обслуживания транзакта.



Для каждого процесса строится своя цепь событий, однако списки событий являются общими для всей модели. Формирование списков начинается с заполнения списка будущих событий. Как было отмечено выше, в этот список помещаются события, время наступления которых превышает текущее значение модельного времени. Очевидно, что на момент заполнения списка время наступления прогнозируемых событий должно быть известно. На первом шаге tм=0, и в список будущих событий заносятся события С11, С21, С31. Затем событие с наименьшим временем наступления — С11 — переносится в список текущих событий; если одновременных с ним событий нет, то оно обрабатывается и исключается из списка текущих событий. После этого вновь корректируется список будущих событий и т.д., пока не истечет заданный интервал моделирования.

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

|  |  |  |
| --- | --- | --- |
| ***t*** | ***Список текущих событий*** | ***Список будущих событий*** |
| ***0*** | *0* | *C11, C21,C31* |

|  |  |  |
| --- | --- | --- |
| ***t11*** | *C11* | *C21, C31, C12* |
| ***t21*** | *C21* | *C31, C12, C22* |
| ***t31*** | *C31* | *C12, C22, C32* |
| ***t12*** | *C12* | *C22, C32, C13* |
| ***t22*** | *C22* | *C32, C13, C23* |
| ***t32*** | *C32* | *C13, C23, C33* |
| ***t13*** | *C13* | *C23, C33* |
| ***t23*** | *C23, C33* |  |

Многие авторы книг по имитационному моделированию считают, что знание механизма ведения списков событий просто необходимо разработчику модели; умение проследить в динамике цепь происходящих в модели событий, во-первых, повышает уверенность создателя модели в том, что она работает правильно и, во вторых, существенно облегчает процесс отладки и модификации модели.

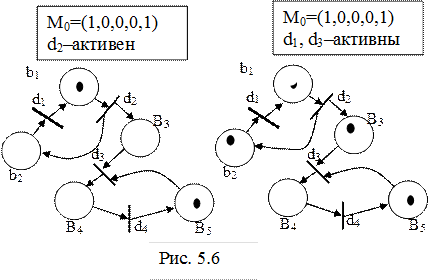
**6.1.3. Применение сетевых моделей для описания параллельных процессов**

Этапу программной реализации модели (т. е. ее описанию на одном из языков программирования) должен предшествовать так называемый этап алгоритмизации. Другими словами, прежде чем превратить имитационную модель в работающую программу, ее создатель должен воспользоваться каким-то менее формальным и более наглядным средством описания логики работы будущей программы. Это требование не является обязательным, т.к. при наличии достаточного опыта программа не очень сложной модели может быть написана сразу. Однако при моделировании более сложных систем даже опытные разработчики бывают вынуждены немного «притормозить» на этапе алгоритмизации. Для описания логики работы модели могут быть использованы различные средства: либо русский язык (устный или письменный), либо традиционные схемы алгоритмов, либо какие-то другие «подручные» средства. Первые два варианта являются, как правило, наиболее знакомыми и наиболее часто используемыми. Однако такие схемы совершенно не приспособлены для описания параллельных процессов.

Одним из наиболее элегантных и весьма распространенных средств описания параллельных процессов — описание **сетями Петри.** Рассмотрим те основные сведения, которые необходимы с точки зрения реализации технологии имитационного моделирования параллельных процессов.

Одно из основных достоинств аппарата сетей Петри заключается в том, что они могут быть представлены как в графической форме (что обеспечивает наглядность), так и в аналитической (что позволяет автоматизировать процесс их анализа).

При графической интерпретации сеть Петри представляет собой граф особого вида, состоящий из вершин двух типов — позиций и переходов, соединенных ориентированными дугами, причем каждая дуга может связывать лишь разнотипные вершины (позицию с переходом или переход с позицией). Вершины-позиции обозначаются кружками, вершины-переходы — черточками. С содержательной точки зрения, переходы соответствуют событиям, присущим исследуемой системе, а позиции — условиям их возникновения. Таким образом, совокупность переходов, позиций и дуг позволяет описать причинно-следственные связи, присущие системе, но в статике. Чтобы сеть Петри «ожила», вводят еще один вид объектов сети — так называемые фишки, или метки позиций. Переход считается активным (событие может произойти), если в каждой его входной позиции есть хотя бы одна фишка. Расположение фишек в позициях сети называется **разметкой сети** (пример перемещения фишек по сети приведен на рис.5.6).



В аналитической форме сеть Петри может быть представлена следующим образом:

P=(B,D,I,0,M),

где **В =** {b i } — конечное непустое множество позиций;

**D** = **{di}** — конечное непустое множество переходов;

**I : BхD ->** 0,1 — входная функция (прямая функция инцидентности), которая для каждого перехода задает множество его входных позиций;

**О : DxB** **->** 0,1 — выходная функция (обратная функция инцидентности), которая для каждого перехода задает множество его входных позиций;

**М** — функция разметки сети, **М : В ->** 0, 1, 2,... — ставит каждой позиции сети в соответствие неотрицательное целое число.

С учетом введенных обозначений необходимое условие срабатывания перехода dj может быть записано следующим образом:

**∀ bi ∈ I (di) {M(bi) ≥ 1}**

(для всех входных позиций разметка должна быть >1).

Срабатывание перехода dj изменяет разметку сети М(В) на разметку М’(В) по следующему правилу:

**M’(B)=M(B)-I(dj)+O(dj),**

то есть переход dj изымает по одной метке из каждой своей входной позиции и добавляет по одной метке в каждую из выходных позиций. Смену разметки обозначают так:

**dj**

**M o⎪⎯M’**

Входная и выходная функции сети Петри (**I** и **О**) позволяют описать любую сеть с помощью двух матриц размера т х п (матриц входных и выходных позиций), имеющих следующую структуру:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | d1 | d2 | ... | dj | ... | dn |
| b1 | 0 | 1 | ... | 0 | ... | 0 |
| b2 | 1 | 1 | ... | 0 | ... | 1 |
| ... | ... | ... | ... | ... | ... | ... |
| bj | 0 | 1 | ... | 0 | ... | 1 |
| ... | ... | ... | ... | ... | ... | ... |
| bm | 1 | 0 | ... | 1 | ... | 0 |

Основные направления анализа сети Петри следующие:

1. Проблема достижимости: в сети Петри с начальной разметкой М 0 требуется определить, достижима ли принципиально некоторая разметка М' из M 0 .

С точки зрения исследования моделируемой системы, эта проблема интерпретируется как проблема достижимости (реализуемости) некоторого состояния системы.

1. Свойство живости. Под живостью перехода понимают возможность его срабатывания в данной сети при начальной разметке М 0 . Анализ модели на свойство живости позволяет выявить невозможные состояния в моделируемой системе (например, неисполняемые ветви в программе).
2. Безопасность сети. Безопасной является такая сеть Петри, в которой ни при каких условиях не может появиться более одной метки в каждой из позиций. Для исследуемой системы это означает возможность функционирования ее в стационарном режиме. На основе анализа данного свойства могут быть определены требования к буферным накопителям в системе.

Итак, достоинства сетей Петри заключаются в том, что они:

1. позволяют моделировать ПП всех возможных типов с учетом вероятных конфликтов между ними;
2. обладают наглядностью и обеспечивают возможность автоматизированного анализа;
3. позволяют переходить от одного уровня детализации описания системы к другому (за счет раскрытия/закрытия переходов).

Вместе с тем, сети Петри имеют ряд недостатков, ограничивающих их возможности. Основной из них — время срабатывания перехода считается равным 0, что не позволяет исследовать с помощью сетей Петри временные характеристики моделируемых систем.

В результате развития аппарата сетей Петри был разработан ряд расширений. Одно из наиболее мощных — так называемые .Е-сети (evaluation — «вычисления», «оценка») — «оценочные сети».

В отличие от сетей Петри, в E-сетях:

1. имеются несколько типов вершин-позиций: простые позиции, позиции-очереди, разрешающие позиции;
2. фишки (метки) могут снабжаться набором признаков (атрибутов);
3. с каждым переходом может быть связана ненулевая задержка и функция преобразования атрибутов фишек;
4. введены дополнительные виды вершин-переходов.
5. в любую позицию может входить не более одной дуги и выходить также не более одной.

В связи с этим любой переход может быть описан тройкой параметров:

**dj=(S,t(dj),ρ(dj)),**

где S — тип перехода, t(dj), — функция задержки, ρ(dj) — функция преобразования атрибутов.

Особенности E-сетей существенно расширяют их возможности для моделирования дискретных систем вообще и параллельных процессов в частности. Технология моделирования систем в виде Е-сетей может быть реализована с помощью инструмента SIMULINK, входящего в состав па

**(С) БГУИР**

# 9) Моделирование случайных факторов

**Тема 4. МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ ФАКТОРОВ**

Имитационная модель позволяет исследовать поведение различных систем с учетом влияния случайных факторов. Эти факторы в зависимости от их природы могут быть отражены в модели как случайные события, случайные величины (дискретные или непрерывные), или как случайные функции (процессы).

В основе всех методов и приемов моделирования случайных факторов лежит использование базовой случайной величины -- случайных чисел, имеющих равномерное распределение на интервале [0; 1].

**4.1.Построение датчиков БСВ**

**4.1.1. Датчики БСВ**

**Базовой** случайной величиной (БСВ) в статистическом моделировании называют непрерывную случайную величину Z, равномерно распределенную на интервале (0,1). Ее плотность распределения вероятностей имеет вид:

Изображение выглядит как ночное небо

Автоматически созданное описание

Мат.ожидание СВ равно

Математическое ожидание квадрата СВ равно

И Дисперсия равна

БСВ моделируется на ЭВМ с помощью датчиков БСВ. Датчик БСВ - это устройство или программа, выдающая по запросу одно или несколько независимых значений z1, ..., zn БСВ.

Датчики БСВ могут быть трех типов: табличные, физические и программные.

**Табличный** датчик БСВ - это просто таблица случайных чисел. Основной недостаток такого датчика - ограниченное количество случайных чисел в таблицах. А в статистическом эксперименте часто требуется не ограниченное заранее их количество.

**Физический** датчик БСВ - это специальное радиоэлектронное устройство в ЭВМ, содержащее источник электронного шума. Шум преобразуется в случайные числа с распределением. Недостатки физического датчика БСВ: невозможность повторения каких-либо ранее полученных реализаций z1, ... , zn без их предварительной записи в память ЭВМ, схемная нестабильность и сложность тиражирования датчика.

**Программный** датчик БСВ обычно вычисляет значения z1, z2,..., по какой-либо рекуррентной формуле типа

zi = f ( zn),

при заданном стартовом значении z0.

Заданное значение z0 полностью определяет всю последовательность реализаций z1, z2,..., поэтому z часто называют псевдослучайной величиной. Но ее статистические свойства идентичны свойствам "чисто случайной" последовательности, что и обеспечивает успех статистического моделирования.

Программный датчик БСВ имеет следующие преимущества: простота создания датчика, простота применения, простота тиражирования, надежность, быстродействие, высокая точность достижения необходимых статистических свойств, сравнимая с точностью представления вещественных чисел, компактность, повторяемость, когда это нужно, любых последовательностей случайных значений без их предварительного запоминания.

В дальнейшем мы будем рассматривать только программные датчики БСВ.

Имея датчик БСВ Z, можно промоделировать любые случайные факторы: непрерывные или дискретные случайные величины (как простые, так и многомерные), случайные события, случайные процессы и поля и т.д. Для этого достаточно соответствующим образом преобразовать последовательность z1, z2, ... . Поэтому БСВ Z и называют базовой.

Теоретически в качестве базовой можно было бы взять почти любую случайную величину. Использование СВ Z с равномерным распределением обусловлено технологическими соображениями: простотой и экономичностью датчика, простотой преобразования Z в другие случайные факторы, относительной простотой тестирования датчика.

**4.1.2.Метод середины квадрата**

Метод середины квадрата предложен для получения псевдослучайных чисел Д.

фон Нейманом в 1946 г. Один из вариантов этого метода заключается в следующем.

1. Возьмем произвольное n-разрядное число.
2. Возведем полученное число в квадрат и, если необходимо, добавим к результату слева нули до 2n-разрядного числа.
3. Возьмем четыре цифры из середины 2n-разрядного в качестве нового случайного n-разрядного числа.
4. Если нужны еще случайные числа, то перейдем к пункту 2.

Например, если взять в качестве начального числа 1994, то из него получается следующая последовательность псевдослучайных чисел: 9760 2576 6357 4114 9249 5440 5936 2360 5696 4444 7491 1150 3225 4006 0480 2304 3084 5110 1121 2566 ...

Сам по себе метод середины квадрата не получил широкого распространения, так как выдает "больше чем нужно малых значений". Но открытый в нем принцип используется во многих, если не во всех, более поздних датчиках БСВ. Этот принцип состоит в вырезании нескольких цифр из результата какой-либо операции над числами.

**4.1.3. Мультипликативный конгруэнтный метод**

Так называемый мультипликативный конгруэнтный датчик БСВ задается двумя параметрами: модулем m и множителем k. Обычно это достаточно большие целые числа.

При заданных m, k числа z1, z2, ..., вычиcляются по рекуррентной формуле:

A i = (kAi-1) mod m, i = 1, 2,...,

z i = A i / m

где m - модуль,

k - множитель,

A0 - начальное значение,

mod - операция вычисления остатка от деления kAi -1 на m.

Таким образом, A1 определяется как остаток от деления kA0 на m; A2 - как остаток от деления kA1 на m и т.д. Поскольку все числа Ai - это остатки от деления на m, то 0 £ Ai < m. Разделив последнее неравенство на m, видим, что 0 £ Ai / m< 1, т. е. 0 £ zi <1.

Из неравенства 0 £ Ai < m вытекает также, что датчик (6.1) дает периодическую последовательность Ai. Действительно, число всех возможных остатков от 0 до m -

1 равно m и, рано или поздно, на каком-то шаге i обязательно появится значение Ai, уже встречавшееся ранее. С этого момента последовательность Ai “зациклится".

Длина периода T будет не больше m - 1. Например, если встретится остаток

Ai= 0, то далее, согласно (4.1), будет Ai+ 1 = 0, Ai+ 2 = 0, ... , т.е. длина периода T = 1. Ненулевых же остатков в интервале 0£ Ai < m всего m - 1, и, если все они войдут в период, будет T = m - 1. Это имеет место, например, при m = 13, k = 7; в этом случае ряд Ai выглядит так:

1, 7, 10, 5, 9, 11, 12, 6, 3, 8, 4, 2, 1, 7,... .

\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

T = m - 1 = 12

Поскольку в качестве случайной можно использовать лишь подпоследовательность Ai внутри одного периода, то параметры датчика выбирают так, чтобы длина периода T была максимальной. С учетом ограничения T£ m - 1 модуль m берут максимально возможным. Чтобы упростить вычисление остатков по (6.1), для двоичных ЭВМ часто берут m = 2n. Рекомендуется также брать достаточно большой множитель k, причем взаимно простой с m.

В можно найти подробные рекомендации по выбору параметров m, k и начального значения A0 . Заметим, однако, что в настоящее время не известны правила, которые гарантировали бы высокое качество датчика без его специального статистического тестирования.

Датчик (4.1) называют мультипликативно-конгруэнтным потому, что он использует две основные операции - умножение (англ. multiplication) и вычисление остатка (в теории чисел - получение конгруэнтного числа). Можно было бы поэтому перевести его название и как "множительно-остатковый датчик".

В качестве примера рассмотрим таблицу параметров датчиков, предлагаемых в некоторых публикациях и программных продуктах.

|  |  |  |
| --- | --- | --- |
| Место использования датчика  (программный продукт или публикация) | модуль *m* | множитель  *k* |
| Язык моделирования СИМУЛА | 2*35* | 5*16* |
| Пакеты LLRANDOM, IMSL | 2*31* - 1  (простое число) | 16807 |
| Язык моделирования SIMSCRIPT | 2*31* - 1 | 63036001 |

6.2. Характеристики датчиков базовых случайных величин

Практика показывает, что результаты имитационного моделирования существенно зависят от качества используемых последовательностей псевдослучайных чисел. Поэтому используемые в имитационной модели генераторы случайных чисел должны пройти тесты на пригодность. Основные анализируемые характеристики генерируемых датчиком последовательностей:

* равномерность;
* стохастичность (случайность);
* независимость.

Рассмотрим методы проведения такого анализа, наиболее часто применяемые на практике.

**4.2.1. Тестирование равномерности**

Обозначим равномерное распределение вероятностей на интервале (0,1) через R[0,1]. Тогда утверждение, что БСВ Z имеет распределение R[0,1], можно кратко записать в виде z ~ R[0,1].

С помощью статистических тестов проверяют два свойства датчика, делающих его точной моделью идеальной БСВ, - это равномерность распределения чисел Zi, выдаваемых датчиком на интервале (0,1), и их статистическая независимость. При этом числа zi рассматривают как реализации некоторой **СВ.**, т.е. как статистическую выборку.

Достаточно простым методом проверки равномерности распределения является частотный тест. Он основан на законе больших чисел и выполняется по следующему алгоритму.

1. Разобьем интервал (0,1) на K равных отрезков (например, K = 10).
2. Сгенерируем n чисел z1,..., zn с помощью тестируемого датчика БСВ (например, n = 100).
3. Подсчитаем, сколько чисел попало в каждый из k отрезков, т.е. найдем числа попаданий n1,...,nk.

5. Рассчитаем относительные частоты попаданий в отрезки:

1. Построим гистограмму частот на K отрезках интервала (0,1).



1. Повторим действия (2) - (5) для большего значения n (например, для n =10 000).
2. Оценим по полученным гистограммам сходимость каждой частоты к вероятности p = 1/K того, что БСВ попадет в i-й отрезок. Согласно закону больших чисел должно быть

 (6.2.)

Это значит, что высоты столбиков во второй гистограмме должны в целом быть ближе к уровню 1/K, чем в первой.

Тестирование датчика на равномерность можно совместить с оцениванием **математического ожидания m\*** и дисперсии S2. Оценки **m\*** и S2 рассчитываются соответственно по формулам:

Изображение выглядит как внутренний

Автоматически созданное описание (6.3)

Изображение выглядит как внутренний, электроника

Автоматически созданное описание (6.4)

С ростом n оценки должны сходиться по вероятности к точным значениям M(z) = 1/2, D(z) = 1/12 = 0.08333... .



**6.2.2. Тестирование стохастичности**

Рассмотрим один из основных методов проверки – метод комбинаций. Суть его сводится к следующему. Выбирают достаточно большую последовательность случайных чисел xi и для нее определяют вероятность появления в каждом из xi ровно j единиц. При этом могут анализироваться как все разряды числа, так и только l старших. Теоретически закон появления j единиц в l разрядах двоичного числа может быть описан как биномиальный закон распределения (исходя из независимости отдельных разрядов).

Тогда при длине выборки N ожидаемое число появлений случайных чисел xi с j единицами в проверяемых l:



Для полученной последовательности определяется эта же характеристика. Проверка соответствия реального значения теоретическому выполняется с помощью одного из статистических критериев согласия.

**4.2.3. Тестирование независимости**

Простейшую проверку статистической независимости реализаций z1, z2, ..., можно осуществить, оценивая корреляцию между числами zi и zi+s, отстоящими друг от друга на шаг s >1.

Для вывода формулы, по которой можно рассчитать коэффициент корреляции чисел zi и zi+ s , рассмотрим две произвольные **с.в**. x, y. Коэффициент корреляции определяется для них формулой:

Изображение выглядит как текст, ночное небо

Автоматически созданное описание (6.5)

С ростом n оценка R' должна приближаться к нулю, в противном случае датчик БСВ не отвечает требованию независимости.

Конечно, если R' сходится к нулю, то это еще не гарантирует наличие независимости, но все же один из тестов оказывается успешно выдержанным. При желании всегда можно продолжить испытания датчика другими методами.

Еще одна важная характеристика датчика СЧ — **длина отрезка периодичности L**. Если в основу работы датчика положен мультипликативный метод, то оценить L несложно: она определяется величиной константы М.

**4.3. Случайные события и их имитация**

**4.3.1.Имитация случайного события**

Пусть некоторое событие А происходит с вероятностью . Требуется воспроизвести факт наступления события А. Поставим в соответствие событию А событие В, состоящее в том, что х меньше либо равно, где х здесь и в дальнейшем

– случайное число (СЧ) с равномерным на интервале (0,1) законом распределения. Вычислим вероятность события В:

Изображение выглядит как электроника

Автоматически созданное описание

Таким образом, события А и В являются равновероятными. Отсюда следует процедура имитации факта появления события А. Она сводится к проверке неравенства  меньше, либо равно Р, а алгоритм заключается в следующем:

1. С помощью датчика случайных чисел (СЧ) получают СЧ Х;
2. Проверяют выполнение неравенства Х меньше, либо равно ;
3. Если оно выполняется, то событие А – произошло, если нет – то произошло



**4.3.2. Имитация сложного события**

Имитация сложного события, состоящего, например, из двух независимых элементарных событий А и В, заключается в проверке неравенств:

Изображение выглядит как внутренний, ночное небо

Автоматически созданное описание,

где – вероятности событий А и В, а х1 и х2 – СЧ с равномерным законом распределения.

и



В зависимости от исхода проверки неравенств (аналогично алгоритму 4.2.1.) делается вывод какой из вариантов:

имеет место.

**4.3.3. Имитация сложного события, состоящего из зависимых событий.**

В случае, когда сложное событие состоит из элементарных зависимых событий А и В имитация сложного события производится с помощью проверки следующих неравенств:



В зависимости от того, какая из этих четырех систем неравенств выполняется, делается вывод о том, какой из этих четырех возможных исходов  имеет место.

В качестве исходных данных задаются Pa, Pb и условная вероятность

вероятность  может быть вычислена. По формуле полной вероятности:

,

где

, отсюда легко выразить 

**4.3.4. Имитация событий, составляющих полную группу**

Пусть событие А i (i=1,n) составляют полную группу, тогда их вероятности Р i, таковы что:



Имитация факта появления одного из событий А i (i=1,n) сводится к проверке следующих неравенств:



Выполнение К-го неравенства эквивалентно выполнению события АК. Описанный алгоритм называют иногда алгоритмом “розыгрыша по жребию”. Его можно интерпретировать как установление номера К-го отрезка длинной Р K, на который пало СЧ х, при условии разбиения отрезка единичной длины на отрезки с длинами P 1,P2,...Pn (рис 4.3.)

Изображение выглядит как петля

Автоматически созданное описание 

Рис. 4.3

**6.4. Имитация непрерывных случайных величин**

**6.4.1. Метод обратной функции**

Пусть непрерывная случайная величина Y задана своим законом распределения:

,

где f(y)– плотность распределения вероятностей, а F(y)- функция распределения вероятностей. Доказано, что случайная величина



распределена равномерно на интервале (0,1).

Отсюда следует, что искомое значение y может быть определено из уравнения:

Изображение выглядит как ночное небо

Автоматически созданное описание (6.8)

которое эквивалентно уравнению:

 (6.9)

где y – значение случайной величины Y, a x – значение СВ X. Решение уравнения (4.9) можно записать в общем виде через обратную функцию 



Основной недостаток метода заключается в том, что интеграл (4.8) не всегда является берущимся, а уравнение (4.9) не всегда решается аналитическими методами.

**4.4.2. Метод Неймана (режекции)**

Метод Неймана, так же как метод обратной функции, является методом, позволяющим получить значения СВ в соответствии с заданным законом распределения. Этот метод является достаточно универсальным он применим для моделирования всех СВ, значения которых не выходят за пределы ограниченного интервала (a,b), а также для СВ, законы распределения которых можно аппроксимировать усеченными.

Метод Неймана состоит в следующем:

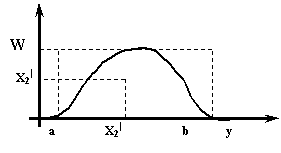
С помощью датчика случайных чисел получают пару чисел, распределенных равномерно на (0,1) x 1 и x2.

Путем преобразований (по методу обратной функции получают два числа x1\* и x2\*, равномерно распределенных соответственно на интервалах (a,b) и (o,w), то есть

и

, где





Из точек с координатами x1\* и x2\* выбирают те, которые попали “под колокол” функции f(x), то есть те точки, для которых f(x1\*)<x2\*.

Если условие выполнено, то искомое значение y полагают равным x1\*.

**4.4.3. Алгоритм получения значения нормально распределенной случайной величины.**

Нормальное распределение является наиболее часто встречающимся. Функция плотности распределения вероятностей для него имеет вид:

Изображение выглядит как ночное небо

Автоматически созданное описание

где m – математическое ожидание, а σ 2– дисперсия. Согласно центральной предельной теореме теории вероятностей



распределена асимптотически нормально, если распределены одинаково.

Для практического получения значений X в качестве и выбирают равномерно распределенные случайные величины. При этом наиболее часто используют преобразование

 (4.10)

где xi – равномерно распределенные на (0,1) случайные числа. При к=12 формула приобретает вид наиболее удобной для расчетов, но она дает достаточно точные результаты уже для k=3,4. Формула (4.10) верна для центрированной (m=0) и нормированной ( =1) случайной величины.

Для получения y\*, распределенного нормально с произвольными m и σ, пользуются дополнительно преобразованием

y\*=m+σy (4.11)

**4.5. Алгоритмы получения значений систем случайных величин (случайных векторов).**

**4.5.1. Метод аналитических преобразований.**

Пусть системы непрерывных случайных величин (x1, x2, …, xn) задана условными законами распределения xi (i=1,n). По теореме умножения плотностей распределения: совместная функция плотности распределения вероятностей

f(x1, x2, . . . xn)=f1(x1) f2(x2|x1) f3(x3|x1x2) . . . f1(xn| x1,x2, . . ., xn-1).

Для системы двух случайных величин (x1,x2), алгоритм получения вектора ее значений сводится к следующему:

Вычисление частной функции плотности для x1:

Изображение выглядит как внутренний, ночное небо

Автоматически созданное описание

Получение значения X1 в соответствии с f1(x1) согласно любому методу, например, одному из описанных в предыдущем разделе.

Вычисление частной функции плотности для второй компоненты x2 системы. Она может быть получена на основании теоремы умножения законов распределения:

Изображение выглядит как ночное небо

Автоматически созданное описание

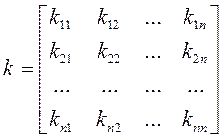
Получение x2 – значения СВ X2 любым известным методом в соответствии с найденным законом ее распределения.

Алгоритм может быть обобщен для любого n. Однако, практические работы, выполняемые по этому методу, связаны с большими вычислительными трудностями, за исключением тех редких случаев, когда интегралы берутся. Поэтому разработаны другие методы, позволяющие решать задачу получения значений системы непрерывных случайных величин.

**4.5.2. Метод разложения по координатным случайным величинам.**

Пусть СНСВ задана в рамках теории корреляций: математическими ожиданиями компонент (m1, m2, . . . mn) и матрицей корреляционных моментов:

,



Доказано, что  можно получить с помощью их разложения по координатам СВ xi:

Изображение выглядит как ночь

Автоматически созданное описание (4.12)

где xi - некоррелированные, центрированные, нормированные нормально

распределенные СВ.

Коэффициенты  могут быть достаточно просто получены решением системы уравнений:

 (4.13)

Алгоритм получения значений СНСВ сводится к следующему: q Решение системы нелинейных уравнений (4.13).

q Получение n значений yi нормированных, центрированных СВ, распределенных нормально.

q Вычисление xi i=(1,…,n) значений СВ, образующих систему непрерывных случайных величин в соответствии с (4.12).

**4.5.3. Алгоритм получения значений системы дискретных случайных величин**

Дискретный двумерный вектор CB задается двумерным законом распределения, т.е.

а) матрицей вероятностей , где P ij – вероятность

совместного появления i-ого и j-ого значений соответственной первой и второй компоненты, причем:

Изображение выглядит как внутренний, ночное небо

Автоматически созданное описание.

б) двумя векторами возможных значений первой и второй компоненты {A i}, {Bi},

Для получения значений двумерной дискретной системы случайных величин вычисляют ряд распределения и функцию распределения составляющей X:

 и F(xi)=k=0i-1PXi

и условные ряды и функции распределения составляющей Y:

PYjXi=PijPXi и FXiyj=k=0j-1PYjXi

Формирование случайной величины происходит в два этапа. На первом этапе разыгрывается значение составляющей Х. Если ξ - равномерно распределенное случайное число из интервала (0,1) такое, что

,

то считают, что X компонента двумерной дискретной случайной величины получила k-ое значение.

Сформированное значение составляющей X определяет условный закон распределения составляющей Y

FXkyj=k=0j-1PYjXk,

которая формируется аналогично составляющей X.