Глава 1

Введение

Данное пособие предназначено для помощи в выполнении лабораторных работ и курсовых проектов по курсу “Моделирование”. В нём рассмотрены вопросы:

1. Генерации псевдослучайных величин.
2. Анализа и редукции графа событий.
3. Построения графа событий.
4. Реализации монитора событий.

1.1 Общая схема моделирования при выполнении курсовых проектов

1.1.1 Этапы имитационного моделирования

В общем случае процесс имитационного моделирования включает:

1. **Формулировку задачи**, включая общее описание системы и определение ее границ.
2. **Определение целей** исследования, - определение вопросов, на которые должна ответить имитация.
3. **Построение модели**, - описание существенных (для данного исследования) свойств системы в терминах сущностей, атрибутов или характеристик каждой сущности, активностей, в которые вовлечены эти сущности и описание множества возможных состояний модели.
4. **Сбор информации** - сбор данных и информации, позволяющих разработчику модели разработку описания свойств сущностей и определения распределений вероятностей для параметров системы.
5. **Кодирование** - процесс перевода описания модели системы в программу для ЭВМ.
6. **Верификацию** - процесс подтверждения правильности функционирования программы (программа работает по своей спецификации).
7. **Валидацию** - процесс подтверждения адекватности модели (модель верно имитирует поведение моделируемой системы). Итерационные шаги валидации и верификации составляют основную часть разработки компьютерной программы.

8. **Планирование эксперимента** - определение альтернатив, которые должны быть рассмотрены в ходе имитационных экспериментов, включает выбор существенных входных переменных, определение их подходящих значений, определение длин экспериментов и их количества.
9. **Рабочие эксперименты** и анализ результатов.
10. **Документирование** программы и результатов моделирования.

Краткость курса не позволяет выполнить все этапы в полном объёме. В частности, формулировка задачи и исходные данные к моделированию входят в задание на проектирование. Также задан и способ представления модели: графы событий. Соответственно, используется событийно-ориентированный подход к моделированию.

1.1.2 Требования к оформлению

Отчёт по курсовому проекту включает в себя следующие обязательные элементы:

1. Титульный лист
2. Задание на проектирование
3. Граф событий для моделируемой системы и его анализ
4. Описание процедур обработки событий
5. Листинг программы моделирования
6. Результаты моделирования и их анализ

Глава 2

Графы событий

Графы событий (ГС) являются удобным средством описания систем с дискретными событиями. Впервые они были предложены в 1983 году Л. Шрубеном [?].

ГС состоит из вершин E_i, соответствующих событиям, и дуг U_{ij}, соответствующих причинно-следственным связям по планированию и отмене событий. В первом случае дуга обозначается сплошной линией, во втором – пунктирной (см. Рис.2.1).

Событие определяется изменением переменных состояния системы, происходящим при наступлении этого события. Множество переменных, которое может изменяться при наступлении события E_i обозначим S_i. Переменные состояния могут быть как детерминированными, так и случайными.

Связь событий может быть *условной* и занимать *время срабатывания перехода*. На Рис. 1 видно, что время срабатывания перехода обозначается как вес дуги, а условие задается его номером, стоящим в скобках возле специального значка на дуге. Время t срабатывания перехода по пунктирной дуге от события j к событию k означает, что требуется отменить *все события k, отстоящие на t и более единиц времени от текущего.*

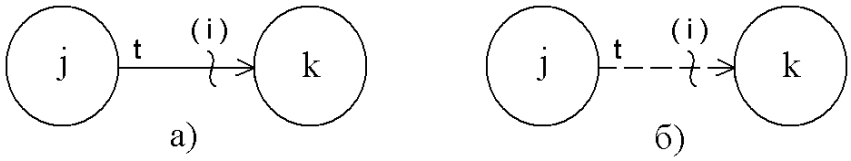

Рис. 2.1: Два типа дуг в графе событий

Введем обозначения:

O_i – множество переменных модели, которые могут быть изменены событием i, $O = \bigcup_i O_i$.

E_i – множество переменных модели, которые вовлечены в принятие решений на дугах, исходящих из вершины i, $E = \bigcup_i E_i$.

L_i – множество переменных модели, которые вовлечены в принятие решений внутри правила обработки события i, $L = \bigcup_i L_i$.

E'_i – объединение E_i и L_i, $E' = \bigcup_i E'_i$.

M_i – множество всех переменных модели, используемых событием i, $M = \bigcup_i M_i$. M_i используются в определении условий и вычислении задержек на дугах, исходящих из вершины i.

Рассмотрим простейший пример системы массового обслуживания G/G/1 (Рис. 2.2).

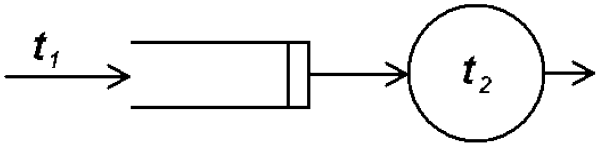

Рис. 2.2: Одноканальная СМО

Можно выделит следующие события:

1. Приход нового требования.
2. Взятие требования на обслуживание.
3. Окончание обслуживания.

Граф событий для этого примера представлен на Рис. 2.3

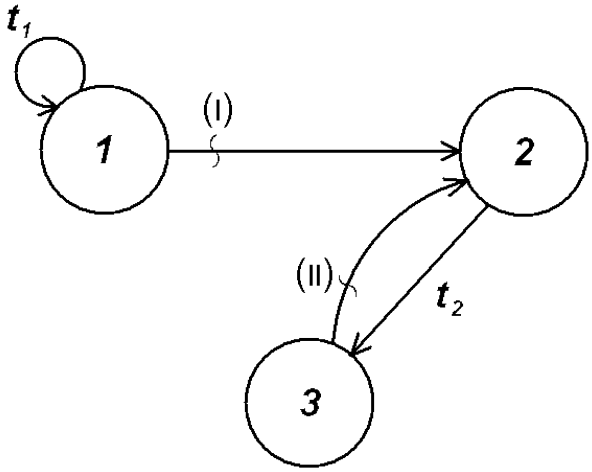

Рис. 2.3: Граф событий для одноканальной СМО

Используются следующие условия:

(I) – прибор свободен;

(II) – очередь не пуста.

Для проверки этих условий необходимо ввести следующие переменные: `Busy` признак занятости, 0 если свободен и 1 если занят; – `Lq` – текущая длина очереди. Тогда первое условие выглядит как `Busy=0`, а второе как `Lq>0`. Соответственно, имеем E_1 =`Busy`, $E_2 = \emptyset$ и E_3 =`Lq`. Эти переменные меняются при наступлении событий следующим образом. При наступлении 1-го события `Lq` увеличивается на единицу, при наступлении 2-го `Lq` уменьшается на единицу и `Busy` принимает значение 1, и при наступлении 3-го `Busy` принимает значение 0. Таким образом, $O_1 = \{$`Lq`$\}$, $O_2 = \{$`Busy,Lq`$\}$ и $O_3 = \{$`Busy`$\}$.

2.0.1 Применение графов событий

Графы событий позволяют ответить на следующие вопросы:

1. какое подмножество переменных состояния абсолютно необходимо для функционирования модели;
2. присутствуют ли в модели неосуществимые события;
3. какие события должны быть запланированны к моменту запуска модели;
4. можно ли уменьшить количество событий и, соответственно, процедур их обработки.

Рассмотрим по пунктам.

Минимальный набор переменных

Правило 1. Для осуществления правильного перехода от состояния к состоянию достаточно ввести и описать переменные из $E \cup L$. В самом деле, для осуществления безусловных переходов достаточно знать значения задержек на дугах, тогда как верные условные переходы невозможны без знания значений переменных, входящих в соответствующие условия, также как невозможно верное вычисление переменных состояния из O_i без знания значений переменных из L_i. Естественно, при написании имитационной модели могут понадобиться и другие переменные, как для промежуточных вычислений, так и для сбора необходимых для ответов на поставленные вопросы статистик.

Минимальный набор событий, которые необходимо запланировать ДО запуска модели и возможные неосуществимые события

Напомним, что в теории графов *сильносвязной компонентой* ориентированного графа называется подграф, в котором из каждой вершины в каждую существует хотябы один путь.

Удалим из графа событий все пунктирные дуги (дуги отмены событий). Тогда справедиво следующее

Правило 2. Для осуществления правильного функционирования модели необходимо, чтобы хотя бы одно событие в каждой сильносвязной компоненте, не имеющей входных дуг, было запланированно до запуска модели.

Это правило достаточно очевидно. В самом деле, если в компонену нет входных дуг, то возможны лишь внутренние переходы между событиями этой компоненты, управление не может прийти извне. Это означает необходимость начального “толчка"внутри самой компоненты. Если же события в компоненте таковы, что невозможно заранее определить момент наступления ни одного из них, это говорит о *недостижимости* событий этой компоненты и, следовательно, о неверной логике модели.

Приоритеты событий

При выполнении имитационной программы, определяемой некоторым графом событий, может возникнуть ситуация, когда несколько событий должны произойти в один и тот же момент системного времени. Иногда порядок вызова соответствующих процедур безразличен, а иногда разный порядок выполнения даёт различное поведение модели. Для разрешения этой проблемы существует следующее

Правило 3. Если пересечение множеств O_k и E_j непусто, то необходимо установить отношение приоритета для событий k и j.

Редукция графа событий

Будем называть два графа событий эквивалентными, если они при одних и тех же начальных условиях порождают одну и ту же фазовую траекторию в пространстве состояний, т.е. одни и те же изменения переменных состояния во времени. Рассмотрим, когда возможно уменьшить получить эквивалентный граф событий с меньшим числом вершин, уменьшив тем самым число процедур обработки событий.

Правило 4. Эквивалентные графы событий возможны с или без вершины k если эта вершина не имеет исходящих условных дуг и если верно одно из следующих условий:

1. все дуги, входящие в вершину k, имеют нулевое время задержки;
2. O_k не содержит переменных, входящих в условия на дугах;
3. все дуги, исходящие из вершины k, имеют нулевое время задержки.

При выполнении первого условия вершина k может быть объединена с одной из вершин, начальных для рёбер, входящих в неё. Изменения переменных состояния, происходившие в вершине k теперь добавляются к изменениям, проводимым в этой исходящей вершине. Затем вершина k удаляется из графа. Отметим, что в случае применения этого варианта редукции требование на отсутствие исходящих условных дуг не является необходимым.

Для приведённого выше примера (Рис. 2.3) имеем что, согласно условию 1, вершина 3 может быть объединена с вершиной 2. Соответственно, имеем редуцированный граф событий, представленный на Рис. 2.4.

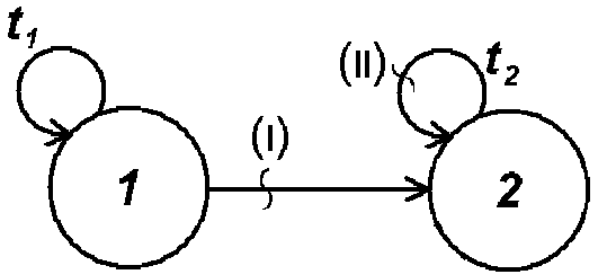

Рис. 2.4: Редуцированный граф событий для примера на Рис. 2.2

Глава 3

Функции и реализация монитора событий

Для реализации программы моделирования необходимо иметь следующие процедуры (в скобках указаны общепринятые, но не обязательные названия):

планирования события (`Schedule(E,t)`) — планирует событие E на момент времени t;

отмены события (`Cancel(E,t)`) — отменяет событие E для всех времён больше или равных t;

запуска модели (`Simulate`) — в бесконечном цикле запускает обработчик ближайшего по времени события.

Реализовать обработку события можно с помощью процедуры (F) или активной фазы программного процесса (P).

Основой для реализации последовательности событий является *календарь событий*. Календарем событий или *управляющим списком* L_c будем называть упорядоченный по модельному (системному) времени список *уведомлений о событиях*, где уведомление о событии есть тройка $E_{time} =< time, F_i[P_i], next >$, где $next$ есть ссылка на следующее в списке уведомление о событии (Рис. 3.1). Далее, для удобства, если не требуется делать специального различия, вместо $F_i[P_i]$ будем использовать обозначение FP_i для обработчика события. На практике, что и нашло отражение на рисунке, уведомление о событии содержит ряд дополнительных полей (например, текстовое имя события), которые могут использоваться для отладки (режим трассировки событий) или более эффективного поиска по календарю.

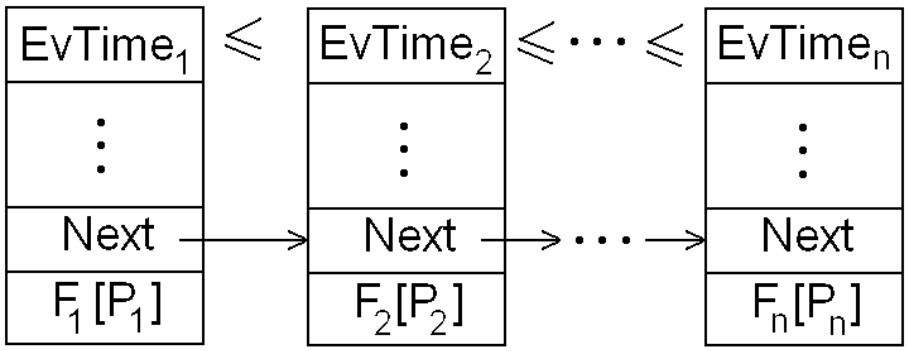

Рис. 3.1: Стандартная структура календаря событий (управляющего списка)

Управляющий алгоритм системы моделирования заключается в последовательном вызове обработчиков событий FP_i с одновременной сменой значения системной переменной **время** на $time_i$.

При выполнении курсовых работ рекомендуется использовать для обработки событий процедуры, так как это проще для небольших моделей, предлагаемых студентам.

3.1 Реализация процедур обработки событий

В общем случае все программы обработки событий имеют одинаковую структуру и состоят из двух частей: части изменения состояния модели (изменяются элементы множества O_i) и части управления событиями, в которой содержатся операторы планирования и отмены событий. Вторая часть в точности соответствует исходящим дугам для соответствующей событию вершины графа событий. Имеем следующие варианты:

Вид дуги	Оператор
сплошная, без условия, время задержки t	`Schedule(<След. событие>,Time+t);`
сплошная, с условием C, время задержки t	`if C {Schedule(<След.событие>,Time+t);}`
пунктирная, без условия, время задержки t	`Cancel(<След.событие>,Time+t);`
пунктирная, с условием C, время задержки t	`if C {Cancel(<След.событие>,Time+t);}`

Глава 4

Датчики случайных чисел

При описании поведения, а иногда и структур сложных систем часто приходится иметь дело со случайными объектами, в частном случае случайными числами и процессами. При их моделировании на ЭВМ, если не использовать специальные устройства, используются алгоритмы получения последовательностей величин которые, не являясь на самом деле случайными, *ведут себя как случайные*. Такие величины или объекты называются псевдослучайными.

В основе всех алгоритмов получения псевдослучайных объектов лежит использование так называемого *базового датчика случайных чисел (д.с.ч.)*, то есть генератора псевдослучайных чисел, равномерно распределенных на отрезке $[0,1]$ (реально, как правило, $[0,1)$). Далее такие числа обозначаются как ξ или ξ_i, в зависимости от контекста.

4.1 Генерация независимых, одинаково распределенных случайных величин

4.1.1 Непрерывные распределения

Существует несколько методов генерации независимых с.в. с заданным законом распределения. Наиболее точные из них основаны на преобразовании с.в. Так, большое количество датчиков получается исходя из известного результата о равномерном на $[0,1)$ распределении функции $F_\eta(\eta)$, где η - произвольная непрерывная случайная величина с функцией распределения $F_\eta(x)$.

Пример 2.1. Требуется получить датчик экспоненциально распределенных с.в. В этом случае

$$F_\eta(x) = 1 - e^{-\lambda x} \tag{4.1}$$

Тогда из решения уравнения

$$\xi = 1 - e^{-\lambda \eta} \tag{4.2}$$

получаем $\eta = -\frac{\log(1-\xi)}{\lambda}$.

Можно заметить, что выражение под знаком логарифма в последней формуле имеет равномерное распределение на отрезке $[0,1)$, что позволяет получать другую, но так же распределённую последовательность по формуле: $\eta = -\frac{\log \xi}{\lambda}$.

Генератор нормальных с.в.

Нормально распределенная с.в. ν имеет функцию плотности распределения вероятности

$$\phi(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{(-\frac{x-a}{\sigma})^2}. \tag{4.3}$$

Связанная с нею нормализованная величина $u = \frac{\nu - a}{\sigma}$ имеет так называемое стандартное нормальное распределение с нулевым математическим ожиданием и единичной дисперсией. Далее речь идет о генерации именно таких с.в.

Для генерации нормально распределенных с.в. метод обратной функции напрямую не применим, поскольку для обратной функции нет аналитического выражения. Возможно приблизить обратную функцию по известным табличным значениям, но достижение хорошей точности потребует достаточно высокой степени аппроксимирующих полиномов. На практике используют кусочную аппроксимацию (три участка) дробно-полиномиальной функцией или используют другие методы. Наиболее простым из точных является метод полярных координат, при котором с использованием пары значений базового датчика получают пару независимых нормально распределенных с.в. Соответствующий алгоритм имеет следующий вид [?]:

Шаг 1. Получить два независимых случайных числа ξ_1 и ξ_2. Положим $v_1 = 2\xi_1 - 1$ и $v_2 = 2\xi_2 - 1$, эти величины равномерно распределены на (-1,1). Величины v_1 и v_2 лучше представить в машине в форме с плавающей запятой.

Шаг 2. Вычислить $S = v_1^2 + v_2^2$.

Шаг 3. Если $S \geq 1$ то повторить шаг 1.

Шаг 4. Присвоить величинам u_1 и u_2 значения, вычисленные по формулам

$$u_1 = v_1\sqrt{\frac{-2\log S}{S}} \quad u_2 = v_2\sqrt{\frac{-2\log S}{S}} \tag{4.4}$$

Другие датчики могут основываться на известных соотношениях между распределениями.

Пример 2.2. Генератор логарифмически нормальных с.в.

По определению, логарифмически нормальной с.в. называется с.в., логарифм которой распределен по нормальному распределению. Соответственно, если иметь генератор независимых нормально распределенных с.в. u(см. выше), требуемый генератор получается по формуле $\eta = e^u$.

Для построения приближенных генераторов можно использовать предельные теоремы теории вероятностей.

Пример 2.3. Генератор нормальных с.в.

Центральная предельная теорема теории вероятностей гласит "распределение суммы независимых, одинаково распределенных случайных величин сходится к нормальному с математическим ожиданием, равным сумме математических ожиданий и дисперсией, равной сумме дисперсий".

Соответственно, при достаточно большом n сумма $\sum_{i=1}^{n} \xi_i$ имеет нормальное распределение. Поскольку $E[\xi] = \frac{1}{2}$ и $D[\xi] = \frac{1}{12}$, удобно ограничить сумму двенадцатью слагаемыми:

$$\tilde{u} = \sum_{i=1}^{12} \xi_i - 6. \tag{4.5}$$

$\tilde{u}$ распределена на интервале (-6,6) и абсолютное отклонение функции плотности вероятностей от истиной не превышает 2%.

Метод отбраковки

В некоторых случаях требуется точное соответствие заданному закону распределения при отсутствии эффективных методов генерации. В такой ситуации для ограниченных с.в. можно использовать следующий метод. Функция плотности распределения вероятностей с.в. $f_\eta(x)$ вписывается в прямоугольник $(a, b) \times (0, c)$, такой, что a и b соответствуют границам диапазона изменения с.в. η, а c – максимальному значению функции плотности её распределения. Тогда очередная реализация с.в. определяется по следующему алгоритму:

Шаг 1. Получить два независимых случайных числа ξ_1 и ξ_2.

Шаг 2. Если $f_\eta(a+(b-a)\xi_1) > c\xi_2$ то выдать $a+(b-a)\xi_1$ в качестве результата. Иначе повторить Шаг 1.

Неэффективность алгоритма для распределений с длинными "хвостами" очевидна (см. Рис. **??**), поскольку в этом случае часты повторные испытания.

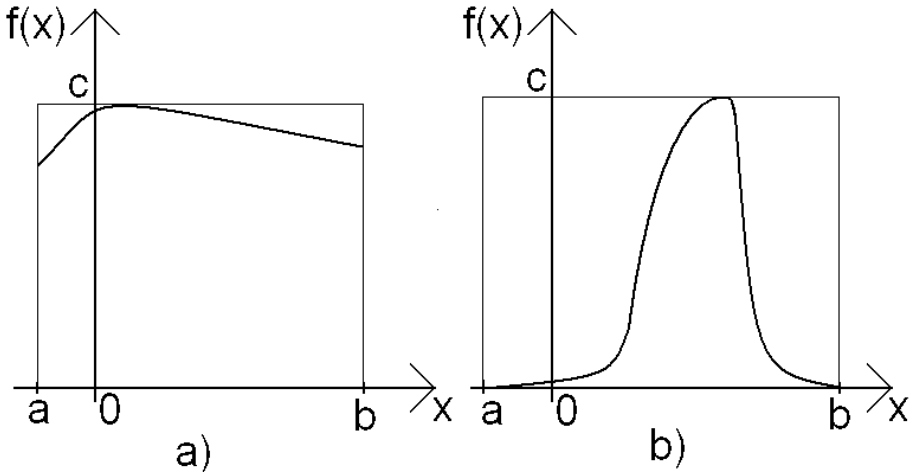

Рис. 4.1: Случаи эффективного(a) и не эффективного (b) применения метода отбраковки

Случай кусочно заданной плотности распределения

Пусть случайная величина η определена на интервале $[a,b)$ и её плотность имеет вид (4.6)

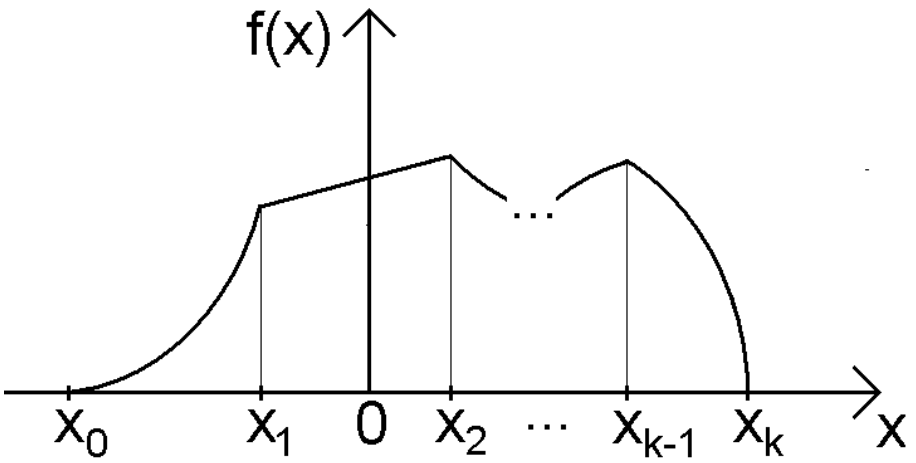

Рис. 4.2: Кусочно определённая плотность распределения вероятностей

$$f(x) = \begin{cases} f_1(x), & \text{при } a = x_0 \le x < x_1, \\ f_2(x), & \text{при } x_1 \le x < x_2, \\ \dots & \\ f_k(x), & \text{при } x_{k-1} \le x < x_k = b, \end{cases} \tag{4.6}$$

где $\forall i \in (0,\dots,k-1) x_i < x_{i+1}$. Обозначив $p_i = \int\limits_{x_{i-1}}^{x_i} f_i(x)dx$ имеем $\sum\limits_{i=1}^{k} p_i = 1$.

Для получения значения η необходимо *сначала* выбрать плотность $f_i(x)$ с вероятностью p_i и *затем* выбрать случайное значение на интервале $[x_{i-1}, x_i)$ согласно выбранному распределению.

Использование линейных преобразований

Часто являются полезными следующие очевидные выражения для плотностей линейных функций случайной величины:

$$\eta = a\nu + b \longrightarrow f_\eta(x) = f_\nu(\frac{x-b}{a}), \tag{4.7}$$

$$\eta = -\nu \longrightarrow f_\eta(x) = f_\nu(-x). \tag{4.8}$$

Пример 2.4. Генератор треугольного распределения Пусть плотность распределения имеет вид, представленный на рисунке 4.3.

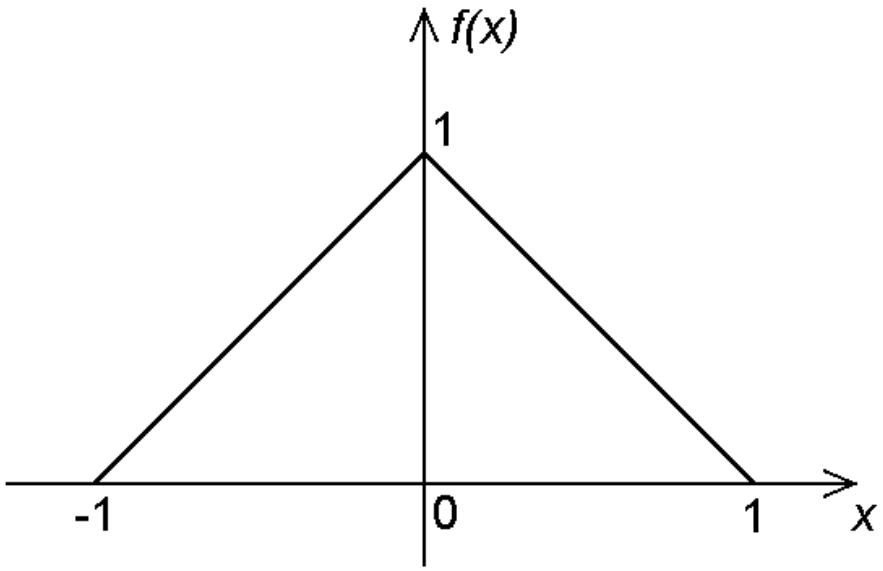

Рис. 4.3: Треугольное распределение

Вариант 1. Прямое применение метода обратной функции.

Сначала получим формулу функции распределения вероятностей как интеграл от плотности распределения:

$$F(x) = \begin{cases} 0, & \text{при } x < -1, \\ (x+1)^2/2, & \text{при } -1 \le x < 0, \\ 1-(1-x)^2/2, & \text{при } 0 \le x < 1, \\ 1, & \text{при } x \ge 1. \end{cases} \tag{4.9}$$

Тогда, решая уравнение $\xi = F^{-1}(\eta)$ относительно η имеем

$$\eta = \begin{cases} \sqrt{2\xi} - 1, & \text{при } \xi < 0.5, \\ 1 - \sqrt{2(1-\xi)}, & \text{при } \xi \ge 0.5. \end{cases} \tag{4.10}$$

Вариант 2. Кусочное рассмотрение функции распределения.

Область изменения рассматриваемой случайной величины можно разбить на 2 равных интервала $[-1, 0)$ и $[0, 1)$, на которые она попадает с равной вероятностью 0.5. Соответствующие плотности усечённых распределений на этих интервалах имеют вид

$$f_1(x) = 2(x+1), \tag{4.11}$$
$$f_2(x) = 2(1-x), \tag{4.12}$$

а функции —

$$F_1(x) = (x+1)^2, \tag{4.13}$$
$$F_2(x) = 1-(1-x)^2. \tag{4.14}$$

Следовательно, генератор должен с вероятностью 0.5 выдавать $\eta = \sqrt{\xi} - 1$ и с той же вероятностью $\eta = 1 - \sqrt{1-\xi}$.
Отметим, что в последней формуле можно заменить $1-\xi$ на ξ, так как эти случайные величины имеют одинаковое распределение.

Вариант 3. Использование функции двух случайных величин.

Рассмотрим распределение суммы двух независимых, равномерно распределённых на отрезке $[0,1)$ случайных величин. Пусть $\eta = \xi_1 + \xi_2$. Тогда $F_\eta(x) = P(\eta < x) = P(\xi_1 + \xi_2 < x)$. Удобнее всего получить эту вероятность геометрически (Рис. 4.4). Ей соответствует площадь части единичного квадрата, лежащей ниже прямой, проходящей через точки $(0,x)$ и $(x,0)$. Для $x < 1$ (двойная штриховка) это $x^2/2$, а для $x \geq 1$ — $1-(2-x)^2/2$. Следовательно,

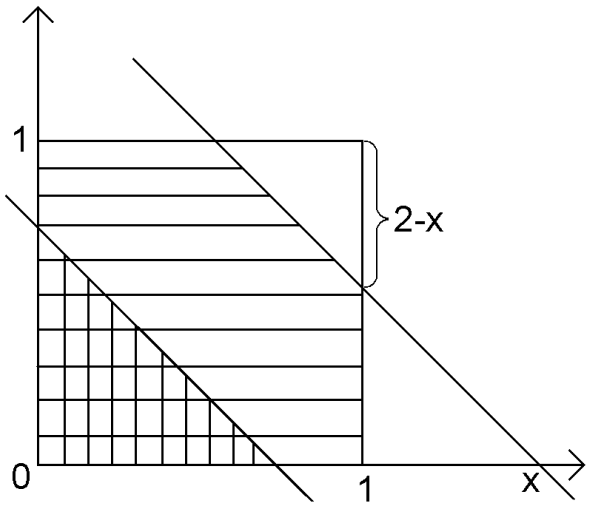

Рис. 4.4: Иллюстрация к выводу распределения суммы двух случайных величин, равномерно распределённых на $[0,1)$

$$F(x) = \begin{cases} 0, & \text{при } x < 0, \\ x^2/2, & \text{при } 0 \leq x < 1, \\ 1-(2-x)^2/2, & \text{при } 1 \leq x < 2, \\ 1, & \text{при } x \geq 2. \end{cases} \tag{4.15}$$

Плотность распределения, соответственно, равна

$$f(x) = \begin{cases} 0, & \text{при } x < 0, \\ x, & \text{при } 0 \leq x < 1, \\ 2-x, & \text{при } 1 \leq x < 2, \\ 0, & \text{при } x \geq 2, \end{cases} \tag{4.16}$$

то есть мы имеем ту же самую треугольную плотность распределения, но сдвинутую по оси абцисс на одну единицу вправо. Следовательно, имеем $\eta = \xi_1 + \xi_2 - 1$!

4.1.2 Дискретные распределения

Наиболее общий случай построения генератора дискретных случайных величин основывается на следующем алгоритме. Пусть имеется таблица пар (x_i, p_i), где x_i – значение случайной величины, а p_i – соответствующая вероятность, $\sum_{i=1}^{n} p_i = 1$ (в общем случае может быть и $n = \infty$). Тогда последнюю сумму можно представить полуинтервалом $[0, 1)$, разбитым на полуинтервалы $[v_{i-1}, v_i)$, $v_0 = 0$, $v_n = 1$ длины p_i. Случайная величина ξ с неизбежностью принадлежит одному из этих полуинтервалов, тем самым определяя индекс дискретного значения. Номер полуинтервала очевидно определяется как

$$\min\{i \mid \xi < v_i\} = \min\{i \mid \xi < \sum_{j=1}^{i} p_j\} \tag{4.17}$$

Замечание. Для эффективности датчика рекомендуется один раз рассчитать все суммы в (4.17) и затем использовать *вектор накопленных вероятностей*.

Пример 2.4. Равномерное дискретное распределение.

В случае равновероятного выбора между $x_i = i$, $i = 1, \ldots, n$ все p_i равны $\frac{1}{n}$ и тем самым $v_i = \frac{i}{n}$. Соответственно имеем номер полуинтервала равным $\min\{i \mid \xi < \frac{i}{n}\}$, откуда получаем просто $i = \lceil n\xi \rceil$.

Пример 2.5. Распределение Пуассона. Используя определение распределения Пуассона и вышеизложенные рассуждения получаем

$$\eta = \min\{i \mid \prod_{j=1}^{i} \xi_i < e^{-\lambda}\} \tag{4.18}$$

4.2 Выборки индексов

Имеем задачу получения случайной выборки объема m значений из множества $\{1, \ldots, n\}$. В случае выборки с возвращением (возможны повторы) просто пользуемся примером 2.4. Для выборки без возвращения наиболее эффективен в программной реализации следующий алгоритм.

Универсальный алгоритм для выборки без повторения.

Входные данные: `n` – число индексов; `m` – длина выборки.

Выходные данные: `L[1..m]` – результирующий вектор индексов.

```
1. for i:=1 to n do
2.   I[i]:=i;
3. endfor;
4. for s:=n downto n-m+1 do
5.   l:=random(1,s);
6.   L[n-s+1]:=I[l];
7.   I[l]:=I[s];
8.   DEC(s);
9. endfor.
```

Замечание: В случае `n=m` мы получаем алгоритм случайных перестановок.

В случае не равных вероятностей индексов (i выбирается с вероятностью p_i) алгоритм изменяется следующим образом:

- кроме вектора I имеется вектор вероятностей $P = \{p_i\}$, этот вектор изменяется после каждого выбора;
- в строке 6 случайный индекс получается по формуле

$$l = \min\{i \mid \xi < \sum_{j=1}^{i} p_j\},$$

где ξ равномерно распределена на (0,1) (процедура `rand()`);

- массив P также изменяется:

```
8a. I[l]:=I[s]; V:=P[l]; P[l]:=P[s];
8b. for i:=1 to n-s do
8c.   P[i]:=P[i]/(1.0-V);
8d. endfor.
```