## § 6. МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ ПРОЦЕССОВ

### 6.1. ТЕХНИКА СТОХАСТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Понятие «случайный» - одно из самых фундаментальных как в математике, так и в повседневной жизни. Моделирование случайных процессов - мощнейшее направление в современном математическом моделировании.

Событие называется случайным, если оно достоверно непредсказуемо. Случайность окружает наш мир и чаще всего играет отрицательную роль в нашей жизни. Однако есть обстоятельства, в которых случайность может оказаться полезной.

В сложных вычислениях, когда искомый результат зависит от результатов многих факторов, моделей и измерений, можно сократить объем вычислений за счет случайных значений значащих цифр. Из теории эволюции следует, что случайность проявляет себя как конструктивный, позитивный фактор. В частности, естественный отбор реализует как бы метод проб и ошибок, отбирая в процессе развития особи с наиболее целесообразными свойствами организма. Далее случайность проявляется в множественности ее результатов, обеспечивая гибкость реакции популяции на изменения внешней среды.

В силу сказанного имеет смысл положить случайность в основу методов получения решения посредством проб и ошибок, путем случайного поиска.

Отметим, что выше, приведя пример имитационного моделирования - игру «Жизнь», мы уже имели по сути дела стохастическую модель. В данном параграфе обсудим методологию такого моделирования более детально.

Итак, пусть в функционале модели значения некоторых входных параметров определены лишь в вероятностном смысле. В этом случае значительно меняется сам стиль работы с моделью.

При серьезном рассмотрении в обиходе появляются слова «распределение вероятностей», «достоверность», «статистическая выборка», «случайный процесс» и т.д.

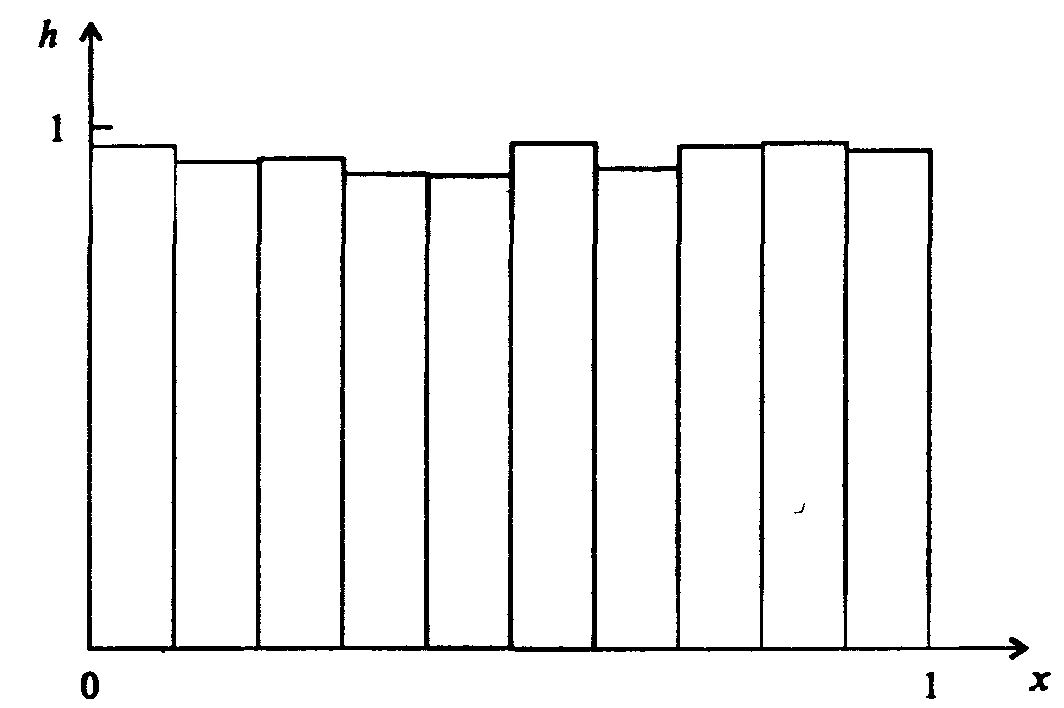
При компьютерном математическом моделировании случайных процессов нельзя обойтись без наборов, так называемых, случайных чисел, удовлетворяющих заданному закону распределения. На самом деле эти числа генерирует компьютер по определенному алгоритму, т.е. они не являются вполне случайными хотя бы потому, что при повторном запуске программы с теми же параметрами последовательность повторится; такие числа называют «псевдослучайными».

Рассмотрим вначале генерацию чисел равновероятно распределенных на некотором отрезке. Большинство программ - генераторов случайных чисел - выдают последовательность, в которой предыдущее число используется для нахождения последующего. Первое из них - начальное значение. Все генераторы случайных чисел дают последовательности, повторяющиеся после некоторого количества членов, называемого периодом, что связано с конечной длиной машинного слова. Самый простой и наиболее распространенный метод - метод вычетов, или линейный конгруэнтный метод, в котором очередное случайное число *xn* определяется«отображением»



где *a*, *с*, *m* - натуральные числа, mod - так называемая, функция деления по модулю (остаток от деления одного числа на другое по модулю). Наибольший возможный период датчика (7.69) равен *т;* однако, он зависит от *а* и *с.* Ясно, что чем больше период, тем лучше; однако реально наибольшее *m* ограничено разрядной сеткой ЭВМ. В любом случае используемая в конкретной задаче выборка случайных чисел должна быть короче периода, иначе задача будет решена неверно. Заметим, что обычно генераторы выдают отношение *,* которое всегда меньше 1, т.е. генерируют последовательность псевдослучайных чисел на отрезке [0,1].

Вопрос о случайности конечной последовательности чисел гораздо сложнее, чем выглядит на первый взгляд Существует несколько статистических критериев случайности, но все они не дают исчерпывающего ответа. Так, последовательно генерируемые псевдослучайные числа могут появляться не идеально равномерно, а проявлять тенденцию к образованию групп (т е. коррелировать) Один из тестов на равномерность состоит в делении отрезка [0, 1] на *М* равных частей - «корзин», и помещения каждого нового случайного числа в соответствующую «корзину». В итоге получается гистограмма, в которой высота каждого столбика пропорциональна количеству попавших в «корзину» случайных чисел (рис. 7.54).



*Рис. 7.54.* Вид гистограммы для равномерно распределенных на отрезке [0,1] чисел при достаточно большой выборке

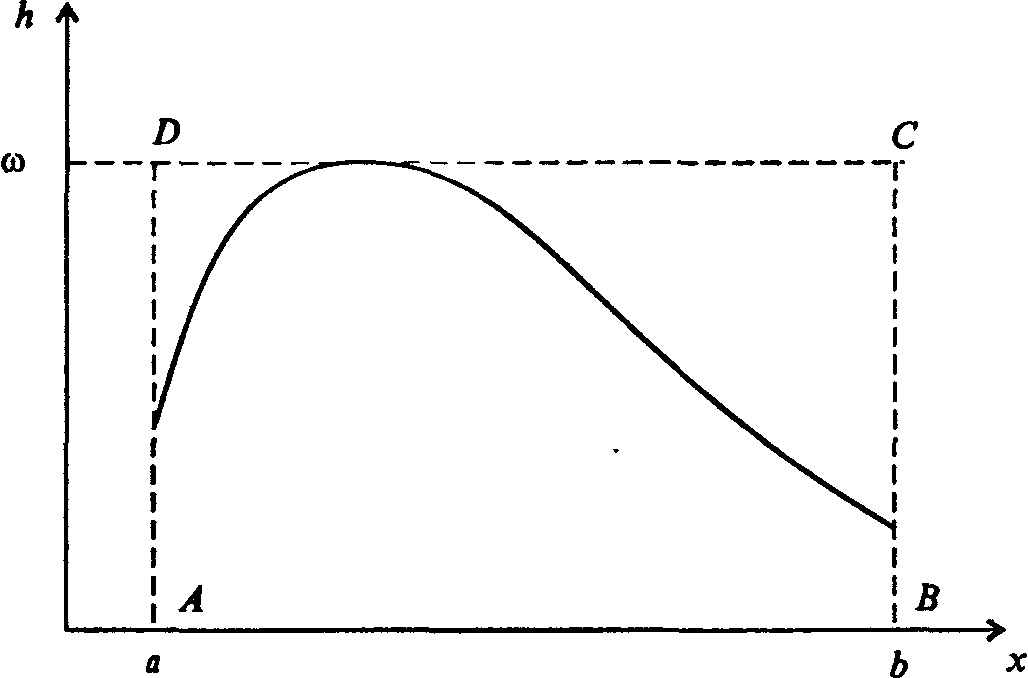
Понятно, что при большом числе испытаний высоты столбиков должны быть почти одинаковыми. Однако, этот критерий является необходимым, но не достаточным; например, он «не замечает» даже очень короткой периодичности Для не слишком требовательного пользователя обычно достаточны возможности датчика (генератора) случайных чисел, встроенного в большинство языков программирования. Так, в PASCAL есть функция random, значения которой - случайные числа из диапазона [0, 1). Ее использованию обычно предшествует использование процедуры randomize, служащей для начальной «настройки» датчика, т.е. получения при каждом из обращений к датчику разных последовательностей случайных чисел. Для задач, решение которых требует очень длинных некоррелированных последовательностей, вопрос осложняется и требует нестандартных решений Равномерно распределенные случайные числа - простейший случай Располагая датчиком случайных чисел, генерирующим числа *r* ∈ [0, 1], легко получить числа из произвольного интервала [*а, b*]*.*

***X = a + (b - a)∙r.***

Более сложные распределения часто строятся с помощью распределения равномерного. Упомянем здесь лишь один достаточно универсальный метод Неймана (часто называемый также методом отбора-отказа), в основе которого лежит простое геометрическое соображение. Допустим, что необходимо генерировать случайные числа с некоторой нормированной функцией распределения *f(x)* на интервале [*а, b*]*.* Введем положительно определенную функцию сравнения *w(x)* такую, что *w(x)* = const и *w(x) > f(x)* на [*а, b*] (обычно *w(x)* равно максимальному значению *f(x)* на [*а, b*]*).* Поскольку площадь под кривой *f(x)* равна для интервала [*х, х* + *dx*]вероятности попадания *х* в этот интервал, можно следовать процедуре проб и ошибок. Генерируем два случайных числа, определяющих равновероятные координаты в прямоугольнике *A BCD* с помощью датчика равномерно распределенных случайных чисел:

***x = a + (b - a)∙r, y = w∙r***

и если точка *М(х, у)* не попадает под кривую *f(x)*, мы ее отбрасываем, а если попадает - оставляем (рис. 7.55). При этом множество координат *х* оставленных точек оказывается распределенным в соответствии с плотностью вероятности *f(x)*.



*Рис. 7.55.* Метод отбора-отказа. Функция *w(x)* = *f*max

Этот метод для ряда распределений не самый эффективный, но он универсален, прост и понятен. Эффективен он тогда, когда функция сравнения *w(x)* близка к *f(х)*. Заметим, что никто не заставляет нас брать *w(x)=* const на всем промежутке [*а, b*]*.* Если *f(x)* имеет быстро спадающие «крылья», то разумнее взять *w(x)* в виде ступенчатой функции.

### 6.2*.* МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ ПРОЦЕССОВ В СИСТЕМАХ МАССОВОГО ОБСЛУЖИВАНИЯ

Комуне случалось стоять в очереди и с нетерпением прикидывать, успеет ли он сделать покупку (или заплатить за квартиру, покататься на карусели и т.д.) за некоторое имеющееся в его распоряжении время? Или, пытаясь позвонить по телефону в справочную и натыкаясь несколько раз на короткие гудки, нервничать и оценивать - дозвонюсь или нет? Из таких «простых» проблем в начале XX века родилась весьма непростая наука - теория массового обслуживания, использующая аппарат теории вероятностей и математической статистики, дифференциальных уравнений и численных методов. Основоположником ее стал датский ученый А.К.Эрланг, исследовавший проблемы функционирования телефонных станций.

Впоследствии выяснилось, что новая наука имеет многочисленные выходы в экономику, военное дело. организацию производства, биологиюи экологию; по нейнаписаны десятки книг, тысячи журнальных статей.

Компьютерное моделирование при решении задач массового обслуживания. реализуемое в виде метода статистических испытаний (метода Монте-Карло), хоть и не является в теории массового обслуживания основным, но играет в ней важную роль. Основная линия в ней - получение результатов аналитических, т.е. представленных формулами. Однако, возможности аналитических методов весьма ограничены, в то время как метод статистических испытаний универсален и весьма прост для понимания (по крайней мере кажется таковым).

Типичная задача: очередь к одному «продавцу». Рассмотрим одну из простейших задач данного класса. Имеется магазин с одним продавцом, в который случайным образом входят покупатели. Если продавец свободен, он начинает обслуживать покупателя сразу, если покупателей несколько, выстраивается очередь.

Вот аналогичные задачи:

• ремонтная зона в автохозяйстве и автобусы, сошедшие с линии из-за поломки;

•травмопункт и больные, пришедшие на прием по случаю травмы (т.е. без системы предварительной записи);

• телефонная станция с одним входом (или одной телефонисткой) и абоненты, которых при занятом входе ставят в очередь (такая система иногда практикуется);

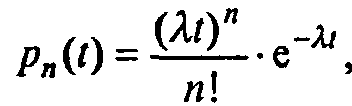
• сервер локальной сети и персональные машины на рабочем месте, которые шлют сообщение серверу, способному воспринять разом и обработать не более одного сообщения.

Будем для определенности говорить о магазине, покупателях и продавце. Рассмотрим возникающие здесь проблемы, которые заслуживают математического исследования и, как выясняется, весьма серьезного.

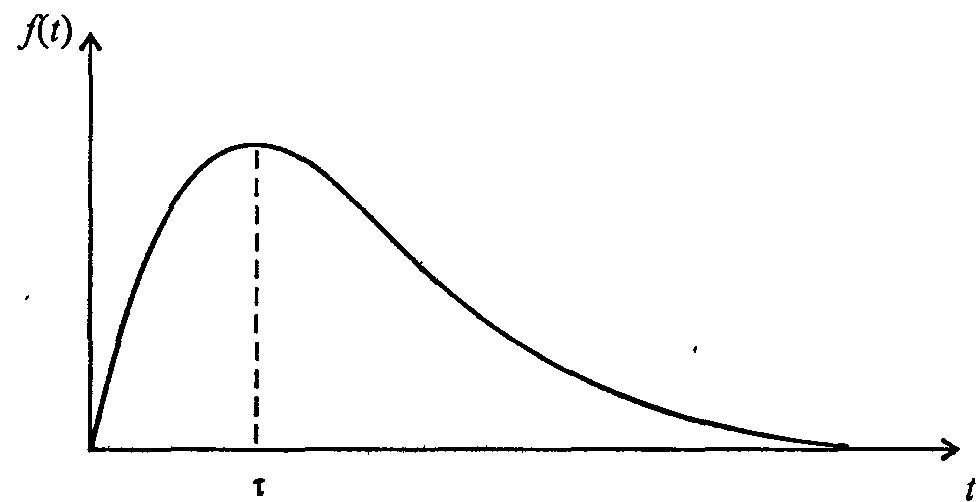
Итак, на входе этой задачи случайный процесс прихода покупателей в магазин. Он является «марковским», т.е. промежутки между приходами любой последовательной пары покупателей - независимые случайные события, распределенные по некоторому закону. Реальный характер этого закона может быть установлен лишь путем многочисленных наблюдений; в качестве простейшей модельной функции плотности вероятности можно взять равновероятное распределение в диапазоне времени от 0 до некоторого *Т -* максимально возможного промежутка между приходами двух последовательных покупателей. При этом распределении вероятность того, что между приходами двух покупателей пройдет 1 минута, 3 минуты или 8 минут одинакова (если *T* > 8).

Такое распределение, конечно, малореалистично; реально оно имеет при некотором значении *t =* *τ* максимум и быстро спадает при больших *t,* т.е. имеет вид, изображенный на рис. 7.56.

Можно, конечно, подобрать немало элементарных функций, графики которых похожи на тот, что изображен на рис. 7.56. Например, семейство функций Пуассона, широко используемых в теории массового обслуживания:

(7.70)

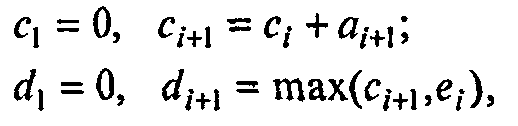
где *λ* - некоторая константа, *п -* произвольное целое. Функции (7.70) имеют максимум при  и нормированы: *pn(t)dt =* 1.



*Рис. 7.56.* Типичная плотность вероятности распределения времени между приходами покупателей

Второй случайный процесс в этой задаче, никак не связанный с первым, сводится к последовательности случайных событий - длительностей обслуживания каждого из покупателей. Распределение вероятностей длительности обслуживания качественно имеет тот же вид, что и в предыдущем случае; при отработке первичных навыков моделирования методом статистических испытаний вполне уместно принять модель равновероятного распределения.

В таблице 7.8 в колонке *А* записаны случайные числа - промежутки между приходами покупателей (в минутах), в колонке *В -* случайные числа - длительности обслуживания (в минутах). Для определенности взято *а*max = 10 и *b*mах = 5. Из этой короткой таблицы, разумеется, невозможно установить, каковы законы распределения приняты для величин *А* и *В;* в данном обсуждении это не играет никакой роли. Остальные колонки предусмотрены для удобства анализа; входящие в них числа находятся путем элементарного расчета. В колонке *С* представлено условное время прихода покупателя, в колонке *D -* момент начала обслуживания, *Е -* момент конца обслуживания, *F -* длительность времени, проведенного покупателем в магазине в целом, *G -* в очереди в ожидании обслуживания, *Н -* время, проведенное продавцом в ожидании покупателя (магазин пуст). Таблицу удобно заполнять по горизонтали, переходя от строчки к строчке. Приведем для удобства соответствующие формулы (в них *i* = 1, 2, 3,...):



так как начало обслуживания очередного покупателя определяется либо временем его прихода, если магазин пуст, либо временем ухода предыдущего покупателя;

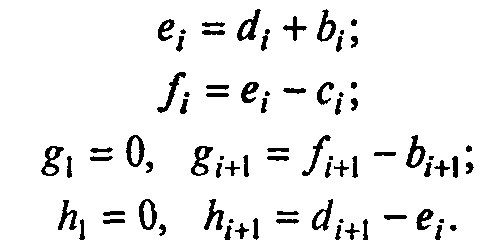


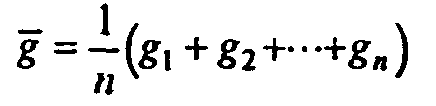
Таблица 7.8

**Моделирование очереди**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | А | В | С | D | Е | F | G | Н |
| 1 | 0 | 4 | 0 | 0 | 4 | 4 | 0 | 0 |
| 2 | 2 | 1 | 2 | 4 | 5 | 3 | 2 | 0 |
| 3 | 10 | 5 | 12 | 12 | 17 | 5 | 0 | 7 |
| 4 | 1 | 2 | 13 | 17 | 19 | 6 | 4 | 0 |
| 5 | 6 | 3 | 19 | 19 | 22 | 3 | 0 | 0 |

Таким образом, при данных случайных наборах чисел в колонках *A* и *В* и покупателям приходилось стоять в очереди (колонка *G),* и продавцу - в ожидании покупателя (колонка *H*).

При моделировании систем такого вида возникают следующие вопросы. Какое среднее время приходится стоять в очереди к прилавку? Чтобы ответить на него, следует найти



в некоторой серии испытаний. Аналогично можно найти среднее значение величины *h.* Конечно, эти выборочные средние сами по себе - случайные величины; в другой выборке того же объема они будут иметь другие значения (при больших объемах выборки, не слишком отличающиеся друг от друга). Доверительные интервалы, в которых находятся точные средние значения (т.е. математические ожидания соответствующих случайных величин) при заданных доверительных вероятностях находятся методами математической статистики.

Сложнее ответить на вопрос, каково распределение случайных величин *G* и *Н* при заданных распределениях случайных величин *А* и *В.* Допустим, в простейшем моделировании мы примем гипотезу о равновероятных распределениях величин *A* и *В -* скажем, для *А* в диапазоне от 0 до 10 минут и *В -* от 0 до 5 минут. Для построения методом статистических испытаний распределений величин *G* и *Н* поступим так: найдем в достаточно длинной серии испытаний (реально - в десятках тысяч, что на компьютере делается достаточно быстро) значения *g*max (для *Н* все делается аналогично) и разделим промежуток [0, gmax] на *т* равных частей - скажем, вначале на 10 - так, чтобы в каждую часть попало много значений *gi*. Разделив число попаданий *nk* в каждую из частей на общее число испытаний *n*, получим набор чисел *pk* =  *(k =* 1, 2,...,*n*). Построенныепо ним гистограммы дают представление о функциях плотностей вероятности соответствующих распределений. По гистограмме можно составить представление о функции плотности распределения соответствующей случайной величины. Для проверки же гипотезы о принадлежности такого эмпирически найденного распределения тому или иному конкретному виду служат известные статистические критерии.

Располагая функцией распределения (пусть даже эмпирической, но достаточно надежной), можно ответить на любой вопрос о характере процесса ожидания в очереди. Например: какова вероятность прождать дольше *т* минут? Ответ будет получен, если найти отношение площади криволинейной трапеции, ограниченной графиком плотности распределения, прямой *х = т* и *у* = 0, к площади всей фигуры.

Следующая программа позволяет моделировать описанный выше процесс. На выходе она дает средние значения и дисперсии случайных величин *g* и *h,* полученные по выборке, максимальный объем которой порядка 10000 (ограничение связано с малой допустимой длиной массива в PASCALe; чтобы его смягчить, использовано динамическое описание массивов *g* и *h*). Кроме того, программа строит гистограммы распределений величин *g* и *h.*

*Программа 152.* Моделирование очереди

Program Cohered;

(входной поток равновероятных событий;

динамические массивы позволяют значительно увеличить объем выборки)

Uses Crt, Graph;

Const N = 10000 (число членов выборки);

W1 = 10 (диапазон времен прихода от 0 дo wl};

W2 = 5 (диапазон времен обслуживания от 0 до w2};

Type Т = Array(l..N] Of Real; U = ^Т;

Var A, B, C, D, E, F, Aa, Bb, Cc, Dd, Ее, Ff, Dg, Dh, M : Real;

Sl, S2 : Double; I, K, J, I1, I2 : Integer;

LI, L2, V : Array [1..11] Of Real; G, H : U; Ch : Char;

Begin

If MaxAvail >= SizeOf(G) Then New(G);

If MaxAvail >= SizeOf(H) Then New(H);

Randomize; (ниже - имитационное моделирование)

Aa := 0; Bb := W2 \* Random; Cc := 0; Ее := Bb; Ff := Bb;

G^[l] = 0; H^[1] := 0;

For К = 1 To 11 Do

Begin L1(K] := 0; L2[K] := 0 End;

For I = 2 To N Do

Begin

A := Wl \* Random; В := W2 \* Random;

С := Cc + A; If С > Ее Then D := С Else D := Ее;

E := D + B; F := E - C; G^[I] := F - B; H^[I] := D - Ее;

Cc := С; Ее := E;

If G^[I] <= 1 Then Ll[l] := Ll[l] + 1; If H^[1] = 0 Then

L2[l] := L2[l] + 1;

For К := 2 To 10 Do

Begin

If (G^[I] > К - 1) And (G^[I] <= K) Then L1[K] := L1[K] + 1;

If (H^[I] > K - 1) And (H^[I] <= K) Then L2[K] := L2[K] + 1;

End;

If G^[I] > 10 Then Ll[l1] := Ll[ll] + 1;

If H^[I] > 10 Then L2[ll] := L2[ll] + 1;

Sl := Sl + G^[l]; S2 := S2 + H^[I];

End;

For I := 1 To 11 Do (ниже - нормировка распределений g и h}

Begin

L1[I] := L1[I] / N; L2[I] := L2[I] / N

End;

(ниже - расчет средних и дисперсий величин g и h}

Sl := Sl / N; S2 := S2 / N; Dg := 0; Dh := 0;

For I := 1 То N Do

Begin

Dg := Dg + Sqr(G^[I] - Sl); Dh := Dh + Sqr(H^[I] - S2)

End;

Dg := Dg / N; Dh := Dh / N;

WriteLn('распределение величины g распределение величины h');

WriteLn;

For K := 1 To 11 Do

WriteLn ('11[', K, ']=', L1[K] : 6 : 4, '' : 20, '12(', К, ']=',

L2[K] : 6 : 4);

WriteLn;

WriteLn('выборочное среднее величины g=', S1 : 6 : 3,

' выборочная дисперсия величины g=', Dg : 6 : 3);

WriteLn('выборочное среднее величины h=', S2 : 6 : 3,

' выборочная дисперсия величины h=', Dh : 6 : 3);

Dispose(G); Dispose(H); WriteLn;

WriteLn('для продолжения нажать любую клавишу');

Repeat Until KeyPressed; Ch := ReadKey;

(ниже - построение гистограмм распределений величин g и h)

DetectGraph(I, К); InitGraph(I, К, '');

I := GetMaxX; К := GetMaxY; J := I Div 2; M :'= Ll[l];

For I1 := 2 То 11 Do If L1[I1] > M Then M := L1[I1];

For I1 := 1 To 11 Do V[I1] := L1[I1] / M;

Line(10, К - 10, J - 20, К - 10); Line[l0, К - 10, 10, 5) ;

OutTextXY(20, 100, 'распределение величины g');

For I1 := 1 To 11 Do

Begin

I2 := Round((K - 20) \* (1 - V[I1])) + 10;

Line(I1 \* 20 - 10, I2, I1 \* 20 + 10, I2);

Line(I1 \* 20 - 10, I2, I1 \* 20 - 10, К - 10);

Line(I1 \* 20 + 10, I2, I1 \* 20 + 10, К - 10);

End;

Line(J + 20, К - 10, I - 10, К - 10);

Line(J + 20, К - 10, J + 20, 5) ;

OutTextXY(J + 30, 100, 'распределение величины h'); M := L2[l];

For I1 := 2 To 11 Do If L2[I1] > M Then M := L2[I1];

For I1 := 1 To 11 Do V[I1] := L2[I1] / M;

For I1 := 1 To 11 Do

Begin

I2 := Round((K - 20) \* (1 - V[I1])) + 10;

Line(J + I1 \* 20, I2, J + I1 \* 20 + 20, I2);

Line(J + I1 \* 20, I2, J + I1 \* 20, К - 10);

Line(J + I1 \* 20 + 20, I2, J + I1 \* 20 + 20, К - 10);

End;

OutTextXY(200, GetMaxY - 10, 'для выхода нажать любую клавишу');

Repeat Until KeyPressed; CloseCraph

End.

Приведем для сравнения результаты расчета средних значений величин *g, h* и соответствующих среднеквадратичных отклонений *Sg,* *Sh*, полученные при одинаковых значениях всех параметров в пяти разных сериях испытании по 10000 событий в серии (табл. 7.9) (входной поток покупателей - процесс равновероятных событий с максимальным временем между приходами 10 мин, длительность обслуживания также распределена равновероятным образом в интервале от 0 до 5мин).

Таблица 7.9

**Сравнение результатов моделирования в разных сериях испытаний**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Испытание | *g* | *Sg* | *h* | *Sh* |
| 1 | 0,738 | 1,568 | 2,508 | 2,588 |
| 2 | 0,746 | 1,511 | 2,500 | 2,571 |
| 3 | 0,765 | 1,529 | 2,446 | 2,582 |
| 4 | 0,753 | 1,524 | 2,451 | 2,589 |
| 5 | 0,765 | 1,573 | 2,482 | 2,572 |

Количество цифр выписано таким образом, чтобы отразить значимую разницу между данными разных серий.

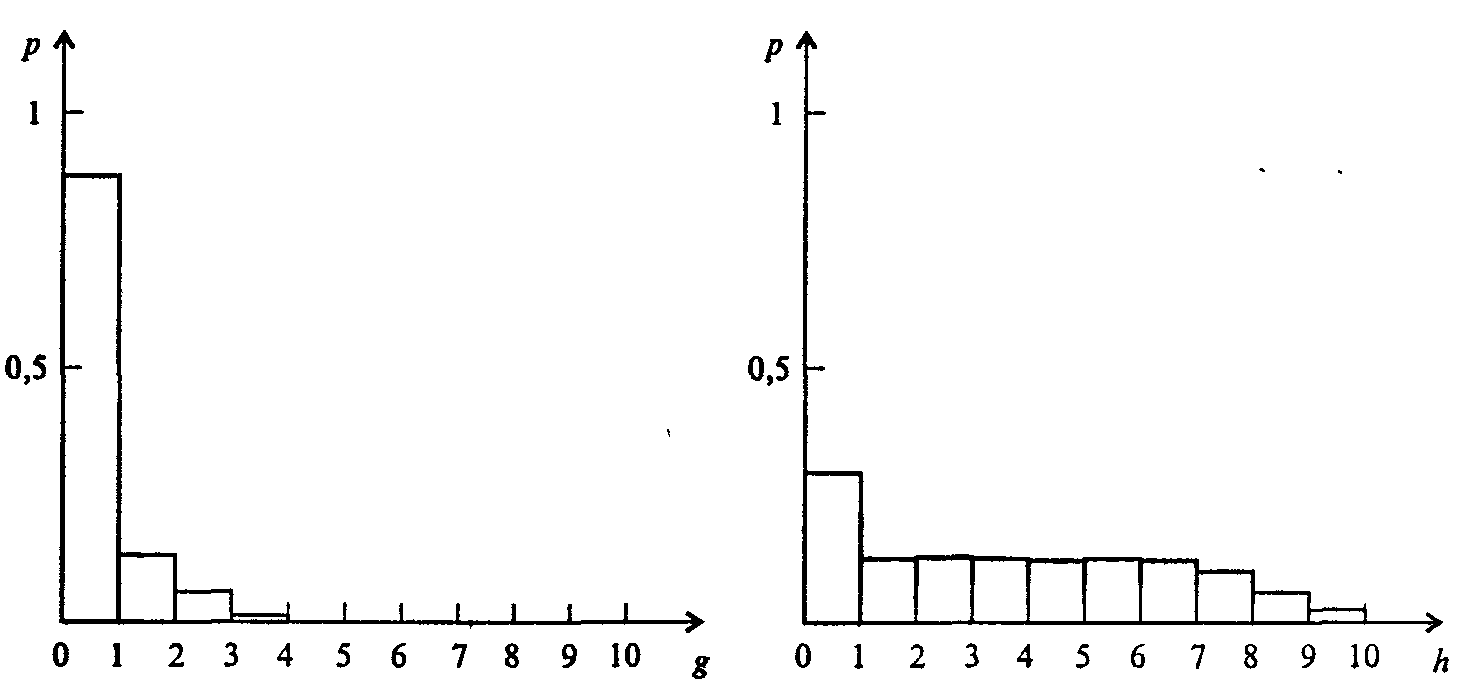
Оценим доверительный интервал математических ожиданий величин *g* и *h* при с достоверности 0,99 по формуле  * < тх < ; ε =* 2,58 ∙ *(* ***-*** среднее значение *х; п -* объем выборки; *S -* среднеквадратичное отклонение). По первой выборке получаем

0,738 - 0,025 < *mg <* 0,738 + 0,025 (округлим: 0,71 < *mg <* 0,77)

2,508 - 0,067 < *mh <* 2,508 + 0,067 (округлим: 2,44 < *mh <* 2,58).

Таким образом, различия в выборочных средних вполне укладываются в указанные доверительные интервалы.

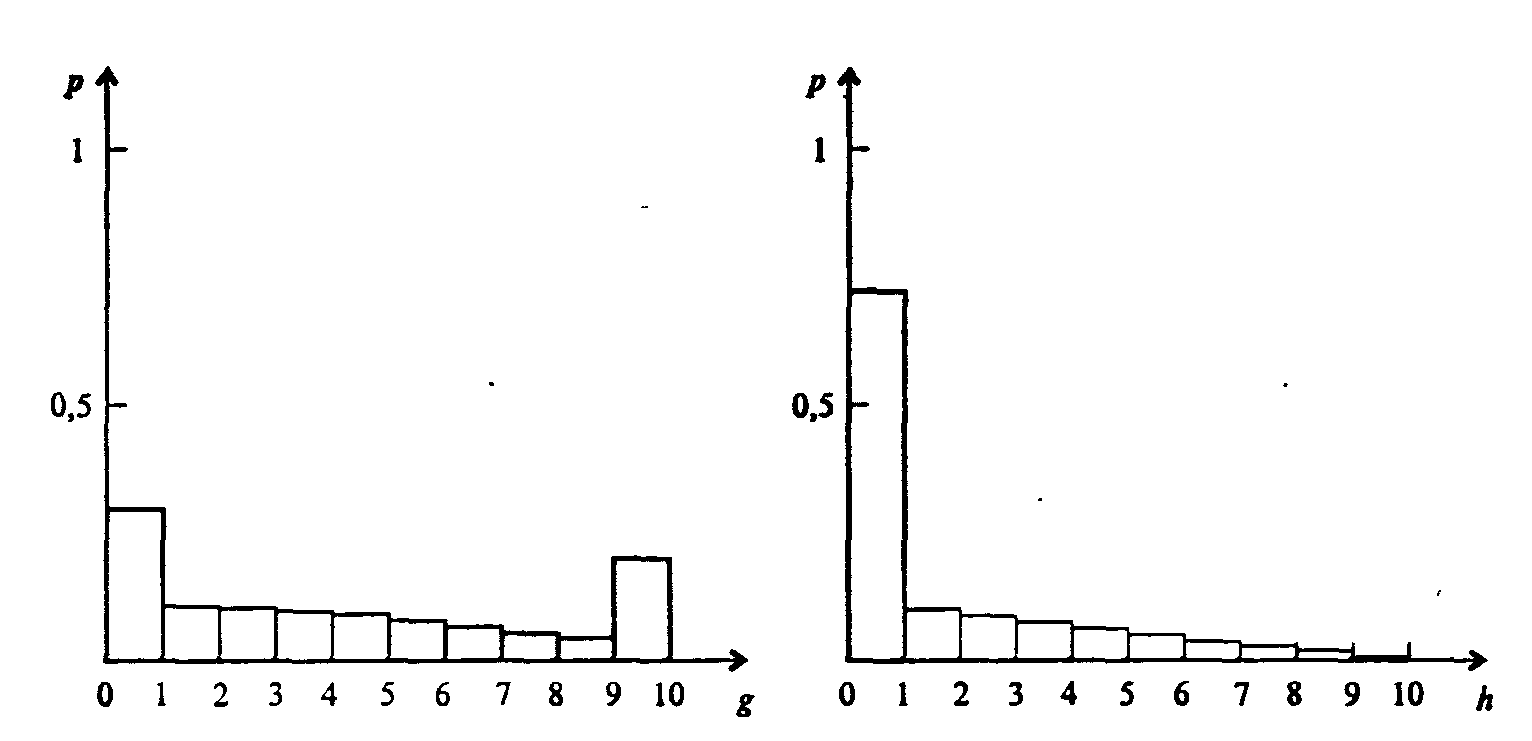
В рассмотренной задаче, как и в любой болеесложной задачеоб очередях,может возникнуть критическая ситуация, когда очередь неограниченно растет со временем. В самом деле, если люди заходят в магазин очень часто (или продавец работает слишком медленно), очередь начинает нарастать, и в любой системе с конечным временем обслуживания наступит кризис. Приведем для иллюстрации динамики этого процесса распределения величина - времени ожидания покупателем в очереди и *h -* времени простоя продавца в ожидании покупателя, при трех наборах параметров *w1, w2,* где *w1 -* максимальный интервал времени между приходами покупателей, *w2 -* максимальная длительность обслуживания покупателя (рис. 7.57 - 7.59).



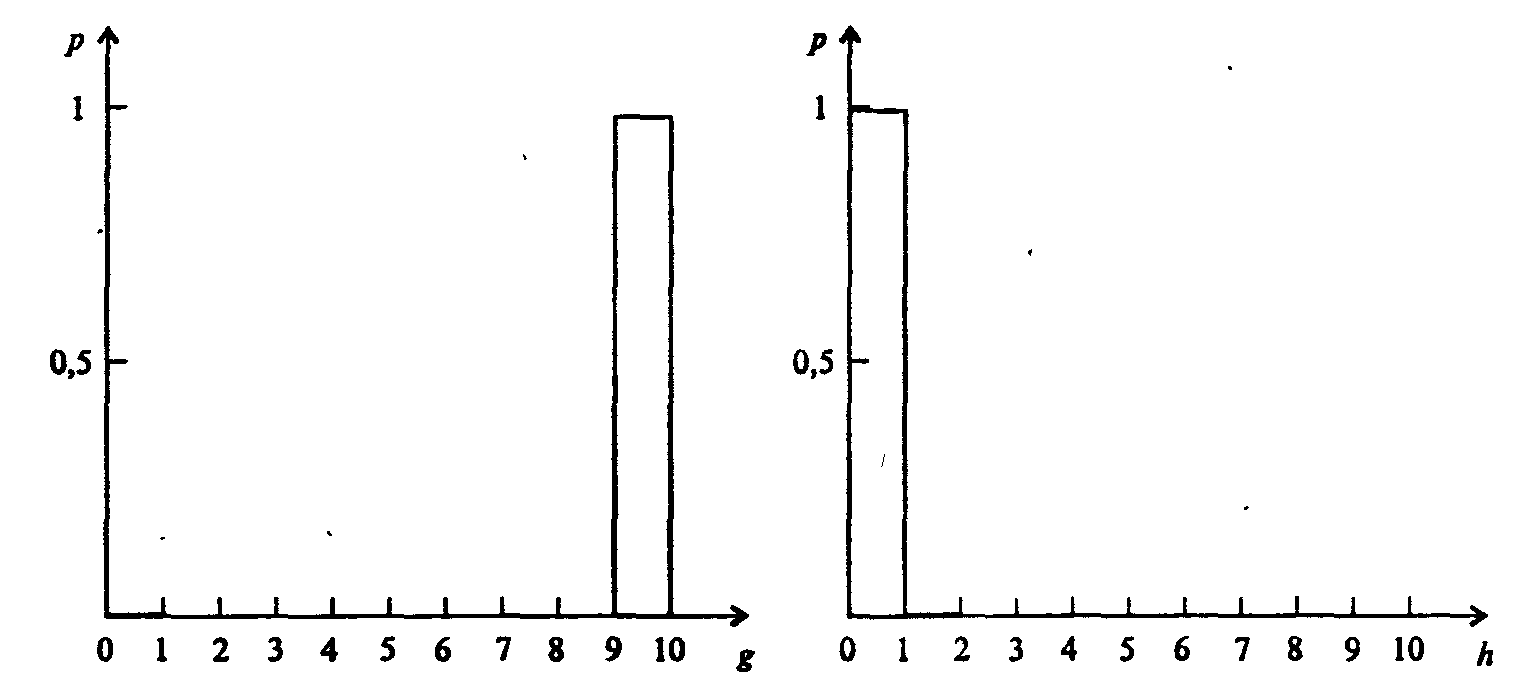
*Рис. 7.57. w1* = 10, *w2 =* 3 (нет проблем с обслуживанием, вероятность долго простоять в очереди мала, вероятность бездеятельности продавца достаточно велика)

В чем практическое значение задач об очередях? Прежде всего, стремление рационально построить обслуживание потребителей. В магазине можно, к примеру, поставить второго продавца, но если при этом продавцы будут мало заняты, возникает ущерб для предприятия. Таким образом, актуальным является вопрос об отыскании оптимального решения при наличии противоречивых требований, т.е. налицо оптимизационная многокритериальная модель.

Визуально проиллюстрировать формирование очереди поможет следующая программа.



*Рис. 7.58. w1 =* 10, *w2 =* 8 (кризис приближается)



*Рис.* 7.59. *wl* = 10, *w2 =* 10 (кризис наступил)

*Программа 153.* Имитационное моделирование очереди

Program Bank;

Uses Graph, Crt;

Var Gm, Gd, P, X, Qq, I, T, V : Integer; St : String[1O];

Begin

Qq := 0; P := 6; V := 2; Randomize; DetectGraph(Gd, Gm);

InitGraph(Gd, Gm, ' ');

SetColor(2); RectAngle(300, 100, 500, 300); Т :- 0;

Repeat

Т := Т + 1; Str(T, St); SetTextStyle(0, 0, 1) ;

SetColor(4); OutTextXY(600, 50, St); X := Random(ll) ;

If X < P Then Qq := Qq + 1; SetColor(15) ;

For I := 0 To Qq Do Circle(490 - I \* 30, 200, 15);

Delay(1000); SetColor(0);

For I := 0 To Qq Do Circle(490 - I \* 30, 200, 15);

If T Mod V = О

Then Begin

Qq := Qq - 1; If Qq < 0 Then Qq :- 0; Setcolor(15) ;

For I := 0 To Qq Do Circle(490 - I \* 30, 200, 15);

End;

SetColor(O); OutTextXY(600, 50, St)

Until KeyPressed Or (Qq > 15); ReadLn;

End.

### 6.3. РАЗЛИЧНЫЕ ПРИМЕРЫ МОДЕЛИРОВАНИЯ СЛУЧАЙНЫХ ПРОЦЕССОВ

Метод статистического моделирования имеет множество приложений. Чаще всего он заключается в том, что для решения математической задачи строится некоторая случайная величина ζ, такая, что математическое ожидание этой случайной величины *E*(ζ) является значением искомого решения. Проводя достаточное количество раз эксперимент со случайной величиной ζ, можно найти приближенное решение как среднее значение результатов эксперимента.

**1. Вычисление площадей.** Найти площадь фигуры *G,* вписанной в прямоугольник с размерами сторон *а* и *b. С* помощью датчика равновероятно распределенных случайных чисел многократно генерируются координаты точки, принадлежащей прямоугольнику. Очевидно, что при большом числе испытаний площадь фигуры *G* приближенно равна отношению числа точек, попавших в область *G,* к числу всех разыгранных точек. В качестве примера приведем программу вычисления числа *π*, находя указанным методом площадь круга, вписанного в квадрат, по 100000 испытаний. Оценку точности полученного результата оставляем читателям.

*Программа 154.* Вычисление числа *π*  методом Монте-Карло

Program Chislo\_Pi;

Uses Crt; Var I, N : Longint; X, Y : Real;

Begin

Randomize; N := 0;

For I := 1 To 100000 Do

Begin

X *:=* Random; Y := Random;

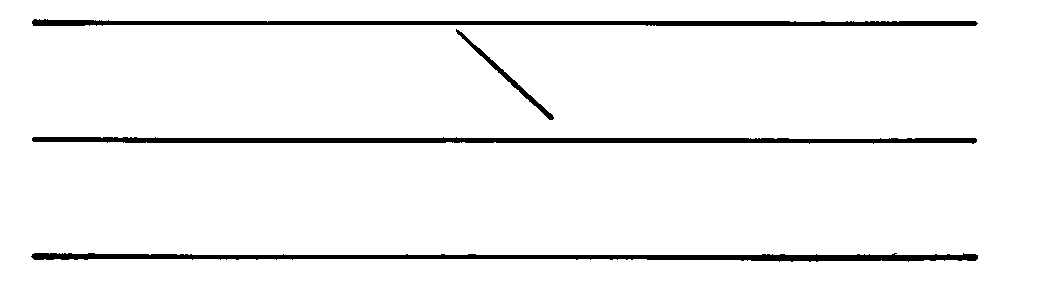
If Sqr(X - 0.5) + Sqr(Y - 0.5) < 0.25 Then N := N + 1

End;

WriteLn ('pi=', (N / 100000 / Sqr(0.5)) : 8 : 5) ;

Repeat Until KeyPressed End.

**2. Задача Бюффона.** На поле, разграфленное параллельными прямыми, расстояние между которыми *L,* бросается наугад игла длиной *l* (рис. 7.60). Какова вероятность того, что игла, упав, пересечег хотя бы одну прямую?



*Рис. 7.60. К.* задаче Бюффона

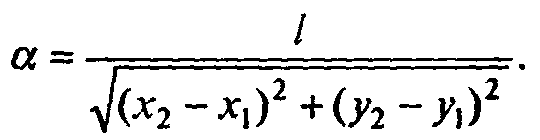
Ж.Бюффон (XVIII в.) подсчитал: *р* = . Таким образом, если *L =* 2*l,* то *р* = . Кроме того, *р* = *,* где *N -* число бросаний, *N1* - число пересечений иглы с линиями.

Относительная доля случаев, когда игла пересечет хотя бы однуиз параллельныхпрямых равно *р* = . Это был одиниз старинных способов опредения числа *π.*

Имитационное моделирование проведем следующим образом. Примем *L* = 1 и *l* = . «Иглу» будем «бросать» в квадрат размером, скажем, 20х20, левый нижний угол которого имеет координаты (0, 0). Положение концов иглы будем задавать с помощью датчика равномерно распределенные, случайных чисел в диапазоне от 0 до 20. Точнее говоря, эти числа определят направление отрезка, вдоль которого находится очередная игла; для того, чтобы ее длина была равна , вторую из случайных точек - концов отрезка - подвинем вдоль него до достижения указанной длины иглы. В математическом отношении это сводится к следующей несложной процедуре;

• генерация координат точек *А(х1, y1), B(x2, у2);*

*•* определение координат точки *В1(х1* + *α(х2 – х1), у1* + *α(у2 – у1)),* где



Поскольку расстояние между горизонтальными линиями взято равным единице, а сами линии имеют целочисленные координаты по *у,* то определить, пересекает ли игла прямую, очень просто - да, если целые части ординат тoчeк *A* и *В1* различны.

*Программа 155* Решение задачи Бюффона.

Program Buffon;

Uses Crt; Var I, J, K, M, N : Integer; XI, X2, Y1, Y2, Al : Real;

Begin

Randomize; M := 30000; N := 1;

For I := 1 To M Do

Begin

X1 := Random \* 20; Yl := Random \* 20; X2 := Random \* 20;

Y2 := Random \* 20;

A1 := 0.5 / Sqrt(Sqr(X2 – X1) + Sqr(Y2 - Yl) ) ;

J := Round(Yl); К := Round(Yl + A1 \* (Y2 – Y1));

If J <> К Then N := H + 1

End;

WriteLn('pi=', W / N) : 8 : 5); Repeat Until KeyPressed

End.

Создание демонстрационной программы, которая выводит на экран несколько параллельных прямых из общего набора и имеющие к ним отношение «иглы», предоставляем читателю.

Рекомендуем провести с предложенной программой несколько экспериментов. Понятно, что чем больше значение *т,* тем, по-видимому, точнее результат. Однако. почему он постоянно слегка занижен? Все ли учтено на краях той зоны, в которой разыгрываются броски иглы? Чтобы почувствовать проблему, следует увеличить число параллельных прямых, что в данной программе очень легко сделать. Почему результат становится лучше? Отметим, что проблема краевых условий, когда события должны по условиям задачи разыгрываться на бесконечном поле, а при имитационном моделировании фактически разыгрываются на конечном (и даже не очень большом), возникает часто и решение ее нетривиально.

**3. Нефтяное месторождение.** Дано нефтяное месторождение, в котором область залегания нефти *G* ограничивается кривой *С.* Дебит скважины, т.е. количество получаемой из нее нефти в единицу времени, зависит от пластового давления нефти *U* в точке скважины. Поэтом) для прогнозирования нефтедобычи важно знать распределение пластового давления на всем месторождении при условии, что оно экспериментально измерено лишь на его границе. В математическом плане функция *U(r)* удовлетворяет уравнению Лапласа  = 0; задача нахождения его решения внутри области при заданном значении *U(r)* на границе - так называемая, краевая задача Дирихле; в данной задаче это решение, которое часто совсем не просто найти аналитически, позволило бы правильно определить точку для скважины.

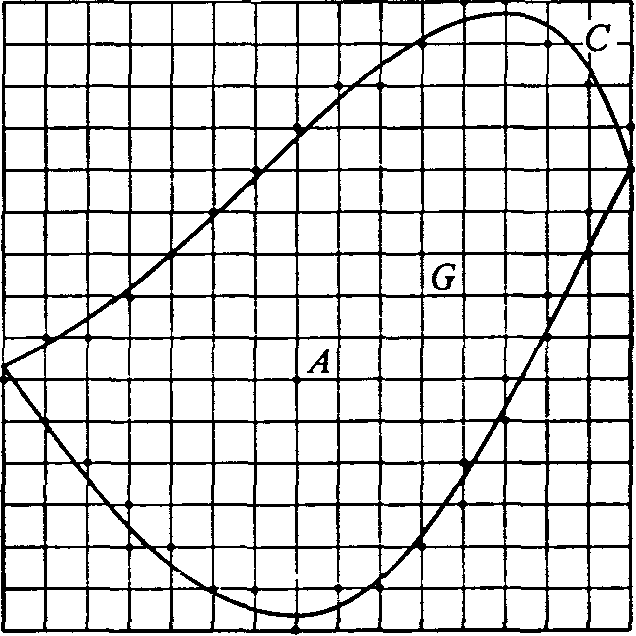
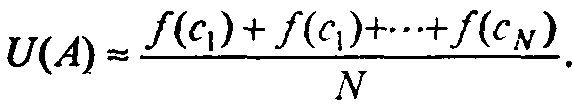


Рис. 7.61. Наложение сетки на заданную область

Покроем область *G* мелкой сеткой. Отметим узлы, наиболее близкие к границе *С,* и будем считать, что значения функции *U* в этих узлах приблизительно равны значениям этой функции в ближайших к ним точках границы. Будем искать значение функции *U(A)* в некотором внутреннем узле *A* (рис. 7.61).

Поместим в точке *А* блуждающую частицу, которая может перемещаться по области в последовательные моменты времени, переходя из одного угла в соседний. Направления перемещений случайны, равновероятны и не зависят от ее положения и предыстории блуждания. Случайный эксперимент состоит в наблюдении факта выхода блуждающей частицы в некоторый граничный узел. Когда блуждание прекращается, запоминается значение функции *f*(*сi*) в этой точке, и так далее, *N* раз. Замечательный факт состоит в том, что решение в точке



Другими словами, среднее значение приближенно равно решению задачи Дирихле в точке *А.*

**4. Модель «пьяницы» (модель случайного блуждания).** Зададим блуждание точки (объекта) по горизонтальной линии по правилу: если случайное число из интервала [0, 1] меньше 0,5, то точка делает шаг вправо *x* *= х +* *h*, в противном случае *x* *= х - h.*

*Программа 156.* Модель случайного блуждания

Program Tochka;

Uses Crt, Graph; Var I, J : Integer; Z, P, X, H, Y : Integer;

Begin

X := 320; Y := 240; H := 10; P := 4; DetectGraph(I, J) ;

InitGraphd, J, ");

SetColor(15); Line(10, 312, 630, 312); Randomize;

Repeat

Z := Random(8); If Z >= P Then X := X + H Else X := X - H;

SetColor(Green); Circle(X, Y, 10); Delay(200);

SetColor(0); Circle(X, Y, 10)

Until KeyPressed Or (X >= 640) Or (X <= 0); CloseGraph

End.

В программе шаг является постоянным, но никто не мешает нам сделать его переменным, выбирая из интервала [0, hmax] случайным образом. Для этого зададим максимально возможный шаг **НМах** и в цикле определим **H := Random(HMax).**

Если задать аналогичным образом вероятности движения точки вверх – вниз, вправо - влево (0 < *рх <* 1, 0 *< рy <* 1), получим хаотическое блуждание точки на плоскости. Для моделирования блуждания точки в замкнутом прямоугольном объеме примем абсолютно упругое (зеркальное) отражение от стенок.

*Программа 157.* Хаотическое блуждание точки

Program Broun;

Uses Crt, Graph;

Var I, J, X, Y, HxMax, HyMax, Hx, Ну : Integer; PI, P2, Z1, Z2 : Real;

Begin

X := 320; Y :== 240; HxMax := 30; PI := 0.5; P2 := 0.5; HyMax := 30;

DetectGraph (I, J) ; InitGraph (1, J, ''); SetColor(15);

Randomize; RectAngle(100, 100, 540, 380);

SetColor(Green); Circle(X, Y, 10); Delay(200); SetColor(0);

Circle(X, У, 10);

Repeat

Zl := Random; Z2 := Random; Hx := Random(HxMax);

Ну := Random(HyMax) ;

If (Zl < PI) Then X := X + Hx Else X := X - Hx;

If (Z2 < P2) Then Y := Y + Ну Else Y :" У - Ну;

If X <= 110 Then X := X + 2 \* (110 - X) ;

If X >= 530 Then X := X - 2 \* (-530 + X) ;

If Y <= 110 Then Y := Y + 2 \* (110 - Y) ;

If Y >= 370 Then Y := Y - 2 \* (Y - 370);

SetColor(Green); Circle(X, Y, 10); Delay(100);

SetColor(0); Circle(X, Y, 10)

Until Keypressed; CloseGraph

End.

Подобным (хотя и более сложным) образом происходит броуновское движение, хорошо известное из курса физики. Если след точки не стирать, то можно будет наблюдать на экране траекторию такого движения. Нет большого труда перейти к случаю *п* частиц. Для этого необходимо завести два массива координат точек и аналогично предыдущему примеру организовать их движение.

*Программа 158.* Броуновское движение

Program Gaz;

Uses Crt, Graph;

Var I, J, HxMax, HyMax, Hx, Ну, N, I : Integer;

X, Y : Array[0..500] Of Integer; PI, P2, Z1, Z2 : Real;

Begin N := 100;

For I := 1 To N Do Begin X[I] := 320; Y[I] := 240 End;

HxMax := 10; PI := 0.5; P2 := 0.5; HyMax := 10;

DetectGraph (1, J) ; InitGraphd, J, ' '); SetColor(15);

Randomize; RectAngle(100, 100, 540, 380);

For I := 1 To N Do PutPixel(X[I], Y[I], White); Delay(200);

For I := 1 To N Do PutPixel(X(I], Y[I], 0) ;

Repeat

For I := 1 To N Do

Begin

Zl := Random; Z2 := Random;

Hx := Random(HxMax); Ну := Random(HyMax);

If Zl < PI Then X[I] := X[I] + Hx Else X[I] := X[I]— Hx;

If Z2 < P2 Then Y[I] := Y[I] + Ну Else Y[I] := Y[I] - Ну;

If X[I] <= 110 Then X[I] := X[I] + 2 \* (110 - X[I]);

If X(I] >= 530 Then X[I] := X[I] - 2 \* (-530 + Х[I];

If Y[I] <= 110 Then Y(I] := Y[I] + 2 \* (110 - Y[I]);

If Y[I] >= 370 Then Y[I] := Y[I] - 2 \* (Y[I] - 370);

PutPixel (X[I], Y[I], 15)

End; Delay(100);

For I := 1 To N Do PutPixel(X[I], Y[I], 0)

Until KeyPressed; CloseGraph

End.

Построенная компьютерная модель в первом приближении может позволить моделировать многие явления и процессы, происходящие в газах: рассеивание облака, диффузия газов. С ее помощью можно получить многие зависимости параметров газа друг от друга. В частности, давление (число соударений частиц на стенки) от длины свободного пробега (величин **HxMax** и **HyMax)** или от числа частиц.

Представляет значительный интерес имитационное моделирование явлений в сплошных средах, удовлетворяющих законам идеального газа, таких, как истечение газа в вакуум, ударная волна, волны разрежения и т.п. Для модернизации модели можно ввести в алгоритм упругое столкновение частиц друг с другом, возникновение кластерных ансамблей и многое другое.

При вероятностном моделировании используютразличные методы, которыепозволяют решать задачи из различных областей. Ниже перечислены сферы применения вероятностных методов.

Метод статистического моделирования: решение краевых задач математической физики, решение систем линейных алгебраических уравнений, обращение матриц и сводящиеся к ним сеточные методы решения систем дифференциальных уравнений, вычисление кратных интегралов, решение интегральных и интегродифференциальных уравнений, задач ядерной физики, газовой динамики, фильтрации, теплотехники.

Метод имитационного моделирования: моделирование систем массового обслуживания, задачи АСУ, АСУП и АСУТП, задачи защиты информации, моделирование сложных игровых ситуаций и динамических систем.

Метод стохастической аппроксимации: рекуррентные алгоритмы решения задач статистического оценивания.

Метод случайного поиска: решение задач оптимизации систем, зависящих от большого числа параметров, нахождение экстремумов функции большого числа переменных.

Другие методы: вероятностные методы распознавания образов, модели адаптации, обучения и самообучения.