Майер Р.В., Глазовский пединститут

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ:
5. ДИСКРЕТНО-СТОХАСТИЧЕСКИЕ МОДЕЛИ

В некоторых случаях функционирование системы не определяется полностью ее параметрами, начальным состоянием и внешними воздействиями, а зависит от каких-то случайных факторов. Такие системы называются стохастическими. Для их изучения используется **метод статистического моделирования,** который состоит в многократном проведении испытаний с последующей статистической обработкой получающихся результатов [11, 13]. Настоящая глава посвящена рассмотрению так называемой P-схемы, в рамках которой исследуемая система заменяется **вероятностным автоматом** [2, 7, 10]. Этот подход позволяет исследовать целый класс стохастических процессов, среди которых: передача сообщений по каналу связи, обучение вероятностного автомата, поведение дискретно-детерминированных систем, на вход которых поступают случайные сигналы и т.д. [8, 9, 12]. **Вероятностные клеточные автоматы** позволяют создать модель размножения бактерий, автоволновых процессов в активной среде, построить стохастические фракталы [1].

5.1. Метод статистического моделирования

Сущность метода статистического моделирования некоторой системы заключается в создании компьютерной модели (то есть программы), имитирующей поведение элементов системы и их взаимодействие друг с другом и внешней средой, и проведение серии вычислительных экспериментов. В результате получается массив данных, соответствующих выходным сигналам, которые подвергаются статистической обработке. При достаточно большом количестве испытаний получающиеся значения приобретают статистическую устойчивость и могут рассматриваться как характеристики функционирования исследуемой системы [11].

Метод **статистического моделирования** или метод **Монте-Карло** применяется для изучения как стохастических, так и динамических систем. Во втором случае детерминированная система заменяется соответствующей стохастической системой, характеристики выходных сигналов которой соответствуют результатом анализа детерминированной системы. Сущность метода состоит в следующем (рис. 1): 1. Создается компью-

терная модель исследуемой системы, позволяющая получать вектор выходных сигналов $(y_1, y_2, ..., y_m)$ при заданных векторе входных сигналов $(x_1, x_2, ..., x_n)$ и векторе внешних воздействий $(F_1, F_2, ..., F_k)$. 2. Автоматически выполняется серия из $10^3 - 10^5$ испытаний компьютерной модели, в ходе которых случайным образом изменяются входные сигналы и внешние воздействия. 3. Результаты вычислительного эксперимента обрабатываются статистическими методами, интерпретируются и анализируются. Выполнение этой программы не требует запоминания всех входных и выходных величин. В памяти ЭВМ запоминаются общие суммы различных исходов и количество испытаний, что позволяет рассчитать вероятности того или иного исхода, средние значения, дисперсию и т.д.

Основное преимущества статистического моделирования состоит в том, что оно позволяет исследовать поведение сложных систем, описывающихся громоздкими соотношениями, содержащих как непрерывные, так и дискретные элементы, на которые влияют различные случайные факторы. Результаты моделирования позволяют установить статистические характеристики изучаемой системы, закономерности ее поведения, оптимизировать процесс автоматического управления [13].

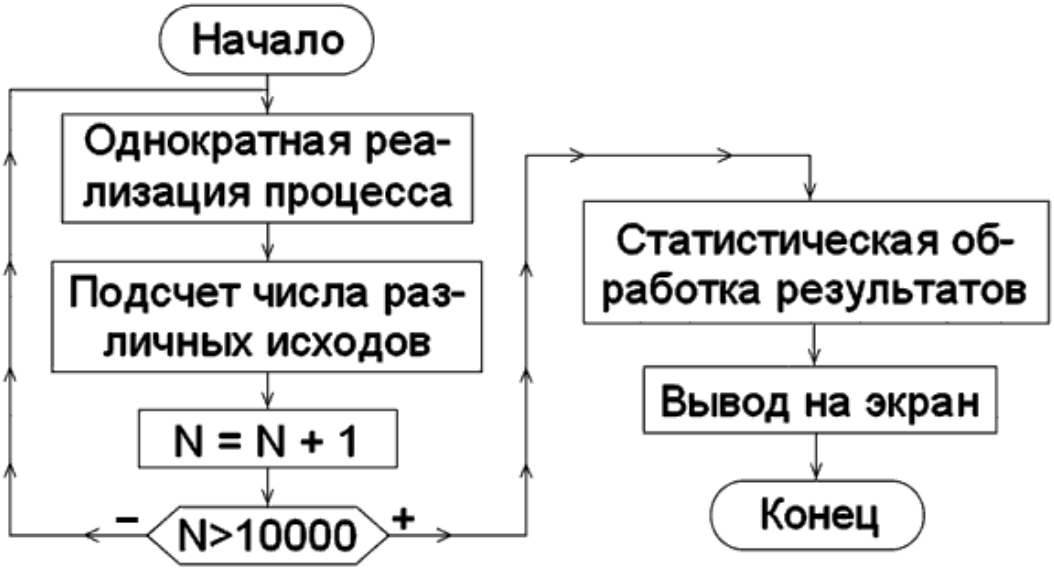

Рис. 1. Метод статистических испытаний.

Теоретической основой этого метода являются предельные теоремы теории вероятностей: теоремы Бернулли, Пуассона, Чебышева, Маркова, Лапласа, центральная предельная теорема [10, с. 109]. Не будем приводить здесь все формулировки, рассмотрим лишь утверждение, известное как **закон больших чисел**: если опыт с несколькими исходами повторяется много раз, то частота n_A / N исхода A стремится к его вероятности:

$$p_A = \lim_{N \to \infty} \frac{n_A}{N}.$$

Важным свойством случайного процесса является **эргодичность**. Оно состоит в том, что все реализации процесса имеют одинаковые статистические характеристики и поэтому одна реализация характеризует весь процесс. То есть математическое ожидание $M(x)$ и среднее квадратическое отклонение (СКО) $s(x)$ случайной величины x могут быть получены не только усреднением по множеству реализаций моделируемой системы, но и усреднением по времени одной реализации. В основе дискретно-стохастического подхода (Р-схемы) к формализации функционирования исследуемой системы лежит понятие вероятностного автомата, -- некоторого гипотетического устройства, которое при поступлении входного сигнала переходит из одного состояния в другое с заданной вероятностью. Вероятностный автомат можно рассматривать как дискретный потактный преобразователь информации с памятью; его переход в следующее состояние и выходной сигнал определяется состоянием памяти на данном шаге и стохастической матрицей переходов.

Введем понятие **вероятностного автомата** (ВА) [7, 10]. Допустим, имеется автомат, с множеством внутренних состояний Q и множествами входных и выходных сигналов X и Y. Введем множество A, состоящее из всевозможных пар (x_i, q_s) где x_i берется из множества входных сигналов X, а q_s -- из множества состояний Q. Зададим функцию перехода $q^{t+1} = \Theta(x^t, q^t)$ и функцию выходов $y^{t+1} = \Psi(x^t, q^{t+1})$. Тогда пятерка $F = < Q, X, Y, \Theta, \Psi >$ задает автомат детерминированного типа (F-автомат).

Теперь рассмотрим более общий случай. Пусть множество B включает в себя всевозможные пары (q_k, y_j). Предположим, что элементы множества A связаны с элементами из множества B случайным образом в соответствии со стохастической матрицей, представленной ниже.

Из множества A	Элементы из множества B				
	(q_1, y_1)	(q_1, y_2)	(q_2, y_3)	…	(q_K, y_J)
(x_1, q_1)	p_{11}	p_{12}	p_{13}	…	p_{1n}
(x_1, q_2)	p_{21}	p_{22}	p_{23}	…	p_{2n}
…	…	…	…	…	…
(x_I, q_S)	p_{m1}	p_{m2}	p_{m3}	…	p_{mn}

В этой матрице p_{13} -- вероятность перехода автомата в состояние q_2 и появления на его выходе сигнала y_3, если он был в состоянии q_1, и на его вход поступил сигнал x_1. Так как автомат обязательно должен перейти в какое-то состояние из множества Q и выдать выходной сигнал из множества Y, то сумма элементов каждой строки p_{ij} равна 1. Четверка $P =< Q, X, Y, B >$ задает вероятностный автомат (Р-автомат).

5.2. Способы получения случайных величин

Исследование стохастических моделей на ЭВМ требует получения случайных величин, имеющих тот или иной закон распределения. Для этого используются специальные **генераторы случайных чисел**, вырабатывающие равномерно распределенную случайную величину x из заданного интервала (например, $[0, 1]$). От качества генерируемых случайных чисел, характеризуемого степенью близости получающегося распределения к равномерному, часто зависит точность оценки определяемой характеристики изучаемого процесса. Поэтому необходимо найти и использовать простой способ формирования последовательности случайных чисел требуемого качества. Существуют три способа: аппаратный, табличный и алгоритмический [10].

Аппаратный или **физический способ** предполагает использование специального электронного блока, вырабатывающего хаотически изменяющееся напряжение, которое после оцифровки дает последовательность случайных чисел. В основе таких генераторов лежит случайный физический процесс, например, шум в электронных и полупроводниковых приборах, радиоактивных распад и другие. Допустим, рядом с радиоактивным препаратом находится датчик, регистрирующий пролет элементарной частицы и вырабатывающий импульс напряжения. Время t_j между соседними импульсами является случайной величиной. Преимущества состоят в том, что: 1) запас чисел не ограничен; 2) малы затраты машинного времени; 3) не требуется оперативная память ЭВМ. Недостатки метода: 1) необходимо специальное устройство, которое приходится периодически проверять, обеспечивать стабильность работы; 2) метод не позволяет воспроизводить одинаковые последовательности чисел.

Табличный или **файловый способ** состоит в создании таблицы случайных чисел и помещении ее в оперативную память ЭВМ в виде файла.

В процессе работы программа обращается к файлу и считывает случайное число из заданного диапазона. Метод имеет недостатки: 1) запас получаемых таким способом чисел ограничен; 2) нахождение данных в оперативной памяти уменьшает ее объем, который требуется для выполнения вычислительного эксперимента; 3) использование жесткого диска или другого внешнего ЗУ увеличивает временные затраты на исполнение программы, моделирующей изучаемую систему. Преимущества состоят в том, что: 1) достаточно однократной проверки качества случайных чисел; 2) метод позволяет воспроизводить их одинаковые последовательности.

Алгоритмический или **программный способ** предусматривает использование специальной программы, вырабатывающей случайное число из заданного интервала. Он позволяет многократно воспроизводить последовательность чисел, требует малого объема памяти и не использует внешних устройств. Недостатки заключаются в том, что: 1) через некоторый период происходит повторение генерируемых чисел; 2) использование этого метода требует определенных затрат машинного времени.

Моделирование систем на ЭВМ требует генерирования некоторых базовых случайных процессов, подчиняющихся тому или иному закону распределения. Так, при имитации функционирования систем массового обслуживания (СМО) обычно генерируют пуассоновский поток заявок. Для получения случайной величины, подчиняющейся тому или иному закону распределения, необходимо получить последовательность равномерно распределенных в интервале [0; 1] случайных чисел. У таких чисел **функции плотности** $f(x)$ и **распределения** $F(x)$ имеют вид:

$$f(x)=\begin{cases}1, & 0\le x\le 1,\\ 0, & x<0,\quad x>1;\end{cases}\qquad F(x)=\begin{cases}0, & x<0,\\ x, & 0\le x\le 1,\\ 1, & x>1.\end{cases}$$

В силу дискретности ЭВМ на ней можно оперировать с конечным набором величин и поэтому получить последовательность случайных величин, идеально соответствующую равномерному распределению, невозможно. Для генерации последовательности чисел на компьютере используется алгоритм, поэтому вычислительный процесс является детерминированным, а получающиеся числа **псевдослучайными**.

В идеале **генератор псевдослучайных чисел** должен выдавать последовательность чисел которые должны быть: 1) статистически независимыми; 2) воспроизводимыми; 3) неповторяющимися; 4) иметь квазиравномерное распределение; 5) получаться с минимальными затратами машинного времени; 6) занимать минимальный объем машинной памяти

[10]. Часто для получения последовательности случайных чисел используют рекуррентные соотношения вида $x_{i+1} = f(x_i)$ при заданном значении x_0. Качественные случайные числа получаются тогда, когда функция $x_{i+1} = f(x_i)$ достаточно плотно заполняет единичный квадрат.

5.3. Генераторы псевдослучайных величин

Рассмотрим основные методы генерации псевдослучайных чисел и кратко проанализируем их:

1. Метод серединных квадратов состоит в следующем: берут $2n$-разрядное число $x_i = 0, b_1 b_2 ... b_{2n}.$, которое меньше 1, и возводят его в квадрат, получая $4n$-разрядное число. Из него вырезают средние $2n$ разрядов и получают число $x_{i+1} = 0, b_{n+1} b_{n+1} ... b_{3n}$. Эта процедура повторяется требуемое количество раз. Недостаток метода состоит в наличии закономерности между числами получающейся последовательности.

2. Мультипликативный метод описывается рекуррентным соотношением $x_{i+1} = a x_i \bmod M$. В результате получается последовательность натуральных чисел из интервала от 0 до M. Удобно в качестве M брать P^r, где P -- основание системы счисления, используемой в ЭВМ, а r -- число бит в машинном слове.

3. Смешанный метод позволяет получить последовательность псевдослучайных целых чисел из интервала от 0 до M по формуле $x_{i+1} = (a x_i + b) \bmod M$.

Последние два метода используют к**онгруэнтную процедуру генерации** псевдослучайных чисел. Это название связано с понятием конгруэнтности по модулю M: два числа x и y обладают этим свойством, если существует целое k такое, что $x - y = kM$.

4. Метод Фибоначчи с запаздываниями состоит в использовании следующей рекуррентной формулы:

$$x_i = \begin{cases} x_{i-a} - x_{i-b}, & x_{i-a} \geq x_{i-b}, \\ x_{i-a} - x_{i-b} + 1, & x_{i-a} < x_{i-b}. \end{cases}$$

Здесь x_i -- действительные числа из диапазона [0; 1], а a и b -- лаги (целые положительные числа), причем $a > b$. Лаги специально подбираются так, чтобы в результате получалась равномерно распределенная случайная величина: $(a,b) = (55,24)$, $(17,5)$, $(97,33)$. Чем больше a, тем

выше качество получающейся последовательности. Если этот алгоритм реализовать через целые числа, то подойдет формула $x_i = x_{i-a} - x_{i-b}$; в этом случае будут происходить арифметические переполнения, обусловленные конечным числом разрядов, отводимых для записи числа в ОЗУ. Для работы этого генератора псевдослучайных чисел необходимо в памяти ЭВМ хранить $m = \max(a,b) = a$ предыдущих случайных чисел: x_{i-m-1}, x_{i-m}, x_{i-m+1}, … , x_{i-1}. Преимущество этого метода в том, что он не требует умножения, а недостатки, -- необходимость большого объема памяти и исходного массива чисел для запуска программы.

Качество получающихся псевдослучайных чисел может быть оценено только путем проведения вычислительного эксперимента. Начиная с некоторого момента, генератор переходит в исходное состояние и затем начинает повторяться. Важной характеристикой программных генераторов псевдослучайных чисел является длина периода L (длина отрезка апериодичности). Если в статистическом эксперименте использовать последовательность чисел, длина которой превышает отрезок апериодичности L, то испытания компьютерной модели будут просто повторяться. С целью увеличения длины периода L используются рекуррентные формулы вида $x_{i+1} = F(x_i, x_{i-1}, x_{i-2}, ..., x_{i-k+1})$, в которых последующее значение вычисляется исходя из нескольких предыдущих, как в методе Фибоначчи. Иногда, чтобы получить неповторяющуюся последовательность случайных чисел, работу генератора связывают с показаниями системного таймера ЭВМ.

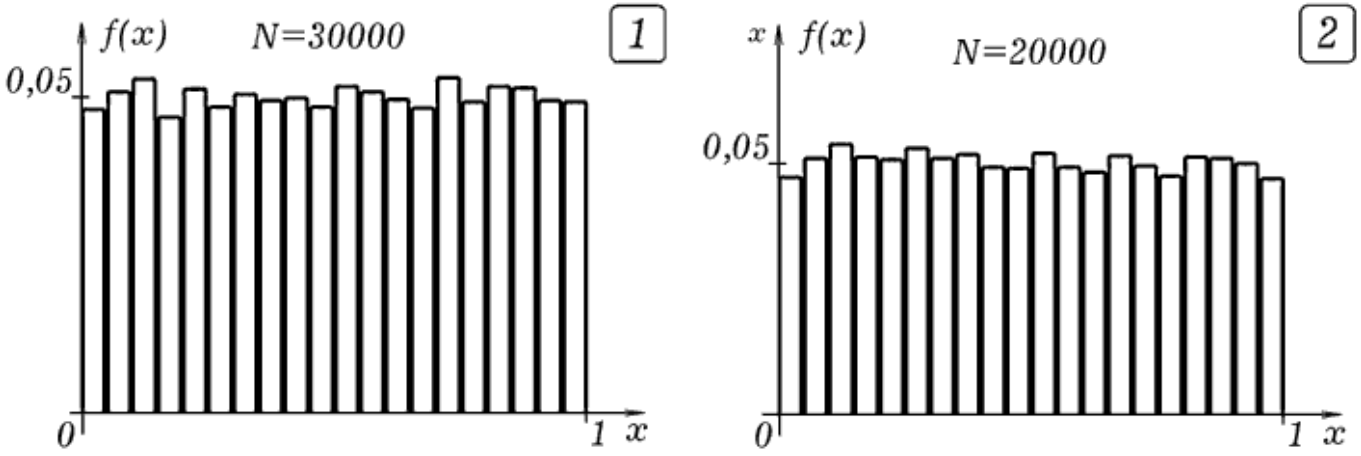

Рис. 2. Практически равномерные распределения случайных чисел.

Рассмотрим фибоначчиевый генератор псевдослучайных чисел с лагами $a = 53$ и $b = 24$ (программа ПР-1). В первом цикле с помощью оператора **random()** генерируется последовательность из 55 чисел, которые сохраняются в массиве **x[i].** Во втором цикле реализуется метод Фибо-

наччи, каждое следующее случайное число сохраняется в переменной **y**. Чтобы оценить качество случайных чисел, получим их распределение чисел в интервале $[0,\ 1]$. Для этого диапазон от 0 до 1 разобьем на 20 поддиапазонов шириной $0{,}05$ и будем подсчитывать количество "попаданий" случайного числа y внутрь каждого поддиапазона. Последний цикл в программе ПР-1 выводит гистограмму распределения на экран (рис 2.1).

Важный практический метод получения псевдослучайных чисел состоит в использовании системного таймера ЭВМ. В качестве примера рассмотрим программу ПР-2 (Приложение). Она содержит цикл, в котором выполняется последовательность действий: 1) с помощью оператора **GetTime(h,m,s,hs)** определяется системное время ЭВМ с точностью до сотых секунды hs; 2) рассчитывается значение случайной величины y и отбрасывается ее целая часть с помощью операторов:

```
y:=1.11*y+hs/100+(hs/100)*random(1000)/1000; y:=y-int(y);.
```

Программа строит гистограмму распределения получающейся случайной величины y (рис. 2.2); видно, что оно близко к равномерному.

5.4. Моделирование случайных событий и процессов

Чтобы промоделировать свершение случайного события A с вероятностью p_A необходимо рассмотреть попадание значения x_i случайной величины x, равномерно распределенной в интервале $[0;\ 1]$, в числовой интервал $[0;\ p_A]$ (рис. 3.1). Для моделирования наступление полной группы несовместимых случайных событий A, B, C, … с вероятностями $p_A,\ p_B,\ p_C,\ \ldots$, используют **метод определения исходов по жребию**. Он состоит в следующем (рис. 3.2): 1) интервал $[0;\ 1]$ разбить на n частей с длинами $p_j,\ j=1,2,...,n$ (сумма всех p_j равна 1); 2) сгенерировать значения x_i равномерно распределенной случайной величины x; 3) для каждого x_i определять номер j отрезка, в который она попала. При этом выполняется условие $l_{j-1} < x_i \le l_j$, где $l_j = p_1 + p_2 + p_3 + ... + p_j$.

Для получения последовательности из N значений дискретной случайной величины x, принимающей значения $x_1 < x_2 < \ldots < x_i < \ldots$ с вероятностями p_i $(i=1,2,3,...n)$, необходимо организовать цикл, совершающий N итераций. В результате каждой итерации реализуется метод определения исходов по жребию и находится следующее значение x_i.

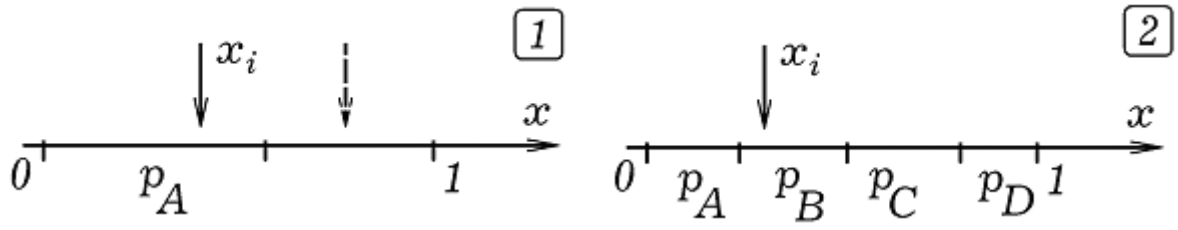

Рис. 3. Метод определения исходов по жребию.

В общем случае, вероятность перехода в следующее состояние на шаге $t+1$ зависит от текущего состояния системы на шаге t. В этом случае говорят о **стохастическом процессе**. В качестве примера рассмотрим такую игру [8, с. 221] : бросают монету, если получается орел, то ее бросают снова. В случае, когда решка, бросают игральную кость (кубик с пронумерованными гранями). Если выпало четное число очков, то снова бросается монета, а если нечетное, -- игральная кость. Промоделируем поведение данной системы и с помощью метода статистических испытаний определим эмпирические частоты различных исходов опыта.

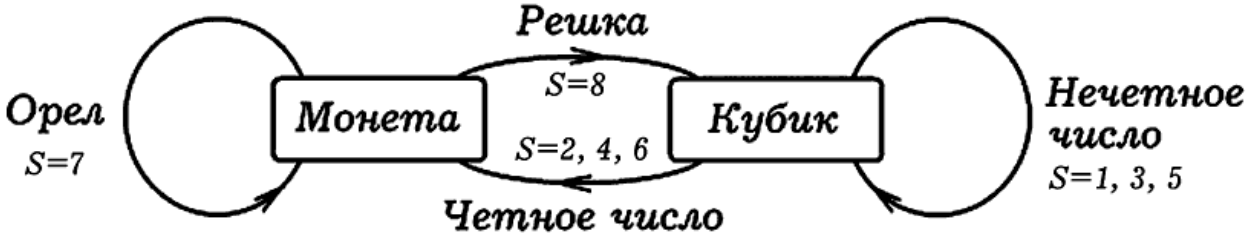

Рис. 4. Граф, описывающий смену состояний системы.

Поведение системы описывается графом, представленном на рис. 4. На его основе может быть составлена программа ПР-3 (приложение), в которой, исходя из записанного выше правила, определяется состояние системы на следующем шаге $t+1$. Вот типичный результат работы программы: "OREL – RESHKA – 6 – RESHKA – 1 – 4 – OREL – OREL – RESHKA – 2 – OREL …". В программе в массиве n[k] подсчитывается количества различных исходов опыта и находятся их эмпирические вероятности p_i ($i=1,2,...,8$), которые затем выводятся на экран. При 10000 испытаний могут получиться следующие эмпирические значения p_i: $0{,}0911$, $0{,}0843$, $0{,}085$, $0{,}0805$, $0{,}0877$, $0{,}0824$, $0{,}241$, $0{,}247$. Они соответствуют теоретически ожидаемым вероятностям.

Часто для изучения стохастического процесса используют марковские цепи. **Простая однородная цепь Маркова**, состоящая из событий A_1, A_2, A_3, …, A_k может быть задана стохастической матрицей переходов:

$$P=\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ p_{21} & p_{22} & \cdots & p_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ p_{k1} & p_{k2} & \cdots & p_{kk} \end{pmatrix}, \quad 0 \le p_{ij} \le 1, \quad \sum_{j=1}^{k} p_{ij} = 1, \quad i = 1,2,...,k.$$

Матрица P определяет вероятностный автомат с k внутренними состояниями, которые можно пронумеровать числами от 1 до k. Элемент матрицы p_{ij} равен условной вероятности перехода в j-ое состояние (наступления события A_j) в случае, когда исходом предыдущего испытания было i-ое состояние (событие A_i). Моделирующая программа должна содержать цикл, в котором методом определения исходов по жребию, учитывая текущее состояние автомата в момент t, устанавливается его состояние в момент $t+1$. Из начальных вероятностей p_1^0, p_2^0, p_3^0, ..., p_k^0 случайно выбирается номер r и соответствующее ему начальное состояние q_r^0. Исходя из вероятностей p_{r1}, p_{r2}, p_{r3}, ..., p_{rk}, по жребию определяют следующее состояние q_r^1. Каждый номер r определяет не только очередное событие q_r^t на шаге t, но и распределение вероятностей p_{r1}, p_{r2}, p_{r3}, ..., p_{rk} для выбора следующего состояния на шаге $t+1$.

Марковские цепи позволяют промоделировать процессы, которые имеют такие свойства: 1) конечное множество состояний; 2) дискретное время; 3) вероятностный характер переходов; 3) вероятность перехода из текущего состояния в следующее не зависит от предыстории. Цель моделирования заключается в нахождении вероятности того, что исследуемый процесс на $k-$ом шаге окажется в каком-то конкретном состоянии q_r^k. В качестве примера рассмотрим задачу о поведении игрока, владеющего двумя стратегиями A и B. Допустим, известно, что если на предыдущем шаге (в предыдущей игре) игрок выбрал стратегию A, то вероятность выбора стратегии A на текущем шаге равна $0{,}62$, а стратегии B -- $0{,}38$. Если же игрок выбрал стратегию B, то на следующем шаге он с вероятностью $0{,}43$ выбирает стратегию A, а с вероятностью $0{,}57$ -- снова B. Несложно нарисовать граф переходов и создать компьютерную модель анализируемой системы. Смена состояний системы описывается стохастической матрицей переходов:

$$P=\begin{pmatrix} 0{,}62 & 0{,}38 \\ 0{,}43 & 0{,}57 \end{pmatrix}.$$

Проведя серию из 1000 испытаний, можно установить, с какой вероятностью игрок выбирает стратегию A или B.

Рассмотрим еще один пример использования цепей Маркова. Для того, чтобы промоделировать диффузию некоторого газа, наполняющего сосуд, представим себе две урны, в которых лежат n шаров, что соответствует n молям вещества. Пусть в первой урне k шаров, тогда во второй $n-k$ шаров. Это означает, что в левой половине сосуда k молей газа, а в правой -- $(n-k)$ молей газа. Данное состояние обозначим так: $\{1:k;\ 2:(n-k)\}$. Предположим, что поведение системы описывается правилом: в следующий дискретный момент времени $t+1$ система переходит в состояние $\{1:k+1;\ 2:(n-k-1)\}$ с вероятностью $(n-k)/n$, либо в состояние $\{1:k-1;\ 2:(n-k+1)\}$ с вероятностью k/n. Этот алгоритм реализован в программе ПР- 4. При ее запуске происходит постепенное выравнивание числа шаров (молей, молекул) в урнах (половинках сосуда).

5.5. Примеры использования метода статистических испытаний

Рассмотрим несколько задач, решение которых требует создания компьютерной модели объекта и проведения серии из N испытаний с последующей статистической обработкой результатов [2, 7, 9, 12]. Последнее предполагает нахождение математического ожидания случайной величины x_i и среднеквадратического отклонения (СКО) s по формулам:

$$x_{cp} = (x_1 + x_2 + x_3 + ...x_N)/N, \qquad s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - x_{cp})^2}.$$

Задача 1. Технологический процесс состоит из 5 операции. Длительность и вероятность правильного выполнения каждой операции задаются матрицами: $\tau_i = \{1{,}6;\ 2{,}7;\ 1{,}4;\ 3{,}8;\ 2{,}6\}$ и $p_i = \{0{,}6;\ 0{,}7;\ 0{,}4;\ 0{,}8;\ 0{,}6\}$. Если операция выполнена неверно, то она повторяется снова. Необходимо вычислить среднее время выполнения технологического процесса и дисперсию этой величины.

Используется программа ПР- 5. Она содержит цикл, в котором моделируется однократное выполнение технологического процесса. Выполнение операции моделируется с помощью генератора случайных чисел. В случае правильного выполнения операции, номер выполняемой

операции увеличивается на 1. Если операция выполнена неверно, то осуществляется повторная попытка ее выполнения. В результате определяется суммарное время i-ой реализации технологического процесса. Затем осуществляется следующая $(i+1)$-ая реализация и так 2000-5000 раз. В результате вычисляется суммарное время выполнения всех M реализаций, среднее значение и СКО времени τ_p одной реализации. Программа также позволяет понять характер распределения величины τ_p и, после небольшой доработки, получить соответствующую гистограмму.

Задача 2. На вход канала связи поступает последовательность символов из трехбуквенного алфавита $A=\{a_1, a_2, a_3\}$, вероятности использования которых известны. На выходе канала связи получается поток символов из алфавита $A'=\{a_1, a_2, a_3, b\}$, где b -- ошибочный символ. Статистические свойства канала связи задаются стохастической матрицей. Необходимо промоделировать передачу информации по каналу связи и методом статистических испытаний определить относительную ошибку.

Допустим, что вероятности использования букв алфавита A определяются матрицей $p_i = (0{,}3;\ 0{,}25;\ 0{,}45)$, а стохастическая матрица, характеризующая канал связи, имеет вид:

$$P = \begin{pmatrix} 0{,}7 & 0{,}1 & 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}8 & 0{,}1 & 0{,}0 \\ 0{,}2 & 0{,}2 & 0{,}5 & 0{,}1 \end{pmatrix}.$$

Ее элементами являются вероятности $p_{i,j}$ того, что при входе в канал связи i-ой буквы из алфавита A на его выходе появится j-ая буква из алфавита A' ($i=1,2,3;\ j=1,2,3,4$). Для изучения передачи сообщений по такому каналу связи используется программа ПР-6. Она содержит цикл, в котором методом выбора по жребию определяется входной символ, а затем с помощью матрицы вероятностей P разыгрывается выходной символ. При этом осуществляется подсчет относительного числа ошибок $n_{ош}/n$, и результат выводится на экран. Экспериментируя с программой, можно установить, что после 2000 – 3000 испытаний результаты вычисления $n_{ош}/n$ приобретают статистическую устойчивость.

В качестве еще одного примера рассмотрим задачу о **ячеистой перколяции** [1, 4], имеющую большое значение для теории перколяции (просачивания или протекания). Эта теория описывает физические процессы, происходящие в неоднородных средах со случайными свойствами. К ней относятся исследования электропроводности композиционных материа-

лов, состоящих из проводящих и непроводящих компонентов, изучения фазовых переходов газа в жидкость, образования магнитных доменов, диффузии в неупорядоченных средах и т.д.

Задача 3. Представим себе металлическую сетку $M \times M$, расположенную между двумя электродами, из которой случайным образом с заданной вероятностью q удалены некоторые узлы или ячейки (а с вероятностью $p = 1 - q$ заняты). При больших q будет вырезано слишком много узлов, и исчезнет путь, соединяющий нижний электрод с верхним, -- сетка перестанет проводить ток. Если вырезано мало узлов, то будет существовать один или несколько **перколяционных кластеров**, пронизывающих данную структуру насквозь и соединяющих электроды. Этот переход из состояния, при котором нет соединяющего кластера в состояние, содержащее соединяющий кластер, является **фазовым переходом.** Проблема состоит в нахождении **порога перколяции** p_c, то есть критического значения вероятности наличия узла $p = 1 - q$, при котором возникает хотя бы один кластер из бесконечного числа узлов.

Совокупность занятых ячеек, связанных с ближайшими по стороне, образуют **кластер**. Две ячейки относятся к одному кластеру, если существует путь из занятых ячеек, соединяющий их друг с другом. На рис. 5.1 ячейки, входящие в один кластер закрашены одним цветом. Красные ячейки образую перколяционный кластер, пронизывающий всю структуру и соединяющий верхний и нижний электроды. Найдем зависимость вероятности P образования перколяционного кластера от вероятности p наличия узла.

Для этого используется метод статистических испытаний, реализованный в программе ПР-7. Сначала, исходя из заданной вероятности p наличия занятой ячейки, случайным образом формируется ячеистая структура (рис. 5.1), после чего определяется, содержит она перколяционный кластер или нет. Эта процедура многократно повторяется, что позволяет определить вероятность перколяции P при данном значении p. Затем проводится аналогичный вычислительный эксперимент при других p и строится график зависимости $P(p)$.

Для установления факта наличия или отсутствия перколяции создается **список связности**, в котором существующим (не вырезанным) узлам сетки, входящим в один кластер, присваивается один и тот же номер. Программа находит существующий узел (i, j) и присваивает ему номер кластера $N_{кл} = 1$, который записывается в специальный массив g[k]. По-

сле этого перебираются все существующие узлы (ячейки), и если они являются смежными, то есть удалены от узла (i, j) на 1, то им тоже присваивается номер $N_{кл}=1$. Если узел $(i, j+1)$ существует (является смежным), то для него повторяют эту процедуру, находя все узлы смежные с ним. В результате все узлы, относящиеся кластеру с номером $N_{кл}=1$, получают номер 1, который хранится в массиве g[k].

После этого программа, перебирая все существующие узлы, находят любой узел, у которого номер $N_{кл}=0$, то есть он не отнесен к первому кластеру. Ему присваивается номер $N_{кл}=2$, и этот же номер получают все остальные существующие узлы, образующие с ним один кластер. После этого, снова перебирая все существующие узлы сетки, находят любой узел, у которого номер $N_{кл}=0$, то есть он не отнесен ни к первому, ни ко второму кластеру. Ему присваивают номер $N_{кл}=3$ и т.д.

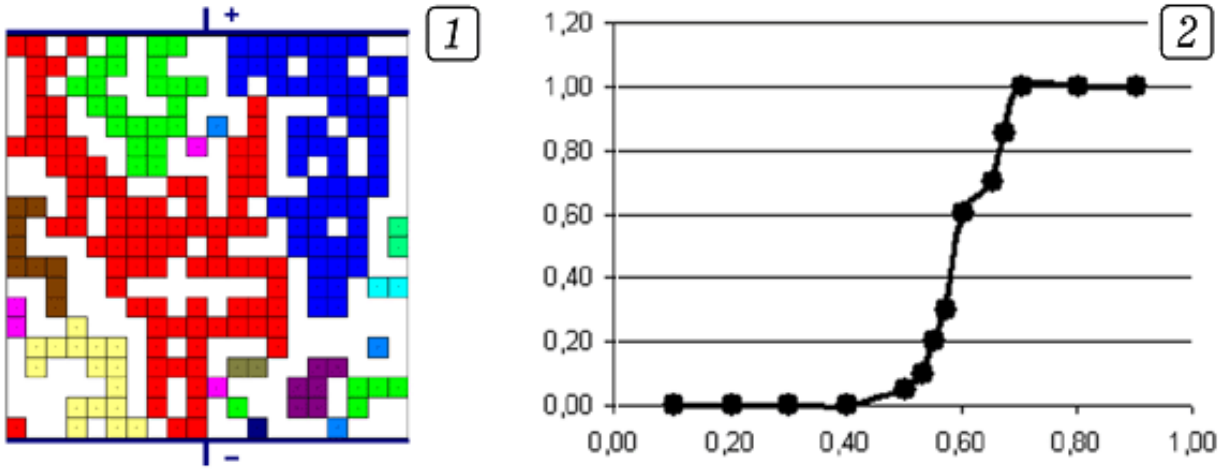

Рис. 5. Изучение перколяции методом статистического моделирования.

После того, как все занятые ячейки отнесены к тому или иному кластеру и имеют номера, записанные в массиве g[k], устанавливается факт существования перколяционного кластера. Перебирая все узлы, ищут такие два, для которых выполняются условия: 1) один узел контактирует с верхней пластиной ($j=1$), а другой с нижней ($j=M$); 2) оба узла относятся к одному и тому же кластеру, то есть имеют одинаковые номера $N_{кл}$. При перколяции на экран выводится 1, при ее отсутствии -- 0.

Для нахождения вероятности P образования пронизывающего кластера процедура формирования новой ячеистой структуры и установления факта существования перколяции многократно повторяется. В программе ПР-7 выполняется $S=20-50$ испытаний, получающиеся результаты выводятся на экран. Это позволяет оценить вероятность P перколяции при

различных p и построить график (рис. 5.2). Из него видно, что с ростом p от $0{,}4$ до $0{,}7$ резко увеличивается вероятность перколяции P. Для увеличения статистической значимости результатов следует провести большее количество испытаний.

5.6. Модель обучения вероятностного автомата

Создание роботов и систем искусственного интеллекта потребовало решения проблемы обучения и самообучения нейросетей и вероятностных автоматов. Рассмотрим использование Р-схемы для моделирования процесса формирования навыка у человека или робота. Допустим, "ученик" должен научиться выполнять определенную последовательность операций: операция 1, затем операция 2, после этого операция 3, затем снова операция 1 и т.д. Формирование этого навыка у человека происходит аналогично обучению вероятностного автомата (ВА) с тремя состояниями, которые соответствуют операциям 1, 2 и 3. До обучения ВА случайным образом выполняет различные операции, совершая при этом ошибки. Алгоритм его работы имеет вид стохастического (вероятностного) графа -- совокупности вершин, соединенных стрелками, которые соответствуют переходам от одной операции к другой. Вероятности p_{ij} перехода от i-ой операции к j-ой образуют двумерную матрицу вероятностей. Если автомат не обучен, то вероятности всех переходов равны:

$$P = \begin{pmatrix} 0{,}33 & 0{,}33 & 0{,}33 \\ 0{,}33 & 0{,}33 & 0{,}33 \\ 0{,}33 & 0{,}33 & 0{,}33 \end{pmatrix},$$

то есть он выбирает следующую операцию совершенно произвольно.

За работой обучаемого ВА следит "учитель" -- детерминированный автомат, знающий правильную последовательность действий и функционирующий по следующему жесткому алгоритму. Если "ученик" совершил правильное действие, то его "поощряют" высокой оценкой 1, в результате чего вероятность повторения этого действия увеличивается, а остальных -- уменьшается. В случае ошибки "ученика" "наказывают" оценкой 0, что приводит к уменьшению вероятности ее повторения. В результате обучения вероятности правильных переходов p_{12}, p_{23}, p_{31} возрастают, стремясь к 1, а вероятности остальных ошибочных действий -- уменьшаются, приближаясь к 0. Матрица вероятностей стремится к виду:

$$P=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{pmatrix}.$$

В конце обучения автомат практически без ошибок выполняет требуемую последовательность $1 \to 2 \to 3 \to 1 \to 2 \to 3 \to 1$... Аналогичным образом осуществляется обучение нейросети: ей многократно предъявляют различные объекты, в случае их правильного узнавания веса активных связей увеличивают, а в случае неправильного -- уменьшают.

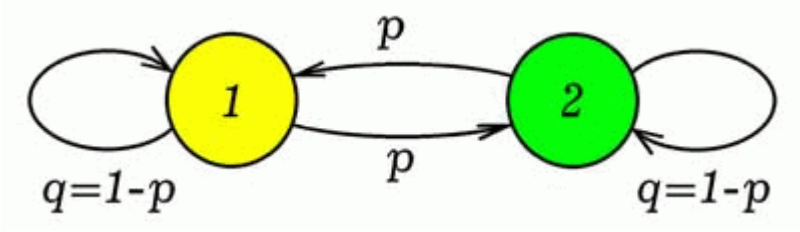

Рис. 6. Диаграмма Мура автомата с двумя состояниями.

Рассмотрим ВА с двумя состояниями [5], соответствующими операциям 1 и 2 (рис. 6). Будем считать, что автомат обучен, когда из состояния 1 он переходит в состояние 2, а из состояния 2 -- в состояние 1 и т.д.: $1 \to 2 \to 1 \to 2 \to 1 \to$... Переходы $1 \to 1$ и $2 \to 2$ являются ошибочными. Вероятность правильного действия обозначим через p, тогда вероятность ошибки равна $q = 1 - p$. Можно изучать работу вероятностного автомата с большим числом состояний, но при этом на любом шаге t один из переходов будет правильным, а остальные -- неправильными.

Для моделирования процесса обучения используется программа ПР-8. Изначально автомат необучен, вероятность правильного действия мала ($p = 0{,}01$). Программа содержит цикл, в котором выбор каждой операции осуществляется с помощью генератора случайных чисел. Если случайное число x из интервала $[0;\ 1]$ меньше p, то "ученик" совершает правильное действие, если нет, -- делает ошибку. Процесс обучения приводит к изменению матрицы вероятностей: вероятность правильного выбора p увеличивается на aq, где a -- коэффициент научения ($0 < a < 1$), а вероятность ошибки уменьшается на ту же величину: **`p:=p+a*q; q:=q-a*q;`**. Уровень сформированности навыка можно считать равным вероятности p выбора правильной операции.

При обучении человека часть информации забывается. Чтобы это учесть, на каждом временном шаге будем уменьшать вероятность правиль-

ного действия p на gp, где g -- коэффициент забывания ($0<g<1$), и на такую же величину увеличивать вероятность ошибки q: **p:=p-g*p; q:=q+g*p;**. Результаты работы программы представлены на рис. 3. При этом промоделированы следующие ситуации:

1. Обучение с поощрением: при выполнении правильного действия ученика "поощряют", пересчитывая вероятности p и q. Так как сначала ученик ошибается гораздо чаще ($q>p$), то при малых t обучение происходит медленно (рис. 7.1). Зато по мере увеличения "знаний" вероятность совершения правильного действия растет. Акты обучения происходят все чаще, вероятность p увеличивается почти до 1. Программа должна содержать оператор:

```
If (x<p)and(t<4000) then
    begin p:=p+a*q; q:=q-a*q; end;
```

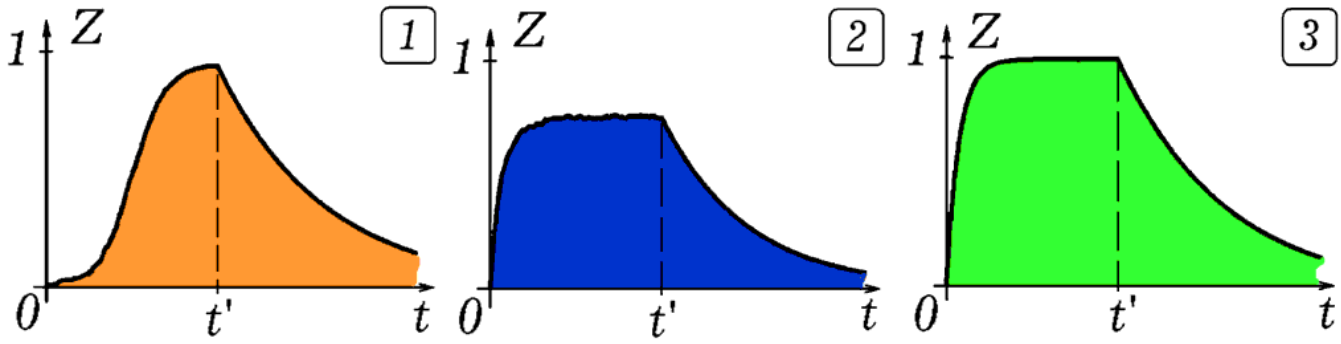

Рис. 7. Зависимости уровня знаний "ученика" от времени.

2. Обучение с наказанием: в случае ошибки ученика "наказывают", подсказывая ему правильный ответ, что приводит к росту p и уменьшению q. Сначала ученик ошибается часто, поэтому уровень его "знаний" быстро растет, вероятность ошибки q падает (рис. 7.2). Акты обучения происходя реже, уровень знаний за счет забывания не достигает 1. Программа должна содержать условный оператор:

```
If (x>p)and(t<4000) then
    begin p:=p+a*q; q:=q-a*q; end;
```

3. Обучение с поощрением и наказанием: при правильном ответе ученика поощряют, а при неправильном наказывают, подсказывая правильный ответ. В обоих случаях вероятность правильного действия p растет, а вероятность ошибки q снижается. Так как при любом действии

учащегося его учат, то уровень знаний быстро растет и достигает 1 (рис. 7.3). Программа должна содержать оператор:

```
If t<4000 then begin p:=p+a*q; q:=q-a*q; end;
```

Дискретно-стохастическое моделирование позволяет исследовать и более сложные обучающиеся системы.

5.7. Вероятностные клеточные автоматы

В отличие от детерминированных КА вероятностные клеточные автоматы подчиняются правилам, в которые внесен элемент случайности [1]. Выше была проанализирована простейшая компьютерная модель диффузии. Ее можно усложнить, рассмотрев деление длинного сосуда не на две, а на $M = 100$ частей. В этом случае система может быть промоделирована одномерным вероятностным клеточным автоматом, состоящим из M ячеек. Состояние каждой i-ой ячейки равно числу молекул (молей) k_i, находящихся в i-ой части сосуда и принимает значения от 0 до 3000 (или больше). На каждом дискретном временном шаге перебираются все соседние КА и определяется сумма $S_{i,i+1} = k_i + k_{i+1}$ молекул в соседних i-ой и $(i+1)$-ой частях сосуда. Переменной d, равной числу молекул, переходящих из i в $i+1$ части или наоборот, присваивается результат округления $a\,|\,k_i - k_{i+1}\,|$. Коэффициент a $(0 < a < 1)$ определяет скорость диффузии.

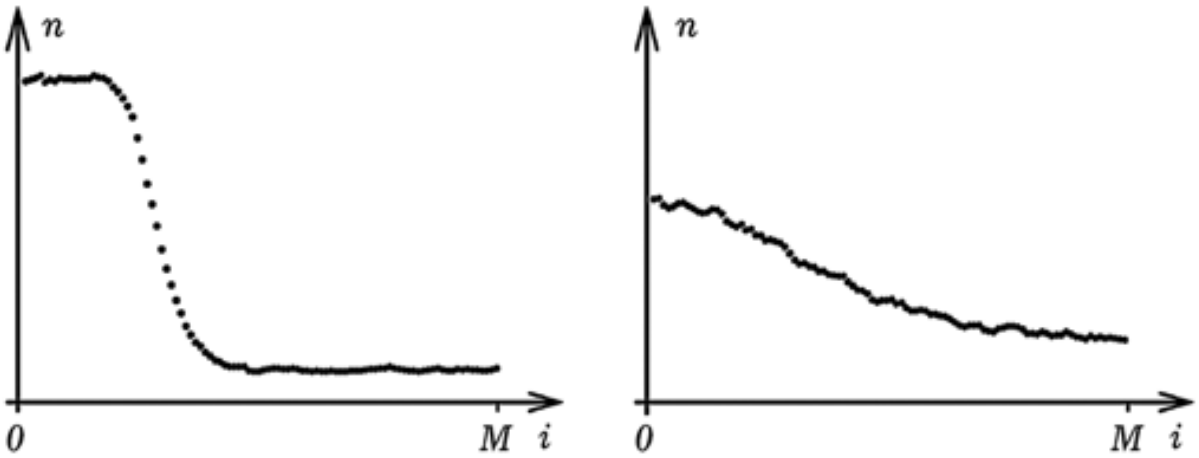

Рис. 8. Моделирование диффузии с помощью вероятностных КА.

После этого генерируется случайное число x и рассчитывается состояния КА с номерами i и $(i+1)$ с помощью оператора:

```
If x<k[i]/S then begin k[i]:=k[i]-d; k[i+1]:=k[i+1]+d; end
      else begin k[i]:=k[i]+d; k[i+1]:=k[i+1]-d; end; end
```

Периодически результаты вычислений усредняются и выводятся на экран (рис. 8). Аналогичным способом можно промоделировать теплопроводность стержня. Чтобы сымитировать источник тепла в узле с номером $i=80$, необходимо периодически k_{80} увеличивать на 1.

В главе 4 обсуждалась программа "Жизнь", созданная на основе детерминированных клеточных автоматов. В ней реализуется жесткий алгоритм, поэтому любая начальная конфигурация живых клеток через одно и тоже число шагов всегда приведет к одной и той же конечной конфигурации. На самом деле процесс размножения бактерий является вероятностным и может быть промоделирован методом вероятностных клеточных автоматов с помощью программы ПР-9.

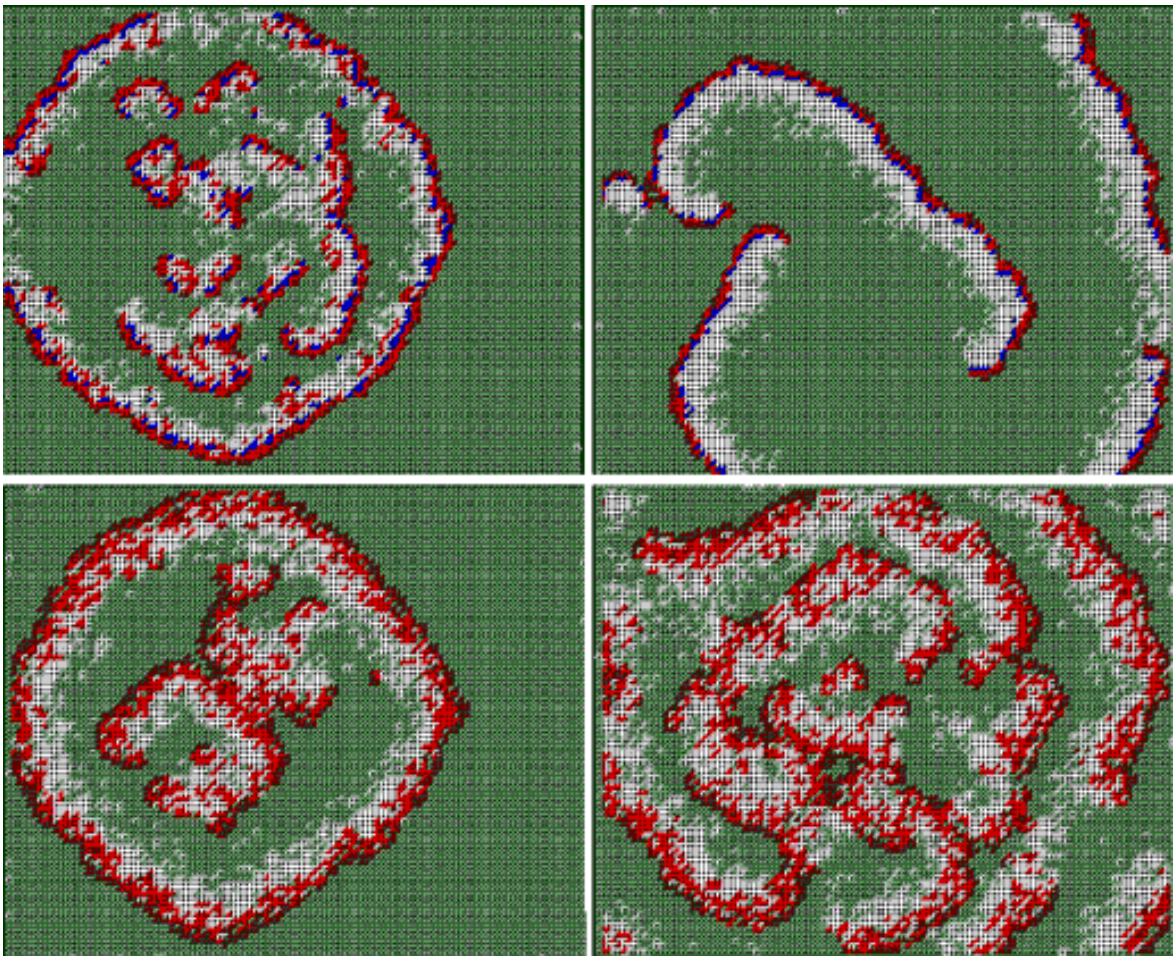

Рис. 9. Рост колонии бактерий: вероятностный КА.

Рассмотрим правила, которые реализуются в этой программе. Каждая ячейка может находиться в 6 живых (возбужденных) состояниях $s=1,2,...,6$ или в мертвом состоянии $s=0$. Живая клетка потребляет полезные вещества из питательной среды (их количество в каждой ячейке записы-

вается в массиве p[i, j]), и остается живой 6 тактов, после чего умирает от старости. При этом количество полезных веществ p_{ij} в данной ячейке с каждым тактом уменьшается на Δp_1. Если $p_{ij} = 0$, то клетка погибает от голода. Если ячейка мертва $a_{ij} = 0$, а количество питательных веществ меньше заданного уровня U_2, то оно с каждым шагом повышается на некоторую положительную случайную величину Δp_2: $p_{ij} = p_{ij} + \Delta p_2$. Чтобы ячейка a_{ij} ожила, ей необходимо иметь 3 или более живых соседа, а уровень питательных веществ p_{ij} не должен быть меньше U_1. Запишем эти правила a_{ij} в математическом виде для детерминированного КА:

$$a_{ij}^{t+1} = \begin{cases} 0, & \textit{если} \quad a_{ij}^t = 0 \quad u \quad p_{ij}^t < U_1, \\ 0, & \textit{если} \quad a_{ij}^t > 0 \quad u \quad p_{ij}^t = 0, \\ 0, & \textit{если} \quad a_{ij}^t = 6, \\ 1, & \textit{если} \quad a_{ij}^t = 0 \quad u \quad p_{ij}^t \geq U_1 \quad u \quad S > 2, \\ a_{ij}^t + 1, & \textit{если} \quad a_{ij}^t < 6 \quad u \quad p_{ij}^t > 0. \end{cases}$$

В обсуждаемом нами вероятностном КА предпоследняя строчка, соответствующая "рождению" клетки должна выполняться случайно, например, с заданной вероятностью $0{,}7$.

Рассмотренный выше алгоритм реализован в программе ПР-10. Результаты моделирования приведены на рис. 9. При определенных параметрах модели происходит автоволновой процесс, при котором в рассматриваемой области возникают спиральные автоволны. В других случаях по среде распространяется одиночная волна возбуждения, либо все живые ячейки быстро погибают от недостатка питания.

Теперь рассмотрим другую ситуацию. Пусть имеется двумерная питательная среда, разбитая на квадратные ячейки, в которых могут жить бактерии (насекомые, черви). Допустим, что бактерии из клетки (i, j) потребляют питательные вещества из соседних четырех клеток $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$. Если в ячейке находятся бактерии, то ее будем считать возбужденной или живой (возраст $a_{ij} = 1,2,3,...$), а в противном случае -- мертвой ($a_{ij} = 0$). Перейдя в живое состояние, клетка будет находиться в нем сколь угодно долго, при этом ее возраст a_{ij} с каждым шагом увеличивается на 1. Бактерии с некоторой вероятностью могут переходить в смежную клетку, например, в ячейку $(i+1, j)$, которая при этом

тоже станет живой или возбужденной (была $a^{t}_{i+1,j}=0$, стала $a^{t+1}_{i+1,j}=1$). Если ячейка имеет несколько живых соседей, "высасывающих" из нее питательные вещества, то с каждым шагом t вероятность перехода ее в возбужденное состояние уменьшается.

В программе ПР-11 перебираются все ячейки, для каждой находится сумма S возрастов 4 возбужденных соседей, и результаты накапливаются в массиве b_{ij} ($b^{t+1}_{ij}=b^{t}_{ij}+S$). Если $b_{ij}=0$, то есть у клетки нет причин оживать, вероятность перехода в живое состояние p_{ij}. Если b_{ij} больше 0, но не превышает 3, то клетка (i,j) на следующем шаге возбуждается с вероятностью p_{ij}, которая уменьшается с ростом b_{ij}. Если $b_{ij}>3$, то клетка не может возбудиться, так как в ней мало питательных веществ, вероятность $p_{ij}=0$. Рассчитанные вероятности сохраняются в массиве p_{ij}.

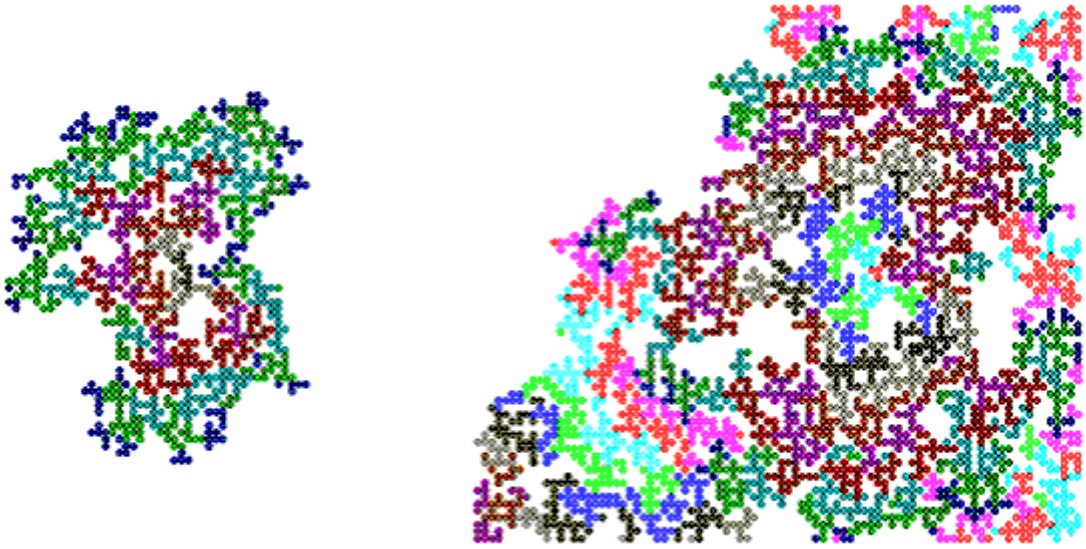

Рис. 10. Стохастический фрактал.

После этого ЭВМ перебирает все мертвые ячейки и, исходя из найденных вероятностей p_{ij}, с помощью генератора случайных чисел определяет их состояние на следующем временном шаге. При этом состояние a_{ij} живых клеток, соответствующий возрасту, увеличивается на 1. Если сделать так, чтобы цвет клетки зависел от ее возраста, то на экране получится стохастический фрактал, переливающийся различными цветами (рис. 10).

Приложение

В приложении представлены тексты программ, позволяющие решить рассмотренные выше задачи. Они написаны в средах Borland Pascal 7.0 и Free Pascal 1.0.10.

ПР - 1

```
uses crt, graph;
var a,b,i,j,k,DV,MV,EC: integer; y: real;
n: array[0..25]of integer; x: array[1..55]of real;
BEGIN
DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi'); EC:=GraphResult;
For i:=1 to 55 do x[i]:=random(10000)/10000; a:=53; b:=24;
Repeat inc(k);
  If x[56-a]>=x[56-b] then Y:=x[56-a]-x[56-b]
                      else Y:=x[56-a]-x[56-b]+1;
  For i:=1 to 54 do x[i]:=x[i+1]; x[55]:=y;
  For j:=0 to 20 do If (y>=j*0.05)and(y<(j+1)*0.05)
                                    then inc(n[j]);
until (k>30000)or(KeyPressed);
For i:=0 to 22 do rectangle(50+20*i,
                    450-round(n[i]/6),68+20*i,450);
Repeat until KeyPressed; CloseGraph;
END.
```

ПР-2

```
uses crt, dos,graph;
var a,b,i,j,DV,MV,EC: integer; k:longint;
h,m,s,hs: word; y: real;
n:array[0..25]of integer;
BEGIN randomize;
DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi'); EC:=GraphResult;
For j:=0 to 22 do n[j]:=0;
Repeat inc(k); GetTime(h,m,s,hs);
  y:=1.11*y+hs/100+(hs/100)*random(1000)/1000;
  y:=y-int(y);
  For j:=0 to 22 do If (y>=j*0.05)and(y<(j+1)*0.05)
                                    then inc(n[j]);
until (k>20000)or(KeyPressed);
For i:=0 to 22 do rectangle(50+20*i,
                  450-round(n[i]/5),68+20*i,450);
Repeat until KeyPressed; CloseGraph;
END.
```

ТТР-3

```
uses crt;
var x: real; q,S,k,L: integer;
n:array[1..8]of integer; Label m;
BEGIN clrscr; Randomize; q:=1; L:=1000;
Repeat  x:=random(1000)/1000;
  If q=1 then If x>0.5 then S:=7 else
            begin S:=8; q:=2; goto m; end;
  If q=2 then S:=round(0.5+random(600)/100);
  If S mod 2=0 then q:=1;
  m: If S=7 then Writeln(' OREL ');
  If S=8 then Writeln(' RESHKA ');
  If S<7 then Writeln(S,' '); inc(k); inc(n[S]);
until (KeyPressed)or(k=L);
For k:=1 to 8 do writeln(n[k]/L); Readkey;
END.
```

ТТР-4

```
uses crt;
const N=200; Var x: real; k,t: integer;
BEGIN clrscr; Randomize; k:=20;
Repeat x:=random(1000)/1000; inc(t);
  If x<k/N then k:=k-1 else k:=k+1;
  writeln(k,' = ',n-k,' | ');
until (KeyPressed)or(t>300); ReadKey;
END.
```

ТТР-5

```
uses crt, graph;
const M=5000;
p_op:array[1..5]of real=(0.6,0.7,0.4,0.8,0.6);
t_op:array[1..5]of real=(1.6,2.7,1.4,3.8,2.6);
Var x,D,t,pr,S,tsr,SP:real;
 i,j,n,DV,MV,k:integer;
 nn : array[1..30]of integer;
BEGIN clrscr; k:=0; tsr:=9.97; D:=0;
Repeat n:=0; t:=0; inc(k);
  Repeat t:=t+t_op[n+1];
    x:=random(1000)/1000;
    If x<p_op[n+1] then inc(n);
  until (n=5)or(KeyPressed);
  S:=S+t;D:=D+sqr(tsr-t);
  For i:=6 to 30 do
    If (t>i)and(t<i+1) then inc(nn[i]);
until (KeyPressed)or(k>M);
For i:=6 to 30 do writeln(i,' ',nn[i]);
```

```
writeln(S/M,' disp=',sqrt(D/(M-1)));
Repeat until KeyPressed;
END.
```

ТТР-6

```
uses crt, graph;
const p: array[1..3,1..4]of real=((0.7,0.1,0.1,0.1),
(0.1,0.8,0.1,0.0),(0.2,0.2,0.5,0.1));
pa: array[1..3]of real=(0.3,0.25,0.45);
var x,y,error,n,n1,i: integer; sl: real;
BEGIN randomize;
Repeat inc(n); sl:=random(100)/100;
  If sl<pa[1] then x:=1;
  If (sl>pa[1])and(sl<pa[1]+pa[2]) then x:=2;
  If (sl>pa[1]+pa[2])and(sl<1) then x:=3;
  sl:=random(100)/100;
  If sl<p[x,1] then y:=1;
  If (sl>=p[x,1])and(sl<p[x,1]+p[x,2]) then y:=2;
  If (sl>=p[x,1]+p[x,2])and(sl<p[x,1]+p[x,2]+p[x,3])
                                              then y:=3;
  If (sl>=p[x,1]+p[x,2]+p[x,3])and(sl<1) then y:=4;
  If x<>y then inc(error); If y=4 then inc(n1);
  writeln(n,' vhod ',x,' vihod ',y,
                ' otn_osh ',error/n,' ',n1/n);
until (n>3000)or(KeyPressed); Readkey;
END.
```

ТТР-7

```
uses crt, graph;
const M=20;
var l,p,xx : real; h,ii,i,j,k,n,z,t,f,DV,MV : integer;
x,y,g: array[1..M*M] of integer;
BEGIN Randomize; p:=0.6; clrscr;
For h:=1 to 10 do begin
For i:=1 to N do begin x[i]:=0; y[i]:=0; g[i]:=0; end; n:=0;
For i:=1 to M do For j:=1 to M do begin
xx:=random(1000)/1000;
if xx<p then begin n:=n+1; x[n]:=i; y[n]:=j; end; end;
g[1]:=1; j:=1; k:=1; t:=0;
Repeat t:=t+1;
For z:=1 to N do For i:=1 to N do For j:=1 to N do begin
If (g[j]=0)and(g[i]=k) then begin
l:=sqr(x[j]-x[i])+sqr(y[j]-y[i]); If l=1 then g[j]:=k; end;
end;
{For ii:=1 to N do begin write(ii,' ',g[ii],' | '); end;}
i:=1; Repeat i:=i+1; until(g[i]=0)or(i=N); k:=k+1; g[i]:=k;
```

```
f:=0; For i:=1 to N do if g[i]=0 then f:=1;
until (t>N+1)or(f=0)or(Keypressed);
f:=0;
For i:=1 to N do For j:=1 to N do begin
if (g[i]=g[j])and(y[i]=1)and(y[j]=20) then f:=1;
end; write(f); end;
readkey;
DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
For i:=1 to N do begin
circle(15+20*x[i],15+20*y[i],6);
circle(15+20*x[i],15+20*y[i],4);
rectangle(5+20*x[i],5+20*y[i],25+20*x[i],25+20*y[i]); end;
rectangle(25,25,425,425);
Repeat until KeyPressed; CloseGraph;
END.
```

ТТР-8

```
uses crt, graph;
var t,Gd,Gm : integer; x,p,q,a,g : real;
BEGIN
  Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
  Randomize;
  line(0,450,640,450); line(10,0,10,480);
  p:=0.01; q:=1-p; a:=0.003; g:=0.0004;
  Repeat
    inc(t); x:=random(1000)/1000;
     If (x>p)and(t<4000) then
     begin  p:=p+a*q; q:=q-a*q; end;
     p:=p-g*p; q:=q+g*p; {забывание}
     circle(10+round(t/15),450-round(400*p),2);
   until (t>10000)or(KeyPressed);
   Repeat until KeyPressed; CloseGraph;
  END.
```

ТТР-9

```
uses crt,graph;
const M=100;
var S,x: real; Gd,Gm,t,i,d: integer;
k: array[1..M]of integer;
BEGIN
Gd:= Detect; InitGraph(Gd, Gm, 'c:\bp\bgi\');
Randomize;
For i:=1 to M do if i<30 then k[i]:=2000 else k[i]:=200;
Repeat inc(t);
{if t mod 10=0 then k[80]:=k[80]+1;}
For i:=1 to M-1 do begin
x:=random(1000)/1000; S:=k[i]+k[i+1]+0.001;
```

```
d:=round(0.02*abs(k[i]-k[i+1])+1);
If x<k[i]/S then begin k[i]:=k[i]-d; k[i+1]:=k[i+1]+d; end
       else begin k[i]:=k[i]+d; k[i+1]:=k[i+1]-d; end; end;
if t =2000 then begin cleardevice; t:=0;
For i:=2 to M-1 do
  circle(10+6*i,round(450-(k[i-1]+k[i]+k[i+1])/15),2);
line(0,450,640,450); end;
until KeyPressed; CloseGraph;
END.
```

ТТР-10

```
uses crt, graph;
const  N=110; M=90; Ur=8;
var a,b,p : array[1..N,1..M] of integer;
    Gd,Gm,t,S,i,j: integer; x: real;
Procedure Draw;
begin
For i:=1 to N do For j:=1 to M do begin
  If a[i,j]=0 then setcolor(black);
  If a[i,j]>0 then setcolor(red);
  If a[i,j]>4 then setcolor(blue);
  circle(5*i,5*j,1); circle(5*i,5*j,2);
  If p[i,j]>Ur/2 then setcolor(8);
  circle(5*i,round(5*j),3); end;
end;
Procedure Raschet;
var i1,j1: integer;
begin S:=0;
If a[i,j]=0 then begin
  For i1:=i-1 to i+1 do For j1:=j-1 to j+1 do
    If a[i1,j1]>0 then inc(S); x:=random(100)/100;
    If (p[i,j]>Ur/4)and(S>2)and(x<0.7) then b[i,j]:=1; end;
If (a[i,j]>0)and(a[i,j]<6) then b[i,j]:=a[i,j]+1;
If (a[i,j]=6)or(p[i,j]=0) then b[i,j]:=0;
end;
BEGIN Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
setbkcolor(15); Randomize;
For i:=40 to 43 do For j:=40 to 43 do a[i,j]:=1;
For i:=1 to N do For j:=1 to M do begin
  x:=random(800)/100; p[i,j]:=Ur*(round(0.2+x)); end;
Repeat inc(t);
For i:=2 to N-1 do For j:=2 to M-1 do begin
  If (t mod 3=0)and(p[i,j]<Ur) then
    p[i,j]:=p[i,j]+round(random(400)/100);
  If p[i,j]>Ur then p[i,j]:=Ur; end;
  For i:=2 to N-1 do For j:=2 to M-1 do Raschet;
  For i:=2 to N-1 do For j:=2 to M-1 do
```

```
    begin a[i,j]:=b[i,j];
  If a[i,j]>0 then If p[i,j]>1 then p[i,j]:=p[i,j]-2
    else p[i,j]:=0; end; If t>100 then Draw;
until keypressed; CloseGraph;
END.
```

ПР-11

```
uses crt, graph;
const   N=120; M=120;
var a,b,p: array[1..N,1..M]of integer;
    Gd,Gm,i,j,S,x: integer;
BEGIN Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
Randomize; setbkcolor(15); a[60,40]:=1;
Repeat delay(100);
For i:=2 to N-1 do For j:=2 to M-1 do begin
S:=a[i-1,j]+a[i,j-1]+a[i+1,j]+a[i,j+1]
{+a[i-1,j-1]+a[i+1,j+1]+a[i+1,j-1]+a[i-1,j+1]};
  b[i,j]:=0*b[i,j]+S; If b[i,j]=0 then p[i,j]:=0;
  If b[i,j]<>0 then p[i,j]:=round(60/b[i,j]/b[i,j]); end;
  For i:=2 to N-1 do For j:=2 to M-1 do begin
    If a[i,j]>0 then a[i,j]:=a[i,j]+1; x:=random(100);
    If (a[i,j]=0)and(x<p[i,j]) then a[i,j]:=1; end;
  For i:=2 to N-1 do For j:=2 to M-1 do
    If a[i,j]>0 then begin If a[i,j]/6>12 then a[i,j]:=1;
      setcolor(round(a[i,j]/6+1));
      circle(5+5*i,5+5*j,2); circle(5+5*i,5+5*j,1); end;
until keypressed; CloseGraph;
END.
```

ЛИТЕРАТУРА

1. Булавин Л.А., Выгорницкий Н.В., Лебовка Н.И. Компьютерное моделирование физических систем. -- Долгопрудный: Издательский Дом "Интеллект", 2011. – 352 с.
2. Бусленко Н.П. Моделирование сложных систем. -- М.: Наука, 1968. -- 356 с.
3. Википедия [Электронный ресурс]: URL: http://ru.wikipedia.org (дата обращения 12.09.2012).
4. Гулд Х., Тобочник Я. Компьютерное моделирование в физике: В 2-х частях. Часть 2. -- М.: Мир, 1990. -- 400 с.
5. Майер Р.В. Задачи, алгоритмы, программы. [Электронный ресурс] -- URL: http://maier-rv.glazov.net, http://komp-model.narod.ru.

6. Майер Р.В. Компьютерное моделирование физических явлений. -- Глазов, ГГПИ: 2009. -- 112 с.
7. Поспелов В.А. Вероятностные автоматы. -- М: Энергия, 1970. -- 88 с.
8. Робертс Ф.С. Дискретные математические модели с приложениями к социальным, биологическим и экологическим задачам. – М.: Наука, Гл. ред. физ. –мат. лит., 1986. -- 496 с.
9. Рубанов В.Г., Филатов А.Г. Моделирование систем: Учебное пособие. -- Белгород: Изд-во БГТУ, 2006. -- 349 с.
10. Советов Б.Я., Яковлев С.А. Моделирование систем: Учебник для вузов. -- М.: Высш. Шк., 2001. -- 343 с.
11. Харин Ю.С. и др. Основы имитационного и статистического моделирования. Учебное пособие // Харин Ю.С., Малюгин В.И., Кирилица В.П., Лобач В.И., Хацкевич Г.А. -- Мн.: Дизайн ПРО, 1997. -- 288 с.
12. Цетлин М. Л. Исследования по теории автоматов и моделированию биологических систем. -- М.: Наука, 1969. -- 316 с.
13. Шеннон Р. Имитационное моделирование систем: Искусство и наука. -- М.: Мир, 1978. -- 302 с.