НАЗАД ОГЛАВЛЕНИЕ ВПЕРЁД

Глава 6
НЕПРЕРЫВНО-СТОХАСТИЧЕСКИЕ МОДЕЛИ

При изучении систем массового обслуживания используются непрерывно-стохастические модели, которые также называются Q-схемами (от англ. queueing system) [5; 9]. Анализ сложных систем требует применения комбинированных моделей, получающихся в результате агрегатного подхода [3]. Эти модели в силу своей универсальности позволяют исследовать поведение систем, которые содержат как дискретные, так и непрерывные подсистемы, как детерминированные, так и стохастические элементы. Изучение непрерывно-стохастических имитационных моделей осуществляется методом статистических испытаний [1; 7-12], который был рассмотрен в главе 5.

6.1. Имитационное моделирование систем

Обсуждая дискретные и непрерывные стохастические модели, часто говорят об **имитационном моделировании**. Одним из создателей теории имитационного моделирования является Р. Шеннон. В своей книге "Имитационное моделирование – искусство и наука" [12] он сформулировал основные принципы построения имитационных моделей, проанализировал методы и привел примеры их использования в конкретных случаях.

Хотя имитационные модели можно создать на аналоговых вычислительных машинах, в основном используются цифровые ЭВМ. Метод имитационного моделирования в англоязычной литературе обозначается термином "computer simulation" или "digital simulation", что означает "цифровая симуляция". Его сущность состоит в том, что испытаниям подвергается не объект исследования, а особым образом запрограммированная ЭВМ

(компьютерная модель), которая имитирует функционирование исследуемой системы. Имитационное моделирование не следует путать с численным моделированием, предусматривающим численное решение алгебраических и дифференциальных уравнений (главы 2 и 3).

При изучении поведения детерминированных систем достаточно провести однократную компьютерную имитацию, так как все последующие приведут к тому же результату. Если система стохастическая, то воспроизведение единственной реализации исследуемого процесса из-за случайных факторов не может достаточно полно характеризовать исследуемый объект. Поэтому применяют метод статистических испытаний (или метод Монте-Карло). Используемая компьютерная модель производит большое количество реализаций процесса и сохраняет получающиеся значения выходных величин. В этом случае результаты приобретают статистическую устойчивость и после соответствующей обработки могут рассматриваться как характеристики изучаемой системы.

Н. П. Бусленко отмечает, что "при статистическом моделировании реализация моделирующего алгоритма является, в некотором смысле, имитацией элементарных явлений, составляющих исследуемый процесс, с сохранением их логической структуры, последовательности протекания во времени и особенно характера и состава информации о состояниях процесса" [3, с. 57]. Имитационные модели используются при исследовании различных социологических и экономических процессов, изучения поведения сложных технических систем (например, ядерный реактор), а также в процессе обучения [1; 3; 5; 7-12].

6.2. Получение непрерывных случайных величин

Для создания непрерывно-стохастической модели требуется последовательность непрерывных случайных величин с заданным законом распределения $p = f(x)$. Для получения этой последовательности исполь-

зуется **метод обратной функции,** который состоит в следующем. Берут случайную равномерно распределенную на интервале $[0;\ 1]$ величину t и находят такое x, при котором выполняется равенство:

$$t = \int_0^x f(x)dx = F(x),$$

где $F(x)$ - интегральная функция распределения (рис. 6.1).

Другой способ получения непрерывной случайной величины с заданным законом распределения $f(x)$ состоит в следующем. Генерируют случайное равномерно распределенное число x_i и по формуле $p = f(x)$ определяют вероятность его появления в результирующей последовательности. После этого генерируют случайное равномерно распределенное число y из интервала $[0;\ 1]$. Если $y < p$, то число x_i принимается и поступает на выход, если нет, - отвергается. После этого генерируется другое x_i и все повторяется снова [1; 9].

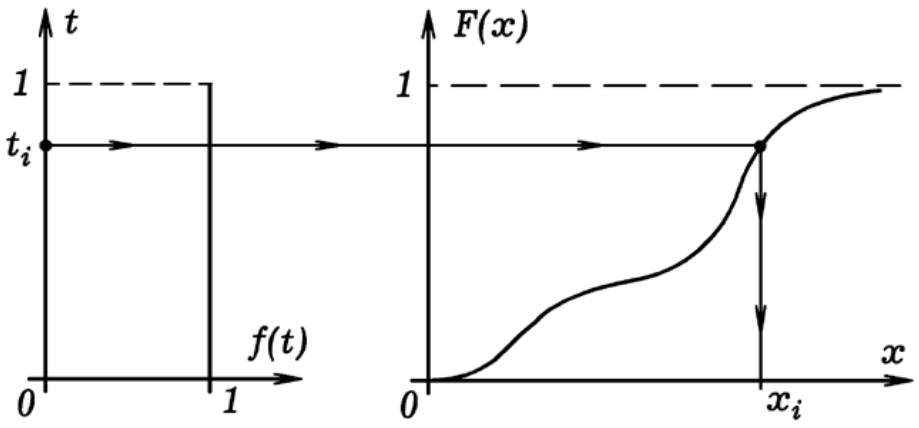

Рис. 6.1. Получение непрерывной случайной величины

В качестве примера промоделируем непрерывную случайную величину, распределенную по закону Пуассона: $p = \lambda \exp(-\lambda x)$ методом обратной функции. Запишем интегральную функцию распределения:

$$F(x) = \int_0^x \lambda \exp(-\lambda x) dx = 1 - \exp(-\lambda x).$$

Чтобы получить случайные числа, распределенные по закону Пуассона, необходимо из уравнения: $t = F(x) = 1 - \exp(-\lambda x)$ выразить x: $x = -\ln(1-t)/\lambda$. Для решения задачи построим генератор псевдослучайных чисел в соответствии с соотношением: $t_{i+1} = (t_i + 157{,}3) \bmod 200$. Он выдает псевдослучайные числа t, имеющие равномерное распределение. Подставляя их в формулу $x = -\ln(1-t)/\lambda$, получаем числа с законом распределения Пуассона. Используется программа ПР-1, после ее запуска получается гистограмма распределения $f(x) = \lambda \exp(-\lambda x)$ (рис. 6.2).

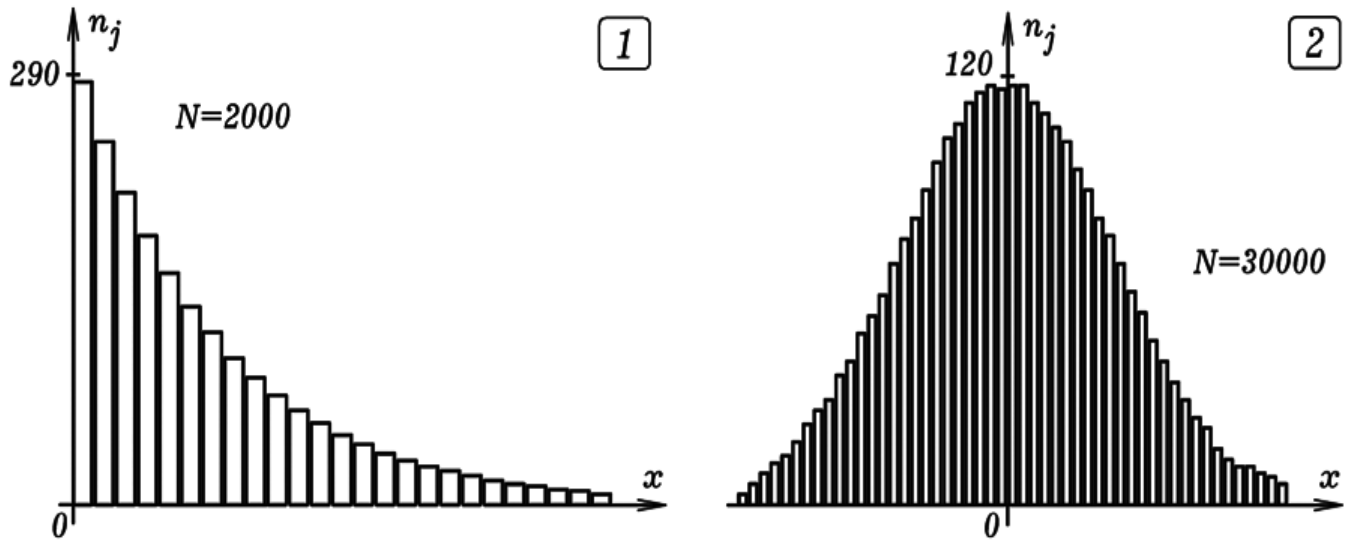

Рис. 6.2. Распределения Пуассона и Гаусса (программы ПР-1 и ПР-2)

Для получения последовательности случайных чисел, распределенных по закону Гаусса

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-x^2/2\sigma^2) \quad \text{при} \quad \sigma = 1,$$

удобно использовать второй метод. Применяемая программа ПР-2 содержит цикл, в котором выполняется следующая последовательность действий. С помощью рекурсивной процедуры генерируется случайное число

x_i из интервала [-5; 5] с равномерным законом распределения и определяется вероятность $p_i = f(x_i) = \exp(-x_i^2/2)/\sqrt{2\pi}$ его появления. После этого генерируется еще одно случайное число y из интервала $[0;\ 1]$ и сравнивается с p_i. Если $y < p$, то число x_i поступает на выход, а если нет – отбрасывается. Затем все повторяется снова. Распределение получающейся при этом случайной величины показано на рис. 6.2.2.

Как уже отмечалось в главе 5, распределение одномерной случайной величины характеризуется **математическим ожиданием** и **среднеквадратическим отклонением**. При большом числе испытаний математическое ожидание случайной величины примерно равно ее среднему значению: $x_{cp} = (x_1 + x_2 + x_3 + \ldots x_N)/N$. Среднеквадратическое отклонение характеризует разброс случайной величины относительно среднего значения и находится по формуле:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - x_{cp})^2}\ .$$

В некоторых случаях, используя частоты отклика системы, полученные в результате многократных испытаний, имеет смысл установить приближенный закон распределения $f(x)$. При этом с помощью метода обратных квадратов или иным способом подбирается функция $f(x)$, наиболее близко соответствующая эмпирическому распределению частот.

6.3. Проблема моделирования времени

При моделировании сложных систем возникает проблема задания времени. В общем случае исследуемая дискретная система состоит из совокупности взаимодействующих между собой автоматов, функционирование которой не синхронизировано. Можно представить себе сложную цифровую схему, состоящую из логических элементов, сумматоров, шифраторов, ячеек памяти, которые переключаются не одновременно. В

подобных случаях создание имитационной модели сложной системы требует организации квазипараллелизма обслуживания активностей. Проблема состоит в том, что различные элементы системы работают одновременно и параллельно, в то время как моделирующая их компьютерная программа состоит из операторов, выполняющихся последовательно друг за другом.

Существуют два подхода решения этой проблемы [3, 5, 7-12]: 1) с помощью постоянных интервалов времени, или "способ Δt-моделирования" (способ фиксированного шага), при котором определяется последовательность состояний системы в дискретные моменты $t_i = i\Delta t$; 2) с помощью переменных интервалов времени, или способ особых состояний (способ шага до следующего события, или "δz-моделирования"), при котором определяется момент перехода системы в другое состояние.

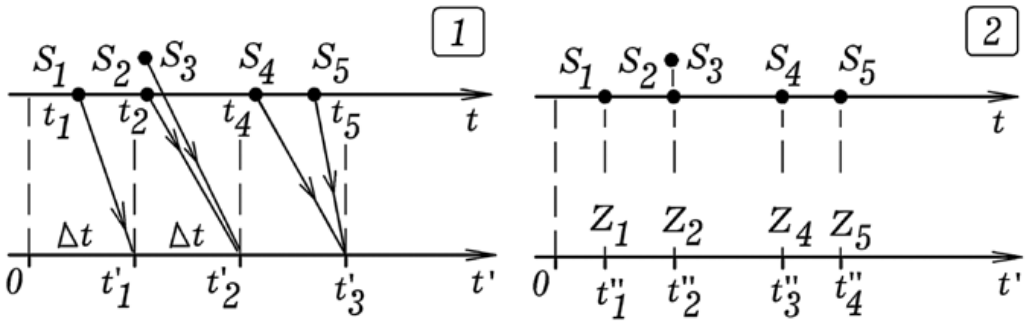

Рис. 6.3. Два подхода к имитации поведения системы

Пусть в системе происходит последовательность событий S_1, S_2, S_3, S_4, S_5 в моменты t_1, t_2, $t_3 = t_2$, t_4, t_5. При этом система совершает последовательность переходов из одного состояния в другое $Z_1 \to Z_2 \to Z_3 \to Z_4 \to Z_5$. Рассмотрим способ управления временем в модели, построенной по принципу Δt (рис. 6.3.1). Моменты системного времени последовательно принимают значения $t_i' = i\Delta t$ ($i = 0, 1, 2, \ldots$), которые не связаны с моментами t_1, t_2, $t_3 \ldots$ Шаг Δt выбирается в начале проведения имитационного эксперимента и должен быть достаточно мал. Рис. 6.3.1 соответствует ситуации, когда Δt слишком велико. ЭВМ отслеживает со-

стояние компьютерной модели в моменты $t_i' = i\Delta t$; при этом получается, что событие S_1 произошло в момент t_1', одновременные события S_2 и S_3 – в момент t_2', неодновременные события S_4 и S_5 – в момент t_3'. Из-за большого шага по времени возникает иллюзия, что события S_4 и S_5 одновременны, поэтому следует предусмотреть обработку группы событий. Чтобы такая модель "различала" события S_4 и S_5, необходимо уменьшить шаг Δt. Это приведет к увеличению затрат машинного времени.

В модели, построенной по принципу δz, время определяется в моменты смены состояния системы (рис. 6.3.2). Поэтому моменты системного времени t_1'', t_2'', t_3'', t_4'', в которые изменяется состояние модели, практически равны действительным моментам t_1, t_2, t_3, t_4 наступления событий S_1, S_2, S_3, S_4, S_5. При этом события обрабатываются последовательно, после чего время смещается до момента начала следующего события. Одновременная обработка событий необходима лишь тогда, когда эти события действительно происходят одновременно.

Выбор способа моделирования системы зависит от цели моделирования, назначения модели, ожидаемой точности, затрат машинного времени и памяти, трудоемкости моделирования. Обычно выделяют следующие **методы описания системы** при компьютерной симуляции: 1) описание активностями; 2) составление расписания событий; 3) транзактный способ; 4) агрегатный способ; 5) описание процессов [8].

6.4. Моделирование систем массового обслуживания

Одно из важных направлений прикладной математики связано с изучением **систем массового обслуживания** (СМО) [4]. Примерами СМО являются магазины, телефонные станции, кассы, ремонтные мастерские, автозаправочные станции, ЭВМ, обрабатывающая запросы от удаленных терминалов, и т. д. Каждая из перечисленных систем состоит из **каналов**

(или приборов) **обслуживания**, на которые в случайные моменты времени поступает **поток заявок** или **требований**. После приема заявки канал оказывается занят на некоторое **среднее время обслуживания** T_{ob}, после чего он освобождается и ожидает следующей заявки. На входе СМО может накапливаться несколько заявок, они либо становятся **в очередь**, либо покидают СМО **необслуженными**.

Последовательность событий, происходящих друг за другом в случайные моменты времени, называется **потоком событий**. Если поток событий задается только моментами времени $0 < t_1 < t_2 < t_3 < \ldots < t_n$ наступления этих событий, то он называется **однородным**. Поток **неоднородных** событий характеризуется: 1) совокупностью вызывающих моментов времени t_i, $i = 1,2,\ldots,n$; 2) набором признаков события, к которым относятся принадлежность заявки к тому или иному источнику, приоритет заявки, возможность обслуживания тем или иным каналом и т. д. **Интенсивность** потока рассчитывается как отношение числа событий ко времени наблюдения: $\lambda = \Delta N / \Delta T_H$. Если вероятность появления заданного числа событий в течение интервала $\Delta\tau$ зависит исключительно от продолжительности интервала $\Delta\tau$ и не зависит от времени τ, прошедшего с начала запуска системы, поток событий называется **стационарным**.

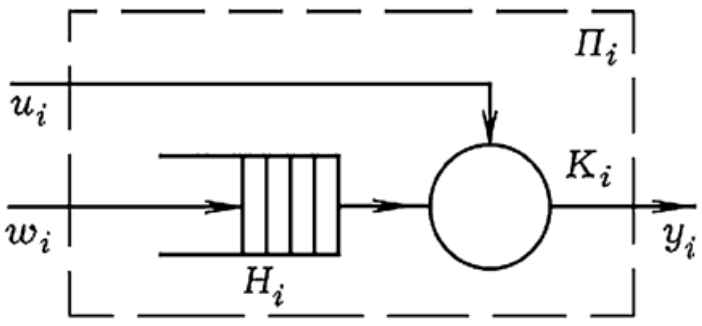

Рис. 6.4. Прибор обслуживания заявок как элемент СМО

Любая СМО состоит из **приборов обслуживания** $П_i$ (рис. 6.4), каждый из которых имеет **накопитель заявок** H_i и **канал обслуживания заявок** K_i [4]. В накопителе заявок может одновременно находиться $n_i = 0, 1,$

$2,...,N_i$ заявок, где N_i – емкость i-го накопителя. В накопитель H_i поступает поток заявок w_i, а на канал K_i – поток обслуживаний u_i. При изучении сложных систем массового обслуживания рассматриваются Q-схемы, образующие **многоканальные** и **многофазные сети** массового обслуживания. Связи между элементами таких СМО изображают в виде стрелок, которые показывают направления движения заявок. В некоторых случаях говорят о **замкнутых** СМО, имеющих обратную связь, по которой выходной поток обслуженных заявок снова поступает на вход того или иного прибора обслуживания. В общем случае процесс функционирования СМО любой сложности можно однозначно задать с помощью Q-схемы, учитывающей: 1) множество входящих потоков W; 2) множество потоков обслуживания U; 3) правила R сопряжения элементов СМО; 4) множество собственных параметров H; 5) оператор алгоритмов обслуживания заявок A; 6) вектор состояний Z, элементы которого характеризуют состояния всех приборов обслуживания и их накопителей.

Для изучения функционирования СМО методом статистических испытаний (методом Монте-Карло) строится имитационная модель процесса и с помощью генератора случайных чисел производится "розыгрыш" случайных событий (входных сигналов и внешних воздействий) в соответствии с заданными законами их распределения. Компьютер моделирует $10^3 - 10^6$ реализаций исследуемого процесса, выходные сигналы сохраняются и подвергаются статистической обработке.

6.5. Моделирование СМО с отказами

Рассмотрим **задачу Эрланга** - классическую задачу теории массового обслуживания [4]. Имеется СМО с отказами, состоящая из 4 каналов обслуживания: когда все каналы обслуживания заняты, заявка покидает СМО и в дальнейшем процессе обслуживания не участвует. В эту СМО поступает простейший поток заявок с интенсивностью λ. Поток обслужива-

ний имеет интенсивность $\mu = 1/T_{ob}$, где T_{ob} - среднее время обслуживания одной заявки. Необходимо найти вероятности состояний СМО, а также абсолютную пропускную способность A, относительную пропускную способность Q, вероятность отказа p_{otk} или вероятность обработки $p_{об}$, среднее число занятых каналов k. Желательно также изучить зависимость этих величин от приведенной интенсивности потока заявок $\rho = \lambda / \mu$, равной среднему числу заявок, приходящих за время T_{ob}.

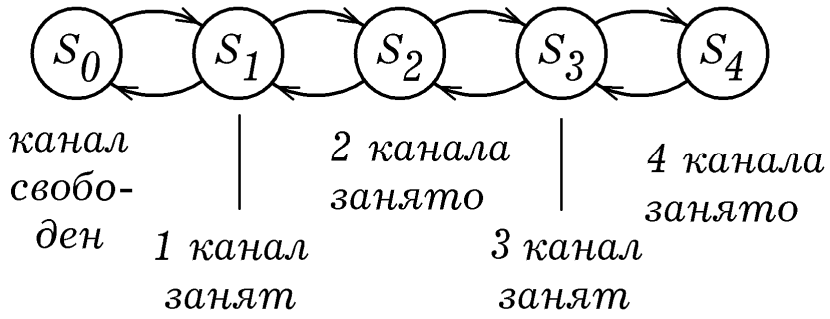

Рис. 6.5. Граф состояний СМО с четырьмя каналами обслуживания

Граф состояний анализируемой системы соответствует "схеме гибели и размножения" (рис. 6.5). Его вершины символизируют следующие состояния: S_0 - в системе нет заявок, S_1 - в системе одна заявка (один канал занят); S_2 - две заявки; S_3 - три заявки; S_4 - четыре заявки (заняты все 4 канала). На вход поступает простейший поток с интенсивностью λ. Это означает, что интервал $\tau > 0$ между соседними событиями имеет показательное распределение с плотностью $f(\tau) = \lambda \exp(-\lambda\tau)$.

Каждый канал обслуживания занимается заявкой в течение времени T_{ob}, после этого освобождается, состояние системы S уменьшается на 1. При поступлении новой заявки, когда $S = 0, 1, 2, 3,$ состояние S увеличивается на 1, число свободных каналов уменьшается. Если свободных каналов нет ($S = 4$), то при поступлении новой заявки состояние системы остается прежним, заявка не принимается.

Будем использовать способ Δt-моделирования. Компьютерная модель анализируемой СМО должна работать по следующему алгоритму:

1. Задается шаг по времени **dt**, интенсивность потока заявок **lambda**, среднее времени обработки заявки **Tobr**.

2. Начало цикла по времени с шагом **dt**. Увеличение **t** на **dt**.

2.1. Увеличение **t1** на **dt**. Если **t1>tau**, то сгенерировать следующее случайное число **tau**. На вход системы подать заявку (**zayavka:=1;**), обнулить переменную **t1** (**t1:=0;**).

2.2. Если **zayavka=1** и **S=0,1,2** или **3**, то **t1, t2, t3** или **t4:= Tob; S:=S+1**. Если **zay=1** и **S=4**, то напечатать "отказ". Увеличить счетчик отказов на 1 (**otkaz:= otkaz+1;**).

2.3. Если **S=1** (1 канал занят), то **t1:=t1-dt**.

2.4. Если **S=2** (2 канала занято), то **t1:=t1-dt, t2:=t2-dt**.

2.5. Если **S=3**, то **t1:=t1-dt, t2:=t2-dt, t3:=t3-dt**.

2.6. Если **S=4** (4 канала занято), то **t1:=t1-dt, t2:=t2-dt, t3:=t3-dt, t4:=t4-dt.**

2.7. Если **t1<=0** (освободился канал 1), то **S:=S-1; tt1:= tt2; tt2:=tt3; tt3:= tt4; tt4:=0**. Увеличить счетчик выполненных заявок **VZ:= VZ+1**, перейти к п. 2.11.

2.8. Если **t2<=0** (освободился канал 2), то **S:=S-1; tt2:= tt3; tt3:=tt4; tt4:=0**. Увеличить счетчик выполненных заявок на 1: **VZ:=VZ+1**, перейти к п. 2.11.

2.9. Если **t3<=0** (освободился канал 3), то **S:=S-1; tt3:= tt4; tt4:=0**. Увеличить счетчик выполненных заявок **VZ:= VZ+1**, перейти к п. 2.11.

2.10. Если **t4<=0** (освободился канал 4), то **S:=S-1, tt4:=0**. Увеличить счетчик выполненных заявок **VZ:=VZ+1**.

2.11. Распечатать значения **Z, S, L, VZ, otkaz**. Вернуться к началу цикла (п. 2). Если **t>500**, выйти из цикла.

3. Напечатать **VZ/L, VZ/t, 1-otkaz/L**. Конец алгоритма.

Здесь tau – случайное число, равное времени между заявками и имеющее показательный закон распределения Пуассона. Переменные `t1, t2, t3, t4` нужны для хранения времени, в течение которого остаются занятыми первый, второй, третий и четвертый каналы соответственно. Для моделирования анализируемой четырехканальной системы массового обслуживания без очереди используется программа ПР-3. Созданная компьютерная модель позволяет изучить зависимости пропускной способности СМО υ, вероятности обработки заявки $P_{об}$, среднего числа занятых каналов $N_{ср}$, доли свободного времени η от приведенной интенсивности потока заявок ρ. Для этого необходимо провести серию компьютерных имитаций при различных значениях ρ. По полученным результатам для $T_{обр} = 1$ построены графики, изображенные на рис. 6.6. Видно, что с ростом интенсивности потока заявок увеличивается вероятность отказов, уменьшается вероятность обработки заявки от 1 до 0, среднее число занятых каналов и пропускная способность растут, стремясь к 4. Предложенная компьютерная модель после соответствующей доработки позволяет исследовать функционирование СМО с отказами, имеющую другое количество каналов.

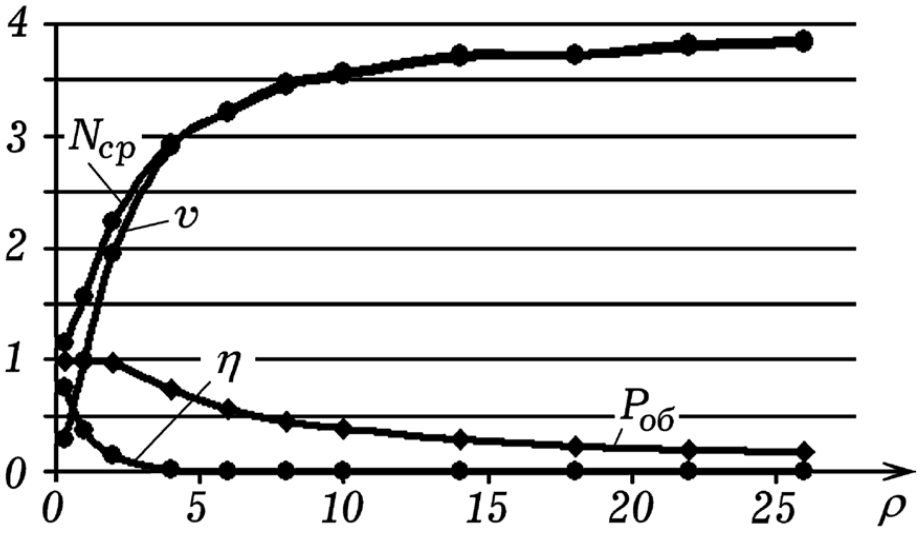

Рис. 6.6. Характеристики СМО с отказами (4 канала)

6.6. Моделирование СМО с очередью

В некоторых случаях СМО может иметь **ограниченную** или **неограниченную очередь**. Например, к автозаправочной станции подъезжают автомобили; если все колонки заняты, то они выстраиваются в очередь, длина которой не может превышать десяти. Подобные системы также могут быть исследованы на ЭВМ методом статистических испытаний.

Рассмотрим СМО с двумя каналами обслуживания и 3-местной очередью. На вход поступает простейший поток событий, имеющий экспоненциальное распределение вероятностей. Если заняты оба канала и три заявки в очереди, то с приходом новой заявки она получает отказ. После завершения обслуживания соответствующий прибор обслуживания освобождается, и система автоматически переходит к заявке, находящейся в очереди. Очередь сдвигается, в ней освобождается одно место. Можно методом имитационного моделирования определить основные характеристики эффективности этой СМО: вероятность выполнения и отклонения заявки, среднее время нахождения заявки в системе, время, пока система не занята, и среднюю длину очереди при различных интенсивностях потока заявок.

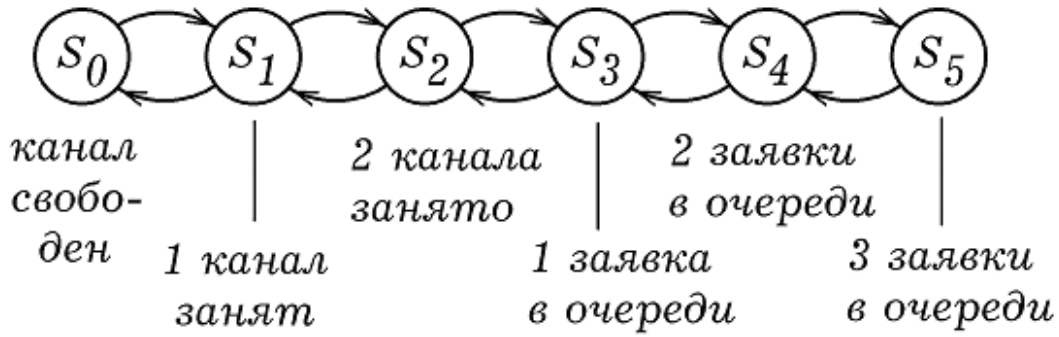

Рис. 6.7. Граф состояний СМО с двумя каналами и 3-местной очередью

Граф состояний анализируемой системы представлен на рис. 6.7. Его вершины соответствуют следующим состояниям: S_0 - в системе нет заявок, S_1 - в системе одна заявка (один канал занят); S_2 - две заявки; S_3 -

две заявки, одна в очереди; S_4 - две заявки, две в очереди; S_5 - в системе две заявки, три в очереди. На вход поступает простейший поток с интенсивностью λ, то есть интервал τ между соседними заявками – случайная величина, распределенная по закону: $f(\tau) = \lambda \exp(-\lambda\tau)$. Каждый канал обслуживания занимается заявкой в течение времени T_{ob}. После ее обработки он освобождается, состояние системы S уменьшается на 1.

Если при поступлении новой заявки $S = 0$ или 1, то она занимает первый или второй канал обслуживания, и S увеличивается на 1. Если $S = 2$, 3 или 4, то новая заявка становится в очередь, S увеличивается на 1. Если свободных каналов нет ($S = 5$), то при поступлении новой заявки состояние системы остается прежним, заявка не принимается.

Алгоритм, моделирующий функционирование анализируемой СМО во многом подобен рассмотренному в предыдущем параграфе. Используется программа ПР-4. На рис. 6.8 показаны графики изменения средней длины очереди $L_{оч}$, среднего времени $T_з$ нахождения заявки в системе, вероятности отказа $P_{отк}$ и обработки $P_{об}$ заявки в зависимости от интенсивности потока заявок ρ.

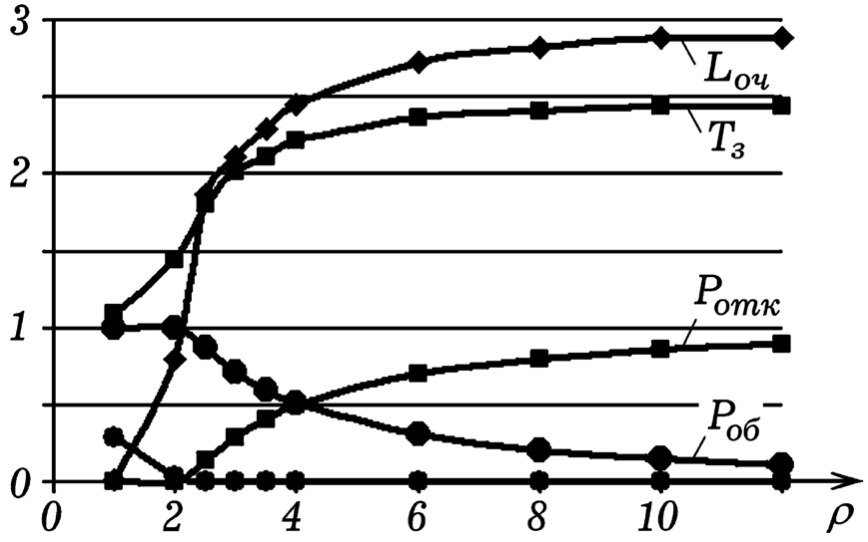

Рис. 6.8. Графики, характеризующие анализируемую СМО с очередью

6.7. Имитационное моделирование более сложных СМО

Метод имитационного моделирования позволяет изучать функционирование замкнутой системы массового обслуживания с несколькими источниками. Рассмотрим следующую ситуацию. В цеху работают 20 станков, которые изредка выходят из строя или требуют наладки. Их обслуживают двое рабочих. Интенсивность потока требований каждого работающего станка равна λ, среднее время наладки (обработки заявки) составляет $T_{ob} = 1/\mu$. Необходимо создать компьютерную модель этой СМО и определить, как изменяются ее характеристики при различных приведенных интенсивностях потока заявок $\rho = \lambda / \mu$.

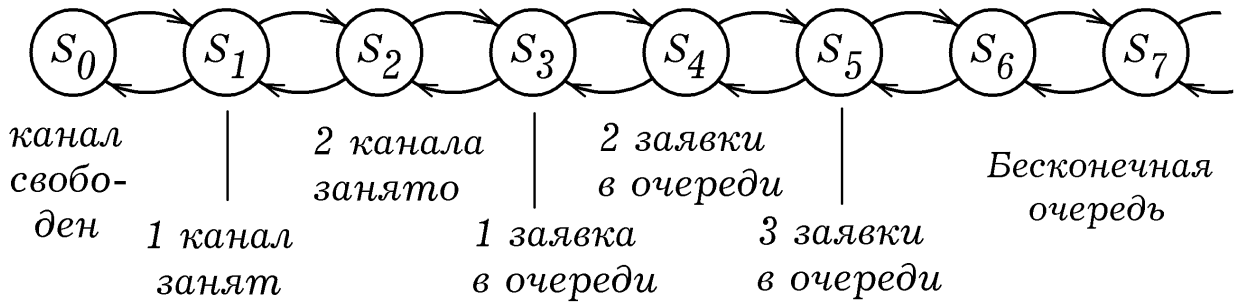

Рис. 6.9. Граф состояний СМО с двумя каналами и бесконечной очередью

Интенсивность потока поступающих заявок пропорциональна числу работающих станков k. Граф состояний анализируемой системы представлен на рис. 6.9. Можно считать, что система имеет бесконечную очередь. Вершины графа соответствуют следующим состояниям: S_0 - в системе нет заявок, S_1 - в системе одна заявка (один рабочий занят); S_2 - две заявки (оба рабочих заняты); S_3 - две заявки обслуживаются, одна в очереди; S_4 - две заявки обслуживаются, две в очереди; S_5 - две заявки обслуживаются, три в очереди и т. д. На вход поступает простейший поток с интенсивностью λ.

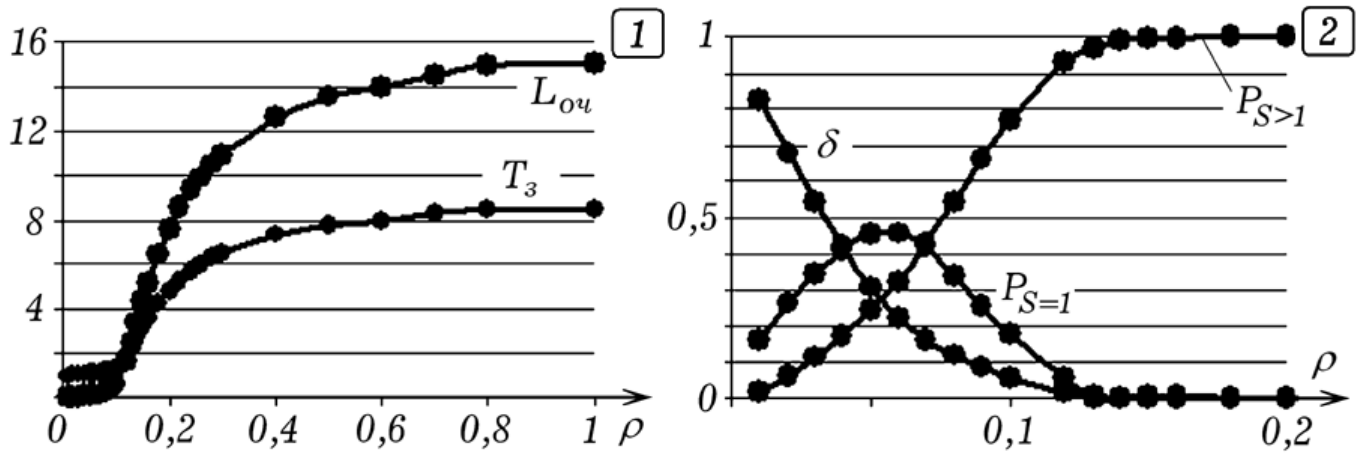

Рис. 6.10. Графики, характеризующие СМО с двумя каналами

Алгоритм, моделирующий функционирование анализируемой СМО, во многом подобен рассмотренному в предыдущих параграфах. Используется программа ПР-5 (Приложение). Задавая различные значения ρ, можно проделать серию вычислительных экспериментов. Типичные результаты – на рис. 6.10. Видно, что при увеличении интенсивности потока заявок ρ среднее время заявки в системе $T_з$ и длина очереди $L_{оч}$ растут, стремясь к некоторым установившимся значениям. Это объясняется тем, что больше 20 станков сломаться не могут. При увеличении ρ от 0 до $0{,}13$ доля времени δ, пока система не занята, уменьшается до 0. Вероятность нахождения СМО в состоянии $S=1$ (1 рабочий занят) при малых ρ растет, достигает максимума, а затем уменьшается до 0. Вероятность нахождения СМО в состоянии $S>1$ (занято оба рабочих) в том же интервале изменения ρ увеличивается до 1 и остается постоянной.

Промоделируем функционирование конвейера с N рабочими местами (рис. 6.11). На конвейер поступает ведущий полуфабрикат (основа сборного узла); рабочим подают присоединяемые к узлу ведомые полуфабрикаты (детали) с номерами $i = 1,2,...,N$. После перемещения конвейера рабочий осуществляет проверку i-й детали, которая длится $t_i^{пр}$. С вероятностью $p_i^{бр}$ i-я деталь может быть признана бракованной. Она исключается, и выбирается новая деталь, которая снова проверяется. Если

деталь небракованная, i-й рабочий присоединяет ее к основе сборного узла. Эта операция длится $t_i^{сб}$; после ее окончания i-й рабочий нажимает на i-ю кнопку, подавая сигнал о своей готовности. Когда все рабочие нажимают на свои кнопки, конвейер передвигается. Необходимо создать имитационную модель этого производственного процесса, определить среднее время сборки одного изделия.

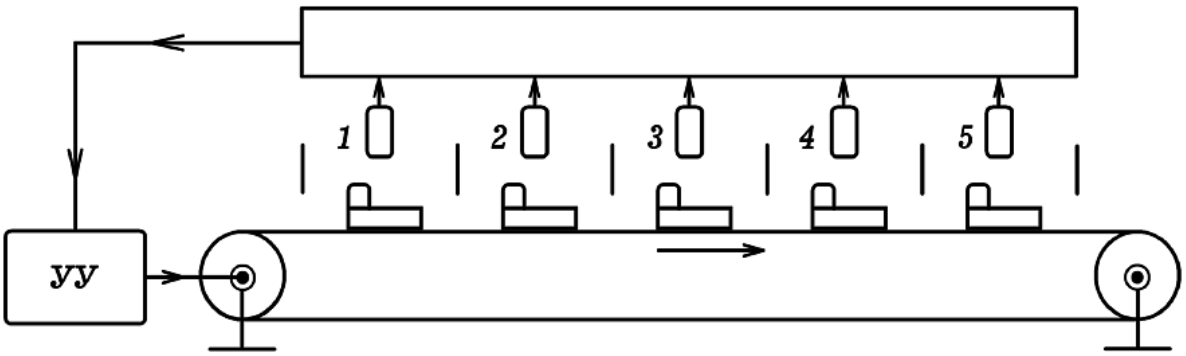

Рис. 6.11. К вопросу о моделировании работы конвейера

Решением задачи для случая, когда число рабочих $N = 5$, является программа ПР-6. В массивах **p_br[i], vr_sb[i], vr_pr[i]** хранятся соответственно вероятность брака, среднее время проверки и среднее время присоединения i-й детали в минутах. Процедура RND генерирует случайное число, расчет состояния i-го места сборки производится в процедуре Mesto_Sborki. При этом различаются три состояния S_i: 1) $S_i = 1$, - осуществляется проверка детали; 2) $S_i = 2$, - деталь присоединяется к основе сборного узла; 3) $S_i = 3$, - присоединение детали закончено, рабочий нажал на кнопку и ожидает передвижения конвейера. Состояние i-го места сборки в предыдущий дискретный момент времени записывается в массив **sp[i]**. Программа должна содержать цикл, в котором определяются состояния S_i в последовательные моменты времени $t = j\Delta t$ ($j = 1,2,3,...$), подсчитывается число перемещений конвейера (равное количеству изделий), вычисляется время изготовления 300 изделий и

среднее время изготовления одного изделия. В нашем случае (при заданных в программе значениях $p_i^{бр}$, $t_i^{сб}$, $t_i^{пр}$) оно составляет около 5,9 минут.

6.8. Стохастическое моделирование физических явлений

Метод статистических испытаний может быть использован не только для изучения систем массового обслуживания, но и для исследования физических, социальных, технических и иных систем [2]. Например, рассмотрим задачу о движении снаряда, выпущенного из ствола пушки с заданной скоростью $\upsilon \pm \Delta\upsilon$ под углом $\alpha \pm \Delta\alpha$. Значения υ и α являются случайными, распределение нормальное, СКО известны. Будем считать, что на летящий снаряд действует возмущающий фактор – случайные порывы ветра одинаковой силы, направленные горизонтально под некоторым углом к траектории. Промежуток времени τ между порывами ветра - случайная величина, закон распределения которой известен. Необходимо определить распределение точек попадания снаряда. Для решения этой задачи методом статистических испытаний необходимо создать компьютерную модель полета снаряда, которая учитывала бы влияние всех случайных событий. Затем следует провести серию вычислительных экспериментов, в которых будет рассчитываться траектория движения снаряда при случайной реализации возмущающих факторов (ветра, небольших изменений начальной скорости и угла). Результаты большого числа экспериментов следует обработать статистическими методами так же, как если бы они были результатами реальных измерений. Это позволит определить закон распределения, средние значения координат точки падения снаряда и их среднеквадратические отклонения.

Рассмотрим еще одну задачу. Нитяной маятник, состоящий из подвешенного на нити тела, находится в горизонтальном потоке воздуха.

Скорость движения воздуха в потоке изменяется случайным образом, а направление остается неизменным. Необходимо изучить движения маятника, получить кривую распределения его угловой координаты φ, найти ее среднее значение и среднеквадратическое отклонение (СКО).

Задача сформулирована не совсем корректно, так как неизвестны закон изменения скорости воздуха и параметры системы. Пусть масса тела $m = 0{,}4$ кг, длина нити $l = 0{,}5$ м, момент инерции $I = ml^2 = 0{,}1 \ \ кг \cdot м^2$, коэффициент сопротивления воздуха $r = 0{,}1 \ \ Н \cdot м \cdot с$. Будем считать, что порывы ветра имеют равную длительность $t_в = 0{,}8$ с, силу $F = 2 \ \ Н$ и следуют один за другим через промежутки времени $t_{пр} = 2 + t_{сл}$ с, где $t_{сл}$ - случайная величина, распределенная по нормальному закону:

$$t_{сл} = \frac{1}{\sigma\sqrt{2\pi}} e^{(-x^2/2\sigma)} \ (с),$$

где $\sigma = 0.4$, а x - равномерно распределенная в интервале $[0;\ 1]$ величина. Ветер создает вращающий момент $Fl\cos\varphi$. Из основного уравнения динамики вращательного движения следует:

$$I\varepsilon = F(\tau)l\cos\varphi - r\dot{\varphi} - mgl\sin\varphi, \quad \varepsilon^{t+1} = (F^t l\cos\varphi^t - r\omega^t - mgl\sin\varphi^t)/I,$$

$$\omega^{t+1} = \omega^t + \varepsilon^{t+1}\Delta\tau, \quad \varphi^{t+1} = \varphi^t + \omega^{t+1}\Delta\tau.$$

Используемая программа ПР-7 содержит цикл по времени, в котором случайным образом разыгрываются промежутки времени между порывами ветра и с шагом $\Delta\tau$ рассчитываются ускорение, скорость и координата маятника. На экране строится график $\varphi(t)$. Кроме того, для изучения распределения угла φ определяется количество раз n_j, когда значение случайной величины φ попадает в один из интервалов $[\varphi_j, \varphi_{j+1}]$, где $\varphi_j = \varphi_0 + j\Delta\varphi$, $j = 0,1,2,3,...$, а $\Delta\varphi$ - ширина интервала. Результаты подсчетов сохраняются в массиве n[j] и используются для построения гистограммы относительных частот (рис. 6.12.2).

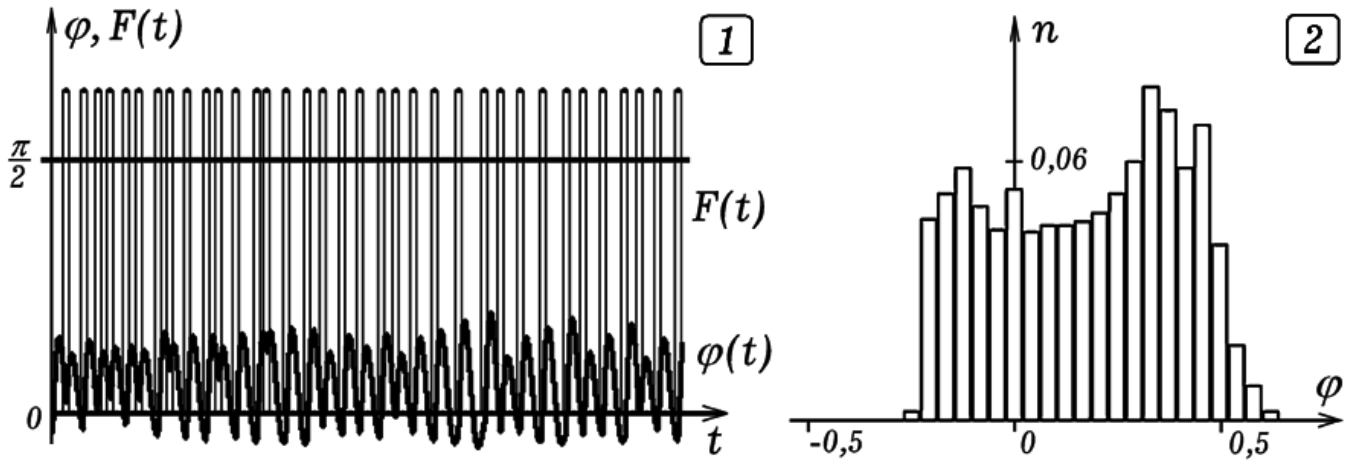

Рис. 6.12. Изучение колебаний маятника в потоке воздуха

Чтобы получить среднее значение угла и среднее квадратическое отклонение (СКО), необходимо закомментировать операторы, требующие использования графического режима, и открыть операторы, выводящие на экран переменные `t,fi,N_izm,Sum_fi/N_izm,Sum,Sum_fi/N_izm, sqrt(Sum/(N_izm-1))`. Сначала необходимо определить среднее значение угла φ_{cp} (величина `Sum_fi/N_izm`) по результатам `N_izm` $=50000$ испытаний, а затем присвоить полученное значение константе `fi_sr` и повторить вычисления для нахождения СКО по формуле:

$$s=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\varphi_i-\varphi_{cp})^2}\,.$$

Типичные результаты решения задачи при некоторых параметрах системы представлены на рис. 6.12.2. Здесь среднее значение угла $\varphi_{cp}=0{,}186$ рад, а среднее квадратическое отклонение СКО $=0{,}216$ рад.

6.9. Агрегатный подход к моделированию сложных систем

Систему называют сложной, если: 1) она имеет достаточно много взаимосвязанных элементов; 2) целевая функция, на оптимизацию кото-

рой направлено функционирование системы, не совпадает с целевыми функциями составляющих ее элементов; 3) имеется управление и разветвленная информационная сеть, связывающая элементы системы между собой и со средой [10, с. 9]. Примерами **сложных систем** являются телефонная или компьютерная сеть, различные электронные устройства, системы здравоохранения и образования, процессы производства бумаги, изготовления самолетов и т. д.

Изучение сложных технических систем и производственных процессов потребовало разработки универсального подхода, позволяющего промоделировать как дискретные, так и непрерывные процессы. Примерами дискретных процессов являются поточное производство штучных изделий: радиодеталей, часов, бетонных плит. В непрерывных производственных процессах используются компоненты сырья (газ, нефть, песок) и готовой продукции (бензин, ткань), которые непрерывными потоками поступают к технологическим установкам либо выходят из них. В случае когда производственный процесс имеет дискретную и непрерывную составляющие, для его моделирования не достаточно отдельно F-, P- или Q-схемы. Н. П. Бусленко [3] предложил универсальный подход к формальному описанию сложных систем, основывающийся на понятии **агрегативной системы** и предусматривающий построение комбинированной модели или А-схемы (от англ. aggregate system). Он позволяет анализировать как непрерывные, так и дискретные процессы, как детерминированные, так и стохастические системы.

Агрегатный подход состоит в следующем [3; 9]. При построении А-модели сложная система разбивается на конечное число взаимосвязанных элементов (подсистем). Этот процесс производится до тех пор, пока не получается совокупность агрегатов. **Агрегат** - математическая абстракция, позволяющая унифицировать построение имитационной модели и обработку результатов моделирования системы, состоящей из большого числа разнотипных элементов. Чтобы задать агрегат, необходимо указать: 1) вектор состояний $z=(z_1, z_2, ..., z_n)$; 2) вектор параметров $\theta=(\theta_1,$

$\theta_2, ..., \theta_m)$, например, среднее время выполнения той или иной операции; 3) вектор воздействий внешней среды $\upsilon = (\upsilon_1, \upsilon_2, ..., \upsilon_k)$; 4) вектор входных воздействий (управляющих сигналов) $u = (u_1, u_2, ..., u_l)$; 5) вектор начальных состояний $z^0 = (z_1^0, z_2^0, ..., z_n^0)$; 6) оператор переходов $z(t) = H(z_0, \upsilon_0, u_0, t)$, задающий состояние агрегата в следующий момент времени, исходя из начального состояния, воздействий внешней среды и входных воздействий; 7) вектор выходных сигналов $y = (y_1, y_2, ..., y_p)$; 8) оператор выходов G, связывающий вектор выходных сигналов с вектором состояний z и вектором входных воздействий u: $y(t) = G(z(t), u(t))$. Оператор переходов может учитывать случайные факторы. Последовательность входных (выходных) сигналов, расположенных в порядке их поступления, называется входным (выходным) сообщением.

В каком-то смысле, агрегат – это усложненный автомат, к которому добавили дополнительные параметры так, что он стал способен описывать всевозможные элементы сложных систем. Как утверждал Н. П. Бусленко [3, с. 211-212], **агрегативная система** представляет собой совокупность агрегатов, передача информации между которыми происходит мгновенно и без искажений. При этом каждый элемент А-системы, являясь агрегатом в общем случае, не должен обязательно обладать всеми свойствами агрегата; он может быть частным случаем, например, автоматом. В то же время А-система должна включать в себя хотя бы один элемент, являющийся агрегатом.

Итак, **агрегат** - математическая модель любого объекта, определяемого упорядоченной совокупностью множеств $<X, \theta, \upsilon, U, X^0, Y>$ и случайных операторов H и G. Каждому состоянию агрегата соответствует точка в n-мерном пространстве, образованном переменными z_1, z_2, …, z_n. Считается, что [9, с. 75-83]: 1) взаимодействие агрегативной системы с внешней средой и входящих в нее агрегатов друг с другом осуществляется только посредством сигналов; 2) любой сигнал характеризуется ко-

нечным набором характеристик; 3) элементарные сигналы передаются от агрегата к агрегату мгновенно; 4) вход любого агрегата соединен с не более чем одним каналом передачи сообщений, а к выходу может быть подключено любое конечное число элементарных каналов.

Для анализа функционирования системы внешнюю среду также представляют в виде агрегата A_0. Каждый агрегат A_i имеет k_i входных контактов X_j^i ($j = 1,2,3,...,k_i$) и l_i выходных контактов Y_j^i ($j = 1,2,3,...,l_i$). Можно задать оператор сопряжения R, который сопоставляет выходному контакту $Y_{j'}^{i'}$ входной контакт $X_{j''}^{i''}$, связанный с ним элементарным каналом. При поступлении на агрегат A воздействий внешней среды υ и управляющих сигналов u агрегат изменяет свое состояние z в соответствии с оператором переходов H. Выделяют **особые состояния**, которыми называются состояния агрегата в моменты поступления входного или управляющего сигнала, воздействия внешней среды или выдачи выходного сигнала. Все остальные состояния считаются **неособыми**. При наступлении особых состояний оператор перехода H изменяется скачкообразно, то есть имеет разрыв. Во временных интервалах между особыми состояниями функция H остается непрерывной.

6.10. Пример использования агрегатного подхода

В качестве простого примера рассмотрим систему, состоящую из генератора сигналов, нелинейного усилителя и RLC-фильтра (рис. 6.13). Генератор сигналов может работать в трех режимах, при этом на выходе будет получаться: 1) синусоидальное напряжение (состояние $S = 1$); 2) прямоугольные импульсы (состояние $S = 2$); 3) линейно-импульсное напряжение (состояние $S = 3$). Генератор позволяет изменять частоту колебаний f и амплитуду A. Усилитель имеет коэффициент усиления, зависящий от входного напряжения: $K_U = u_{вых} / u_{вх} = K_{\max} \exp(-a\,|\,u_{вх}\,|)$. В уси-

лителе имеется источник случайных помех с амплитудой $U'=10$ В. Чтобы это учесть, к выходному напряжению $u_2(t)$ следует прибавить случайную величину, имеющую, например, равномерный закон распределения. Параметры R, L, C фильтра известны.

На выходы системы поступают: 1) напряжение u_1 с выхода генератора; 2) напряжение u_2 с выхода усилителя; 3) напряжение u_3 с выхода фильтра (например, с конденсатора). Процессы в рассматриваемом последовательном колебательном контуре описываются уравнением: $L\ddot{q}+R\dot{q}+q/C=u_2(t)$. Для их моделирования на компьютере используется конечно-разностная схема:

$$i^{t+1}=i^t+(u_2^t-Ri^t-q^t/C)\Delta t/L, \qquad q^{t+1}=q^t+i^{t+1}\Delta t.$$

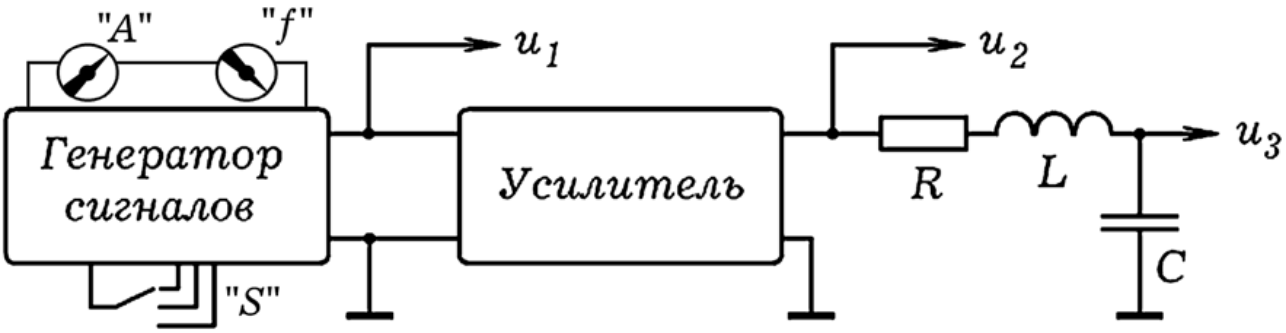

Рис. 6.13. Система из генератора, усилителя и колебательного контура

Заменим каждый из трех элементов агрегатом и рассмотрим структуру анализируемой системы (рис. 6.14). Внешнюю среду представим в виде фиктивного элемента A_0. Он имеет четыре выходных контакта Y_1^0, Y_2^0, Y_3^0, Y_4^0, которые соединены с входными контактами X_1^1, X_2^1, X_3^1 и X_2^2 агрегатов $A1$ и $A2$, соответствующих переключателю режима S, регулятору частоты f, регулятору амплитуды A генератора, а также регулятору коэффициента усиления усилителя. Выход усилителя (контакт Y_2^2) соединен с входом RLC-фильтра (контакт X_1^3). Выходы генератора (агрегат $A1$), усилителя (агрегат $A2$) и фильтра (агрегат $A3$) соединяются

с входными контактами X_1^0, X_2^0, X_3^0 среды (фиктивного элемента A_0). Этой агрегативной системе соответствует оператор сопряжения R, задаваемый в виде таблицы 6.1.

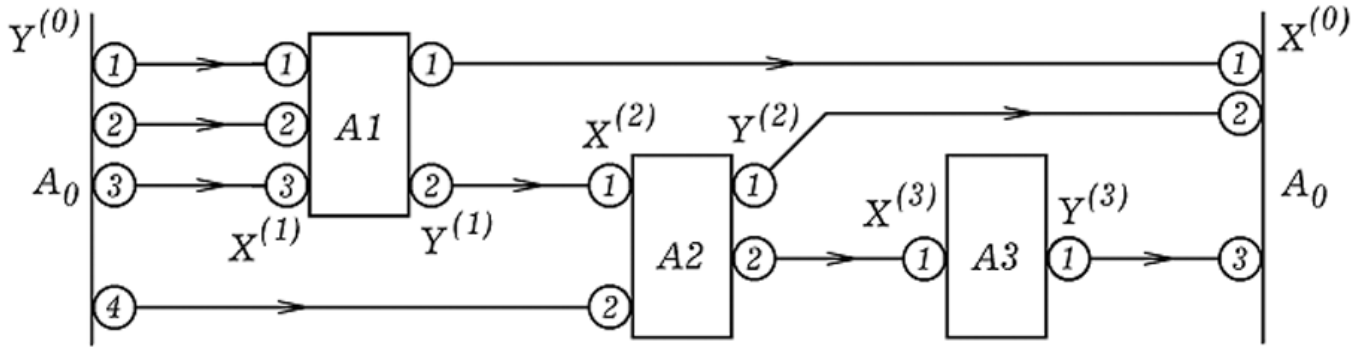

Рис. 6.14. Структура агрегативной системы, изображенной на рис. 6.13

Промоделировать функционирование рассматриваемой системы можно с помощью программы ПР-8 (Приложение). При запуске программа запрашивает режим S работы генератора, частоту f, амплитуду A и коэффициент $K_{\max}$ усилителя. На экране компьютера получаются осциллограммы напряжений $u_1(t)$, $u_2(t)$ и $u_3(t)$.

Таблица 6.1

Номер агрегата	Номер входа i		
	1	2	3
0	1, 1	2, 1	3, 1
1	0, 1	0, 2	0, 3
2	1, 2	0, 4	
3	2, 2		

Приложение к главе 6

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

Программа ПР-1

```
uses crt, graph;
const lambda=1.5; dt=0.01;
var C,b,R,D,t, x,x1,y,y1,p : real; i,j,DV,MV : integer;
n : array[0..25] of integer; M,L : longint;
BEGIN DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
x:=24; M:=200; L:=2000;
For i:=1 to L do begin x:=x+157.3;
  If x>M then Repeat x:=x-M; until x<M;
  x1:=x/M; t:=x1; {ravnomernoe}
  t:=-ln(1-x1)/lambda; {raspred Puassona}
  For j:=0 to 24 do begin
  If (t>=j*0.1)and(t<(j+1)*0.1) then inc(n[j]); end;
end;
For i:=0 to 24 do rectangle(50+20*i,
                      450-round(n[i]*1.3),68+20*i,450);
For i:=0 to 24 do rectangle(51+20*i,
                      451-round(n[i]*1.3),67+20*i,451);
Repeat until keypressed; CloseGraph;
END.
```

Программа ПР-2

```
uses crt, graph;
var t,x,x1,y,y1,p: real; i,j,DV,MV,M,M1: integer;
n: array[-26..26] of integer; L: longint;
BEGIN DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
x:=24; M:=210; M1:=183; L:=30000;
For i:=1 to L do begin
  x:=x+7.312;
  If x>M then Repeat x:=x-M; until x<M;
  x1:=10*x/M-5;
  y:=y+157.1;
  If y>M1 then Repeat y:=y-M1; until y<M1;
```

```
  y1:=y/M1; p:=exp(-x1*x1/2)/sqrt(2*3.14);
  If y1<p then For j:=-25 to 25 do begin
  If (x1>=j*0.1)and(x1<(j+1)*0.1) then inc(n[j]);
end; end;
For i:=-25 to 25 do rectangle(290+10*i,
                      450-round(n[i]*3),298+10*i,450);
For i:=-25 to 25 do rectangle(291+10*i,
                      451-round(n[i]*3),297+10*i,451);
Repeat until keypressed; CloseGraph;
END.
```

Программа ПР-3

```
uses crt;
const rho=2.5;  dt=0.01;
var lambda,tt,tt1,tt2,tt3,tt4,tau,t1,t,tz,tsv,Tobr,b,R,D,
x,x1,y,y1,p : real; otkaz,VZ,S,zay,i,j,DV,MV,z: integer;
k,M,L:longint; Label m1;
BEGIN x:=24; M:=200; Tobr:=1; lambda:=rho/Tobr;
Repeat t:=t+dt; t1:=t1+dt;
  If t1>tau then begin x:=x+157.3;
  If x>M then Repeat x:=x-M; until x<M;
  x1:=x/M; tau:=-ln(1-x1)/lambda; zay:=1; t1:=0; end;
  If zay=1 then begin inc(L);
  writeln('Z A Y A V K A ', L,' ',t); end;
  If S>0 then begin k:=k+S; tz:=tz+1; end;
  If S=0 then begin tsv:=tsv+1; end;
  If (zay=1)and(S=4) then begin inc(otkaz);
             writeln(' otkaz ',otkaz); end;
  If (zay=1)and(S=3) then begin S:=4; tt4:=Tobr; end;
  If (zay=1)and(S=2) then begin S:=3; tt3:=Tobr; end;
  If (zay=1)and(S=1) then begin S:=2; tt2:=Tobr; end;
  If (zay=1)and(S=0) then begin S:=1; tt1:=Tobr; end;
  If S=1 then tt1:=tt1-dt;
  If S=2 then begin tt1:=tt1-dt; tt2:=tt2-dt; end;
  If S=3 then begin tt1:=tt1-dt; tt2:=tt2-dt;
                                 tt3:=tt3-dt; end;
  If S=4 then begin tt1:=tt1-dt; tt2:=tt2-dt;
                    tt3:=tt3-dt; tt4:=tt4-dt; end;
  If tt1<0 then begin S:=S-1; tt1:=tt2; tt2:=tt3;
```

```
          tt3:=tt4; tt4:=0; inc(VZ); goto m1; end;
  If tt2<0 then begin S:=S-1; tt2:=tt3; tt3:=tt4;
                    tt4:=0; inc(VZ); goto m1; end;
  If tt3<0 then begin S:=S-1; tt3:=tt4; tt4:=0;
                             inc(VZ); goto m1; end;
  If tt4<0 then begin S:=S-1; inc(VZ); goto m1; end; m1:
  writeln('vremya ',t,' zayavka ',zay,' SOST ',S,' L= ',
              L,' Vip_Z= ',VZ,' otkaz= ',otkaz); zay:=0;
until (keypressed)or(t>100);
writeln('VEROYATN VIPOLN ',VZ/L,' ',1-otkaz/L);
writeln('PROPUSKN SPOSOB ',VZ/t);
writeln('ZANYAT KANALOV ',k/tz);
writeln('DOLYA VREMYA SVOBODNOI SYSTEMI ',(t-tz*dt)/t,
' ',tsv*dt/t); readkey;
END.
```

Программа ПР-4

```
uses crt;
const rho=4; dt=0.01;
var tt1,tt2,tau,t1,t0,t,tnz,C,b,R,D,Tobr,toch,lambda,
x,x1,y,y1,p : real;
Noch,otkaz,VZ,S,zay,i,j,DV,MV,z: integer;
Dloch,k,M,L :Longint; Label m1;
BEGIN x:=24; M:=200; Tobr:=1; lambda:=rho/Tobr;
Repeat t:=t+dt; t1:=t1+dt; zay:=0;
  If t1>tau then begin x:=x+157.3;
  If x>M then Repeat x:=x-M; until x<M;
  x1:=x/M; tau:=-ln(1-x1)/lambda; zay:=1; t1:=0; end;
  If zay=1 then begin inc(L);
    writeln('Z A Y A V K A ', L); end;
  If (zay=1)and(S=5) then begin inc(otkaz);
    writeln(' O T K A Z ',otkaz); goto m1; end;
  If (zay=1)and(S=4) then begin S:=5; end;
  If (zay=1)and(S=3) then begin S:=4; end;
  If (zay=1)and(S=2) then begin S:=3; end;
  If (zay=1)and(S=1) then begin S:=2; tt2:=Tobr; end;
  If (zay=1)and(S=0) then begin S:=1; tt1:=Tobr; end;
    t0:=t0+S*dt; If S=0 then tnz:=tnz+dt;
  If S>2 then begin Dloch:=Dloch+(S-2); k:=k+1; end;
```

```
  If S=1 then tt1:=tt1-dt;
  If S>=2 then begin tt1:=tt1-dt; tt2:=tt2-dt; end;
  If (S=5)and(tt1<0) then begin S:=4; tt1:=tt2;
    tt2:=Tobr; inc(VZ); goto m1; end;
  If (S=5)and(tt2<0) then begin
    S:=4; tt2:=Tobr; inc(VZ); goto m1; end;
  If (S=4)and(tt1<0) then begin S:=3; tt1:=tt2;
    tt2:=Tobr; inc(VZ); goto m1; end;
  If (S=4)and(tt2<0) then begin
    S:=3; tt2:=Tobr; inc(VZ); goto m1; end;
  If (S=3)and(tt1<0) then begin
    S:=2; tt1:=tt2; tt2:=Tobr; inc(VZ); goto m1; end;
  If (S=3)and(tt2<0) then begin
    S:=2; tt2:=Tobr; inc(VZ); goto m1; end;
  If (S=2)and(tt1<0) then begin S:=1; tt1:=tt2;
    tt2:=0; inc(VZ); goto m1; end;
  If (S=2)and(tt2<0) then begin S:=1; inc(VZ); goto m1; end;
  If (S=1)and(tt1<0) then begin S:=0; inc(VZ); goto m1; end;
    m1: {writeln(zay,' SOST=',S,' L=',L,' VZ=',VZ,
       ' OTK=',otkaz,' OCHERED ',S-2); delay(100);}
until (keypressed)or(t>100);
writeln('VER VIPOLN ',VZ/L,' ',1-otkaz/L);
writeln('VER OTKAZA ',otkaz/L);
writeln('VREMYA ZAY V SYST ',t0/VZ,' ',t0);
writeln('DOLYA VREMYA SYST NE ZANYATA ',tnz/500);
writeln('DLINA OCHEREDI ',Dloch/k); readkey;
END.
```

Программа ПР-5

```
uses crt, graph;
const rho=0.12; dt=0.02;
var tt1,tt2,tau,t1,t0,t,ts12,ts2,tnz,C,b,R,D,Tobr,toch,
lambda,x,x1,y,y1,p : real;
Ksl,Noch,otkaz,VZ,S,i,j,DV,MV,z: integer;
Dloch,k,M,L:Longint; Label m1;
BEGIN x:=24; M:=200; Tobr:=1; lambda:=rho/Tobr; ksl:=0;
Repeat t:=t+dt; t1:=t1+dt; Z:=0;
  If ksl<20 then begin k:=20-ksl;
    If t1>tau then begin x:=x+157.3;
```

```
    If x>M then Repeat x:=x-M; until x<M;
    x1:=x/M; tau:=-ln(1-x1)/lambda/k; Z:=1; t1:=0; end;
  end;
  If Z=1 then begin inc(L); inc(ksl);
    writeln('Z A Y A V K A ', L); end;
  If (Z=1)and(S>1) then S:=S+1;
  If (Z=1)and(S=1) then begin S:=2; tt2:=Tobr; end;
  If (Z=1)and(S=0) then begin S:=1; tt1:=Tobr; end;
  t0:=t0+S*dt;
  If S=0 then tnz:=tnz+dt; if S=1 then ts12:=ts12+dt;
  If S>1 then ts2:=ts2+dt;
  If S>2 then begin Dloch:=Dloch+(S-2); end;
  If S=1 then tt1:=tt1-dt;
  If S>=2 then begin tt1:=tt1-dt; tt2:=tt2-dt; end;
  If (S>2)and(tt1<0) then begin S:=S-1; tt1:=tt2;
     tt2:=Tobr; Ksl:=Ksl-1; inc(VZ); goto m1; end;
  If (S>2)and(tt2<0) then begin S:=S-1; tt2:=Tobr;
                Ksl:=Ksl-1; inc(VZ); goto m1; end;
  If (S=2)and(tt1<0) then begin S:=1; tt1:=tt2; tt2:=0;
                Ksl:=Ksl-1; inc(VZ); goto m1; end;
  If (S=2)and(tt2<0) then begin S:=1; Ksl:=Ksl-1;
                            inc(VZ); goto m1; end;
  If (S=1)and(tt1<0) then begin S:=0; Ksl:=Ksl-1;
                            inc(VZ); goto m1; end;
  m1: writeln(Z,' SOST=',S,' L=',L,' VZ=',VZ,' OTK=',
     otkaz,' OCHERED ',S-2); delay(10);
until (keypressed)or(t>100);
writeln('VER VIPOLN ',VZ/L,' ',1-otkaz/L);
writeln('VER OTKAZA ',otkaz/L);
writeln('VREMYA ZAYAVKI V SYSTEME ',t0/VZ,' ',t0);
writeln('DOLYA VREMENI SYSTEMA NE ZANYATA S=0 ',tnz/100);
writeln('DLINA OCHEREDI ',Dloch*dt/t);
writeln('VER S=1 and S>1 ',ts12/t,' ',ts2/t); readkey;
END.
```

Программа ПР-6

```
uses crt, graph;
const lambda=2.5; dt=0.1;
p_br:array[1..5]of real=(0.1,0.2,0.17,0.13,0.08);
```

```
vr_pr:array[1..5]of real=(1.4,1.2,2,1.5,2.7);
vr_sb:array[1..5]of real=(3.4,2.9,2.2,2.7,1.3);
var sl,tt1,tt2,tau,t,C,b,R,D,Tobr,x,x1,y,y1 : real;
sum,konv,i,j,DV,MV,z: integer;
n: array[0..20] of integer; M:longint;
s,sp : array[0..10]of integer;
tsb,tpr,p: array[0..10]of real;
Procedure RND;
begin sl:=sl+0.173*random(500)/100;
  If sl>1 then Repeat sl:=sl-1; until sl<1;
end;
Procedure Mesto_Sborki;
begin If (sp[i]=3)and(s[i]=1) then tpr[i]:=0;
  tpr[i]:=tpr[i]+dt; { proverka detali }
  If (s[i]=1)and(tpr[i]>vr_pr[i]) then begin RND;
  If sl<p_br[i] then begin s[i]:=1; tpr[i]:=0;
     writeln('BRAK ',i,' '); end else s[i]:=2; end;
  If (sp[i]=1)and(s[i]=2) then tsb[i]:=0;
  tsb[i]:=tsb[i]+dt; { sborka detali }
  If (s[i]=2)and(tsb[i]>vr_sb[i]) then
                    begin s[i]:=3; tsb[i]:=0; end;
end;
BEGIN x:=24; M:=200; randomize;
For i:=1 to 5 do begin s[i]:=1; sp[i]:=3; end;
Repeat t:=t+dt; x:=x+157.3;
  If x>M then Repeat x:=x-M; until x<M;
  x1:=x/M; tau:=-ln(1-x1)/lambda;
  For i:=1 to 5 do Mesto_Sborki; writeln(s[1],' ',s[2],
  ' ',s[3],' ',s[4],' ',s[5],' vremya ',t:2:3,' ',konv);
  For i:=1 to 5 do sp[i]:=s[i]; sum:=0;
  For i:=1 to 5 do sum:=sum+s[i]; delay(50);
  If sum = 15 then begin
  For i:=1 to 5 do s[i]:=1; inc(konv); t:=t+0.2;
  writeln('PEREDVINULI KONVEYER'); delay(50); end;
until (konv=300)or(KeyPressed); writeln(t/konv); Readkey;
END.
```

Программа ПР-7

```
uses crt, graph;
const dt=0.001; r=0.1; m=0.4; l=0.5; I=0.1;
      dlit=0.8; fi_sr=0.186;
var t,t1,tau,x,y,x1,y1,p,M1,M2,fi,w,eps,F,Sum,
    SKO,Sum_fi: real; j,DV,MV: integer;
    N_izm: Longint; n : array[0..100] of longint;
Procedure RND; Label metka;
begin metka: x:=x+7.312;
  If x>M1 then Repeat x:=x-M1; until x<M1;
  x1:=10*x/M1-5; y:=y+157.1;
  If y>M2 then Repeat y:=y-M2; until y<M2;
  y1:=y/M2; p:=exp(-6.25*x1*x1/2)/sqrt(3.14)/0.4;
  If y1<p then tau:=2+x1 else goto metka;
end;
BEGIN DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
      x:=24; M1:=210; M2:=183; x:=24; RND;
Repeat t:=t+dt; t1:=t1+dt;
  If (t1>tau)and(t1<tau+dlit) then begin F:=2; end;
  If (t1>tau+dlit) then begin F:=0; t1:=0; RND; end;
  w:=w; eps:=(F*l*cos(fi)-r*w-m*9.8*l*sin(fi))/I;
  w:=w+eps*dt; fi:=fi+w*dt;
  circle(round(t*8),350-round(100*fi),1);
  circle(round(t*8),350-round(F*50),1);
  If round(t/dt)mod 4=0 then begin Sum_fi:=Sum_fi+fi;
     inc(N_izm); Sum:=Sum+(fi_sr-fi)*(fi_sr-fi);
{writeln(t,' ',fi,' ',N_izm,' ',Sum_fi/(N_izm),Sum);}
  For j:=1 to 60 do begin
  If (fi>=-1+2*j/50)and(fi<-1+2*(j+1)/50) then inc(n[j]);
  end; end;
until (KeyPressed)or(N_izm>50000); cleardevice;
For j:=1 to 60 do rectangle(8*j,
                      450-round(n[j]/25),7+8*j,450);
line(0,450-round(3000/25),640,450-round(3000/25));
{writeln(Sum_fi/(N_izm),' SKO ',sqrt(Sum/(N_izm-1)));}
Repeat until keypressed; CloseGraph;
END.
```

Программа ПР-8

```
uses crt, graph;
const dt=0.00005; R=20; C=0.001; L=0.01; pi=3.1415;
var DV,MV,S: integer;
t,t1,f,w,A,U1,U2,UC,pr2,i,q,Period,U_gen,K_us,K_max:real;
Procedure Gen_impulsov;
begin Period:=1/f; If S=1 then U_gen:=A*sin(w*t);
  If S=2 then If sin(w*t)>0 then U_gen:=A else U_gen:=0;
  If S=3 then begin t1:=t1+dt; U_gen:=A*t1/Period;
  If U_gen>A then t1:=0; end; end;
Procedure Usilitel;
begin K_us:=K_max*exp(-abs(U1/20));
  U2:=U1*K_us+10*random(100)/100; end;
BEGIN writeln('Vvedite regim raboti'); readln(S);
writeln('Vvedite chastotu (1-10)'); readln(f);
writeln('Vvedite amplitudu (5-20)'); readln(A);
Randomize; w:=2*pi*f; K_max:=12;
DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
Repeat t:=t+dt; Gen_impulsov; U1:=U_gen; Usilitel;
  pr2:=(U2-R*i-q/C)/L; i:=i+pr2*dt; q:=q+i*dt; UC:=q/C;
  circle(round(200*t),80-round(3*U_gen),1);
  circle(round(200*t),240-round(1.5*U2),1);
  circle(round(200*t),400-round(UC),1); delay(1);
until (KeyPressed)or(t>20); CloseGraph;
END.
```

Список литературы

1. Боев В. Д., Сыпченко Р. П. Компьютерное моделирование. ИНТУ-ИТ.РУ, 2010. 349 с.
2. Булавин Л. А., Выгорницкий Н. В., Лебовка Н. И. Компьютерное моделирование физических систем. Долгопрудный: Издательский Дом "Интеллект", 2011. 352 с.
3. Бусленко Н. П. Моделирование сложных систем. М.: Наука, 1968. 356 с.

4. Венцель Е. С. Исследование операций: задачи, принципы, методология. М.: Наука, 1988. 208 с.
5. Дворецкий С. И., Муромцев Ю. Л., Погонин В. А. Моделирование систем. М.: Изд. центр "Академия", 2009. 320 с.
6. Майер Р. В. Задачи, алгоритмы, программы: электронное учеб. пособие. Глазов: Глазов. гос. пед. ин-т, 2012. URL: http://maier-rv.glazov.net
7. Паничев В. В., Соловьев Н. А. Компьютерное моделирование: учеб. пособие. Оренбург: ГОУ ОГУ, 2008. 130 с.
8. Рубанов В. Г., Филатов А. Г. Моделирование систем: учеб. пособие. Белгород: Изд-во БГТУ, 2006. 349 с.
9. Советов Б. Я., Яковлев С. А. Моделирование систем: учебник для вузов. М.: Высш. шк., 2001. 343 с.
10. Харин Ю. С., Малюгин В. И., Кирлица В. П. и др. Основы имитационного и статистического моделирования: учеб. пособие. Мн.: Дизайн ПРО, 1997. 288 с.
11. Шевченко Н. Ю. Моделирование систем: учеб. пособие. Томск: Томский межвузовский центр дистанционного образования, 2002. 175 с.
12. Шеннон Р. Имитационное моделирование систем - искусство и наука / пер. с англ. М.: Мир, 1978. 302 с.

ВВЕРХ