



*Томский межвузовский центр
дистанционного образования*

Г.Н. Решетникова

МОДЕЛИРОВАНИЕ СИСТЕМ

Часть 2

Учебное пособие

ТОМСК – 2004

Министерство образования и науки Российской Федерации

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

**Кафедра компьютерных систем в управлении
и проектировании (КСУП)**

Г.Н. Решетникова

МОДЕЛИРОВАНИЕ СИСТЕМ

Часть 2

Учебное пособие

2004

Корректор: Осипова Е.А.

Решетникова Г.Н.

Моделирование систем: Учебное пособие. В 2-х частях. – Томск: Томский межвузовский центр дистанционного образования, 2004. – Ч.2. – 169 с.

Учебное пособие предназначено для студентов вузов, обучающихся по специальностям, которые связаны с проектированием сложных технических комплексов и управлением техническими и экономическими объектами и процессами.

© Решетникова Г.Н., 2004

© Томский межвузовский центр
дистанционного образования, 2004

СОДЕРЖАНИЕ

Предисловие	7
7 ЧИСЛЕННЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ.....	8
7.1 Элементы теории погрешности.....	9
7.2 Приближение данных функциями	12
7.2.1 Интерполирование	13
7.2.2 Сплайн-функции	24
7.2.3 Аппроксимация данных методом наименьших квадратов	29
7.3 Численное дифференцирование.....	37
7.4 Численное интегрирование	39
7.4.1 Интерполяционные квадратурные формулы	40
7.4.2 Квадратурные формулы наивысшей алгебраической степени точности. Квадратурные формулы Гаусса.....	46
7.5 Решение нелинейных уравнений и систем	49
7.5.1 Решение нелинейных уравнений.....	50
7.5.2 Метод Лобачевского при решении алгебраических уравнений	59
7.5.3 Решение систем нелинейных уравнений	64
7.6 Решение задач матричной алгебры.....	65
7.6.1 Некоторые понятия матричной алгебры	66
7.6.2 Обусловленность систем и матриц	70
7.6.3 Метод Гаусса	74
7.6.4 Итерационные методы решения систем линейных алгебраических уравнений.....	76
7.6.5 Метод Данилевского при решении полной проблемы собственных значений. Построение канонической формы Фробениуса.....	81
7.7 Решение обыкновенных дифференциальных уравнений и систем	85
7.7.1 Методы Эйлера и Рунге-Кутта при решении задачи Коши для дифференциального уравнения первого порядка.....	86
7.7.2 Правило Рунге	88
7.7.3 Методы Эйлера и Рунге-Кутта при решении задачи Коши для систем дифференциальных уравнений	90

7.7.4	Решение задачи Коши для систем линейных дифференциальных уравнений.....	91
7.7.5	Методы Адамса при решении задачи Коши для обыкновенных дифференциальных уравнений	92
7.7.6	Метод сеток решения линейных краевых задач	94
7.7.7	Метод коллокации решения краевых задач	97
8	МОДЕЛИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ.....	100
8.1	Описание систем в пространстве состояний	100
8.2	Аналитическое конструирование оптимальных регуляторов.....	104
8.2.1	Оптимальное управление при минимизации классического квадратичного функционала	106
8.2.2	Оптимальное управление при минимизации функционала обобщенной работы	108
8.2.3	Оптимальное управление при минимизации локального квадратичного критерия	112
8.2.4	Синтез следящей системы управления	113
8.3	Моделирование систем оптимального управления	115
8.3.1	Основные понятия цифровых систем управления	115
8.3.2	Моделирование поведения управляемого объекта.....	117
8.3.3	Синтез оптимального управления	120
8.4	Моделирование систем управления при случайных внешних воздействиях.....	125
8.4.1	Моделирование поведения объекта при наличии внешних возмущений.....	125
8.4.2	Описание математической модели измерительного комплекса.....	127
8.4.3	Оценивание состояния модели объекта.....	128
8.4.4	Синтез управляющих воздействий по оценкам состояния	132
8.5	Синтез адаптивной следящей системы	135
8.5.1	Основные понятия адаптивных систем управления.....	136
8.5.2	Одновременное оценивание состояния и параметров модели объекта.....	137
8.6	Учет ограничений и запаздываний по управлению.....	141
8.7	Общая схема синтеза адаптивных систем управления.....	142

9 ПРИМЕРЫ ПОСТРОЕНИЯ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ	145
9.1 Построение математических моделей прямолинейного и вращательного движения	145
9.1.1 Математическая модель прямолинейного движения	145
9.1.2 Математическая модель вращательного движения	147
9.2 Макроэкономическая модель динамики фондов производственного накопления и потребления	148
9.3 Построение математических моделей производства, хранения и сбыта товара повседневного спроса	149
10 ПРОГРАММНЫЕ СРЕДСТВА МОДЕЛИРОВАНИЯ	154
10.1 Языки программирования	155
10.1.1 Язык Fortran	155
10.1.2 Язык C	155
10.1.3 Язык C++	156
10.1.4 Языки Pascal и Object Pascal	159
10.2 Системы разработки программного обеспечения	159
10.2.1 Microsoft Visual C++ и MFC	159
10.2.2 Borland Delphi	160
10.2.3 Borland C++Builder	160
10.3 Специализированные математические пакеты	161
10.3.1 Система Maple	162
10.3.2 Система Mathematica	162
10.3.3 Система MatLAB	164
10.3.4 Система MathCAD	165
10.3.5 Система STATISTICA	167
10.3.6 Система EXCEL	167
ЛИТЕРАТУРА	169

ПРЕДИСЛОВИЕ

При компьютерном моделировании возникает необходимость в численном решении достаточно большого количества разнообразных задач. В связи с этим рассматриваются основные подходы к методам решения таких задач, как приближение данных, численное дифференцирование и интегрирование, решение нелинейных уравнений и систем, решение задач матричной алгебры, решение задачи Коши и краевой для обыкновенных дифференциальных уравнений. Приводятся условия сходимости итерационных алгоритмов, обсуждаются способы реализации и обращается внимание на погрешность результата.

Подробно рассматривается моделирование систем управления в пространстве состояний методами аналитического конструирования оптимальных регуляторов. При этом рассматривается формирование следящей системы адаптивного управления с неполным измерением при построении оценок состояния и параметров модели объекта дискретными фильтрами Калмана. Построение такой системы управления осуществляется путем постепенного усложнения с помощью добавления новых алгоритмов и изменения существующих.

Так как одной из первых и основных задач при формировании систем управления является построение математической модели управляемого объекта, то рассматриваются примеры построения математических моделей, описывающих прямолинейное и вращательное движение, изменение фондов производственного накопления и потребления, математических моделей процессов производства, сбыта и хранения продукции.

В последней главе приводятся характеристики и возможности программных средств, используемых для моделирования систем, как инструментальных, так и специализированных математических пакетов. Эта глава написана в соавторстве с Борисовым Сергеем Ивановичем, программистом ЛИСМО ТУСУР, ассистентом кафедры КСУП ТУСУР.

7 ЧИСЛЕННЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ

Настоящее время характеризуется резким расширением применения математики практически во всех сферах деятельности человека. Это во многом связано с созданием и быстрым развитием средств вычислительной техники. Расширение возможностей приложения математики обусловило математизацию различных разделов науки: химии, экономики, биологии, геологии, психологии, медицины, конкретных разделов техники и др. Процесс *математизации* состоит в построении математических моделей процессов и явлений и в разработке методов их исследования. В физике или механике, например, построение математических моделей для описания различных явлений природы и изучения этих моделей с целью объяснения старых или предсказания новых эффектов явлений являются традиционными. Современные успехи в решении таких важных для общества проблем, как атомные, космические, экономические, вряд ли были бы возможны без применения вычислительной техники и численных методов. Быстрое проникновение математики во многие области знания, в частности, объясняется тем, что математические модели и методы их исследования применимы сразу ко многим явлениям, сходным по своей формальной структуре. Часто математическая модель, описывающая какое-либо явление, появляется при изучении других явлений или при абстрактных математических построениях задолго до конкретного рассмотрения данного явления.

Математические модели реальных объектов и процессов являются, как правило, достаточно сложными. Поэтому моделирование необходимо осуществлять с помощью вычислительной техники с использованием вычислительных алгоритмов.

К вычислительным алгоритмам, которые используются для моделирования, предъявляются следующие основные требования:

- 1) реализуемость – возможность решения задачи за допустимое время;
- 2) экономичность – получение решения за минимальное время среди эквивалентных по точности алгоритмов;
- 3) отсутствие аварийных остановов ЭВМ в процессе вычислений;

4) сходимость итерационных алгоритмов, применяемых для решения конкретных задач;

5) вычислительная устойчивость – отсутствие накопления суммарной погрешности за счет влияния погрешности округления.

Для решения практически каждой задачи существует несколько численных методов. Выбор метода определяется конкретной задачей и условиями ее реализации. В настоящем учебном пособии будут рассмотрены лишь некоторые численные методы, позволяющие осуществлять моделирование с помощью вычислительной техники.

7.1 Элементы теории погрешности

Результаты вычислений особенно достаточно большого объема, что характерно для моделирования реальных процессов, всегда являются неточными. Это вызвано накоплением различного рода погрешностей, влияние которых необходимо учитывать.

Основными источниками погрешностей являются:

1) исходные данные, которые для вычислений часто берутся из эксперимента, а каждый эксперимент дает результат с ограниченной точностью;

2) использование иррациональных величин, которые в ЭВМ представляются приближенно;

3) применение итерационных методов решения задач, которые дают только приближенные результаты;

4) необходимость округления результатов при умножении и делении.

Общепринятой является следующая *классификация погрешностей*:

1) неустраняемая погрешность, которая возникает за счет неточности исходных данных;

2) погрешность метода, возникающая в результате решения задачи;

3) погрешность округления, которая всегда присутствует в вычислениях.

В вычислительной математике большинство задач может быть записано в виде:

$$y = A(x), \quad (7.1.1)$$

где y и x принадлежат заданным пространствам R_1 и R_2 , $A(x)$ – некоторая заданная функция. Задача состоит либо в отыскании y по заданному x , либо в отыскании x по заданному y . Основным методом, при помощи которого в вычислительной математике решают поставленные задачи, является замена пространств R_1 , R_2 и функции A некоторыми другими пространствами \bar{R}_1 , \bar{R}_2 и функцией \bar{A} , при этом замена должна быть сделана таким образом, чтобы решение новой задачи

$$\bar{y} = \bar{A}(\bar{x}), \quad (7.1.2)$$

где $\bar{x} \in \bar{R}_1$, $\bar{y} \in \bar{R}_2$, было в каком то смысле близким к точному решению исходной задачи и его можно было бы практически отыскать. За счет погрешности исходных данных и округления фактически была решена задача

$$\bar{\bar{y}} = \bar{\bar{A}}(\bar{\bar{x}}). \quad (7.1.3)$$

При этом полная погрешность решения задачи может быть записана в виде:

$$y - \bar{\bar{y}} = (y - \bar{y}) - (\bar{y} - \bar{\bar{y}}). \quad (7.1.4)$$

Первая скобка в (7.1.4) характеризует погрешность метода, а вторая – погрешность, возникающую за счет неточности исходных данных и округления, которая называется вычислительной погрешностью. И полная погрешность результатов вычислений складывается из вычислительной погрешности и погрешности метода.

Обозначим точные значения некоторых величин через x^* , y^* , z^* , ..., а соответствующие им приближенные значения через x , y , z ,

Абсолютной погрешностью величины x называется разность между точным и приближенным значением этой величины:

$$\alpha_x = x^* - x. \quad (7.1.5)$$

Предельной абсолютной погрешностью A_x величины x называется наименьшая из верхних границ $|\alpha_x|$, которая может быть найдена, исходя из способа получения числа x .

Относительной погрешностью величины x называется отношение модуля абсолютной погрешности к абсолютному значению приближенной величины x :

$$\delta_x = \frac{|\alpha_x|}{|x|}. \quad (7.1.6)$$

Предельной относительной погрешностью Δ_x величины x называется отношение предельной абсолютной погрешности A_x к абсолютному значению величины x :

$$\Delta_x = \frac{\delta_x}{|x|}. \quad (7.1.7)$$

Рассмотрим зависимость абсолютной и относительной погрешностей функции от соответствующих погрешностей ее аргументов. При этом функцию можно рассматривать как модель вычислительного процесса.

Пусть $y = f(x_1, x_2, \dots, x_n)$ – заданная функция n аргументов x_1, x_2, \dots, x_n , по значениям которых требуется определить y . Будем предполагать, что:

1) функция $f(x_1, x_2, \dots, x_n)$ непрерывно дифференцируема в области определения своих аргументов;

2) величины A_f и Δ_f необходимо определить с небольшой точностью;

3) погрешности аргументов настолько меньше значений соответствующих аргументов, что в сумме ими можно пренебречь.

По определению имеем:

$$\begin{aligned} \alpha_f &= f(x_1^*, x_2^*, \dots, x_n^*) - f(x_1, x_2, \dots, x_n) = \\ &= \sum_{i=1}^n f'_{x_i}(\xi_1, \xi_2, \dots, \xi_n) \alpha_{x_i}, \end{aligned} \quad (7.1.8)$$

где $\xi = (\xi_1, \xi_2, \dots, \xi_n)$ – некоторая точка отрезка, соединяющего $(x_1^*, x_2^*, \dots, x_n^*)$ и (x_1, x_2, \dots, x_n) , а $f'_{x_i}(\xi)$ – частная производная функции f по x_i в точке ξ , $i = \overline{1, n}$. Последнее равенство в (7.1.8) получено согласно формуле конечных приращений. Учитывая предположение относительно малости погрешностей аргументов, заменим $f'_{x_i}(\xi)$ на $f'_{x_i}(x)$ и получим:

$$\alpha_f \approx \sum_{i=1}^n f'_{x_i}(x) \alpha_{x_i} \quad (7.1.9)$$

и

$$A_f \approx \sum_{i=1}^n \left| f'_{x_i}(x) \right| A_{x_i}. \quad (7.1.10)$$

Относительные погрешности функции будут определяться следующим образом:

$$\delta_f \approx \sum_{i=1}^n \left| \frac{f'_{x_i}(x)}{f(x)} \right| \alpha_{x_i} = \sum_{i=1}^n \left| \frac{x_i f'_{x_i}(x)}{f(x)} \right| \delta_{x_i}, \quad (7.1.11)$$

$$\Delta_f \approx \sum_{i=1}^n \left| \frac{f'_{x_i}(x)}{f(x)} \right| A_{x_i} = \sum_{i=1}^n \left| \frac{x_i f'_{x_i}(x)}{f(x)} \right| \Delta_{x_i}. \quad (7.1.12)$$

Примеры.

1. Пусть $f(x) = x_1 + x_2 + x_3 + \dots + x_n$. Тогда

$$A_f \approx \sum_{j=1}^n A_{x_j}, \quad \Delta f \approx \sum_{j=1}^n \left| \frac{x_j}{f(x)} \right| \Delta x_j.$$

2. Пусть $f(x) = x_1 \cdot x_2 \cdot x_3 \cdot \dots \cdot x_n$. Тогда

$$A_f \approx \sum_{j=1}^n \left| \frac{f(x)}{x_j} \right| A_{x_j}, \quad \Delta f \approx \sum_{j=1}^n \Delta x_j.$$

Так как погрешности суммируются, то может случиться так, что через достаточно большое число операций погрешности станут столь большими, что полностью исказят результаты вычислений. В связи с этим необходимо всегда учитывать влияние погрешностей, возникающих при моделировании.

7.2 Приближение данных функциями

При моделировании систем достаточно часто приходится осуществлять приближения данных, заданных таблично, непрерывными функциями. Такие задачи возникают, например, при моделировании систем управления для нестационарных моделей объектов (моделей движения самолета на взлете и посадке, движения судна при изменении курса и т.д.). Основными методами,

которые используются для построения приближающих функций, являются интерполирование алгебраическими многочленами, сплайн-функции, метод наименьших квадратов.

7.2.1 Интерполирование

Пусть задана таблица значений неизвестной функции $y = f(x)$ в точках x_0, x_1, \dots, x_n ($x_0 < x_1 < \dots < x_n$), и необходимо определить значение этой функции в некоторой точке x , не совпадающей с $x_i, i = \overline{0, n}$. Это типичная постановка задач интерполяции (если $x_0 < x < x_n$) или экстраполяции (если $x < x_0$ или $x > x_n$). Точки x_0, x_1, \dots, x_n называются узлами интерполирования.

В математике под *интерполированием* понимается задача нахождения неизвестного значения какой-либо величины по нескольким известным ее значениям и, может быть, по нескольким значениям других величин, с нею связанных (например, ее производных). Без дополнительных предположений о функции $y = f(x)$ эта задача является неопределенной и за $f(x)$ можно принять любое число, если только x не является одним из узлов таблицы.

Предположим, что рассматривается множество F функций f , обладающих некоторыми свойствами. Рассмотрим семейство функций $\{\varphi_n(x, a_0, a_1, \dots, a_n)\}$, определенных на интервале $[a, b]$, содержащем все узлы интерполирования, и зависящих от параметров $a_j, j = \overline{0, n}$. Будем полагать, что

$$f(x) \approx \varphi_n(x, a_0, a_1, \dots, a_n). \quad (7.2.1)$$

Относительно функций φ_n известны следующие предположения:

- 1) функции известны при всяких $n = \overline{0, \infty}$ и определены на отрезке $[a, b]$ для любых значений параметров a_0, a_1, \dots, a_n ;
- 2) с увеличением n семейство функций φ_n расширяется;
- 3) при всяком n функции семейства φ_n обладают теми же свойствами, что и функции F ;

4) для всякой функции $f \in F$ и любого числа $\varepsilon > 0$ существует такое n и такие значения параметров $a_0^*, a_1^*, \dots, a_n^*$, что при всех $x \in [a, b]$ выполняется неравенство:

$$|f(x) - \varphi_n(x, a_0^*, a_1^*, \dots, a_n^*)| < \varepsilon$$

(это условие обеспечивает возможность равномерного на $[a, b]$ сколь угодно точного приближения любой функции $f \in F$ с помощью функций из последовательности семейств φ_n , при этом семейство функций φ_n называется *полным* для множества F);

5) параметры $a_i, i = \overline{0, n}$ выбираются из условия совпадения функций f и φ_n в узлах $x_i, i = \overline{0, n}$. Это приводит к системе уравнений:

$$f(x_i) = \varphi_n(x_i, a_0, a_1, \dots, a_n), \quad i = \overline{0, n}. \quad (7.2.2)$$

Если φ_n – нелинейная функция относительно a_0, a_1, \dots, a_n , то система (7.2.2) будет нелинейной и ее решение может быть только приближенным. Поэтому чаще всего задают функцию φ_n линейной относительно параметров a_0, a_1, \dots, a_n , то есть в виде:

$$\varphi_n(x, a_0, a_1, \dots, a_n) = a_0 u_0(x) + a_1 u_1(x) + \dots + a_n u_n(x), \quad (7.2.3)$$

где $u_i(x), i = \overline{0, n}$ – линейно независимые *координатные* функции, определенные на $[a, b]$. При этом система (7.2.2) примет вид:

$$\begin{aligned} a_0 u_0(x_0) + \dots + a_n u_n(x_0) &= f(x_0), \\ &\dots \end{aligned} \quad (7.2.4)$$

$$a_0 u_0(x_n) + \dots + a_n u_n(x_n) = f(x_n).$$

Система (7.2.4) имеет единственное решение тогда и только тогда, когда ее определитель

$$\Delta = \begin{vmatrix} u_0(x_0) & \dots & u_n(x_0) \\ \dots & \dots & \dots \\ u_0(x_n) & \dots & u_n(x_n) \end{vmatrix} \neq 0. \quad (7.2.5)$$

Последовательности функций $u_i(x), i = \overline{0, n}$, для которых определитель (7.2.5) отличен от нуля при всяких несовпадающих значениях $x_i \in [a, b], i = \overline{0, n}$, называются *последовательностями Чебышева*.

Часто в качестве функций φ_n используются алгебраические многочлены:

$$P_n(x) = a_0 + a_1x + \dots + a_nx^n. \quad (7.2.6)$$

Система (7.2.4) в этом случае имеет вид:

$$\begin{aligned} a_0 + a_1x_0 + \dots + a_nx_0^n &= f(x_0), \\ &\dots \\ a_0 + a_1x_n + \dots + a_nx_n^n &= f(x_n), \end{aligned} \quad (7.2.7)$$

а ее определитель является *определителем Вандермонда*

$$\Delta = \begin{vmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & \dots & x_1^n \\ \dots & \dots & \dots & \dots \\ 1 & x_n & \dots & x_n^n \end{vmatrix},$$

который отличен от нуля для любых несовпадающих узлов $x_i, i = \overline{0, n}$. Таким образом, система (7.2.7) имеет единственное решение и интерполирование функции $f(x)$ по ее значениям в узлах $x_i, i = \overline{0, n}$ с помощью многочлена $P_n(x)$ всегда возможно и единственно. Кроме того, согласно теореме Вейерштрасса, если отрезок $[a, b]$ конечный и замкнутый и функция $f(x)$ непрерывна на нем вместе с производными первых m порядков, то для всякого $\varepsilon > 0$ существует алгебраический многочлен $P_n(x)$ некоторой степени n , для которого при любых $x \in [a, b]$ выполняются неравенства

$$|f^{(i)}(x) - P_n^{(i)}(x)| \leq \varepsilon, \quad i = \overline{0, m}.$$

Таким образом, если удачно расположить узлы на $[a, b]$, то можно получить достаточно хорошие результаты при вычислении значений функции $f(x)$ и ее производных в любой точке отрезка $[a, b]$.

Решим систему (7.2.7) по правилу Крамера. Тогда

$$a_i = \frac{\Delta_i}{\Delta} = \sum_{j=0}^n f(x_j) \frac{\Delta_{ij}}{\Delta}, \quad i = \overline{0, n}, \quad (7.2.8)$$

где Δ_i – определитель, полученный из Δ заменой i -го столбца столбцом свободных членов, Δ_{ij} – алгебраическое дополнение для j -го элемента в Δ_i при разложении по элементам i -го столбца.

Найденные таким образом параметры a_0, a_1, \dots, a_n подставим в (7.2.6), приведем подобные относительно $f(x_j), j = \overline{0, n}$ и получим:

$$P_n(x) = \sum_{j=0}^n f(x_j) \Phi_j(x), \quad (7.2.9)$$

где $\Phi_j(x), j = \overline{0, n}$, – алгебраические многочлены n -ой степени. Учитывая условие совпадения $f(x)$ и $P_n(x)$ в узлах интерполирования $x_i, i = \overline{0, n}$, получим:

$$\Phi_j(x_i) = \begin{cases} 0, & i \neq j, \\ 1, & i = j. \end{cases}$$

Таким образом, $\Phi_j(x)$ есть многочлен n -ой степени, равной нулю в n точках $x_0, \dots, x_{j-1}, x_{j+1}, \dots, x_n$, и, следовательно, представимый в виде:

$$\Phi_j(x) = c_j (x - x_0) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n).$$

Константу c_j определим из условия

$$\Phi_j(x_j) = 1.$$

Тогда

$$c_j(x) = \frac{1}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}$$

и

$$\Phi_j(x) = \frac{(x - x_0) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n)}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}.$$

Таким образом получим функцию, которая называется *интерполяционным многочленом Лагранжа* и обозначается $L_n(x)$:

$$L_n(x) = \sum_{j=0}^n f(x_j) \frac{(x - x_0) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n)}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}. \quad (7.2.10)$$

Обозначим через $\omega_n(x)$ многочлен n -ой степени вида:

$$\omega_n(x) = (x - x_0)(x - x_1) \dots (x - x_n).$$

Тогда многочлен Лагранжа запишется в виде:

$$L_n(x) = \sum_{j=0}^n f(x_j) \frac{\omega_n(x)}{(x - x_j)\omega'_n(x_j)}. \quad (7.2.11)$$

Интерполяция многочленом Лагранжа имеет существенный недостаток: если выяснится, что полученная точность интерполирования недостаточна и для улучшения точности потребуется увеличить число узлов, то все вычисления придется проделать заново. От этого недостатка избавлен другой интерполяционный многочлен – многочлен Ньютона.

Введем следующие обозначения: разностными отношениями первого порядка назовем величины

$$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i},$$

разностными отношениями второго порядка –

$$f(x_i, x_{i+1}, x_{i+2}) = \frac{f(x_{i+1}, x_{i+2}) - f(x_i, x_{i+1})}{x_{i+2} - x_i}$$

и т.д., и разностными отношениями k -го порядка –

$$f(x_i, x_{i+1}, \dots, x_{i+k}) = \frac{f(x_{i+1}, \dots, x_{i+k}) - f(x_i, \dots, x_{i+k-1})}{x_{i+k} - x_i}.$$

Разностные отношения являются симметрическими функциями своих аргументов и

$$f(x_0, x_1, \dots, x_n) = \sum_{j=0}^n \frac{f(x_j)}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}.$$

Для построения многочлена Ньютона запишем многочлен Лагранжа $L_n(x)$ следующим образом:

$$L_n(x) = L_0(x) + [L_1(x) - L_0(x)] + \dots + [L_n(x) - L_{n-1}(x)]. \quad (7.2.12)$$

Разность $L_k(x) - L_{k-1}(x)$ есть многочлен k -ой степени, равный нулю в точках $x_i, i = \overline{0, k-1}$, и, следовательно, представимый в виде:

$$L_k(x) - L_{k-1}(x) = c_k(x - x_0) \dots (x - x_{k-1}).$$

Константу c_k определим, полагая $x = x_k$, и учитывая, что $L_k(x_k) = f(x_k)$, то есть

$$\begin{aligned}
c_k &= \frac{f(x_k)}{(x_k - x_0) \dots (x_k - x_{k-1})} - \\
&\quad - \frac{\sum_{j=0}^{k-1} f(x_j) \frac{(x_k - x_0) \dots (x_k - x_{j-1})(x_k - x_{j+1}) \dots (x_k - x_{k-1})}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_{k-1})}}{(x_k - x_0) \dots (x_k - x_{k-1})} = \\
&= \sum_{j=0}^k \frac{f(x_j)}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_k)} = f(x_0, \dots, x_k).
\end{aligned}$$

Тогда

$$L_k(x) - L_{k-1}(x) = (x - x_0) \dots (x - x_{k-1}) f(x_0, \dots, x_k).$$

Подставляя полученное выражение в (7.2.12) для $k = \overline{1, n}$, получим *интерполяционный многочлен Ньютона*:

$$\begin{aligned}
P_n(x) &= f(x_0) + (x - x_0)f(x_0, x_1) + (x - x_0)(x - x_1)f(x_0, x_1, x_2) + \\
&\quad + \dots + (x - x_0) \dots (x - x_{n-1})f(x_0, x_1, \dots, x_n).
\end{aligned} \tag{7.2.13}$$

Несмотря на более сложное строение, формула Ньютона удобнее для вычислений, чем формула Лагранжа, так как позволяет не фиксировать заранее число узлов, а подключать новые узлы до тех пор, пока не будет достигнута требуемая точность вычислений.

Величина остатка или погрешности интерполирования

$$R(x) = f(x) - P_n(x) \tag{7.2.14}$$

зависит от свойств функции f , выбора узлов x_k , $k = \overline{0, n}$ и от положения точки x на $[a, b]$. Формула Лагранжа для погрешности интерполирования $R(x)$ имеет вид:

$$R(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_n(x), \tag{7.2.15}$$

где ξ – некоторая точка отрезка $[a, b]$. Если обозначить

$$M_{n+1} = \sup_{x \in [a, b]} |f^{(n+1)}(x)|,$$

то

$$R(x) \leq \frac{M_{n+1}}{(n+1)!} |\omega_n(x)|. \tag{7.2.16}$$

Для того, чтобы погрешность в (7.2.16) была наименьшей, необходимо выбрать узлы интерполирования таким образом,

чтобы значение $\sup_{x \in [a, b]} |\omega_n(x)|$ было минимальным. Для этого мож-

но воспользоваться *многочленами Чебышева*:

$$T_n(t) = \cos[n \arccos(t)]. \quad (7.2.17)$$

Многочлены Чебышева обладают следующими свойствами:

1) они являются алгебраическими многочленами соответствующей степени, так как $T_1(x) = x$, $T_2(x) = 2x^2 - 1$, а для построения многочленов более высокой степени можно воспользоваться рекуррентной формулой:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x); \quad (7.2.18)$$

2) коэффициент при старшей степени x в $T_n(x)$ равен 2^{n-1} ;

3) области определения и изменения многочленов Чебышева равны $[-1, 1]$;

4) многочлены $T_n(x)$ имеют n корней на $[-1, 1]$, которые располагаются в точках

$$x_k = \cos \frac{(2k+1)\pi}{2n}, \quad k = \overline{0, n-1}; \quad (7.2.19)$$

5) многочлены $T_n(x)$ достигают $n+1$ экстремум, равный ± 1 , на $[-1, 1]$ в точках

$$x_k = \cos \frac{k\pi}{n}, \quad k = \overline{0, n}; \quad (7.2.20)$$

6) они являются четными или нечетными в зависимости от n , так как

$$T_n(-x) = (-1)^n T_n(x);$$

7) многочлены Чебышева ортогональны с весом $p(x) = \frac{1}{\sqrt{1-x^2}}$ на $[-1, 1]$, так как

$$\int_{-1}^1 p(x) T_n(x) T_m(x) dx = \begin{cases} \pi, & n = m = 0, \\ \frac{\pi}{2}, & n = m \neq 0, \\ 0, & n \neq m; \end{cases}$$

8) графическое представление многочленов Чебышева совпадает с проекциями синусов, нанесенных на цилиндр;

9) многочлены Чебышева являются наименее отклоняющимися от нуля, так как

$$\sup_{x \in [-1,1]} |T_n(x)| = \frac{1}{2^{n-1}},$$

при этом справедливо утверждение, что какой бы другой многочлен $P_n(x)$ степени n со старшим коэффициентом, равным единице, мы ни взяли

$$\sup_{x \in [-1,1]} |P_n(x)| \geq \frac{1}{2^{n-1}}.$$

Таким образом, для того чтобы значение $\sup_{x \in [-1,1]} |\omega_n(x)|$ было наименьшим, необходимо в качестве $\omega_n(x) = (x - x_0)(x - x_1) \dots (x - x_n)$ взять приведенный многочлен Чебышева $\frac{1}{2^n} T_{n+1}(x)$, то есть в качестве узлов интерполирования взять корни многочлена Чебышева:

$$x_i = \cos \frac{(2i+1)\pi}{2n+2}, \quad i = \overline{0, n}. \quad (7.2.21)$$

Тогда для оценки погрешности будет верно неравенство:

$$R(x) \leq \frac{M_{n+1}}{2^n (n+1)!}. \quad (7.2.22)$$

Если интерполирование производится на интервале $[a, b]$, то значения $x_i \in [-1, 1]$ с помощью линейного преобразования

$$t_i = \frac{1}{2} [(b-a)x_i + (b+a)], \quad i = \overline{0, n}, \quad (7.2.23)$$

можно перевести в интервал $[a, b]$, а для оценки погрешности будет верно неравенство:

$$R(x) \leq \frac{M_{n+1}(b-a)^{n+1}}{2^{2n+1}(n+1)!}. \quad (7.2.24)$$

Достаточно часто при интерполировании приходится иметь дело с равноотстоящими узлами. Тогда полагают:

$$x_0 = a, \quad x_n = b, \quad h = \frac{b-a}{n}, \quad x_i = x_0 + ih, \quad i = \overline{0, n}.$$

В этом случае интерполяционные формулы можно значительно упростить, если ввести понятие *конечных разностей*.

Конечной разностью k -го порядка называется величина

$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i, \quad k = 1, 2, \dots,$$

где

$$y_i = f(x_i), \quad \Delta^0 y_i = y_i, \quad i = \overline{0, n}.$$

Так как узлы, лежащие вблизи точки интерполирования x , оказывают большее влияние, чем узлы, лежащие дальше, то целесообразно при построении интерполяционной формулы привлекать сначала ближний узел, а затем остальные в порядке их удаленности от точки x . При этом абсолютные величины слагаемых будут уменьшаться с увеличением их порядкового номера и, начиная с некоторого слагаемого, выйдут за пределы требуемой точности. В соответствии с этим построим конкретные интерполяционные формулы.

Формула Ньютона для интерполирования вперед

Пусть точка x находится ближе к левому концу отрезка $[a, b]$ или слева от него. Для построения интерполяционной формулы Ньютона за x_0 принимается ближайший к x узел интерполирования, а затем узлы привлекаются в следующем порядке:

$$x_0, \quad x_0 + h, \quad x_0 + 2h, \quad \dots, \quad x_0 + nh,$$

при этом, если это возможно, за x_0 выбирается лежащий слева от точки x узел, чтобы не попасть в ситуацию экстраполирования. Тогда многочлен Ньютона (7.2.13) запишется в виде:

$$\begin{aligned} P(x) = & f(x_0) + (x - x_0)f(x_0, x_0 + h) + \\ & + (x - x_0)(x - x_0 - h)f(x_0, x_0 + h, x_0 + 2h) + \dots \\ & + (x - x_0)(x - x_0 - h) \dots (x - x_0 - (n-1)h)f(x_0, x_0 + h, \dots, x_0 + nh). \end{aligned}$$

Если ввести новую переменную $t = \frac{x - x_0}{h}$, то

$$(x - x_0)(x - x_0 - h) \dots (x - x_0 - ih) = h^{i+1}t(t-1) \dots (t-i),$$

и учитывая, что

$$f(x_0, x_0 + h, \dots, x_0 + ih) = \frac{\Delta^i y_0}{i! h^i},$$

получим

$$P(x_0 + th) = y_0 + \frac{t}{1!} \Delta y_0 + \frac{t(t-1)}{2!} \Delta^2 y_0 + \frac{t(t-1)(t-2)}{3!} \Delta^3 y_0 + \dots + \frac{t(t-1)\dots(t-n+1)}{n!} \Delta^n y_0. \quad (7.2.25)$$

Эта формула называется формулой Ньютона для интерполирования вперед.

Формула Ньютона для интерполирования назад

Пусть точка x находится ближе к правому концу отрезка $[a, b]$ или справа от него. За первый узел для построения интерполяционной формулы примем ближайший к x узел интерполирования и обозначим его x_n , при этом, если возможно, следует избегать ситуации экстраполирования. Далее узлы для построения интерполяционной формулы следует привлекать в следующем порядке:

$$x_n, \quad x_n - h, \quad x_n - 2h, \quad \dots, \quad x_n - nh.$$

Введем новую переменную $t = \frac{x - x_n}{h}$. Тогда, преобразовывая аналогично предыдущему, получим формулу Ньютона для интерполирования назад:

$$P(x_n + th) = y_n + \frac{t}{1!} \Delta y_{n-1} + \frac{t(t+1)}{2!} \Delta^2 y_{n-2} + \dots + \frac{t(t+1)(t+2)}{3!} \Delta^3 y_{n-3} + \dots + \frac{t(t+1)\dots(t+n-1)}{n!} \Delta^n y_0. \quad (7.2.26)$$

Интерполяционные формулы Гаусса

Если точка интерполирования x находится в середине отрезка $[a, b]$ вблизи узла x_k , причем правее его, то есть $x > x_k$, то узлы привлекаются в следующем порядке:

$$x_k, \quad x_k + h, \quad x_k - h, \quad \dots, \quad x_k + ih, \quad x_k - ih.$$

Тогда, после введения новой переменной $t = \frac{x - x_k}{h}$ и соответствующих преобразований, получается формула Гаусса для интерполирования вперед, которая имеет вид:

$$\begin{aligned}
P(x_k + th) = & y_k + \frac{t}{1!} \Delta y_k + \frac{t(t-1)}{2!} \Delta^2 y_{k-1} + \frac{t(t^2-1^2)}{3!} \Delta^3 y_{k-1} + \dots \\
& + \frac{t(t^2-1^2) \dots (t^2-(i-1)^2)}{(2i-1)!} \Delta^{2i-1} y_{k-i+1} + \\
& + \frac{t(t^2-1^2) \dots (t^2-(i-1)^2)(t-i)}{(2i)!} \Delta^{2i} y_{k-i}.
\end{aligned} \tag{7.2.27}$$

Если точка x находится вблизи узла x_k , причем $x < x_k$, то для построения интерполяционной формулы узлы привлекаются в следующем порядке:

$$x_k, \quad x_k - h, \quad x_k + h, \quad \dots, \quad x_k - ih, \quad x_k + ih,$$

и, после введения переменной $t = \frac{x - x_k}{h}$ и соответствующих преобразований, получим формулу Гаусса для интерполирования назад:

$$\begin{aligned}
P(x_k + th) = & y_k + \frac{t}{1!} \Delta y_{k-1} + \frac{t(t+1)}{2!} \Delta^2 y_{k-1} + \frac{t(t^2-1^2)}{3!} \Delta^3 y_{k-2} + \dots \\
& + \frac{t(t^2-1^2) \dots (t^2-(i-1)^2)}{(2i-1)!} \Delta^{2i-1} y_{k-i} + \\
& + \frac{t(t^2-1^2) \dots (t^2-(i-1)^2)(t+i)}{(2i)!} \Delta^{2i} y_{k-i}.
\end{aligned} \tag{7.2.28}$$

Если функция $y(x)$ достаточно гладкая, то разности $\Delta^j y_i$, $j = 1, 2, \dots$ убывают с ростом j и для некоторого j они практически равны нулю. Но, так как с ростом порядка разностей растет погрешность их вычисления, то привлечение таких разностей для построения интерполяционной формулы может существенно исказить результат. Существует следующее правило определения максимального порядка разностей, которые ведут себя *правильно*, то есть не вносят дополнительной погрешности в значение интерполяционного многочлена.

Пусть значения y_i , $i = \overline{0, n}$ вычислены с абсолютной погрешностью ε . Так как конечные разности определяются по формуле:

$$\Delta^{j+1}y_i = \Delta^j y_{i+1} - \Delta^j y_i, \quad j = 0, 1, 2, \dots, \quad \Delta^0 y_i = y_i, \quad i = \overline{0, n},$$

то абсолютная погрешность разностей первого порядка равна 2ε , второго порядка – 4ε , ..., s -го порядка – $2^s \varepsilon$. Тогда, если

$$\max_i |\Delta^j y_i| \geq 2^j \varepsilon,$$

$$\max_i |\Delta^{j+1} y_i| < 2^{j+1} \varepsilon,$$

то максимальный порядок разностей, которые ведут себя правильно, равен j . Разности $(j+1)$ -го порядка уже меньше погрешности, поэтому их использование приведет к искажению результата и при построении интерполяционной формулы они отбрасываются.

7.2.2 Сплайн-функции

Пусть на отрезке $[a, b]$ задано разбиение $a = x_0 < x_1 < x_2 < \dots < x_n = b$, в узлах которого известны значения достаточно гладкой функции $y = f(x)$: $y_i = f(x_i)$, $i = \overline{0, n}$. Узлы разбиения делят отрезок $[a, b]$ на n отрезков $[x_0, x_1]$, $[x_1, x_2]$, ..., $[x_{n-1}, x_n]$.

Сплайном называется составная функция $P(x)$, которая вместе с производными нескольких порядков непрерывна на всем отрезке $[a, b]$, а на каждом частичном отрезке $[x_{i-1}, x_i]$ в отдельности является составляющей функцией:

$$P_i(x) = F(i, x_0, \dots, x_n, y_0, \dots, y_n), \quad i = \overline{1, n}.$$

Рассмотрим частный случай, когда функции $P_i(x)$ являются алгебраическими многочленами вида:

$$P_i(x) = \sum_{k=0}^N \alpha_k^{(i)} (x - x_{i-1})^k, \quad i = \overline{0, n},$$

где $\alpha_k^{(i)}$ – коэффициенты, определяемые для каждого частичного отрезка.

Максимальная по всем частичным отрезкам степень многочлена называется *степенью сплайна*, а разность между степенью сплайна и порядком наивысшей непрерывной на $[a, b]$ производной – *дефектом сплайна*.

Среди существующих сплайнов наиболее широкое распространение получили сплайны следующих типов: линейные, параболические, кубические, В-сплайны.

Рассмотрим алгоритмы построения коэффициентов трех первых из перечисленных сплайнов.

Линейный сплайн

Сплайн $P(x)$ состоит из линейных многочленов вида:

$$P_i(x) = a_i + b_i(x - x_{i-1}), \quad x \in [x_{i-1}, x_i], \quad i = \overline{1, n}. \quad (7.2.29)$$

Параметры сплайна $a_i, b_i, i = \overline{1, n}$ определим из условия непрерывности $P(x)$ на $[a, b]$ и требования совпадения значений сплайна с функцией $f(x)$ в узловых точках:

$$P_i(x_i) = P_{i+1}(x_i), \quad i = \overline{0, n}, \quad (7.2.30)$$

$$P_i(x_i) = y_i, \quad i = \overline{0, n}. \quad (7.2.31)$$

Обозначим $h_i = x_i - x_{i-1}$. Тогда для каждого из многочленов можно записать

$$P_i(x_i) = a_i + b_i h_i = y_i,$$

$$P_i(x_{i-1}) = a_i = y_{i-1}, \quad i = \overline{1, n},$$

и для определения коэффициентов линейного сплайна (7.2.29) получим уравнения:

$$a_i = y_{i-1}, \quad i = \overline{1, n}, \quad (7.2.32)$$

$$b_i = \frac{y_i - y_{i-1}}{h_i}, \quad i = \overline{1, n}. \quad (7.2.33)$$

Параболический сплайн

Сплайн $P(x)$ состоит из парабол, то есть $P_i(x)$ имеют вид:

$$P_i(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2, \quad (7.2.34)$$

$$x \in [x_{i-1}, x_i], \quad i = \overline{1, n},$$

Для определения коэффициентов сплайна дополнительно к условиям (7.2.30), (7.2.31) потребуем непрерывности первой производной сплайна на интервале $[a, b]$, то есть:

$$P'_i(x_i) = P'_{i+1}(x_i), \quad i = \overline{1, n-1}. \quad (7.2.35)$$

Обозначив $h_i = x_i - x_{i-1}$, $i = \overline{1, n}$, и, учитывая, что

$$P'_i(x) = b_i + 2c_i(x - x_{i-1}), \quad (7.2.36)$$

получим:

$$a_i = y_{i-1}, \quad (7.2.37)$$

$$a_i + b_i h_i + c_i h_i^2 = y_i, \quad (7.2.38)$$

$$b_{i+1} = b_i + 2c_i h_i. \quad (7.2.39)$$

Тогда коэффициенты a_i , $i = \overline{1, n}$ определяются согласно (7.2.37), а (7.2.38) можно записать в виде:

$$b_i = \frac{y_i - y_{i-1}}{h_i} - h_i c_i, \quad i = \overline{1, n}. \quad (7.2.40)$$

Если теперь записать выражения для b_i и b_{i+1} в виде (7.2.40) и подставить их в (7.2.39), то получим

$$c_i h_i + c_{i+1} h_{i+1} = g_i, \quad i = \overline{1, n-1}, \quad (7.2.41)$$

где

$$g_i = \frac{(y_{i+1} - y_i)}{h_{i+1}} - \frac{(y_i - y_{i-1})}{h_i}, \quad i = \overline{1, n-1}. \quad (7.2.42)$$

Таким образом, уравнения (7.2.37), (7.2.40) и (7.2.41) образуют систему из $3n - 1$ уравнений для определения $3n$ коэффициентов сплайна. Недостающее уравнение получается из дополнительного условия, которое накладывается на значение производной сплайна на конце интервала $[a, b]$, то есть:

$$P'(x_n) = 0 \quad (7.2.43)$$

или

$$b_n + 2c_n h_n = 0. \quad (7.2.44)$$

Если подставить в (7.2.44) выражение для b_n из (7.2.40), то формулу (7.2.44) можно переписать в виде:

$$c_n h_n = g_n,$$

где

$$g_n = \frac{y_{n-1} - y_n}{h_n}.$$

Тогда

$$c_n = \frac{g_n}{h_n} \quad (7.2.45)$$

и

$$c_i = \frac{g_i - c_{i+1}h_{i+1}}{h_i}, \quad i = \overline{n-1, 1}. \quad (7.2.46)$$

Таким образом, параметры параболического сплайна вычисляются в следующем порядке: сначала в обратном порядке вычисляются коэффициенты c_i , $i = \overline{n, 1}$ по (7.2.45), (7.2.46), затем b_i , $i = \overline{1, n}$ по (7.2.39) и a_i , $i = \overline{1, n}$, по (7.2.37).

Кубический сплайн

Сплайн $P(x)$ состоит из гипербола вида:

$$P_i(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, \quad (7.2.47)$$

$$x \in [x_{i-1}, x_i], \quad i = \overline{1, n}.$$

Для определения параметров сплайна потребуем дополнительно к (7.2.30), (7.2.31), (7.2.35) выполнения условия непрерывности второй производной:

$$P_i''(x_i) = P_{i+1}''(x_i), \quad i = \overline{1, n-1}. \quad (7.2.48)$$

Первая и вторая производные отдельных многочленов сплайна соответственно равны:

$$P_i'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2, \quad (7.2.49)$$

$$P_i''(x) = 2c_i + 6d_i(x - x_{i-1}). \quad (7.2.50)$$

Тогда, обозначив $h_i = x_{i-1} - x_i$, получим:

$$P_i(x_{i-1}) = a_i = y_{i-1}, \quad i = \overline{1, n}, \quad (7.2.51)$$

$$P_i(x_i) = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_i, \quad i = \overline{1, n}, \quad (7.2.52)$$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2, \quad i = \overline{1, n-1}, \quad (7.2.53)$$

$$c_{i+1} = c_i + 3d_i h_i, \quad i = \overline{1, n-1}. \quad (7.2.54)$$

Последние два уравнения получены из (7.2.35) и (7.2.48).

Уравнения (7.2.51)–(7.2.54) составляют систему из $4n - 2$ уравнений для определения $4n$ параметров сплайна. Для получения двух недостающих уравнений потребуем выполнения дополнительных условий на концах интервала $[a, b]$:

$$P''(x_0) = 0, \quad (7.2.55)$$

$$P''(x_n) = 0. \quad (7.2.56)$$

Тогда из (7.2.55) следует

$$c_1 = 0, \quad (7.2.57)$$

а из (7.2.56) получим:

$$c_{n+1} = c_n + 3d_n h_n = 0. \quad (7.2.58)$$

Из уравнения (7.2.54) выразим d_i

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \quad i = \overline{1, n}, \quad (7.2.59)$$

и, подставляя его в (7.2.52), с учетом (7.2.51), получим

$$b_i = \frac{y_i - y_{i-1}}{h_i} - \frac{h_i(c_{i+1} + 2c_i)}{3}, \quad i = \overline{1, n}. \quad (7.2.60)$$

Если выражения для b_i , b_{i+1} и d_i подставить в (7.2.53), то получим:

$$h_i c_i + 2(h_i + h_{i+1})c_{i+1} + h_{i+1}c_{i+2} = g_{i+1}, \quad (7.2.61)$$

где

$$g_{i+1} = 3 \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right), \quad i = \overline{1, n-1}. \quad (7.2.62)$$

Уравнения (7.2.61) образуют систему из $n-1$ уравнения относительно неизвестных c_2, c_3, \dots, c_n . Эта система является системой с трехдиагональной матрицей вида:

$$\begin{pmatrix} 2(h_1 + h_2) & h_2 & \dots & 0 & 0 \\ h_2 & 2(h_2 + h_3) & \dots & 0 & 0 \\ 0 & h_3 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & h_{n-1} & 2(h_{n-1} + h_n) \end{pmatrix} \quad (7.2.63)$$

и вектором свободных членов $(g_2, g_3, \dots, g_n)^T$, где T – символ транспонирования.

Решение таких систем осуществляется *методом прогонки*, согласно которому решение представляют в виде:

$$c_{i+1} = \xi_{i+1}c_{i+2} + \eta_{i+1}, \quad i = \overline{n-1, 1}, \quad (7.2.64)$$

где ξ_{i+1}, η_{i+1} – неизвестные коэффициенты. Чтобы получить выражения для этих коэффициентов, запишем формулы для определения c_{i+1} и c_i согласно (7.2.64) и, подставив их в (7.2.61), сравним полученное выражение с (7.2.64). При сравнении получим,

что выражения для определения неизвестных коэффициентов ξ_{i+1}, η_{i+1} будут иметь вид:

$$\begin{aligned}\xi_{i+1} &= \frac{-h_{i+1}}{h_i \xi_i + 2(h_i + h_{i+1})}, \\ \eta_{i+1} &= \frac{g_{i+1} - h_i \eta_i}{h_i \xi_i + 2(h_i + h_{i+1})}, \quad i = \overline{1, n-1}.\end{aligned}\tag{7.2.65}$$

При этом, учитывая (7.2.57), полагают $\eta_1 = 0, \quad \xi_1 = 0$.

Таким образом, порядок вычисления коэффициентов кубического сплайна следующий: сначала определяют коэффициенты $c_i, \quad i = \overline{2, n}$, для чего необходимо, осуществляя прямой ход метода прогонки по формулам (7.2.65), найти значения $\eta_{i+1}, \quad \xi_{i+1}, \quad i = \overline{1, n-1}$, при $\eta_1 = 0, \quad \xi_1 = 0$, а затем, обратным ходом, считая $c_{n+1} = 0$, по формуле (7.2.64), вычислить c_n, c_{n-1}, \dots, c_2 . При этом, согласно (7.2.57), $c_1 = 0$. Остальные коэффициенты сплайна определяются по следующим формулам: $a_i - (7.2.51)$, $d_i - (7.2.59)$, $b_i - (7.2.60)$, $i = \overline{1, n}$.

Полиномиальные сплайны имеют ряд существенных недостатков:

1) при увеличении степени составных многочленов вычисление их коэффициентов значительно усложняется из-за увеличения числа уравнений (условий непрерывности функций и их производных);

2) требуется достаточно большой объем памяти для хранения информации о сплайне (точек разбиения $x_i, \quad i = \overline{0, n}$ и значений коэффициентов сплайнов).

От этих недостатков в значительной мере избавлен другой вид сплайнов, который основан на базисных функциях и называется *B-сплайном*.

7.2.3 Аппроксимация данных методом наименьших квадратов

Пусть x_j — узлы исходной таблицы данных, а $f(x_j), \quad j = \overline{0, n}$ — значения экспериментальных данных или некоторой неизвестной

функции в узловых точках. Введем непрерывную функцию $\varphi(x)$ для аппроксимации дискретных значений $f(x_j)$ и обозначим

$$\xi_j = \varphi(x_j) - f(x_j), \quad j = \overline{0, n},$$

отклонения в узлах x_j . Тогда сумма квадратов отклонений аппроксимирующей функции $\varphi(x)$ от неизвестной функции $f(x)$ в узловых точках x_j , $j = \overline{0, n}$ запишется в виде:

$$Q = \sum_{j=0}^n \xi_j^2 = \sum_{j=0}^n (\varphi(x_j) - f(x_j))^2. \quad (7.2.66)$$

Метод построения аппроксимирующей функции $\varphi(x)$ из условия минимизации суммы квадратов отклонений Q называется *методом наименьших квадратов (МНК)*.

Наиболее часто аппроксимирующую функцию $\varphi(x)$ задают в виде:

$$\varphi(x) = c_0 \varphi_0(x) + c_1 \varphi_1(x) + \dots + c_m \varphi_m(x), \quad (7.2.67)$$

где $\varphi_i(x)$, $i = \overline{0, m}$, $m \leq n$ – линейно независимые *базисные* функции, c_0, \dots, c_m – неизвестные коэффициенты, определяемые из условия минимума Q , т. е. из условий равенства нулю частных производных Q по c_0, \dots, c_m :

$$\frac{\partial Q}{\partial c_k} = 2 \sum_{j=0}^n (c_0 \varphi_0(x_j) + \dots + c_m \varphi_m(x_j) - f(x_j)) \varphi_k(x_j) = 0, \quad (7.2.68)$$

$$k = \overline{0, m}.$$

Таким образом, получаем систему линейных алгебраических уравнений вида $A \cdot c = b$ для определения коэффициентов c_k , $k = \overline{0, m}$. Эта система называется системой *нормальных* уравнений. Матрица системы имеет вид

$$A = \begin{pmatrix} (\varphi_0, \varphi_0) & \dots & (\varphi_0, \varphi_m) \\ \dots & \dots & \dots \\ (\varphi_m, \varphi_0) & \dots & (\varphi_m, \varphi_m) \end{pmatrix} \quad (7.2.69)$$

и называется *матрицей Грама*.

Элементами матрицы Грама являются скалярные произведения базисных функций:

$$(\varphi_i, \varphi_k) = \sum_{j=0}^n \varphi_i(x_j) \varphi_k(x_j). \quad (7.2.70)$$

Вектор свободных членов системы нормальных уравнений имеет вид:

$$b = ((\varphi_0, f), (\varphi_1, f), \dots, (\varphi_m, f))^T, \quad (7.2.71)$$

элементами этого вектора являются скалярные произведения

$$(\varphi_k, f) = \sum_{j=0}^n \varphi_k(x_j) f(x_j). \quad (7.2.72)$$

Матрица Грама обладает следующими основными свойствами:

1) она симметрична, что позволяет сократить объем вычислений при заполнении матрицы;

2) матрица является положительно определенной, поэтому при решении системы нормальных уравнений методом исключения Гаусса можно отказаться от процедуры выбора главного элемента, а сразу брать в качестве ведущего диагональный элемент;

3) определитель матрицы Грама отличен от нуля, если в качестве базиса выбраны линейно независимые функции $\varphi_k(x)$, $k = \overline{0, m}$.

Для аппроксимации экспериментальных данных, определенных с погрешностью ε в каждой узловой точке, обычно сначала функцию $\varphi(x)$ задают в виде линейной комбинации из одной или двух базисных функций и, если после определения коэффициентов c_k окажется, что $\sqrt{Q} > \varepsilon$, то расширяют базис добавлением новых функций $\varphi_k(x)$, так до тех пор, пока не выполнится условие $\sqrt{Q} \approx \varepsilon$.

Выбор конкретных базисных функций зависит от свойств аппроксимируемой функции $f(x)$ таких, как периодичность, экспоненциальный или логарифмический характер, свойства симметрии, наличие асимптотики и т.д.

Аппроксимация алгебраическими полиномами

Пусть аппроксимирующая функция задана в виде алгебраического полинома степени m :

$$\varphi(x) = c_0 + c_1x + \dots + c_mx^m, \quad (7.2.73)$$

или

$$\varphi_0(x) = 1, \quad \varphi_1(x) = x, \quad \dots, \quad \varphi_m(x) = x^m.$$

Степень полинома m выбирают обычно меньше n . Аппроксимирующая кривая в этом случае не проходит через экспериментальные точки, т.е. экспериментальные данные «сглаживаются» с помощью функции $\varphi(x)$. Если взять $m = n$, то $\varphi(x)$ совпадает с многочленом Лагранжа, аппроксимирующая кривая пройдет через все экспериментальные точки и $Q = 0$. Это обстоятельство часто используется для отладки и тестирования программ, реализующих алгоритмы МНК.

Матрица A системы нормальных уравнений и вектор свободных членов для функции $\varphi(x)$ вида (7.2.73) записываются следующим образом:

$$A = \begin{pmatrix} n+1 & \sum_{j=0}^n x_j & \sum_{j=0}^n x_j^2 & \dots & \sum_{j=0}^n x_j^m \\ \sum_{j=0}^n x_j & \sum_{j=0}^n x_j^2 & \sum_{j=0}^n x_j^3 & \dots & \sum_{j=0}^n x_j^{m+1} \\ \dots & \dots & \dots & \dots & \dots \\ \sum_{j=0}^n x_j^m & \sum_{j=0}^n x_j^{m+1} & \sum_{j=0}^n x_j^{m+2} & \dots & \sum_{j=0}^n x_j^{2m} \end{pmatrix}, \quad (7.2.74)$$

$$b = \left(\sum_{j=0}^n f_j, \sum_{j=0}^n x_j f_j, \sum_{j=0}^n x_j^2 f_j, \dots, \sum_{j=0}^n x_j^m f_j \right)^T, \quad (7.2.75)$$

Для решения систем уравнений с матрицей Грама разработаны *методы сингулярного разложения*. Если же $m \leq 4$, то такие системы можно решать и более простым методом исключения Гаусса.

Система нормальных уравнений обычно бывает *плохо обусловленной*, что приводит к дополнительным сложностям при реализации МНК: вычисление с двойной (расширенной) точностью, введение весовых коэффициентов и т.д. Весовые коэффициенты выбираются из различных соображений, например, полагают:

$$w_j = \frac{1}{f(x_j)}, \quad j = \overline{0, n}.$$

Такой выбор весовых коэффициентов называется *методом статистического взвешивания*.

В этом случае:

$$Q = \sum_{j=0}^n w_j [\varphi(x_j) - f(x_j)]^2$$

и скалярные произведения в матрице A и векторе b будут соответственно иметь вид

$$(\varphi_i, \varphi_k) = \sum_{j=0}^n w_j \varphi_i(x_j) \varphi_k(x_j),$$

$$(\varphi_k, f) = \sum_{j=0}^n w_j \varphi_k(x_j) f(x_j).$$

Аппроксимация ортогональными полиномами

Лучшие по точности результаты при аппроксимации можно получить, если использовать в качестве базисных функций $\varphi_i(x)$ классические ортогональные полиномы Чебышева, Лежандра, Лагерра, Якоби и других.

Полиномы называются ортогональными, если существует некоторый интервал $[a, b]$, на котором

$$\int_a^b p(x) \varphi_j(x) \varphi_k(x) dx = 0, \quad j \neq k, \quad (7.2.76)$$

где $p(x)$ – весовая функция.

В случае большого количества узлов x_j на $[a, b]$ значения интегралов (7.2.76) будут близки к дискретным скалярным произведениям (7.2.70), так как интегрирование можно приближенно заменить суммированием. В этом случае недиагональные элементы матрицы Грама будут небольшими по абсолютной величине, что уменьшает погрешность решения системы нормальных уравнений.

Для наиболее гладкого представления экспериментальных данных (с минимальным числом и амплитудой выбросов) в каче-

стве базисных функций $\varphi_k(x)$ выбирают ортогональные полиномы Чебышева $T_k(x)$, которые определены и ортогональны на интервале $[-1, 1]$ с весовой функцией $p(x) = \frac{1}{\sqrt{1-x^2}}$.

Для задания полиномов Чебышева используется рекуррентная формула:

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad (7.2.77)$$

где $T_0(x) = 1$, $T_1(x) = x$.

Так как в многочленах Чебышева коэффициент при старших степенях x равен 2^k , то это не всегда удобно при оценке вклада в аппроксимирующую функцию $\varphi(x)$ старших степеней x по величине коэффициентов c_k . В этом случае полиномы Чебышева можно ввести и по другой рекуррентной формуле, позволяющей построить приведенные многочлены Чебышева:

$$T_{k+1}^*(x) = xT_k^*(x) - \frac{1}{4}T_{k-1}^*(x), \quad (7.2.78)$$

где $T_0^*(x) = 1$, $T_1^*(x) = x$.

Полиномы $T_k^*(x)$ ортогональны на интервале $[-1, 1]$ с такой же весовой функцией, что и $T_k(x)$.

Весовую функцию, равную единице на интервале $[-1, 1]$, имеют полиномы Лежандра, которые определяются по следующей рекуррентной формуле:

$$L_{k+1}(x) = \frac{1}{k+1}[(2k+1)xL_k(x) + kL_{k-1}(x)], \quad (7.2.79)$$

где $L_0(x) = 1$, $L_1(x) = x$.

Интервал $[x_0, x_n]$, где заданы узлы таблицы данных x_j , переводится в интервал $[-1, 1]$, где определены и ортогональны полиномы Чебышева и Лежандра с помощью линейного преобразования:

$$z = 2 \frac{x - x_0}{x_n - x_0} - 1. \quad (7.2.80)$$

Аппроксимация ортогональными полиномами дискретной переменной

Если построить систему базисных функций $\varphi_k(x)$ таким образом, чтобы обращались в нуль скалярные произведения на дискретном множестве узловых точек, то матрица Грама будет диагональной и можно избежать численного решения системы нормальных уравнений. В зависимости от распределения погрешности обрабатываемых данных можно построить ортогональные полиномы дискретной переменной с соответствующими дискретными весовыми функциями $p(x_j)$, $j = \overline{0, n}$. Из классических *ортогональных полиномов дискретной переменной* известны полиномы Хана, Мейкснера, Кравчука, Шарлье.

Рассмотрим алгоритм построения полиномов Чебышева $t_k(x)$ дискретной переменной, которые являются частным случаем полиномов Хана с единичной весовой функцией.

Полагаем:

$$t_0(x) = 1, \quad (7.2.81)$$

$$t_1(x) = x - a_{11}, \quad (7.2.82)$$

и неизвестный коэффициент a_{11} определим из условия ортогональности $t_0(x)$ и $t_1(x)$, то есть

$$(t_0(x), t_1(x)) = 0$$

или

$$\sum_{j=0}^n 1 \cdot (x_j - a_{11}) = \sum_{j=0}^n x_j - a_{11} \sum_{j=0}^n 1 = \sum_{j=0}^n x_j - a_{11}(n+1). \quad (7.2.83)$$

Откуда

$$a_{11} = \frac{1}{n+1} \sum_{j=0}^n x_j. \quad (7.2.84)$$

Полином второй степени также представляется в общем виде с неопределенными коэффициентами a_{21} и a_{20} :

$$t_2(x) = x^2 + a_{21}x + a_{20}. \quad (7.2.85)$$

Коэффициенты a_{21} и a_{20} найдем из условия ортогональности полиномов $t_0(x)$, $t_1(x)$, $t_2(x)$, то есть $(t_0, t_2) = 0$, $(t_1, t_2) = 0$ и т.д.

Для полиномов Чебышева дискретной переменной существует *двухслойная рекуррентная* формула, по которой можно вычислить полином любой степени, зная $t_0(x)$:

$$t_{k+1}(x) = (x - a_{k+1})t_k - b_{k+1}t_{k-1}(x), \quad (7.2.84)$$

где

$$a_{k+1} = \frac{\sum_{j=0}^n x_j t_k^2(x_j)}{\sum_{j=0}^n t_k^2(x_j)}, \quad b_{k+1} = \frac{\sum_{j=0}^n t_k^2(x_j)}{\sum_{j=0}^n t_{k-1}^2(x_j)}. \quad (7.2.85)$$

Аппроксимирующая функция $\varphi(x)$ определяется, как и ранее, в виде линейной комбинации базисных функций, в качестве которых берутся полиномы Чебышева дискретной переменной $t_k(x)$:

$$\varphi(x) = \sum_{k=0}^m c_k t_k(x). \quad (7.2.86)$$

Тогда, так как матрица Грама является диагональной, коэффициенты c_k этой линейной комбинации определяются как частное от деления правых частей получающейся системы нормальных уравнений на диагональные элементы этой матрицы, то есть:

$$c_k = \frac{\sum_{j=0}^n f(x_j) t_k(x_j)}{\sum_{j=0}^n t_k^2(x_j)}. \quad (7.2.87)$$

Заметим, что если для улучшения качества аппроксимации возникает необходимость в увеличении числа базисных функций, то не придется пересчитывать коэффициенты c_k , определенные с меньшим значением m .

На практике достаточно часто при обработке экспериментальных данных можно ограничиться построением линейной аппроксимирующей функции (*линии регрессии*), то есть:

$$\varphi(x) = a + bx.$$

Для коэффициентов a и b из общего алгоритма МНК получаются выражения:

$$a = \bar{f} - b\bar{x}, \quad b = \frac{\sum_{j=1}^n (x_j - \bar{x})(f_j - \bar{f})}{\sum_{j=1}^n (x_j - \bar{x})^2},$$

где

$$\bar{x} = \frac{1}{n} \sum_{j=1}^n x_j, \quad \bar{f} = \frac{1}{n} \sum_{j=1}^n f_j.$$

Погрешность вычисления коэффициентов a и b определяется по формулам:

$$\Delta b = T_{\alpha_n} \sqrt{\frac{1}{n-2} \left[\frac{\sum_{j=1}^n (f_j - \bar{f})^2}{\sum_{j=1}^n (x_j - \bar{x})^2} - b^2 \right]},$$

$$\Delta a = \Delta b \sqrt{\bar{x}^2 + \frac{1}{n} \sum_{j=1}^n (x_j - \bar{x})^2},$$

где T_{α_n} – коэффициент Стьюдента для n измерений и доверительной вероятности α .

7.3 Численное дифференцирование

К численному дифференцированию приходится прибегать в том случае, когда функция $f(x)$, для которой нужно найти производную, задана таблично или же функциональная зависимость x и $f(x)$ имеет очень сложное аналитическое выражение. В первом случае методы дифференциального исчисления просто неприменимы, а во втором случае их использование вызывает значительные трудности.

В тех случаях, когда численное дифференцирование неприменимо, вместо функции $f(x)$ рассматривают интерполяционный многочлен $P_n(x)$ и считают производную от $f(x)$ приближенно равной производной от $P_n(x)$. Естественно, что при этом производная от $f(x)$ будет найдена с некоторой погрешностью.

Пусть требуется найти производную функции $f(x)$ в некоторой точке x , если таблица значений функции $f(x)$ задана в произвольных точках x_j , $j = \overline{0, n}$. В этом случае наиболее удобным является использование в качестве $P_n(x)$ интерполяционного многочлена Ньютона. При этом для построения многочлена Ньютона в качестве первой точки привлекается ближайшая к x табличная точка, а затем остальные, в порядке их удаленности от точки x . Например, если x находится вблизи табличной точки x_k , то для вычисления значения первой производной функции $f(x)$ в точке x получаются следующие выражения:

$$\begin{aligned} f'(x) &\approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}, \\ f'(x) &\approx \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}. \end{aligned} \quad (7.3.1)$$

Функцию $f(x)$ можно записать в виде:

$$f(x) = P_n(x) + r_n(x) \quad (7.3.2)$$

и, дифференцируя это тождество k раз (в предположении, что $f(x)$ и $P_n(x)$ имеют производные k -го порядка), имеем

$$f^{(k)}(x) = P_n^{(k)}(x) + r_n^{(k)}(x). \quad (7.3.3)$$

Так как за приближенное значение $f^{(k)}(x)$ принимается $P_n^{(k)}(x)$, то погрешность дифференцирования есть $r_n^{(k)}(x)$. При замене $f(x)$ интерполяционным многочленом предполагается, что остаточный член $r_n(x)$ мал, но из этого вовсе не следует, что мало $r_n^{(k)}(x)$, так как производные от малой функции могут быть весьма велики. На самом деле практика показывает, что при таком способе вычисления производных $f^{(k)}(x)$ получается сравнительно большая погрешность, особенно при вычислении производных высших порядков.

Если табличные значения являются равноотстоящими, то есть $x_j = x_0 + jh$, $h = (x_n - x_0)/n$, $j = \overline{0, n}$, то для вычисления значения первой производной в точке x , лежащей вблизи табличной точки x_k , получаются следующие выражения:

$$\begin{aligned}
 f'(x) &= \frac{f(x_{k+1}) - f(x_k)}{h} + O(h), \\
 f'(x) &= \frac{f(x_k) - f(x_{k-1}))}{h} + O(h), \\
 f'(x) &= \frac{f(x_{k+1}) - f(x_{k-1}))}{2h} + O(h^2),
 \end{aligned}
 \tag{7.3.4}$$

где $O(h)$, $O(h^2)$ используются для обозначения выражений, имеющих порядок малости, превышающий величины h и h^2 соответственно. Несмотря на то, что с помощью последнего выражения в (7.3.4) получается более точный результат, использовать это выражение для вычисления первой производной нужно с большой осторожностью, так как достаточно часто это приводит к потере устойчивости решения.

Для вычисления второй производной в случае, если точка x находится вблизи табличной точки x_k , используется следующее выражение:

$$f''(x) = \frac{f(x_{k+1}) - 2f(x_k) + f(x_{k-1}))}{h^2} + O(h^2). \tag{7.3.5}$$

Аналогично можно получить выражения для вычисления производных и более высоких порядков. Кроме того, можно использовать и другие подходы для вычисления значений производных функций, заданных таблично, например, *метод неопределенных коэффициентов*.

Заметим, что замена производных *разностными отношениями* типа (7.3.4) и (7.3.5) часто используется при моделировании задач математической физики.

7.4 Численное интегрирование

Будем рассматривать задачу вычисления интеграла при помощи некоторого числа значений интегрируемой функции. Достоинство этого метода состоит в его простоте и универсальности.

Пусть $[a, b]$ есть любой конечный или бесконечный отрезок числовой оси и требуется найти приближенное значение интеграла

$$I = \int_a^b F(x) dx \quad (7.4.1)$$

по n значениям функции $F(x)$ в точках $x_i, i = \overline{0, n}$. Многие правила численного интегрирования основаны на замене интегрируемой функции $F(x)$ на всем отрезке $[a, b]$ или на его частях на более простую функцию, близкую к $F(x)$, легко интегрируемую точно и принимающую в точках $x_i, i = \overline{0, n}$ те же значения, что и $F(x)$. В качестве такой функции достаточно часто используют алгебраический многочлен или рациональную функцию. В том случае, если интегрируемая функция $F(x)$ является достаточно гладкой, то можно рассчитывать хорошо приблизить ее многочленом невысокой степени или несложной рациональной функцией. Если же сама функция $F(x)$ имеет особенности, то это затруднит такое приближение или сделает его вообще невозможным. В этом случае заранее освобождаются от этих особенностей путем их выделения. Для этого функцию $F(x)$ представляют в виде произведения двух функций:

$$F(x) = p(x)f(x), \quad (7.4.2)$$

где $p(x)$ имеет те же особенности, что и $F(x)$, и называется *весовой функцией* или *весом*, а $f(x)$ является достаточно гладкой функцией. Тогда задача заключается в вычислении интеграла вида:

$$I = \int_a^b p(x)f(x) dx. \quad (7.4.3)$$

Правила вычисления интегралов в большинстве своем являются специализированными, предназначенными для численного интегрирования функций, имеющих те же особенности, что и весовая функция $p(x)$. Поэтому при вычислении интеграла (7.4.3) функция $p(x)$ считается фиксированной функцией, а $f(x)$ – любой достаточно гладкой функцией на $[a, b]$.

7.4.1 Интерполяционные квадратурные формулы

Пусть $f(x)$ – достаточно гладкая функция, а интервал $[a, b]$ – конечный и замкнутый. Правило вычисления интеграла будем задавать в виде:

$$\int_a^b p(x)f(x)dx = \sum_{k=0}^n A_k f(x_k) + R_n(f). \quad (7.4.4)$$

Такое правило называется *методом механических квадратур*, сумма – *квадратурной суммой*; A_k , $k = \overline{0, n}$ – *квадратурными коэффициентами*; x_k , $k = \overline{0, n}$ – *квадратурными узлами*.

Функцию $f(x)$ можно приблизить интерполяционным многочленом Лагранжа, который строится по заданным значениям $f(x_i)$, $i = \overline{0, n}$, то есть

$$f(x) = L_n(x) + r_n(x), \quad (7.4.5)$$

где

$$L_n(x) = \sum_{k=0}^n f(x_k) \frac{\omega_n(x)}{(x - x_k)\omega'_n(x_k)}, \quad (7.4.6)$$

$$r_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_n(x), \quad (7.4.7)$$

$$\omega_n(x) = (x - x_0)(x - x_1) \dots (x - x_n), \quad (7.4.8)$$

ξ – некоторая точка интервала $[a, b]$.

Тогда

$$\begin{aligned} \int_a^b p(x)f(x)dx &= \int_a^b p(x)L_n(x)dx + \int_a^b p(x)r_n(x)dx = \\ &= \sum_{k=0}^n A_k f(x_k) + R_n(f) \end{aligned} \quad (7.4.9)$$

и

$$A_k = \int_a^b p(x) \frac{\omega_n(x)}{(x - x_k)\omega'_n(x)} dx, \quad (7.4.10)$$

$$R_n(f) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \int_a^b p(x)\omega_n(x)dx. \quad (7.4.11)$$

Квадратурное правило, коэффициенты которого вычисляются согласно (7.4.10), называется *интерполяционным*. Оно является точным для всех алгебраических многочленов степени не выше n . Поэтому *степень точности* интерполяционного квадратурного правила равна n .

Частным случаем интерполяционного квадратурного правила является квадратурное правило Ньютона-Котеса. Это правило используется в том случае, когда $f(x)$ является достаточно гладкой функцией и узлы квадратурного правила x_k , $k = \overline{0, n}$ являются равноотстоящими. В этом случае считают весовую функцию $p(x) = 1$ и квадратурное правило (7.4.4) записывают в виде:

$$\int_a^b f(x)dx \approx (b-a) \sum_{k=0}^n B_k^n f(x_k). \quad (7.4.12)$$

Коэффициенты B_k^n в (7.4.12), учитывая выражение для A_k (7.4.10), вычисляются по формуле:

$$B_k^n = \frac{A_k}{(b-a)} = \frac{(-1)^{n-k}}{n k! (n-k)!} \int_0^1 \frac{t(t-1) \dots (t-n)}{(t-k)} dt \quad (7.4.13)$$

и имеют следующие конкретные значения:

$$n=0, B_0^0 = 1,$$

$$n = 1, B_0^1 = B_1^1 = \frac{1}{2},$$

$$n = 2, B_0^2 = B_2^2 = \frac{1}{6}, B_1^2 = \frac{4}{6},$$

$$n = 3, B_0^3 = B_3^3 = \frac{1}{8}, B_1^3 = B_2^3 = \frac{3}{8}$$

и т.д.

Коэффициенты B_k^n вычислены до $n = 20$. Они являются рациональными числами и обладают следующими свойствами:

1) при каждом $n \sum_{k=0}^n B_k^n = 1$, в чем легко убедиться, если в (7.4.12) положить $f(x) \equiv 1$;

$$2) B_j^n = B_{n-j}^n;$$

3) при $n = 8$ и для всех $n \geq 10$ среди B_n^k встречаются отрицательные, причем абсолютные величины B_k^n быстро растут с ростом n .

Последнее свойство коэффициентов B_k^n является существенным при определении погрешности вычисления интеграла с

помощью квадратурной суммы. Так, если значения подинтегральной функции $f(x_k)$, $k = \overline{0, n}$ известны с абсолютной погрешностью ε , то неустранимая погрешность вычисления интеграла в (7.4.12) может быть оценена величиной

$$\varepsilon(b-a) \sum_{k=0}^n |B_k^n|, \quad (7.4.14)$$

при этом значения $\sum_{k=0}^n |B_k^n|$ при увеличении n быстро растут. На-

пример, при $n = 20$ эта сумма равна 560. Поэтому при больших значениях n незначительные ошибки в значениях функций $f(x_k)$, $k = \overline{0, n}$ могут привести к большой погрешности в квадратурной сумме (7.4.12). В связи с этим формулы Ньютона-Котеса используются только при малых значениях n . Для уменьшения погрешности результата отрезок $[a, b]$ разбивают на достаточно большое число m интервалов, затем к каждому из них применяют квадратурную формулу с малым числом узлов, и результаты суммируют. При этом, так как погрешность метода для формулы Ньютона-Котеса можно представить в виде $(b-a)^p C(a, b)$, где $C(a, b)$ – медленно изменяющаяся функция на $[a, b]$, а погрешность той же формулы, применённой к отрезку, полученному делением интервала $[a, b]$ на m частей, есть $\left(\frac{b-a}{m}\right)^p C(a, b)$, то по-

сле суммирования погрешность результата примет вид: $\frac{(a-b)^p}{m^{p-1}} C(a, b)$. Таким образом, в результате разбиения интервала интегрирования на m частей погрешность результата уменьшается в m^{p-1} раз.

Заметим, что если средняя точка интервала $[a, b]$ является узлом квадратурного правила, то алгебраическая точность правила увеличивается на единицу, то есть правило становится точным для многочленов степени $n+1$.

Приведем конкретные формулы Ньютона-Котеса для $n = 0, 1, 2$, которые используются чаще всего. При этом приведем сразу обобщенные формулы, полученные делением интервала

$[a, b]$ на m частей и суммированием результатов. При получении формулы для погрешности в случае, когда квадратурный узел является серединой интервала интегрирования, строится интерполяционный многочлен на порядок выше при условии равенства первых производных интерполяционного многочлена и подынтегральной функции в средней точке каждого интервала, полученного при делении. Кроме того, для записи обобщенных квадратурных формул будем использовать следующее обозначение: $f_i = f(x_i)$, $i = \overline{0, n}$.

Квадратурные формулы прямоугольников ($n = 0$).

На интервале $[a, b]$ необходимо выбрать одну любую точку в качестве узла квадратурного правила. Обычно выбирают среднюю или крайние точки: левую или правую, и в соответствии с этим получают следующие формулы:

1) формула левых прямоугольников:

$$\int_a^b f(x) dx \approx \frac{b-a}{m} [f_0 + f_1 + \dots + f_{m-1}], \quad (7.4.15)$$

$$R_{0,лев}^{об}(f) = -\frac{(b-a)^2}{2m} f'(\xi), \quad (7.4.16)$$

2) формула правых прямоугольников:

$$\int_a^b f(x) dx \approx \frac{b-a}{m} [f_1 + f_2 + \dots + f_m], \quad (7.4.17)$$

$$R_{0,пр}^{об}(f) = \frac{(b-a)^2}{2m} f'(\xi), \quad (7.4.18)$$

3) формула средних прямоугольников:

$$\int_a^b f(x) dx \approx \frac{b-a}{m} \left[f\left(a + \frac{h}{2}\right) + f\left(a + \frac{3h}{2}\right) + \dots + f\left(a + \frac{2m-1}{2}h\right) \right], \quad (7.4.19)$$

$$R_{0,ср}^{об}(f) = \frac{(b-a)^3}{24m^2} f''(\xi). \quad (7.4.20)$$

$$\text{В (7.4.19) } h = \frac{b-a}{m}.$$

Квадратурная формула трапеций ($n = 1$):

$$\int_a^b f(x)dx \approx \frac{b-a}{2m} [f_0 + 2(f_1 + f_2 + \dots + f_{m-1}) + f_m], \quad (7.4.21)$$

$$R_1^{об}(f) = -\frac{(b-a)^3}{12m^2} f''(\xi). \quad (7.4.22)$$

Квадратурная формула Симпсона ($n = 2$):

$$\int_a^b f(x)dx \approx \frac{b-a}{3m} [f_0 + f_m + 4(f_1 + f_3 + \dots + f_{m-1}) + 2(f_2 + f_4 + \dots + f_{m-2})], \quad (7.4.23)$$

$$R_2^{об}(f) = -\frac{(b-a)^5}{180m^4} f^{(4)}(\xi). \quad (7.4.24)$$

При использовании квадратурной формулы Симпсона значение m должно быть четным.

В порядке убывания точности вычисления интеграла квадратурные формулы Ньютона-Котеса при фиксированном m располагаются в следующем порядке: формула Симпсона, средних прямоугольников, трапеций, крайних прямоугольников.

Обычно требуется вычислить значение интеграла с некоторой заданной точностью ε . В этом случае строится последовательность значений интеграла I_1, I_2, \dots , для которой справедливо соотношение:

$$\lim_{n \rightarrow \infty} I_n \rightarrow I, \quad (7.4.25)$$

где I_n — значение интеграла, вычисленное на n -ом шаге. Тогда интеграл считается вычисленным с заданной точностью ε , если для некоторого шага k будет справедливо неравенство:

$$|I_{k+1} - I_k| \leq \varepsilon. \quad (7.4.26)$$

Для формул Ньютона-Котеса I_1 соответствует значению интеграла, вычисленному при некотором значении m , I_2 — значению интеграла, вычисленному при увеличении значения m в два раза и т.д. При этом, так как узлы квадратурного правила Ньютона-Котеса являются равноотстоящими, то увеличение m в два раза приведет просто к добавлению к старым узлам новых. Поэтому можно организовать процесс вычисления интеграла таким

образом, чтобы не пересчитывать полученные ранее значения подынтегральной функции, а только добавлять в квадратурную сумму новые значения.

7.4.2 Квадратурные формулы наивысшей алгебраической степени точности. Квадратурные формулы Гаусса

Если вопрос о точности вычисления интеграла стоит очень остро, то можно воспользоваться другими квадратурными формулами, в частности, *квадратурными формулами наивысшей алгебраической степени точности*, которая равна $2n - 1$.

Пусть в квадратурном правиле

$$\int_a^b p(x)f(x)dx \approx \sum_{k=1}^n A_k f(x_k) \quad (7.4.27)$$

$[a, b]$ — есть любой конечный или бесконечный отрезок и весовая функция $p(x)$ такова, что ее произведение на любую неотрицательную степень x абсолютно интегрируемо на $[a, b]$:

$$\int_a^b |p(x)x^i| dx < \infty.$$

Кроме того, будем считать функцию $p(x)$ не эквивалентной нулю, то есть

$$\int_a^b p(x)dx > 0.$$

Квадратурное правило (7.4.27) при фиксированном значении n содержит $2n$ параметров $x_k, A_k, k = \overline{1, n}$, и выбрать их можно так, чтобы равенство (7.4.27) выполнялось точно для всех алгебраических многочленов степени не выше $2n - 1$ или, что равносильно, чтобы выполнялись равенства:

$$\int_a^b p(x)x^i dx = \sum_{k=1}^n A_k x_k^i, \quad i = \overline{0, 2n-1}. \quad (7.4.28)$$

Равенства (7.4.28) образуют систему из $2n$ уравнений относительно $2n$ неизвестных $x_k, A_k, k = \overline{1, n}$, но в силу того, что данная система является нелинейной, ее решение весьма затруднительно.

Введем многочлен, корнями которого являются узлы квадратурного правила:

$$\omega(x) = (x - x_1)(x - x_2) \dots (x - x_n). \quad (7.4.29)$$

Для определения параметров квадратурного правила (7.4.27) доказаны следующие теоремы.

Теорема 1. Для того, чтобы квадратурное правило (7.4.27) было точным для всех алгебраических многочленов степени не выше $2n - 1$, необходимо и достаточно, чтобы выполнялись следующие условия:

1) правило (7.4.27) было интерполяционным, то есть коэффициенты A_k определялись по формулам:

$$A_k = \int_a^b p(x) \frac{\omega(x)}{(x - x_k)\omega'(x_k)} dx, \quad (7.4.30)$$

2) многочлен $\omega(x)$ был ортогонален на $[a, b]$ по весу $p(x)$ ко всякому многочлену $Q(x)$ степени меньшей n :

$$\int_a^b p(x)\omega(x)Q(x)dx = 0. \quad (7.4.31)$$

Теорема 2. Если весовая функция $p(x)$ не меняет знак на $[a, b]$, то существует и при этом единственный многочлен $\omega(x) = (x - x_1)(x - x_2) \dots (x - x_n)$ ортогональный на $[a, b]$ по весу $p(x)$ ко всякому многочлену $Q(x)$ степени меньшей n .

Теорема 3. Если весовая функция $p(x)$ не меняет знак на $[a, b]$ и многочлен $\omega(x)$ ортогонален на $[a, b]$ по весу $p(x)$ ко всякому многочлену $Q(x)$ степени, меньшей n , то все корни многочлена $\omega(x)$ действительные, различные и лежат внутри $[a, b]$.

Теорема 4. Если весовая функция $p(x)$ не меняет знак на $[a, b]$, то ни при каком выборе x_k и A_k равенство (7.4.27) не может быть верным для всех многочленов степени $2n$.

Приведенные теоремы доказывают справедливость следующего утверждения: если весовая функция $p(x)$ сохраняет знак на $[a, b]$, то квадратурное правило (7.4.27), верное для всех многочленов степени не выше $2n - 1$, существует при всех n и является единственным для каждого n . При этом для знакопостоянной весовой функции $p(x)$ степень точности $2n - 1$ является

наивысшей возможной. Таким образом, для конкретной весовой функции $p(x)$ существует единственный многочлен $\omega(x)$, корни которого являются узлами квадратурного правила, а выражение (7.4.30) определяет коэффициенты этого правила.

Квадратурное правило наивысшей алгебраической степени точности для постоянной весовой функции $p(x)=1$ называется *квадратурным правилом Гаусса*. Отрезок интегрирования $[a, b]$ будем считать конечным. Всякий конечный отрезок $[a, b]$ линейной заменой переменных

$$x = \frac{b+a}{2} + \frac{b-a}{2}t \quad (7.4.32)$$

может быть преобразован в отрезок $[-1, 1]$. Тогда выражение (7.4.27) может быть преобразовано к виду:

$$\begin{aligned} \int_a^b f(x)dx &= \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b+a}{2} + \frac{b-a}{2}t\right)dt = \dots \\ &= \frac{b-a}{2} \sum_{k=1}^n A_k f\left(\frac{b+a}{2} + \frac{b-a}{2}t_k\right). \end{aligned} \quad (7.4.33)$$

Систему многочленов, ортогональную на $[-1, 1]$, образуют *многочлены Лежандра*:

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n. \quad (7.4.34)$$

Таким образом, в квадратурной формуле Гаусса узлы t_k , $k = \overline{1, n}$ должны располагаться в корнях многочлена Лежандра степени n и в справочниках можно найти значения t_k , A_k , $k = \overline{1, n}$ для различных n .

Процесс вычисления интеграла методом Гаусса с точностью ε можно организовать следующим образом. Задать конкретное значение n и выписать значения узлов t_k и коэффициентов A_k для этого n . Затем строить последовательность значений интеграла $I_1, I_2, \dots, I_m, \dots$, где величина I_m получается путем суммирования значений интегралов, вычисленных на m отрезках, полученных делением отрезка $[a, b]$ на m частей. При этом каждый частичный отрезок переводится в $[-1, 1]$ и интеграл для каждого

частичного отрезка вычисляется при использовании одних и тех же значений t_k , A_k , $k = \overline{1, n}$. Интеграл будем считать вычисленным с заданной точностью ε , если для некоторого m выполнится неравенство:

$$|I_{m+1} - I_m| \leq \varepsilon. \quad (7.4.35)$$

7.5 Решение нелинейных уравнений и систем

Во многих инженерных и научных задачах возникает необходимость решения уравнений или систем вида:

$$F(x, p_1, p_2, \dots, p_k) = 0, \quad (7.5.1)$$

где F – заданная функция либо вектор-функция $F = (F_1, F_2, \dots, F_n)^T$, x – неизвестная переменная либо вектор $x = (x_1, x_2, \dots, x_n)$, p_1, p_2, \dots, p_k – параметры задачи. Как правило, при моделировании возникает необходимость определения решения в зависимости от параметров. При этом каждому фиксированному набору параметров, в соответствии с физическим смыслом конкретной задачи, соответствует либо конечное, либо бесконечное множество решений.

Так, например, в электродинамике при математическом моделировании электромагнитных волновых колебательных процессов в линиях передачи и резонаторах получают так называемое дисперсионное уравнение вида (7.5.1). В этом случае параметрами p_1, p_2, \dots, p_k являются: частота колебаний, геометрические параметры системы и включений, пространственное распределение диэлектрической и магнитной проницаемостей в электродинамической структуре и т.д. В качестве неизвестного x могут быть выбраны коэффициенты распространения и затухания электромагнитных волн в линиях передачи либо собственные частоты и добротности колебательной системы. Бесконечное множество решений дисперсионного уравнения будет соответствовать бесконечному числу собственных типов волн (колебаний) в исследуемой системе.

Только для простейших нелинейных уравнений и систем удастся найти решение в аналитическом виде. В большинстве

случаев приходится решать уравнения и системы вида (7.5.1) численными методами.

7.5.1 Решение нелинейных уравнений

Пусть дано уравнение

$$f(x) = 0, \quad (7.5.2)$$

где функция $f(x)$ определена и непрерывна на некотором конечном или бесконечном интервале $[a, b]$. Всякое значение x^* , обращающее функцию $f(x)$ в нуль, т.е. такое, что $f(x^*) = 0$, называется *корнем уравнения* (7.5.2) или *нулем функции* $f(x)$.

Будем предполагать, что уравнение (7.5.2) имеет лишь *изолированные корни*, т.е. такие корни x^* , для которых существует окрестность, не содержащая других корней этого уравнения.

Численное решение уравнения (7.5.2) обычно проводят в два этапа:

- 1) *отделение корней* – определение таких интервалов изменения переменной x , где находится только один корень;
- 2) определение корня уравнения с заданной точностью ε .

Для отделения корней, т.е. для определения отрезка $[\alpha, \beta]$, содержащего только один корень, полезна следующая теорема математического анализа.

Теорема. Если непрерывная функция $f(x)$ принимает значения разных знаков на концах отрезка $[\alpha, \beta]$, т.е. $f(\alpha)f(\beta) < 0$, то внутри этого отрезка содержится, по крайней мере, один корень уравнения $f(x) = 0$, т.е. найдется хотя бы одно число $x^* \in [\alpha, \beta]$, такое, что $f(x^*) = 0$. Корень x^* будет единственным, если производная $f'(x)$ существует и сохраняет постоянный знак внутри этого отрезка.

Для определения начального приближения x_0 для реализации итерационных методов нахождения корней или для определения корня уравнения с невысокой точностью можно использовать следующие методы.

Табличный метод. Сначала определяют знаки функции $f(x)$ в граничных точках интервала $[a, b]$, затем определяют зна-

ки функции $f(x)$ в ряде промежуточных точек этого интервала $\alpha_1, \alpha_2, \dots, \alpha_n$, выбор которых учитывает особенности функции $f(x)$. Если окажется, что для некоторой пары точек α_k, α_{k+1} выполняется неравенство:

$$f(\alpha_k)f(\alpha_{k+1}) < 0,$$

то в силу приведенной выше теоремы в интервале $[\alpha_k, \alpha_{k+1}]$ имеется хотя бы один корень уравнения $f(x) = 0$. Затем нужно тем или иным способом убедиться, является ли корень единственным. После чего в качестве начального значения x_0 можно взять любую точку интервала $[\alpha_k, \alpha_{k+1}]$.

Метод половинного деления (метод дихотомии). В методе половинного деления интервал $[a, b]$, на концах которого функция $f(x)$ имеет значения разных знаков, делят пополам и из двух полученных интервалов $[a, \frac{a+b}{2}]$, $[\frac{a+b}{2}, b]$ выбирают тот, на концах которого $f(x)$ имеет значения разных знаков. Затем делят пополам выбранный интервал и т.д. Деление интервала продолжается до тех пор, пока длина последнего выбранного интервала не будет превышать заданное значение ε . Число делений интервала k определяется формулой:

$$k = \frac{\ln \frac{b-a}{\varepsilon}}{\ln 2},$$

и в качестве x_0 можно взять любую точку последнего выбранного интервала.

Метод пропорциональных частей (метод хорд) применяют, если функция $f(x)$ на концах $[a, b]$ принимает значения разных знаков, и, кроме того, $f'(x)$ и $f''(x)$ сохраняют постоянные знаки на $[a, b]$. В этом методе за точку деления отрезка $[a, b]$ берут точку

$$x_1 = a_1 - \frac{f(a)}{f(b) - f(a)}(b - a),$$

которая является нулем функции

$$f_1(x) = f(a) + \frac{x-a}{b-a}[f(b) - f(a)].$$

Эта функция является линейным приближением функции $f(x)$ на $[a, b]$, а x_1 – это точка пересечения функцией $f_1(x)$ оси абсцисс. Затем рассматривают отрезки $[a, x_1]$ и $[x_1, b]$ и выбирают тот, на концах которого функция $f(x)$ имеет разные знаки. На выбранном отрезке строят функцию $f_2(x)$ и находят нуль этой функции и т.д. Если требуется найти корень уравнения с невысокой точностью ε , то деление отрезков продолжается до тех пор, пока длина очередного выбранного отрезка не будет меньше ε или не выполнится неравенство:

$$|f(x_i)| < \varepsilon,$$

где x_i – очередная точка деления отрезка на два. Эти же критерии можно использовать и для определения начального приближения x_0 .

Графический метод. Начальное приближение x_0 можно определить графически как точку пересечения функцией $y = f(x)$ оси абсцисс. Если $f(x)$ сложная функция, то $f(x)$ представляется в виде разности двух функций $\psi_1(x)$ и $\psi_2(x)$, т.е.

$$f(x) = \psi_1(x) - \psi_2(x),$$

каждую из которых можно достаточно просто изобразить графически. Тогда за x_0 принимают абсциссу точки пресечения функций $y = \psi_1(x)$ и $y = \psi_2(x)$.

Самыми распространенными методами решения нелинейных уравнений являются методы простой итерации и метод Ньютона.

Метод простой итерации

Сначала требуется привести заданное уравнение $f(x) = 0$ к канонической форме:

$$x = \varphi(x), \quad (7.5.2)$$

причем для одного уравнения можно построить несколько канонических форм.

Пример. Пусть исходное уравнение имеет вид:

$$\cos(x) + x - \ln(x) = 0.$$

Функции $\varphi(x)$ канонических форм для этого уравнения будут следующими:

$$\varphi_1(x) = \ln(x) - \cos(x),$$

$$\varphi_2(x) = \arccos(\ln(x) - x),$$

$$\varphi_3(x) = \exp(\cos(x) + x).$$

Итерационное правило метода простых итераций имеет вид:

$$x_{n+1} = \varphi(x_n), \quad n = 0, 1, \dots \quad (7.5.3)$$

Геометрически это правило означает следующее (рис 7.5.1). Точное решение x^* является точкой пересечения кривой $y = \varphi(x)$ с биссектрисой $y = x$. За очередное приближение x_{n+1} берется точка, значение которой равно $\varphi(x_n)$. Это значение получается следующим образом: на кривой $y = \varphi(x)$ отмечается точка с координатами $(x_n, \varphi(x_n))$ и проводится прямая параллельная оси абсцисс до пересечения с биссектрисой. Учитывая, что биссектриса – это множество точек, равноудаленных от осей, опускают перпендикуляр на ось абсцисс и находят точку x_{n+1} .

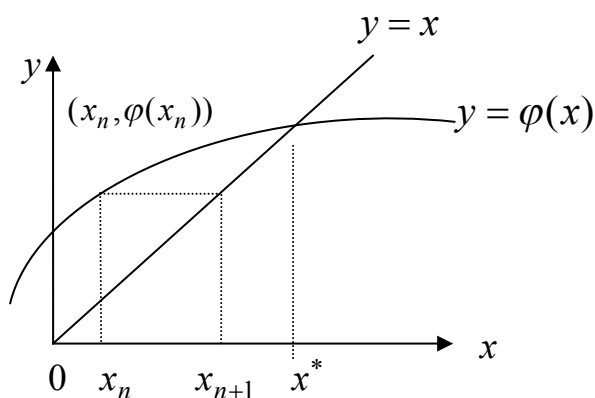


Рис. 7.5.1

Для проверки сходимости метода простых итераций используются следующие теоремы.

Теорема. Пусть выполняются условия:

- 1) функция $\varphi(x)$ определена на отрезке $[x_0 - \delta, x_0 + \delta]$;

2) непрерывна там и удовлетворяет условию Липшица с коэффициентом q , меньшим единицы, т.е. для любых точек $x_1, x_2 \in [x_0 - \delta, x_0 + \delta]$:

$$|\varphi(x_1) - \varphi(x_2)| \leq q|x_1 - x_2|, \quad 0 \leq q < 1; \quad (7.5.4)$$

3) для начального значения x_0 верно неравенство:

$$|x_0 - \varphi(x_0)| \leq m; \quad (7.5.5)$$

4) для чисел δ, q и m выполнено требование:

$$\frac{m}{1-q} \leq \delta. \quad (7.5.6)$$

Тогда

1) уравнение $x = \varphi(x)$ на отрезке $[x_0 - \delta, x_0 + \delta]$ имеет решение;

2) итерационная последовательность приближений $x_n, n=1,2,\dots$, может быть построена, принадлежит отрезку $[x_0 - \delta, x_0 + \delta]$ и является сходящейся, т.е. $\lim_{n \rightarrow \infty} x_n = x^*$, при этом x^* является решением уравнения $x = \varphi(x)$;

3) для x_n выполняется неравенство:

$$|x^* - x_n| \leq \frac{m}{1-q} q^n, \quad (7.5.7)$$

которое характеризует скорость сходимости метода простой итерации.

Теорема (о единственности решения). Уравнение $x = \varphi(x)$ на всяком множестве точек, на котором для $\varphi(x)$ выполняется неравенство

$$|\varphi(x_1) - \varphi(x_2)| < |x_1 - x_2|, \quad (x_1 \neq x_2), \quad (7.5.8)$$

может иметь не более одного решения.

При практическом применении метода простой итерации часто проверяют достаточные условия сходимости итерационного процесса.

Теорема. Пусть функция $\varphi(x)$ определена и дифференцируема на отрезке $[\alpha, \beta]$, причем все ее значения принадлежат этому отрезку. Тогда, если существует q такое, что

$$|\varphi'(x)| \leq q < 1 \quad (7.5.9)$$

для всех $x \in [\alpha, \beta]$, то

1) итерационный метод (7.5.3) сходится независимо от начального приближения $x_0 \in [\alpha, \beta]$;

2) предельное значение $x^* = \lim_{n \rightarrow \infty} x_n$ является единственным корнем уравнения $x = \varphi(x)$ на отрезке $[\alpha, \beta]$.

Замечания.

1. Благодаря тому, что метод простой итерации сходится при любом выборе начального приближения $x_0 \in [\alpha, \beta]$, он является самоисправляющимся, т.е. отдельная ошибка в вычислениях, не выходящая за пределы $[\alpha, \beta]$, не повлияет на конечный результат, так как ошибочное значение можно рассматривать как новое начальное приближение x_0 .

2. Зависимость погрешности на $(n+1)$ -м шаге ε_{n+1} от погрешности на n -м шаге ε_n выражается соотношением

$$\varepsilon_{n+1} \approx \varphi'(x^*)\varepsilon_n,$$

поэтому говорят, что метод простой итерации сходится почти со скоростью геометрической прогрессии со знаменателем

$$q = \max_{x \in [\alpha, \beta]} |\varphi'(x)|.$$

3. Метод простой итерации является одношаговым, т.е. для построения итерационного правила достаточно одного приближения x_0 .

4. Можно построить каноническую форму таким образом, чтобы выполнялись условия сходимости итерационного процесса. Представим функцию $\varphi(x)$ в виде:

$$\varphi(x) = x - \lambda f(x),$$

где параметр λ нужно определить таким образом, чтобы выполнялось условие:

$$\max_{x \in [\alpha, \beta]} |\varphi'(x)| = q < 1.$$

Введем следующие обозначения:

$$M = \max_{x \in [\alpha, \beta]} |f'(x)|, \quad m = \min_{x \in [\alpha, \beta]} |f'(x)|.$$

Тогда параметр λ можно определить следующим образом.

$$1) \lambda = \frac{1}{M} \text{sign}(f'(x_0)), \text{ при этом } 0 \leq \varphi'(x) < 1;$$

$$2) \lambda = \frac{2}{M} \text{sing}(f'(x_0)), \text{ при этом } -1 < \varphi'(x) < 1.$$

Действительно, так как $\varphi'(x) = 1 - \lambda f'(x)$, то в первом случае $q = 1 - \frac{m}{M} < 1$, а во втором $q = 1 - \frac{2m}{M} < 1$.

Существуют видоизменения метода простой итерации. Наиболее известными, увеличивающими скорость сходимости, являются следующие:

1) *метод секущих* (правило линейной интерполяции)

$$x_{n+1} = \frac{x_{n-1}\varphi(x_n) - x_n\varphi(x_{n-1})}{\varphi(x_n) - x_n - \varphi(x_{n-1}) + x_{n-1}}, \quad n = 1, 2, \dots \quad (7.5.10)$$

2) *метод Стеффенсена*

$$x_{n+1} = \frac{x_n\varphi[\varphi(x_n)] - \varphi^2(x_n)}{\varphi[\varphi(x_n)] - 2\varphi(x_n) + x_n}, \quad n = 0, 1, \dots \quad (7.5.11)$$

Метод Ньютона

Метод Ньютона применим к решению широкого класса нелинейных уравнений. Идея этого метода заключается в том, что он позволяет решение нелинейного уравнения свести к решению последовательности линейных задач.

Пусть требуется найти точное решение x^* уравнения $f(x) = 0$ при заданном начальном приближении x_0 . Итерационное правило Ньютона имеет вид:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, \dots \quad (7.5.12)$$

Геометрически метод Ньютона означает следующее (рис. 7.5.2): точное решение x^* является точкой пересечения кривой $y = f(x)$ с осью абсцисс. За очередное приближение x_{n+1} принимается точка пересечения касательной к кривой в точке $(x_n, f(x_n))$ с осью абсцисс.

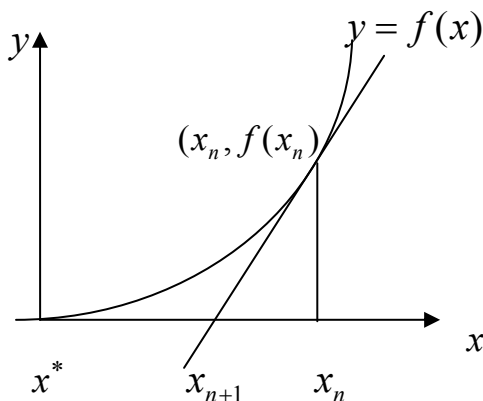


Рис. 7.5.2

Для проверки сходимости метода Ньютона используются следующие теоремы.

Теорема. Пусть выполнены условия:

1) функция $f(x)$ определена и дважды непрерывно дифференцируема на отрезке $[x_0 - \delta, x_0 + \delta]$, при этом $|f''(x)| \leq K$ для всех x на этом отрезке;

2) $f'(x_0) \neq 0$ и $\frac{1}{|f'(x_0)|} \leq B$;

3) $\left| \frac{f(x_0)}{f'(x_0)} \right| \leq \eta$;

4) для B, K, η соблюдено условие $h = BK\eta \leq \frac{1}{2}$;

5) верно неравенство $\frac{1 - \sqrt{1 - 2h}}{h} \eta \leq \delta$.

Тогда:

1) последовательность $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, $n = 0, 1, \dots$, может

быть построена и является сходящейся, т.е. $\lim_{n \rightarrow \infty} x_n = x^*$;

2) предельное значение x^* есть решение уравнения $f(x) = 0$;

3) верна оценка скорости сходимости:

$$|x^* - x_n| \leq t^* - t_n,$$

где t_n – ньютонова последовательность приближений

$$t_{n+1} = t_n - \frac{P(t_n)}{P'(t_n)}$$

к меньшему корню t^* уравнения

$$P(t) = \frac{1}{2}Kt^2 - \frac{1}{B}t + \frac{\eta}{B} = 0,$$

построенная при $t_0 = 0$.

Теорема. При соблюдении условий предыдущей теоремы о сходимости метода Ньютона для разности $x^* - x_n$ верна оценка:

$$|x^* - x_n| \leq \frac{1}{2^{n-1}} (2h)^{2^{n-1}} \eta. \quad (7.5.13)$$

При практическом использовании метода Ньютона часто проверяют достаточные условия сходимости, определяемые следующей теоремой.

Теорема. Если $f(x)$ определена, дважды дифференцируема в $[\alpha, \beta]$ и принимает значения разных знаков на концах интервала $[\alpha, \beta]$, причем $f'(x)$ и $f''(x)$ отличны от нуля и сохраняют постоянные знаки на $[\alpha, \beta]$, то, исходя из начального приближения $x_0 \in [\alpha, \beta]$, удовлетворяющего неравенству

$$f(x_0)f''(x_0) > 0,$$

можно вычислить методом Ньютона единственный корень x^* уравнения $f(x) = 0$ с любой степенью точности.

Замечание. Зависимость погрешности на $(n+1)$ -м шаге ε_{n+1} от погрешности на n -м шаге ε_n в методе Ньютона выражается соотношением

$$\varepsilon_{n+1} \approx -\frac{1}{2} \frac{f''(x^*)}{f'(x^*)} \varepsilon_n^2,$$

поэтому говорят, что сходимость метода Ньютона является почти квадратичной.

Наиболее известными видоизменениями метода Ньютона, которые уменьшают объем вычислений, являются следующие.

1) *метод секущих*

$$x_{n+1} = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}, \quad n = 1, 2, \dots; \quad (7.5.14)$$

2) видоизменение с *постоянным значением производной*

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}, \quad n = 0, 1, 2, \dots \quad (7.5.15)$$

7.5.2 Метод Лобачевского при решении алгебраических уравнений

Особое место среди нелинейных уравнений занимают алгебраические уравнения или полиномы n -ой степени, которые можно представить в виде:

$$P(x) = a_0 x^n + a_1 x^{n-1} + \dots + a_n = 0. \quad (7.5.16)$$

Для нахождения корней алгебраического уравнения можно использовать методы простой итерации и Ньютона, но, во-первых, эти методы предназначены для отыскания действительных простых корней и, во-вторых, даже если нужно найти только действительные корни необходимо для каждого корня указать интервалы, содержащие изолированные корни и определить начальные приближения.

Для нахождения корней полиномов существует метод Лобачевского. Он не требует определения начальных приближений для корней и позволяет одновременно найти все корни полинома (7.5.16). Недостатком этого метода является тот факт, что при вычислениях приходится иметь дело с числами, которые сильно различаются по порядкам величин.

Пусть коэффициенты полинома (7.5.16) a_0, a_1, \dots, a_n являются действительными числами, а корни пронумерованы в следующем порядке:

$$|x_1| \geq |x_2| \geq \dots \geq |x_n|. \quad (7.5.17)$$

Кроме того, будем предполагать, что все корни являются действительными и различными. В основание метода Лобачевского положены следующие соотношения между корнями и коэффициентами алгебраического уравнения (равенства Виетта):

$$\begin{aligned}
x_1 + x_2 + \dots + x_n &= -\frac{a_1}{a_0}, \\
x_1x_2 + x_1x_3 + \dots + x_{n-1}x_n &= \frac{a_2}{a_0}, \\
x_1x_2x_3 + \dots + x_{n-2}x_{n-1}x_n &= -\frac{a_3}{a_0}, \\
&\dots \\
x_1x_2x_3\dots x_n &= (-1)^n \frac{a_n}{a_0}.
\end{aligned} \tag{7.5.18}$$

Будем говорить, что корни x_i , $i = \overline{1, n}$ *сильно разделены в смысле отношения их модулей*, если модуль предыдущего корня во много раз больше модуля последующего корня:

$$\left| \frac{x_i}{x_{i+1}} \right| \gg 1, \quad i = \overline{1, n-1}. \tag{7.5.19}$$

Если выполняются соотношения (7.5.19), то равенства Виетта (7.5.18) значительно упрощаются. Так, первое уравнение в (7.5.18) можно записать в виде:

$$x_1 \left(1 + \frac{x_2}{x_1} + \dots + \frac{x_n}{x_1} \right) = -\frac{a_1}{a_0},$$

и, так как для корней выполняется соотношение (7.5.19), то все отношения, стоящие в скобках, будут величинами, пренебрежимо малыми в сравнении с единицей, и ими можно пренебречь. Тогда получим

$$x_1 \approx -\frac{a_1}{a_0}.$$

Аналогичное будет иметь место и для всех остальных равенств Виетта и (7.5.18) можно заменить следующей системой приближенных равенств, верных лишь в принятой точности вычислений:

$$\begin{aligned}
x_1 &\approx -\frac{a_1}{a_0}, \\
x_1 x_2 &\approx \frac{a_2}{a_0}, \\
x_1 x_2 x_3 &\approx -\frac{a_3}{a_0}, \\
&\dots \\
x_1 x_2 \dots x_n &\approx (-1)^n \frac{a_n}{a_0}.
\end{aligned} \tag{7.5.20}$$

Тогда из (7.5.20) следует, что

$$x_1 \approx -\frac{a_1}{a_0}, \quad x_2 \approx -\frac{a_2}{a_1}, \quad x_3 \approx -\frac{a_3}{a_2}, \quad \dots, \quad x_n \approx -\frac{a_n}{a_{n-1}}. \tag{7.5.21}$$

Таким образом найти сильно разделенные корни алгебраического уравнения достаточно просто. Поэтому решение уравнения (7.6.16) необходимо начинать с разделения корней. Для этого можно воспользоваться *процессом квадрирования*, то есть построением последовательности таких полиномов, у которых корни последующего равны квадратам соответствующих корней предыдущего. Это равносильно вычислению коэффициентов для последовательности полиномов по следующим рекуррентным формулам:

$$\begin{aligned}
a_0^{(k+1)} &= (a_0^{(k)})^2, \\
a_1^{(k+1)} &= (a_1^{(k)})^2 - 2a_0^{(k)} a_2^{(k)}, \\
a_2^{(k+1)} &= (a_2^{(k)})^2 - 2a_1^{(k)} a_3^{(k)} + 2a_0^{(k)} a_4^{(k)}, \\
&\dots \\
a_n^{(k+1)} &= (a_n^{(k)})^2.
\end{aligned} \tag{7.5.22}$$

Процесс квадрирования можно прекратить, если в пределах принятой точности выполняются соотношения:

$$a_i^{(k+1)} \approx (a_i^{(k)})^2, \quad i = \overline{0, n}. \tag{7.5.23}$$

Тогда для полинома

$$P^{(k)}(x) = a_0^{(k)} x^n + a_1^{(k)} x^{n-1} + \dots + a_n^{(k)}, \tag{7.5.24}$$

в силу разделенности его корней, выполняются соотношения аналогичные (7.5.21), а модули приближенных значений корней исходного уравнения (7.5.16) можно определить из следующих равенств:

$$|x_1|^m = \left| \frac{a_1^{(k)}}{a_0^{(k)}} \right|, \quad |x_2|^m = \left| \frac{a_2^{(k)}}{a_1^{(k)}} \right|, \quad \dots, \quad |x_n|^m = \left| \frac{a_n^{(k)}}{a_{n-1}^{(k)}} \right|, \quad (7.5.25)$$

где $m = 2^k$, при этом знаки корней определяются подстановкой в исходное уравнение.

Если корни уравнения (7.5.16) все действительные и среди них есть равные по абсолютной величине, например, x_2 и x_3 , то для уравнения (7.5.24), полученного после квадрирования, будут справедливы следующие приближенные равенства:

$$\begin{aligned} x_1^m &\approx -\frac{a_1^{(k)}}{a_0^{(k)}}, \\ 2x_1^m x_2^m &\approx -\frac{a_2^{(k)}}{a_0^{(k)}}, \\ x_1^m x_2^{2m} &\approx -\frac{a_3^{(k)}}{a_0^{(k)}}, \\ &\dots \\ x_1^m x_2^{2m} \dots x_n^m &\approx (-1)^n \frac{a_n^{(k)}}{a_0^{(k)}}. \end{aligned} \quad (7.5.26)$$

Тогда из второго равенства в (7.5.26) следует, что

$$|x_2|^m = \frac{1}{2} \left| \frac{a_2^{(k)}}{a_0^{(k)}} \right|,$$

а из третьего –

$$|x_2|^{2m} = \left| \frac{a_3^{(k)}}{a_0^{(k)}} \right|,$$

и любое из этих равенств можно использовать для определения модулей корней x_2 и x_3 , знаки которых, как и ранее, определяются подстановкой в исходное уравнение.

Если уравнение имеет комплексные корни, то они будут парно сопряжены, так как все коэффициенты уравнения действи-

тельные. Пусть, например, x_2 и x_3 – пара комплексно сопряженных корней. Эти корни можно представить в виде:

$$x_2 = re^{i\varphi}, \quad x_3 = re^{-i\varphi}. \quad (7.5.27)$$

Так как $x_2^m + x_3^m = 2r^m \cos(m\varphi)$ и $x_2^m x_3^m = r^{2m}$, то для уравнения (7.5.24), полученного после квадрирования, будут справедливы следующие приближенные равенства:

$$\begin{aligned} x_1^m &\approx -\frac{a_1^{(k)}}{a_0^{(k)}}, \\ 2x_1^m r^m \cos(m\varphi) &\approx \frac{a_2^{(k)}}{a_0^{(k)}}, \\ x_1^m r^{2m} &\approx -\frac{a_3^{(k)}}{a_0^{(k)}}, \\ x_1^m r^{2m} x_4^m &\approx \frac{a_4^{(k)}}{a_0^{(k)}}, \\ &\dots \\ x_1^m r^{2m} \dots x_n^m &\approx (-1)^n \frac{a_n^{(k)}}{a_0^{(k)}}. \end{aligned} \quad (7.5.28)$$

Тогда модуль комплексных чисел можно определить из соотношения:

$$r^{2m} = \left| \frac{a_3^{(k)}}{a_1^{(k)}} \right|, \quad (7.5.29)$$

а для определения аргумента можно воспользоваться первым соотношением Виетта в (7.5.18), которое в данном случае будет иметь вид:

$$x_1 + 2r \cos(\varphi) + x_4 + \dots + x_n = -\frac{a_1}{a_0}. \quad (7.5.30)$$

Отсюда определяется значение $\cos(\varphi)$, а $\sin(\varphi) = \sqrt{1 - \cos^2(\varphi)}$.

Таким образом, решение алгебраического уравнения методом Лобачевского осуществляется в несколько этапов.

1. Разделение корней путем квадрирования, при этом по поведению коэффициентов, получаемых в процессе квадрирования, делается вывод о том, какими являются корни уравнения:

- если все коэффициенты стремятся к квадратам соответствующих коэффициентов, полученных на предыдущем шаге процесса квадрирования, то все корни действительные и различные;
- если какой-то коэффициент стремится к квадрату соответствующего коэффициента, деленному на целое число, то номер этого коэффициента указывает на номер первого из равных по модулю корней уравнения в соответствии с нумерацией (7.5.17), а целое число указывает на количество таких корней;
- если какой-то коэффициент меняет знак в процессе квадрирования, то это указывает на наличие комплексно сопряженных корней, причем номер этого коэффициента указывает на первый из таких корней.

2. Вычисление значений корней по формулам, которые соответствуют сделанным выводам о виде корней.

7.5.3 Решение систем нелинейных уравнений

Пусть требуется найти решение $x_1^*, x_2^*, \dots, x_n^*$ системы нелинейных уравнений:

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= 0, \\ f_2(x_1, x_2, \dots, x_n) &= 0, \\ &\dots \\ f_n(x_1, x_2, \dots, x_n) &= 0. \end{aligned} \tag{7.5.31}$$

Введем следующие обозначения:

$$x = (x_1, \dots, x_n), \quad f(x) = (f_1(x), \dots, f_n(x))^T.$$

Тогда система (7.5.31) запишется в виде:

$$f(x) = 0, \tag{7.5.32}$$

Для решения систем нелинейных уравнений используются методы *простой итерации* и *Ньютона*, которые являются естественным распространением этих методов для решения уравнений на многомерный случай.

Для метода простой итерации систему (7.5.32) необходимо привести к канонической форме

$$x = \varphi(x), \quad (7.5.33)$$

где $\varphi(x) = (\varphi_1(x), \dots, \varphi_n(x))^T$. Итерационное правило простой итерации записывается в виде:

$$x^{(k+1)} = \varphi(x^{(k)}), \quad k = 0, 1, \dots \quad (7.5.34)$$

Итерационное правило метода Ньютона имеет вид:

$$x^{(k+1)} = x^{(k)} - [f'(x^{(k)})]^{-1} f(x^{(k)}), \quad k = 0, 1, \dots, \quad (7.5.35)$$

где $[f'(x^{(k)})]^{-1}$ – обратная матрица к матрице Якоби вида:

$$f'(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}. \quad (7.5.36)$$

Для реализации этих методов необходимо отделить корни и задать начальное приближение $x^{(0)}$. Проверка сходимости методов достаточно сложна.

7.6 Решение задач матричной алгебры

В матричной алгебре рассматриваются четыре класса основных задач: решение систем линейных алгебраических уравнений, вычисление определителей, нахождение обратных матриц, определение собственных значений и собственных векторов матриц. Все эти задачи имеют важное прикладное значение при моделировании в различных областях науки и техники. Кроме того, задачи матричной алгебры являются вспомогательными при реализации многих алгоритмов вычислительной математики, математической физики, обработки результатов экспериментальных исследований.

7.6.1 Некоторые понятия матричной алгебры

Пусть $x = (x_1, \dots, x_n)^T \in R^n$ – вектор n -мерного пространства, $A = (a_{ij}) \in R^{n \times n}$ – матрица порядка n .

Одним из важных понятий матричной алгебры является понятие нормы векторов и матриц.

Нормой вектора x называется действительное число, обозначаемое $\|x\|$, удовлетворяющее условиям:

- 1) $\|x\| > 0$, если $x \neq 0$ и $\|0\| = 0$;
- 2) $\|c \cdot x\| = |c| \cdot \|x\|$, при любом численном множителе c ;
- 3) $\|x + y\| \leq \|x\| + \|y\|$.

Из такого определения нормы вектора непосредственно следует, что $\|x - y\| \geq \left| \|x\| - \|y\| \right|$.

Говорят, что последовательность векторов $x^{(k)}$ сходится к вектору x по норме, если $\|x - x^{(k)}\| \rightarrow 0$ при $k \rightarrow \infty$.

Вводить норму вектора можно различными способами, но при этом должны выполняться условия 1) – 3) определения нормы.

Наиболее часто используются следующие определения нормы вектора.

1. *Первая (кубическая) норма:*

$$\|x\|_I = \max_i |x_i|. \quad (7.6.1)$$

Множество векторов вещественного пространства, для которых $\|x\|_I \leq 1$ заполняет единичный куб.

2. *Вторая (октаэдрическая) норма:*

$$\|x\|_{II} = \sum_{i=1}^n |x_i|. \quad (7.6.2)$$

Множество векторов вещественного пространства, для которых $\|x\|_{II} \leq 1$, заполняет n -мерный аналог октаэдра (восьмиугольника).

3. *Третья (сферическая или евклидова) норма:*

$$\|x\|_{III} = \sqrt{(x, x)} = \sqrt{\sum_{i=1}^n |x_i|^2}. \quad (7.6.3)$$

Это длина вектора. Совокупность векторов, для которых $\|x\|_{III} \leq 1$, заполняет шар единичного радиуса.

Скалярное произведение векторов $x = (x_1, \dots, x_n)^T$ и $y = (y_1, \dots, y_n)^T$ вводится по формуле:

$$(x, y) = \sum_{i=1}^n x_i \bar{y}_i,$$

где \bar{y}_i – компоненты вектора, комплексно сопряженного вектору y .

Норма вектора может быть определена многими способами в зависимости от условий задачи и целей исследования, но при всяком определении она должна удовлетворять трем условиям, которые являются аксиомами общей или абстрактной нормы.

Нормой матрицы A называют число $\|A\|$, удовлетворяющее условиям:

- 1) $\|A\| = 0$, если $A = 0$ и $\|0\| = 0$;
- 2) $\|c \cdot A\| = |c| \cdot \|A\|$ для всякого числового множителя c ;
- 3) $\|A + B\| \leq \|A\| + \|B\|$;
- 4) $\|A \cdot B\| \leq \|A\| \cdot \|B\|$.

Для нормы матрицы верно неравенство:

$$\|A - B\| \geq \left| \|A\| - \|B\| \right|.$$

В большинстве случаев приходится одновременно рассматривать матрицы и векторы, потому их нормы рационально вводить так, чтобы они были в какой-то мере согласованными. Обычно говорят, что норма матрицы A *согласована* с нормой вектора, если для всякого вектора x размерности n выполняется неравенство:

$$\|Ax\| \leq \|A\| \|x\|. \quad (7.6.4)$$

Среди согласованных норм матрицы часто (особенно при получении оценок, чтобы сделать их точными) выбирают наименьшую. Так как случай нулевого вектора интереса не имеет, то

неравенство (7.6.4) можно записать в виде $\|Ay\| \leq \|A\|$, где $y = \frac{x}{\|x\|}$, т.е. $\|y\| = 1$.

Последнее означает, что согласованная норма матрицы должна быть верхней границей норм векторов Ay , при условии, что $\|y\| = 1$. Наименьшей согласованной нормой будет точная верхняя граница множества значений $\|Ay\|$ при $\|y\| = 1$, т.е.

$$\|A\| = \max_{\|x\|=1} \|Ax\|. \quad (7.6.5)$$

Эта норма называется нормой матрицы, *подчиненной* норме вектора.

Наиболее распространенными подчиненными нормами матриц являются следующие определения норм матриц.

1. *Первая (кубическая) норма:*

$$\|A\|_I = \max_i \sum_{j=1}^n |a_{ij}|. \quad (7.6.6)$$

2. *Вторая (октаэдрическая) норма:*

$$\|A\|_{II} = \max_j \sum_{i=1}^n |a_{ij}|. \quad (7.6.7)$$

3. *Третья (сферическая или евклидова) норма:*

$$\|A\|_{III} = \sum_{i=1}^n |a_{ij}|^2 = \sqrt{\mu_1}, \quad (7.6.8)$$

где μ_1 – наибольшее по модулю собственное значение матрицы A^*A , где A^* – матрица, сопряженная A .

Важным понятием матричной алгебры является понятие собственного значения матрицы. *Собственным значением* (или *характеристическим числом*) квадратной матрицы A называется такое число λ , что для некоторого ненулевого вектора x имеет место равенство:

$$Ax = \lambda x. \quad (7.6.9)$$

Любой ненулевой вектор x , удовлетворяющий этому равенству, называется *собственным вектором* матрицы A , соответствующим (или *принадлежащим*) собственному значению λ . Все

собственные векторы матриц определены с точностью до числового множителя.

Перепишем систему (7.6.9) в виде:

$$(A - \lambda E)x = 0, \quad (7.6.10)$$

где E – единичная матрица соответствующей размерности.

Условием существования для однородной системы ненулевого решения является равенство нулю ее определителя, т.е.

$$|A - \lambda E| = \begin{vmatrix} a_{11} - \lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - \lambda & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} - \lambda \end{vmatrix} = 0. \quad (7.6.11)$$

Это уравнение обычно называют *вековым* или *характеристическим уравнением* матрицы A .

Левая часть векового уравнения

$$|A - \lambda E| = (-1)^n (\lambda^n - p_1 \lambda^{n-1} - \dots - p_n) \quad (7.6.12)$$

называется *характеристическим многочленом* матрицы A . Многочлен $P(\lambda)$

$$P(\lambda) = \lambda^n - p_1 \lambda^{n-1} - \dots - p_n, \quad (7.6.13)$$

отличающийся от характеристического множителем $(-1)^n$, называется *собственным многочленом* матрицы A . Собственные значения матрицы являются корнями собственного многочлена. Совокупность всех собственных значений $\lambda_1, \lambda_2, \dots, \lambda_n$ матрицы A , где каждое собственное значение выписано столько раз, какова его кратность как корня собственного многочлена, называется *спектром* матрицы A . Собственными векторами x_i матрицы A являются нетривиальные решения однородной системы (7.6.10), в которой вместо λ подставлены собственные значения λ_i , $i = \overline{1, n}$.

Нахождение всех собственных значений и соответствующих собственных векторов матрицы называется решением *полной проблемы собственных значений*, нахождение части собственных значений (чаще всего это максимальное по модулю собственное значение) называется *частичной проблемой собственных значений*.

Полезно знать следующие соотношения между элементами матрицы A и ее собственными значениями:

$$\lambda_1 + \lambda_2 + \dots + \lambda_n = SpA = trA, \quad (7.6.14)$$

$$\lambda_1 \cdot \lambda_2 \cdot \dots \cdot \lambda_n = |A|, \quad (7.6.15)$$

где SpA (или trA) обозначает *след матрицы*, который равен сумме ее диагональных элементов. Из (7.6.15), в частности, следует, что матрица A имеет хотя бы одно собственное значение, равное нулю, если определитель этой матрицы равен нулю, т.е. если она *особенная*.

7.6.2 Обусловленность систем и матриц

Большое значение при решении задач матричной алгебры имеет понятие *обусловленности систем и матриц*.

Пусть δ – погрешность исходных данных, а ε – погрешность решения системы линейных алгебраических уравнений. Оценим зависимость погрешности решения от погрешности исходных данных неравенством:

$$\varepsilon \leq k \delta. \quad (7.6.16)$$

Число k характеризует зависимость максимально возможных ошибок решения от ошибок исходных данных задачи, и, зная его, можно разумно выбрать допустимую погрешность метода и необходимую точность проведения вычислений. Действительно, если исходные данные известны с погрешностью δ , то нецелесообразно искать решение задачи с точностью, намного превосходящей $k \delta$.

Если k «не очень велико», то будем говорить, что система *хорошо обусловлена*, если k «велико» – то система *плохо обусловлена*. Из (7.6.16) видно, что в плохо обусловленной задаче малым погрешностям исходных данных отвечают большие погрешности в решении.

Пусть система линейных алгебраических уравнений задана в виде:

$$Ax = b, \quad (7.6.17)$$

где A – невырожденная вещественная матрица.

Обозначим δA и δb соответственно матрицу и вектор малых погрешностей элементов матрицы A и вектора свободных членов

b , δx – вектор погрешностей решения. Тогда для относительной погрешности решения справедлива следующая оценка:

$$\frac{\|\delta x\|}{\|x\|} \leq \sqrt{\frac{\mu_1}{\mu_n}} \left\{ \frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right\}, \quad (7.6.18)$$

где μ_1 и μ_n – максимальное и минимальное собственные значения матрицы A^*A . Таким образом, обусловленность системы (7.6.17) зависит от величины отношения $\sqrt{\frac{\mu_1}{\mu_n}}$.

Замечание. Если матрица A симметрическая, то собственные значения матрицы A^*A равны квадратам собственных значений матрицы A , т.е.

$$\sqrt{\frac{\mu_1}{\mu_n}} = \frac{\lambda_1}{\lambda_n}, \quad (7.6.19)$$

где λ_1, λ_n – соответственно максимальное и минимальное собственные значения матрицы A .

Пусть δ – величина *невязки* (результат подстановки найденного решения в исходную систему), т.е.

$$\delta = Ax - b. \quad (7.6.20)$$

Если (7.6.20) переписать в виде:

$$Ax = b + \delta, \quad (7.6.21)$$

то, если система плохо обусловлена, малым δ может соответствовать большая погрешность в решении. Таким образом, *оценка точности решения системы по невязкам применима только к хорошо обусловленным системам*. В то же время проверка точности определения обратной матрицы не зависит от ее обусловленности. Действительно, если \tilde{A}^{-1} – приближенное значение обратной к A матрицы, тогда невязка решения задачи обращения матрицы δ равна:

$$\delta = A\tilde{A}^{-1} - E \quad (7.6.22)$$

и

$$\frac{\|\delta A^{-1}\|}{\|A^{-1}\|} \leq \delta, \quad (7.6.23)$$

где $\delta A^{-1} = \tilde{A}^{-1} - A^{-1}$ – погрешность вычисления обратной матрицы.

Из геометрической интерпретации решения системы линейных алгебраических уравнений, где рассматривается решение системы как точка пересечения гиперплоскостей $L_i = \{A_i x - b_i = 0, i = \overline{1, n}\}$, следует, что если

$$|\det A| \approx \prod_{i=1}^n \|A_i\|, \quad (7.6.24)$$

то система хорошо обусловлена, если

$$\frac{\prod_{i=1}^n \|A_i\|}{|\det A|} \gg 1, \quad (7.6.25)$$

то система плохо обусловлена. Здесь A_i – вектор, являющийся i -ой вектор-строкой матрицы A .

Для проверки обусловленности системы и матрицы A можно использовать *число обусловленности* матрицы A :

$$\nu = \|A\| \cdot \|A^{-1}\|. \quad (7.6.26)$$

Из (7.6.26) следует, что если матрица A близка к особенной, то число ν будет велико, и матрица A будет плохо обусловленной, если ν мало, то матрица A будет хорошо обусловленной. Как правило, *система с плохо обусловленной матрицей будет плохо обусловленной системой и наоборот*.

Значение числа ν зависит от выбора нормы матрицы A . Если A – вещественная матрица и используется третье определение нормы, то

$$\nu_{III} = \|A\|_{III} \cdot \|A^{-1}\|_{III} = \sqrt{\frac{\mu_1}{\mu_n}}. \quad (7.6.27)$$

Ввиду важности понятия обусловленности систем и матриц приведем численный пример.

Пример плохо обусловленной системы

Пусть дана система

$$Ax = b,$$

где

$$A = \begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 6 & 7 \\ 6 & 8 & 16 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix}, b = \begin{pmatrix} 23 \\ 32 \\ 39 \\ 31 \end{pmatrix}.$$

Точное решение системы: $x^* = (1 \ 1 \ 1 \ 1)^T$.

Точное значение обратной матрицы:

$$A^{-1} = \begin{pmatrix} 12.065 & -8.097 & -0.548 & 0.129 \\ -8.097 & 5.645 & 6.323 & -0.194 \\ -0.548 & 6.323 & 0.101 & -0.097 \\ 0.129 & -0.194 & -0.097 & 0.258 \end{pmatrix}.$$

Если внести погрешность $\delta_1 = 0.01$ только в один элемент a_{11} матрицы A , то значение обратной матрицы будет равно

$$A^{-1} + \delta A^{-1} = \begin{pmatrix} 10.766 & -7.225 & -0.489 & 0.115 \\ -7.225 & 5.06 & 0.283 & -0.184 \\ -0.489 & 0.283 & 0.159 & 0.096 \\ 0.115 & -0.184 & 0.096 & 0.258 \end{pmatrix}.$$

Решение системы также изменится и будет равно:

$$(A^{-1} + \delta A^{-1})b = (0.892 \ 1.072 \ 1.005 \ 0.225)^T.$$

Если внести погрешность $\delta_2 = 0.01$ в вектор свободных членов, т.е. $(b + \delta b) = (23 + \delta_2 \ 32 - \delta_2 \ 39 - \delta_2 \ 31 - \delta_2)^T$, то решение системы будет следующим:

$$A^{-1}(b + \delta b) = (1.206 \ 0.861 \ 0.991 \ 1.002)^T.$$

Собственные числа матрицы A равны: 5.468; 0.057; 3.08; 32.398, а отношение максимального собственного значения к минимальному есть 568.386. Число обусловленности матрицы A , вычисленное по третьей норме, приблизительно равно 580.

Таким образом, обусловленность системы и обусловленность задачи отыскания обратной матрицы тесно связаны, а именно: *из хорошей обусловленности системы следует хорошая обусловленность задачи отыскания обратной матрицы и, наоборот, из плохой обусловленности системы следует плохая обусловленность задачи отыскания обратной матрицы.*

Приведенные результаты получены в системе MathCAD, которая осуществляет все вычисления с 15 знаками после запятой, т.е. погрешность округления не оказала влияния на результат.

Если система является плохо обусловленной, то необходимо осуществлять ее решение специальными методами, например, методами регуляризации, которые предназначены для решения плохо обусловленных систем.

7.6.3 Метод Гаусса

Достаточно часто для решения систем с действительными элементами используется *метод Гаусса (метод исключения неизвестных)*. Этот метод осуществляет приведение исходной системы к эквивалентной системе с правой треугольной матрицей (*схема единственного деления*) или с диагональной матрицей (*схема оптимального исключения*). При этом не требуется заранее определять, имеет или нет решение данная система.

Рассмотрим схему единственного деления. Систему $Ax = b$ представим в виде:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2, \\ &\dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n. \end{aligned} \tag{7.6.28}$$

Будем считать выбор порядка преобразования, в котором исключаются неизвестные, произвольным. Выберем какое-либо уравнение и неизвестное в этом уравнении. Единственное условие, которое должно быть выполнено при этом выборе, состоит в том, что коэффициент при выбранном неизвестном должен быть отличным от нуля. Переставляя, если необходимо, уравнения и меняя местами неизвестные, можно считать, что выбрано первое уравнение, неизвестное x_1 , и при этом $a_{11} \neq 0$. Разделив выбранное уравнение на a_{11} , приведем его к виду:

$$x_1 + b_{12}x_2 + \dots + b_{1n}x_n = q_1, \tag{7.6.29}$$

где

$$b_{1j} = \frac{a_{1j}}{a_{11}}, \quad j = \overline{2, n}, \quad q_1 = \frac{b_1}{a_{11}}.$$

Исключим x_1 из остальных уравнений системы. Для этого умножим (7.6.29) последовательно на $a_{21}, a_{31}, \dots, a_{n1}$ и вычтем из второго, третьего и т.д. последнего уравнения системы. Преобразованные уравнения будут иметь вид:

$$a_{22.1}x_2 + a_{23.1}x_3 + \dots + a_{2n.1}x_n = b_{2.1},$$

$$\dots$$
(7.6.30)

$$a_{n2.1}x_2 + a_{n3.1}x_3 + \dots + a_{nn.1}x_n = b_{n.1},$$

где $a_{ij.1} = a_{ij} - b_{1j}a_{i1}$, $i, j = \overline{2, n}$, $b_{i.1} = b_i - a_{i1}q_1$.

К полученной системе применим такое же преобразование, т.е. выберем уравнение и неизвестное с коэффициентом, отличным от нуля, приведем этот коэффициент к единице, исключим неизвестное из прочих уравнений и так до тех пор, пока такие преобразования возможны.

В результате придем к одной из двух ситуаций.

1. После n шагов преобразований получим систему вида:

$$x_1 + b_{12}x_2 + b_{13}x_3 + \dots + b_{1n}x_n = q_1,$$

$$x_2 + b_{23}x_3 + \dots + b_{2n}x_n = q_2,$$

$$\dots$$
(7.6.31)

$$x_n = q_n.$$

Решение полученной системы осуществляется снизу вверх следующим образом:

$$x_n = q_n,$$

$$x_i = q_i - \sum_{j=n}^{i-1} b_{ij}x_j, \quad i = \overline{n-1, 1}.$$
(7.6.32)

2. После шага преобразований $m < n$ система приняла вид:

$$x_1 + b_{12}x_2 + b_{13}x_3 + \dots + b_{1n}x_n = q_1,$$

$$x_m + b_{m+1}x_{m+1} + \dots + b_{mn}x_n = q_m,$$

$$0 = b_{m+1.n},$$

$$\dots$$

$$0 = b_{m.n}.$$
(7.6.33)

Тогда, если среди элементов $b_{m+1,n}, \dots, b_{m,n}$ есть отличные от нуля, то система (7.6.28) не имеет решения, если все $b_{j,m} = 0$, $j = \overline{m+1, n}$, то система имеет бесчисленное множество решений (неизвестные x_{m+1}, \dots, x_n могут принимать любые значения, а x_1, \dots, x_m выражаются через них).

Приведение системы к виду (7.6.31) называется *прямым ходом метода Гаусса*, а нахождение ее решения (7.5.32) – *обратным*. Заметим, что на каждом шаге прямого хода метода Гаусса выбирается уравнение и неизвестное, подлежащие исключению из прочих уравнений. Это равносильно выбору коэффициента для очередного шага преобразований. Этот коэффициент называется *ведущим* и он должен быть отличным от нуля. Во избежание большой потери точности рекомендуется осуществлять такую перестановку уравнений, чтобы ведущий коэффициент являлся либо максимальным по модулю коэффициентом во всей системе, либо максимальным по модулю коэффициентом в выбранном уравнении. Такая процедура называется *методом Гаусса с выбором главного элемента*.

Метод Гаусса применим к вычислению определителей и обратных матриц. Так, значение определителя D матрицы A равно произведению ведущих коэффициентов:

$$D = a_{11}a_{22.1} \cdots a_{nn.n-1}, \quad (7.6.34)$$

а вычисление обратной матрицы осуществляется одновременным решением n систем:

$$Ax^{(k)} = e_k, \quad k = \overline{1, n}, \quad (7.6.35)$$

где e_k – единичный вектор (вектор, у которого все элементы, кроме k -го, равны нулю, а k -ый равен единице), $x^{(k)}$ – k -ый столбец матрицы A^{-1} . Невозможность вычисления обратной матрицы проявляется в невозможности решения какой-либо системы из (7.5.35).

7.6.4 Итерационные методы решения систем линейных алгебраических уравнений

Эти методы дают решение системы в виде предела последовательности некоторых векторов, построение которых осуществ-

ляется посредством единообразного процесса, называемого процессом итераций. В современной литературе описано большое количество итерационных методов, основанных на различных принципах. Как правило, вычислительные схемы таких методов просты и удобны для реализации на ЭВМ. Однако каждый итерационный процесс имеет свою ограниченную область применения, так как:

1) процесс итераций может оказаться расходящимся для данной системы;

2) сходимость процесса итераций для данной системы может быть настолько медленной, что практически невозможно достичь удовлетворительной близости к точному решению за приемлемое время.

Пусть дана система линейных алгебраических уравнений

$$Ax = f \quad (7.6.36)$$

с неособенной матрицей A . Точное решение системы x^* является пределом последовательности векторов $x^{(0)}, x^{(1)}, \dots, x^{(k)}, \dots$, которые строятся по рекуррентным формулам:

$$x^{(k)} = x^{(k-1)} + H^{(k)}(f - Ax^{(k-1)}) \quad (7.6.37)$$

или

$$x^{(k)} = C^{(k)}x^{(k-1)} + z^{(k)}, \quad (7.6.38)$$

где $H^{(k)}, C^{(k)}$ – последовательности матриц, $z^{(k)}$ – последовательности векторов, $k = 0, 1, \dots$. При этом для линейных систем в качестве начального приближения $x^{(0)}$ можно взять любой вектор соответствующей размерности.

Итерационные процессы, протекающие по формулам (7.6.37) и (7.6.38), обладают тем свойством, что для каждого из них точное решение x^* является неподвижной точкой. Это значит, что если за начальное приближение $x^{(0)}$ взято x^* , то все последующие приближения будут также равны x^* .

Простейшими среди итерационных процессов являются стационарные, в которых матрицы в (7.6.37) и (7.6.38) не зависят от номера шага k .

Метод последовательных приближений

Под процессом последовательных приближений понимается следующий итерационный процесс. Система уравнений (7.6.36) записывается в виде:

$$x = Bx + g,$$

где $B = E - A$, E – единичная матрица соответствующего порядка, $g = f$, и последовательные приближения вычисляются по формуле:

$$x^{(k+1)} = Bx^{(k)} + g. \quad (7.6.39)$$

Сходимость метода последовательных приближений определяется следующими теоремами.

Теорема. Для сходимости метода последовательных приближений (7.6.39) необходимо и достаточно, чтобы все собственные значения матрицы B были по модулю меньше единицы.

Теорема. Для сходимости метода последовательных приближений достаточно, чтобы какая-либо норма матрицы B была меньше единицы.

Таким образом, для сходимости метода последовательных приближений достаточно, чтобы выполнялось одно из условий:

$$\max_i \sum_{j=1}^n |b_{ij}| < 1;$$

$$\max_j \sum_{i=1}^n |b_{ij}| < 1;$$

$$\sum_{i,j=1}^n b_{ij}^2 < 1,$$

где b_{ij} , $i, j = \overline{1, n}$ – элементы матрицы B .

Метод простых итераций (метод Якоби)

Запишем систему (7.6.36) в развернутом виде:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= f_1, \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= f_2, \\ &\dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= f_n. \end{aligned} \quad (7.6.40)$$

Поделив каждое уравнение этой системы на соответствующий диагональный элемент, получим систему:

$$\begin{aligned} x_1 + \frac{a_{12}}{a_{11}} x_2 + \dots + \frac{a_{1n}}{a_{11}} x_n &= \frac{f_1}{a_{11}}, \\ \frac{a_{21}}{a_{22}} x_1 + x_2 + \dots + \frac{a_{2n}}{a_{22}} x_n &= \frac{f_2}{a_{22}}, \\ &\dots \\ \frac{a_{n1}}{a_{nn}} x_1 + \frac{a_{n2}}{a_{nn}} x_2 + \dots + x_n &= \frac{f_n}{a_{nn}}, \end{aligned} \quad (7.6.41)$$

или в матричной форме:

$$x = Bx + g, \quad (7.6.42)$$

где

$$B = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \dots & -\frac{a_{2n}}{a_{22}} \\ & & \dots & \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \dots & 0 \end{pmatrix}, \quad g = \begin{pmatrix} \frac{f_1}{a_{11}} \\ \frac{f_2}{a_{22}} \\ \dots \\ \frac{f_n}{a_{nn}} \end{pmatrix}.$$

Преобразование исходной системы (7.6.40) в систему (7.6.42) равносильно матричному преобразованию:

$$B = E - D^{-1}A, \quad g = D^{-1}f, \quad (7.6.43)$$

где D – диагональная матрица, на диагонали которой находятся элементы $a_{11}, a_{22}, \dots, a_{nn}$ матрицы A и

$$D^{-1} = \begin{pmatrix} \frac{1}{a_{11}} & 0 & \dots & 0 \\ 0 & \frac{1}{a_{22}} & \dots & 0 \\ & & \dots & \\ 0 & 0 & \dots & \frac{1}{a_{nn}} \end{pmatrix}.$$

На основании приведенных выше теорем для сходимости метода простых итераций необходимо и достаточно, чтобы все собственные значения матрицы $B = E - D^{-1}A$ были по модулю меньше единицы, или, что то же самое, чтобы корни уравнения

$$|A - D + tD| = \begin{vmatrix} a_{11}t & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22}t & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn}t \end{vmatrix} = 0 \quad (7.6.44)$$

были по модулю меньше единицы.

Достаточным условием сходимости является выполнение одного из условий:

$$\begin{aligned} \max_i \sum_{j=1, j \neq i}^n \left| \frac{a_{ij}}{a_{ii}} \right| &< 1; \\ \max_j \sum_{i=1, i \neq j}^n \left| \frac{a_{ij}}{a_{ii}} \right| &< 1; \\ \sum_{i \neq j=1}^n \left(\frac{a_{ij}}{a_{ii}} \right)^2 &< 1. \end{aligned}$$

Метод Зейделя

Пусть система (7.6.36) представлена в виде:

$$x = Bx + g, \quad (7.6.45)$$

где $B = E - A$, $g = f$.

Одношаговый циклический процесс, называемый методом Зейделя, отличается от процесса последовательных приближений тем, что при вычислении k -го приближения для i -ой компоненты учитываются вычисленные ранее k -е приближения для компонент $x_1^{(k)}, \dots, x_{i-1}^{(k)}$. Таким образом, вычисления ведутся по формулам:

$$x_i^{(k)} = \sum_{j=1}^{i-1} b_{ij} x_j^{(k)} + \sum_{j=i}^n b_{ij} x_j^{(k-1)} + g_i, \quad i = \overline{1, n}. \quad (7.6.46)$$

Представим систему (7.6.45) в матричном виде:

$$x = (M + N)x + g, \quad (7.6.47)$$

где

$$M = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ b_{21} & 0 & \dots & 0 & 0 \\ & & \dots & & \\ b_{n1} & b_{n2} & \dots & b_{nn-1} & 0 \end{pmatrix}, \quad N = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n-1} & b_{1n} \\ 0 & b_{22} & \dots & b_{2n-1} & b_{2n} \\ & & \dots & & \\ 0 & 0 & \dots & 0 & b_{nn} \end{pmatrix}.$$

Тогда итерационное правило (7.6.46) можно представить в матричной форме в виде:

$$x^{(k)} = Mx^{(k)} + Nx^{(k-1)} + g \quad (7.6.48)$$

или

$$x^{(k)} = (E - M)^{-1} Nx^{(k-1)} + (E - M)^{-1} g. \quad (7.6.49)$$

Для сходимости метода Зейделя необходимо и достаточно, чтобы все собственные значения матрицы $(E - M)^{-1} N$ были по модулю меньше единицы или, что то же самое, чтобы корни уравнения

$$|N - (E - M)t| = \begin{vmatrix} b_{11} - t & b_{12} & \dots & b_{1n-1} & b_{1n} \\ b_{21}t & b_{22} - t & \dots & b_{2n-1} & b_{2n} \\ & & \dots & & \\ b_{n1}t & b_{n2}t & \dots & b_{nn-1}t & b_{nn} - t \end{vmatrix} = 0$$

были по модулю меньше единицы. Достаточное условие сходимости состоит в том, чтобы правая или вторая норма матрицы B были меньше единицы.

7.6.5 Метод Данилевского при решении полной проблемы собственных значений. Построение канонической формы Фробениуса

Метод Данилевского решения полной проблемы собственных значений основан на том, что преобразование подобия не изменяет характеристического многочлена матрицы. Целью преобразования подобия является приведение исходной матрицы A к канонической форме Фробениуса:

$$\Phi = \begin{pmatrix} 0 & 0 & \dots & 0 & p_n \\ 1 & 0 & \dots & 0 & p_{n-1} \\ 0 & 1 & \dots & 0 & p_{n-2} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & p_1 \end{pmatrix}, \quad (7.6.50)$$

по виду которой можно достаточно просто построить ее собственный многочлен:

$$P(\lambda) = \lambda^n - p_1 \lambda^{n-1} - \dots - p_n, \quad (7.6.51)$$

так как коэффициенты собственного многочлена являются элементами последнего столбца матрицы Фробениуса.

Для получения матрицы Фробениуса строится последовательность матриц $A^{(1)}, A^{(2)}, \dots, A^{(n-1)}$ по следующему правилу:

$$A^{(i)} = S_i^{-1} A^{(i-1)} S_i, \quad i = \overline{1, n-1}, \quad (7.6.52)$$

где каждая последующая матрица $A^{(i)}$ получается из предыдущей $A^{(i-1)}$ преобразованием подобия с помощью матрицы S_i :

$$S_i = \begin{pmatrix} 0 & 0 & \dots & 0 & a_{1n}^{(i-1)} \\ 1 & 0 & \dots & 0 & a_{2n}^{(i-1)} \\ 0 & 1 & \dots & 0 & a_{3n}^{(i-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & a_{nn}^{(i-1)} \end{pmatrix}. \quad (7.6.53)$$

Обратная к S_i матрица имеет вид:

$$S_i^{-1} = \begin{pmatrix} -\frac{a_{2n}^{(i-1)}}{a_{1n}^{(i-1)}} & 1 & 0 & \dots & 0 \\ -\frac{a_{3n}^{(i-1)}}{a_{1n}^{(i-1)}} & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -\frac{a_{nn}^{(i-1)}}{a_{1n}^{(i-1)}} & 0 & 0 & \dots & 1 \\ \frac{1}{a_{1n}^{(i-1)}} & 0 & 0 & \dots & 0 \end{pmatrix}, \quad (7.6.54)$$

где $a_{jn}^{(i-1)}$, $j = \overline{1, n}$ – элементы последнего столбца матрицы $A^{(i-1)}$ и $A^{(0)} = A$. Тогда, если все элементы $a_{1n}^{(j)} \neq 0$, $j = \overline{0, n-2}$ (регулярный случай), то матрица

$$A^{(n-1)} = \begin{pmatrix} 0 & 0 & \dots & 0 & a_{1n}^{(n-1)} \\ 1 & 0 & \dots & 0 & a_{2n}^{(n-1)} \\ 0 & 1 & \dots & 0 & a_{3n}^{(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & a_{nn}^{(n-1)} \end{pmatrix} \quad (7.6.55)$$

будет иметь форму Фробениуса.

Если при построении матрицы $A^{(j+1)}$ элемент $a_{1n}^{(j)} = 0$, то имеем *нерегулярный случай*, так как $A^{(j)}$ имеет вид:

$$A^{(j)} = \left(\begin{array}{cccc|cccc} a_{1,1}^{(j)} & a_{1,2}^{(j)} & \dots & a_{1,n-j-1}^{(j)} & 0 & \dots & 0 & a_{1,n}^{(j)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n-j-1,1}^{(j)} & a_{n-j-1,2}^{(j)} & \dots & a_{n-j-1,n-j-1}^{(j)} & 0 & \dots & 0 & a_{n-j-1,n}^{(j)} \\ \hline a_{n-j,1}^{(j)} & a_{n-j,2}^{(j)} & \dots & a_{n-j,n-j-1}^{(j)} & 0 & \dots & 0 & a_{n-j,n}^{(j)} \\ a_{n-j+1,1}^{(j)} & a_{n-j+1,2}^{(j)} & \dots & a_{n-j+1,n-j-1}^{(j)} & 1 & \dots & 0 & a_{n-j+1,n}^{(j)} \\ a_{n-j+2,1}^{(j)} & a_{n-j+2,2}^{(j)} & \dots & a_{n-j+2,n-j-1}^{(j)} & 0 & \dots & 0 & a_{n-j+2,n}^{(j)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n,1}^{(j)} & a_{n,2}^{(j)} & \dots & a_{n,n-j-1}^{(j)} & 0 & \dots & 1 & a_{n,n}^{(j)} \end{array} \right). \quad (7.6.56)$$

В этом случае возможны две ситуации.

1. Среди элементов $a_{2n}^{(j)}, a_{3n}^{(j)}, \dots, a_{n-jn}^{(j)}$ имеется хотя бы один, отличный от нуля, например, $a_{kn}^{(j)} \neq 0$, $k = \overline{2, n-j}$. Тогда можно прийти к регулярному случаю, если в матрице $A^{(j)}$ поменять местами первую и k -ую строки и столбцы. Такое преобразование матрицы $A^{(j)}$ будет подобным, и, кроме того, столбцы, уже имеющие нужный вид не будут испорчены.

2. Все элементы $a_{kn}^{(j)} = 0$, $k = \overline{2, n-j}$, т.е. матрица $A^{(j)}$ имеет блочный вид:

$$A^{(j)} = \left(\begin{array}{c|c} B & 0 \\ \hline C & \Phi_{j+1} \end{array} \right), \quad (7.6.57)$$

где Φ_{j+1} – квадратная матрица порядка $j+1$, имеющая каноническую форму Фробениуса, B – квадратная матрица порядка $n-j-1$, 0 – нулевая матрица. Тогда, на основании теоремы Лапласа, можно записать:

$$|A^{(j)} - \lambda E| = |B - \lambda E_{n-j-1}| \cdot |\Phi_{j+1} - \lambda E_{j+1}|,$$

т.е. характеристический многочлен матрицы Φ_{j+1} является делителем характеристического многочлена матрицы A . Для отыскания характеристического многочлена матрицы A нужно найти характеристический многочлен матрицы B , что можно сделать путем приведения B к форме Фробениуса.

Таким образом, построив собственный многочлен матрицы A , можно найти собственные значения $\lambda_1, \lambda_2, \dots, \lambda_n$ этой матрицы, которые являются корнями ее собственного многочлена.

Если удалось привести A к виду (7.6.55), то собственный вектор $x^{(i)}$ матрицы A будет равен:

$$x^{(i)} = S y^{(i)}, \quad (7.6.58)$$

где $S = S_1 S_2 \dots S_{n-1}$ – произведение матриц преобразования, а $y^{(i)} = (y_1^{(i)}, y_2^{(i)}, \dots, y_n^{(i)})^T$ – собственный вектор матрицы (7.6.55), соответствующий собственному значению λ_i , компоненты которого $y_1^{(i)}, y_2^{(i)}, \dots, y_n^{(i)}$ определяются следующим образом:

$$\begin{aligned} y_n^{(i)} &= 1, \\ y_{n-1}^{(i)} &= \lambda_i - p_1, \\ y_{n-2}^{(i)} &= \lambda_i^2 - p_1 \lambda_i - p_2, \\ &\dots \\ y_1^{(i)} &= \lambda_i^{n-1} - p_1 \lambda_i^{n-2} - \dots - p_{n-1}, \end{aligned} \quad (7.6.59)$$

где $p_j = a_{n-j+1, n}^{(n-1)}$, $j = \overline{1, n}$.

Если не удалось привести матрицу A к виду (7.6.55), то для отыскания собственных векторов необходимо решить системы линейных алгебраических уравнений:

$$Ax^{(i)} = \lambda_i x^{(i)}, \quad i = \overline{1, n}, \quad (7.6.60)$$

где $x^{(i)} = (x_1^{(i)}, \dots, x_n^{(i)})^T$. Так как системы (7.6.58) решаются с точностью до постоянного множителя, то одну из компонент вектора $x^{(i)}$ можно приравнять некоторому числу.

7.7 Решение обыкновенных дифференциальных уравнений и систем

Среди задач, с которыми приходится иметь дело при моделировании, значительную часть составляют различные задачи для обыкновенных дифференциальных уравнений. Такие задачи возникают как непосредственно при моделировании многих реальных процессов, так и в качестве промежуточных при решении более сложных математических задач. При этом, как правило, точное решение рассматриваемой задачи не удастся выразить через элементарные функции. Доля задач, решаемых в явном виде в случае обыкновенных дифференциальных уравнений, ничтожно мала. Обычно приходится прибегать к помощи приближенных методов решения подобных задач. Такие методы в зависимости от того, ищется ли приближенное решение в аналитическом виде или в виде таблицы значений, разделяют на *аналитические* и *численные*. Конкретный вид метода существенно зависит от типа решаемой задачи. В случае обыкновенных дифференциальных уравнений в зависимости от того, ставятся ли дополнительные условия в одной или в нескольких точках отрезка изменения независимой переменной, задачи обычно подразделяют на *одноточечные* (задачи с начальными условиями, или задачи Коши) и *многоточечные*. Среди многоточечных задач наиболее часто в прикладных вопросах встречаются так называемые *граничные задачи*, когда дополнительные условия ставятся на концах рассматриваемого отрезка. При этом нередко решение многоточечных задач и, в частности, решение граничных задач сводят к решению задач с начальными условиями.

7.7.1 Методы Эйлера и Рунге-Кутты при решении задачи Коши для дифференциального уравнения первого порядка

Задача Коши для дифференциального уравнения первого порядка

$$y'(x) = f(x, y(x)) \quad (7.7.1)$$

состоит в нахождении функции $y(x)$, удовлетворяющей этому уравнению при начальном условии $y(x_0) = y_0$. При численном решении этой задачи обычно задают некоторый шаг h и находят решение в точках x_1, x_2, \dots, x_n , где $x_i = x_0 + ih$, $i = \overline{0, n}$, $h = \frac{x_0 - x_n}{n}$.

Обозначим разность между точными значениями функции $y(x)$ в двух рядом стоящих точках через

$$\Delta y(x_0) = y(x_0 + h) - y(x_0) \quad (7.7.2)$$

и представим приближенно эту же разность в виде некоторой линейной комбинации:

$$\Delta y_0 = p_1 K_1(x_0, y_0) + p_2 K_2(x_0, y_0) + \dots + p_r K_r(x_0, y_0), \quad (7.7.3)$$

где

$$\begin{aligned} K_1(x_0, y_0) &= h f(x_0, y_0), \\ K_2(x_0, y_0) &= h f(x_0 + \alpha_2 h, y_0 + \beta_{2,1} K_1(x_0, y_0)), \\ &\dots \\ K_r(x_0, y_0) &= h f(x_0 + \alpha_r h, y_0 + \beta_{r,1} K_1(x_0, y_0) + \dots \\ &\quad + \beta_{r,r-1} K_{r-1}(x_0, y_0)). \end{aligned} \quad (7.7.4)$$

Коэффициенты $p_j, \alpha_j, \beta_{ji}$, $j = \overline{1, r}$, $i = \overline{1, r-1}$ будем задавать таким образом, чтобы разложение $\Delta y(x_0)$ в ряд Тейлора по степеням h и линейная комбинация Δy_0 совпадали до возможно более высоких степеней h при произвольной функции $f(x, y)$ и произвольном шаге h .

Введем функцию:

$$\varphi_r(h) = y(x_0 + h) - y(x_0) - [p_1 K(x_0, y_0) + \dots + p_r K_r(x_0, y_0)]. \quad (7.7.5)$$

Разложение функции $\varphi_r(h)$ в ряд Тейлора в окрестности $h = 0$ будет иметь вид:

$$\varphi_r(h) = \varphi_r(0) + h \frac{\varphi_r'(0)}{1!} + h^2 \frac{\varphi_r''(0)}{2!} + \dots + h^{m+1} \frac{\varphi_r^{m+1}(0)}{(m+1)!} + O(h^{m+1}), \quad (7.7.6)$$

где $O(h^{m+1})$ – величина, имеющая более высокий порядок малости, чем h^{m+1} . Коэффициенты $p_j, \alpha_j, \beta_{ji}$, $j = \overline{1, r}$, $i = \overline{1, r-1}$ задают таким образом, чтобы функция $\varphi_r(h)$ обладала следующими свойствами:

$$\varphi_r(0) = \varphi_r'(0) = \dots = \varphi_r^{(s)}(0) = 0, \quad \varphi_r^{(s+1)}(0) \neq 0 \quad (7.7.7)$$

при возможно больших s для любых h и $f(x, y)$.

Формула Эйлера

Если в (7.7.3) положить $r = 1$, то получим:

$$\Delta y_0 = p_1 K_1(x_0, y_0). \quad (7.7.8)$$

Тогда

$$\begin{aligned} \varphi_r(h) &= y(x_0 + h) - y(x_0) - p_1 h f(x_0, y_0), \\ \varphi_r'(h) &= y'(x_0 + h) - p_1 f(x_0, y_0), \\ \varphi_r''(h) &= y''(x_0 + h), \end{aligned} \quad (7.7.9)$$

и при $h = 0$

$$\begin{aligned} \varphi_r(0) &= 0, \\ \varphi_r'(0) &= y'(x_0) - p_1 f(x_0, y_0), \\ \varphi_r''(0) &= y''(x_0). \end{aligned}$$

При этом $\varphi_1'(0) = 0$ тогда и только тогда, когда $p_1 = 1$, а $\varphi_1''(0) = y''(x_0)$, вообще говоря, в нуль не обращается. Тогда приближенное решение в точке x_1 согласно (7.7.8) будет определяться следующим образом:

$$y_1 = y_0 + h f(x_0, y_0).$$

Формула Эйлера для значения функции решения в произвольной точке будет иметь вид:

$$y_{i+1} = y_i + h f(x_i, y_i), \quad i = \overline{0, n}. \quad (7.7.10)$$

Формула Рунге-Кутты

Существует несколько формул Рунге-Кутты для решения дифференциального уравнения первого порядка, которые полу-

чены при различных значениях $r > 1$. Наиболее распространенной является формула Рунге-Кутты при $r = 4$, которая имеет следующий вид:

$$y_{i+1} = y_i + \frac{1}{6}(K_1^{(i)} + 2K_2^{(i)} + 2K_3^{(i)} + K_4^{(i)}), \quad (7.7.11)$$

где

$$\begin{aligned} K_1^{(i)} &= h f(x_i, y_i), \\ K_2^{(i)} &= h f\left(x_i + \frac{h}{2}, y_i + \frac{K_1^{(i)}}{2}\right), \\ K_3^{(i)} &= h f\left(x_i + \frac{h}{2}, y_i + \frac{K_2^{(i)}}{2}\right), \\ K_4^{(i)} &= h f(x_i + h, y_i + K_3^{(i)}) \end{aligned} \quad (7.7.12)$$

и $K_j^{(i)} = K_j(x_i, y_i)$, $j = \overline{1, 4}$.

Методы Эйлера и Рунге-Кутты можно использовать также и для моделирования в неравноотстоящих точках.

7.7.2 Правило Рунге

Функция $\varphi_r(h)$ в (7.7.5) имеет следующий смысл: это разность между точным и приближенным приращениями решения дифференциального уравнения в точке x_1 . Тогда, учитывая (7.7.6) и (7.7.7), можно записать уравнение погрешности функции решения в одной точке следующим образом:

$$\varphi_r(h) = h^{s+1} \frac{\varphi_r^{(s+1)}(0)}{(s+1)!} + O(h^{s+1}), \quad (7.7.13)$$

где $h^{s+1} \frac{\varphi_r^{(s+1)}(0)}{(s+1)!}$ — главный член погрешности. Число шагов, ко-

торые необходимо сделать до произвольной точки x , обратно пропорционально шагу h , т.е. $N = O(1/h)$ и погрешность в произвольной точке x будет иметь смысл произведения погрешности на шаге $\varphi_r(h)$ на число шагов N , т.е. $\varphi_r(h)N$.

Если главный член погрешности можно записать в виде $K(x)h^m$, то говорят, что *порядок погрешности метода равен m* .

На практике в качестве погрешности приближенного решения берут главный член погрешности:

$$R_r(x) = K(x)h^m. \quad (7.7.14)$$

Рассмотренные методы имеют *порядок погрешности на шаге*, равный $s+1$, а *порядок погрешности метода* — s . Для метода Эйлера $s=1$, а для метода Рунге-Кутты $s=4$. Таким образом, при моделировании методом Эйлера результаты получаются с погрешностью порядка h , а методом Рунге-Кутты с погрешностью порядка h^4 .

Оценка погрешности методов Эйлера и Рунге-Кутты затруднительна. Грубую оценку погрешности можно получить, используя *правило Рунге*.

Пусть $y(x_i)$ — точное решение в точке x_i , $y_i^{(h)}$ — приближенное решение в точке x_i , найденное с шагом h , $y_i^{(2h)}$ — приближенное значение в точке x_i , найденное с шагом $2h$, и пусть погрешность приближенного решения определяется формулой:

$$R_r(x_i) = K(x_i)h^s.$$

Тогда справедливы следующие равенства:

$$y(x_i) = y_i^{(h)} + K(x_i)h^s,$$

$$y(x_i) = y_i^{(2h)} + K(x_i)(2h)^s.$$

Приравнивая правые части этих равенств, получим:

$$K(x_i) = \frac{y_i^{(h)} - y_i^{(2h)}}{(2h)^s - h^s} = \frac{y_i^{(h)} - y_i^{(2h)}}{h^s(2^s - 1)}$$

и

$$R_r(x_i) = \frac{y_i^{(h)} - y_i^{(2h)}}{2^s - 1}. \quad (7.7.15)$$

Таким образом, грубую оценку погрешности на интервале $[x_0, x_n]$ можно получить с помощью двойного счета по формулам Эйлера и Рунге-Кутты и в качестве оценки погрешностей использовать величины:

- для метода Эйлера

$$R_1(x) \approx \max_{i=1,n} |y_i^{(h)} - y_i^{(2h)}|, \quad (7.7.16)$$

- для метода Рунге-Кутты

$$R_4(x) \approx \max_{i=1,n} \left| \frac{y_i^{(h)} - y_i^{(2h)}}{15} \right|. \quad (7.7.17)$$

Кроме того, для контроля правильности выбора шага h рекомендуется вычислять величину:

$$\theta = \max_{i=1,n} \left| \frac{K_2^{(i)} - K_3^{(i)}}{K_1^{(i)} - K_2^{(i)}} \right|. \quad (7.7.18)$$

7.7.3 Методы Эйлера и Рунге-Кутта при решении задачи Коши для систем дифференциальных уравнений

Ограничимся для сокращения записи системой двух уравнений.

Задача Коши для решения системы из двух уравнений:

$$\begin{aligned} y_1'(x) &= f_1(x, y_1(x), y_2(x)), \\ y_2'(x) &= f_2(x, y_1(x), y_2(x)), \end{aligned} \quad (7.7.19)$$

состоит в том, что требуется найти на интервале $[x_0, x_n]$ решение этой системы, удовлетворяющее условиям:

$$y_1(x_0) = y_1^{(0)}, \quad y_2(x_0) = y_2^{(0)}. \quad (7.7.20)$$

Формулы Эйлера решения задачи Коши для системы (7.7.19) имеют вид:

$$\begin{aligned} y_1^{(i+1)} &= y_1^{(i)} + h f_1(x_i, y_1^{(i)}, y_2^{(i)}), \\ y_2^{(i+1)} &= y_2^{(i)} + h f_2(x_i, y_1^{(i)}, y_2^{(i)}), \end{aligned} \quad (7.7.21)$$

где $h = x_{i+1} - x_i$, $y_1^{(i)}$, $y_2^{(i)}$ – приближенные решения в точке x_i , $i = \overline{0, n}$.

Формулы Рунге-Кутта решения задачи Коши для системы (7.7.19) имеют вид:

$$\begin{aligned} y_1^{(i+1)} &= y_1^{(i)} + \frac{1}{6} [K_1^{(i)} + 2K_2^{(i)} + 2K_3^{(i)} + K_4^{(i)}], \\ y_2^{(i+1)} &= y_2^{(i)} + \frac{1}{6} [L_1^{(i)} + 2L_2^{(i)} + 2L_3^{(i)} + L_4^{(i)}], \end{aligned} \quad (7.7.22)$$

где

$$K_1^{(i)} = h f_1(x_i, y_1^{(i)}, y_2^{(i)}),$$

$$\begin{aligned}
L_1^{(i)} &= h f_2(x_i, y_1^{(i)}, y_2^{(i)}), \\
K_2^{(i)} &= h f_1(x_i + \frac{h}{2}, y_1^{(i)} + \frac{K_1^{(i)}}{2}, y_2^{(i)} + \frac{L_1^{(i)}}{2}), \\
L_2^{(i)} &= h f_2(x_i + \frac{h}{2}, y_1^{(i)} + \frac{K_1^{(i)}}{2}, y_2^{(i)} + \frac{L_1^{(i)}}{2}), \\
K_3^{(i)} &= h f_1(x_i + \frac{h}{2}, y_1^{(i)} + \frac{K_2^{(i)}}{2}, y_2^{(i)} + \frac{L_2^{(i)}}{2}), \\
L_3^{(i)} &= h f_2(x_i + \frac{h}{2}, y_1^{(i)} + \frac{K_2^{(i)}}{2}, y_2^{(i)} + \frac{L_2^{(i)}}{2}), \\
K_4^{(i)} &= h f_1(x_i + h, y_1^{(i)} + K_3^{(i)}, y_2^{(i)} + L_3^{(i)}), \\
L_4^{(i)} &= h f_2(x_i + h, y_1^{(i)} + K_3^{(i)}, y_2^{(i)} + L_3^{(i)}).
\end{aligned} \tag{7.7.23}$$

7.7.4 Решение задачи Коши для систем линейных дифференциальных уравнений

Задача Коши для системы линейных дифференциальных уравнений, записанной в матричной форме, имеет вид:

$$y'(t) = \bar{A}(t)y(t) + \bar{b}(t), \quad y(t_0) = y^{(0)}, \tag{7.7.24}$$

где $y(t) = (y_1(t), y_2(t), \dots, y_n(t))^T$, $y^{(0)} = (y_1^{(0)}, y_2^{(0)}, \dots, y_n^{(0)})^T$, $\bar{A}(t)$ – матрица размерности $n \times n$, $\bar{b}(t)$ – вектор с n элементами.

Системы такого вида обычно используются в теории автоматического управления для моделирования поведения объекта в пространстве состояний.

В силу линейности системы (7.7.24) методы Эйлера и Рунге-Кутта значительно упрощаются и принимают вид разностных уравнений:

$$y(k+1) = A(k)y(k) + b(k), \quad y(0) = y^{(0)}. \tag{7.7.25}$$

При этом для формулы Эйлера:

$$A(k) = E_n + \Delta t \bar{A}(t_k), \quad b(k) = \Delta t \bar{b}(t_k), \tag{7.7.26}$$

а для формулы Рунге-Кутта:

$$A(k) = E_n + \sum_{i=1}^4 \frac{\Delta t^i}{i!} \bar{A}(t_k), \quad b(k) = \sum_{i=1}^4 \frac{\Delta t^i}{(i-1)!} \bar{A}^{i-1}(t_k) \bar{b}(t_k). \tag{7.7.27}$$

В (7.7.26), (7.7.27) k соответствует моменту времени $t_k = t_0 + k \Delta t$, где $\Delta t = t_{k+1} - t_k$, $k = \overline{0, n}$; E_n – единичная матрица порядка n .

Замечание.

Для того, чтобы воспользоваться методами Рунге-Кутты и Эйлера для решения дифференциальных уравнений и систем, содержащих высшие производные относительно искомых функций, обычно, путем введения новых переменных, переходят к системе обыкновенных дифференциальных уравнений.

7.7.5 Методы Адамса при решении задачи Коши для обыкновенных дифференциальных уравнений

Методы Эйлера и Рунге-Кутты являются одношаговыми, так как при вычислении решения в точке $x_i + h$ они используют информацию о значении функции решения только в одной точке x_i . Существуют методы, которые позволяют часть полученной информации использовать повторно на нескольких шагах вычислительного процесса. Это так называемые многошаговые методы. При этом иногда оказывается целесообразным привлекать также и информацию с «забеганием» вперед.

Рассмотрим решение задачи Коши для дифференциального уравнения первого порядка (7.7.1).

Пусть:

- 1) узлы являются равноотстоящими $x_{i+1} = x_i + h$, $i = \overline{0, n}$;
- 2) для нахождения решения в точке x_{i+1} требуется знание приближенного решения в $(k+1)$ предыдущих точках $x_i, x_{i-1}, \dots, x_{i-k}$.

Точки x_0, x_1, \dots, x_k образуют *начальный отрезок*. Решение на начальном отрезке может быть найдено другим методом, например, методом Рунге-Кутты.

Пусть известно решение в точках x_0, x_1, \dots, x_k и требуется найти решение в точке x_{k+1} . Проинтегрируем исходное уравнение $y' = f(x, y)$ на отрезке $[x_k, x_{k+1}]$:

$$\int_{x_k}^{x_{k+1}} y' dx = \int_{x_k}^{x_{k+1}} f(x, y) dx.$$

Тогда

$$y(x_{k+1}) = y(x_k) + \int_{x_k}^{x_{k+1}} f(x, y) dx. \quad (7.7.28)$$

Для подынтегральной функции $f(x, y)$ построим интерполяционный многочлен $P_n(x)$ и подставим его в (7.7.28). Так как значение интерполяционного многочлена вычисляется на отрезке $[x_k, x_{k+1}]$, то в качестве такого многочлена выберем формулу Ньютона для интерполирования назад, и, в зависимости от того, какую точку привлекают первой при построении интерполяционного многочлена, получаются различные формулы для решения дифференциального уравнения.

Если вместо функции $f(x, y)$ в (7.7.28) подставить $P_n(x)$, точные значения $y(x_i)$ заменить на приближенные y_i и проинтегрировать, то получим конкретные формулы Адамса. Наиболее распространенными являются формулы для $n = 4$.

Экстраполяционная формула Адамса

Если для построения интерполяционного многочлена $P_4(x)$ используются значения функции $f(x, y)$ в узлах $x_k, x_{k-1}, \dots, x_{k-4}$, то

$$P_4(x) = f(x_k, y(x_k)) + \frac{t}{1!} \Delta f(x_{k-1}, y(x_{k-1})) + \frac{t(t+1)}{2!} \Delta^2 f(x_{k-2}, y(x_{k-2})) + \dots + \frac{t(t+1)\dots(t+3)}{4!} \Delta^4 f(x_{k-4}, y(x_{k-4})).$$

Тогда для вычисления значения решения дифференциального уравнения (7.7.1) в точке x_{k+1} получим *экстраполяционную формулу Адамса*, которая имеет вид:

$$y_{k+1} = y_k + \frac{h}{24} (55f_k - 59f_{k-1} + 37f_{k-2} - f_{k-3}). \quad (7.7.29)$$

Здесь $f_j = f(x_j, y_j)$, $j = \overline{k, k-3}$.

Интерполяционная формула Адамса

Если для построения интерполяционного многочлена $P_4(x)$ используются значения функции $f(x, y)$ в узлах $x_{k+1}, x_k, \dots, x_{k-3}$, то

$$P_4(x) = f(x_{k+1}, y(x_{k+1})) + \frac{t}{1!} \Delta f(x_k, y(x_k)) + \frac{t(t+1)}{2!} \Delta^2 f(x_{k-1}, y(x_{k-1})) + \dots + \frac{t(t+1)\dots(t+3)}{4!} \Delta^4 f(x_{k-3}, y(x_{k-3}))$$

и для вычисления значения решения дифференциального уравнения (7.7.1) в точке x_{k+1} получим *интерполяционную формулу Адамса*, которая имеет вид:

$$y_{k+1} = y_k + \frac{h}{24} (9f_{k+1} + 19f_k - 5f_{k-1} + f_{k-2}). \quad (7.7.30)$$

Заметим, что интерполяционная формула (7.7.30) представляет собой уравнение относительно y_{k+1} . Одним из методов решения этого уравнения является метод последовательных приближений:

$$y_{k+1}^{(i+1)} = y_k^{(i)} + \frac{h}{24} (9f_{k+1}^{(i)} + 19f_k - 5f_{k-1} + f_{k-2}), \quad (7.7.31)$$

где $f_{k+1}^{(i)} = f(x_{k+1}, y_{k+1}^{(i)})$, начальное значение $y_{k+1}^{(0)}$ задается, например, с помощью экстраполяционной формулы Адамса, $i = 0, 1, 2, \dots$.

Методы Адамса дают результат с погрешностью порядка h^4 . Практическая оценка погрешности приближенного решения может быть получена с помощью правила Рунге.

7.7.6 Метод сеток решения линейных краевых задач

Наряду с задачами Коши для обыкновенных дифференциальных уравнений достаточно часто рассматриваются также *граничные задачи*. В этих задачах дополнительные условия, которым должно удовлетворять решение, задаются в виде уравнений, содержащих комбинации значений решения и его производных, взятых в нескольких точках рассматриваемого отрезка $[a, b]$. Простейшими среди граничных задач являются *краевые задачи*, когда дополнительные условия задаются только на концах отрез-

ка $[a, b]$. Если дифференциальное уравнение и уравнения, определяющие дополнительные условия, являются линейными, то граничная задача называется *линейной*.

Пусть на отрезке $[a, b]$ задана линейная краевая задача для дифференциального уравнения

$$L(y) = y'' + p(x)y' + q(x)y = f(x) \quad (7.7.32)$$

с условиями

$$\begin{aligned} l_a(y) &= \alpha_0 y(a) + \alpha_1 y(b) = A, \\ l_b(y) &= \beta_0 y(a) + \beta_1 y(b) = B, \end{aligned} \quad (7.7.33)$$

где $L(y)$, $l_a(y)$, $l_b(y)$ линейные операторы.

Метод сеток для решения краевой задачи (7.7.32), (7.7.33) состоит в следующем.

1. На отрезке $[a, b]$ выбирается некоторая система точек x_k . Совокупность этих точек называется *сеткой*, а точки x_k называются *узлами сетки*. Достаточно часто используют *равномерную сетку узлов*, когда $x_k = a + kh$, $h = \frac{b-a}{n}$, $k = \overline{0, n}$, n — заданное целое число.

2. Краевая задача (7.7.32), (7.7.33) на множестве узлов, принадлежащих сетке, заменяется некоторой *сеточной задачей*, то есть некоторыми соотношениями между приближенными значениями решения краевой задачи (7.7.32), (7.7.33) в узлах сетки.

3. Сеточная задача решается каким-либо численным методом. Это позволяет найти приближенные значения решения краевой задачи в узлах сетки.

Метод сеток является одним из наиболее универсальных как для обыкновенных дифференциальных уравнений, так и для уравнений в частных производных. Рассмотрим этот метод в применении к решению линейных краевых задач для дифференциальных уравнений второго порядка.

Пусть линейная краевая задача задана в виде:

$$A(x)y'' + B(x)y' + C(x)y = f(x), \quad (7.7.34)$$

$$y(a) = \alpha, \quad y(b) = \beta. \quad (7.7.35)$$

На интервале $[a, b]$ зададим равномерную сетку узлов $x_i = a + ih$, $i = \overline{0, n}$ и обозначим y_i , $i = \overline{0, n}$ значения приближен-

ного решения в точках x_i . В дифференциальном уравнении (7.7.34) производные заменим разностными отношениями:

$$y'(x_i) = \begin{cases} \frac{y_{i+1} - y_i}{h}, \\ \frac{y_i - y_{i-1}}{h}, \\ \frac{y_{i+1} - y_{i-1}}{2h}, \end{cases} \quad (7.7.36)$$

$$y''(x_i) = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}. \quad (7.7.37)$$

Тогда, записав уравнение (7.7.34) для точки x_i и заменив производные в этой точке согласно (7.7.36), (7.7.37), получим систему линейных алгебраических уравнений вида:

$$\begin{aligned} y_0 &= \alpha, \\ a_i y_{i-1} + b_i y_i + c_i y_{i+1} &= g_i, \quad i = \overline{1, i-1}, \\ y_n &= \beta, \end{aligned} \quad (7.7.38)$$

где a_i, b_i, c_i, g_i получены приведением подобных членов после замены производных. Система (7.7.38) является системой с трехдиагональной матрицей. Для решения таких систем используется *метод прогонки*. Согласно этому методу решение системы записывается в виде:

$$y_i = L_i y_{i+1} + K_i, \quad (7.7.39)$$

где L_i и K_i – неизвестные коэффициенты, которые называются *прогоночными*.

Так как, при $i = 0$ $y_0 = \alpha$ и

$$y_0 = L_0 y_1 + K_0,$$

то

$$K_0 = \alpha, \quad L_0 = 0.$$

При $i=1$ имеем

$$y_1 = L_1 y_2 + K_1,$$

а из уравнения системы (7.7.38)

$$a_1 y_0 + b_1 y_1 + c_1 y_2 = g_1$$

или

$$y_1 = \frac{-c_1}{b_1} y_2 + \frac{g_1 - a_1 K_0}{b_1}.$$

Тогда для определения прогоночных коэффициентов получаются следующие рекуррентные соотношения:

$$L_i = -\frac{c_i}{a_i \cdot L_{i-1} + b_i}, \quad K_i = \frac{g_i - a_i \cdot K_{i-1}}{a_i \cdot L_{i-1} + b_i}, \quad i = \overline{1, n-1}. \quad (7.7.40)$$

Решение системы (7.7.38) находится в обратном порядке следующим образом:

$$\begin{aligned} y_n &= \beta, \\ y_{i-1} &= L_{i-1} y_i + K_{i-1}, \quad i = n, n-1, \dots, 2, \\ y_0 &= \alpha. \end{aligned} \quad (7.7.41)$$

Для того, чтобы получить решение задачи (7.7.34), (7.7.35) в аналитической форме, необходимо аппроксимировать значения y_0, y_1, \dots, y_n некоторой функцией.

Существуют и другие методы решения граничных задач, получающие решение, как в виде таблицы, так и в виде аналитической функции, например, в виде отрезка некоторого ряда.

7.7.7 Метод коллокации решения краевых задач

При решении задач физики и механики искомое решение иногда более выгодно находить в аналитическом виде. Часто в таких случаях не преследуют цель получить решение с большой точностью, а ограничиваются отысканием функции, точно удовлетворяющей граничным условиям и некоторым соотношениям, связанным с заданным дифференциальным уравнением. Эти соотношения строятся таким образом, что функция, удовлетворяющая им, удовлетворяет приближенно с возможно хорошей точностью также и заданному дифференциальному уравнению, при этом эти соотношения в различных методах строятся по-разному.

Решение краевой задачи

$$\begin{aligned} L(y) &= y'' + p(x)y' + q(x)y = f(x) \\ \Gamma_a(y) &= \alpha_0 y(a) + \alpha_1 y'(a) = A \\ \Gamma_b(y) &= \beta_0 y(b) + \beta_1 y'(b) = B \end{aligned} \quad (7.7.42)$$

ищется в виде:

$$y_n(x_0) = \varphi_0(x) + \sum_{i=1}^n c_i \varphi_i(x), \quad (7.7.43)$$

где конечная система базисных функций $\{\varphi_i(x)\}$, $i = \overline{0, n}$, заданных на $[a, b]$, выбирается так, чтобы функция $\varphi_0(x)$ удовлетворяла неоднородным краевым условиям:

$$l_a(\varphi_0) = A, \quad l_b(\varphi_0) = B, \quad (7.7.44)$$

а функции $\varphi_i(x)$, $i = \overline{1, n}$ удовлетворяли бы однородным краевым условиям:

$$l_a(\varphi_i) = l_b(\varphi_i) = 0, \quad i = \overline{1, n}. \quad (7.7.45)$$

Подставляя (7.7.43) в дифференциальное уравнение, получим невязку

$$R(x, c, \dots, c_n) = L[\varphi_0] + \sum_{i=1}^n c_i L[\varphi_i] - f(x). \quad (7.7.46)$$

Потребуем, чтобы невязка $R(x, c, \dots, c_n)$ обращалась в нуль на некоторой системе точек x_1, \dots, x_n отрезка $[a, b]$, называемых *точками коллокации*, причем число таких точек должно равняться числу коэффициентов c_i в выражении (7.7.43). Тогда, для определения значений c_1, \dots, c_n , получим систему уравнений

$$\begin{aligned} R(x_1, c_1, \dots, c_n) &= 0, \\ R(x_2, c_1, \dots, c_n) &= 0, \\ &\dots\dots\dots \\ R(x_n, c_1, \dots, c_n) &= 0, \end{aligned} \quad (7.7.47)$$

определитель которой имеет вид:

$$D = \begin{vmatrix} L(\varphi_1(x_1)) & L(\varphi_2(x_1)) & \dots & L(\varphi_n(x_1)) \\ & & \dots & \\ L(\varphi_1(x_n)) & L(\varphi_2(x_n)) & \dots & L(\varphi_n(x_n)) \end{vmatrix}. \quad (7.7.48)$$

Система (7.7.47) имеет единственное решение, если $D \neq 0$, в связи с этим функции $L(\varphi_i(x))$, $i = \overline{1, n}$ должны составлять *систему Чебышева*.

Для достижения хорошего качества приближения $y_n(x)$ к точному решению $y(x)$ функции $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ должны удовлетворять следующим условиям:

- 1) быть линейно независимыми;
- 2) при непрерывных функциях $p(x), q(x), f(x)$ непрерывными на $[a, b]$ вместе со своими производными до второго порядка включительно должны быть и функции $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$;
- 3) для того, чтобы $L(y_n(x))$ могли сколь угодно точно приближать значения $L(y(x))$, система функций $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ должна быть полной в классе дважды непрерывно дифференцируемых функций.

Существуют и другие методы решения граничной задачи (7.7.42) в виде (7.7.43), например, метод Галеркина, наименьших квадратов. Однако метод коллокации приводит к более простым вычислениям.

Замечание.

Для решения краевой задачи (7.7.42) в качестве функций $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ можно выбрать, например, следующие функции:

- 1) $\varphi_0(x) = e + d \cdot x$, где e и d задаются таким образом, чтобы выполнялись неоднородные краевые условия (7.7.44);
- 2) $\varphi_1(x) = (x - a)(x - b)$, $\varphi_2(x) = (x - a)^2(x - b)$, ..., $\varphi_n(x) = (x - a)^n(x - b)$, которые будут удовлетворять однородным краевым условиям (7.7.45).

8 МОДЕЛИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ

Рассмотрим задачи моделирования систем управления в пространстве состояний. При этом будем проектировать следящие системы управления для работы в реальном времени функционирования объекта на основе методов аналитического конструирования оптимальных регуляторов.

В реальных условиях практически любой объект подвержен влиянию случайных внешних воздействий, информация о состоянии объекта часто бывает неполной и измеряется с ошибками, кроме того, в модели объекта присутствуют неизвестные параметры. В связи с этим будем рассматривать синтез систем управления, учитывающий факт неопределенности модели.

Проектирование системы управления осуществляется постепенным усложнением структурной схемы с помощью добавления новых алгоритмов. Решение всех задач доведено до вычислительно реализуемых алгоритмов.

8.1 Описание систем в пространстве состояний

Описание систем во временной области лежит в основе современной теории управления. *Временная область* – это область, в которой поведение системы рассматривается как функция переменной t (времени). Анализ и синтез систем управления во временной области основан на понятии состояния системы. *Состояние системы* – это совокупность таких переменных, знание которых наряду с входными функциями и уравнениями, описывающими динамику системы, позволяет определить ее будущее состояние и выходную переменную. Для динамической системы ее состояние описывается набором переменных состояния $x_1(t), x_2(t), \dots, x_n(t)$. Это такие переменные, которые определяют будущее поведение системы, если известно ее текущее состояние и все внешние воздействия. Например, для системы, структурная схема которой изображена на рис. 8.1.1, переменные $x_1(t), x_2(t), \dots, x_n(t)$ имеют следующий смысл: если в момент времени t_0 известны начальные значения $x_1(t_0), x_2(t_0), \dots, x_n(t_0)$ и входные сигналы $U_1(t), \dots, U_m(t)$ для $t \geq t_0$, то этой информации

достаточно, чтобы определить будущие значения всех переменных состояния и выходных переменных. Таким образом, *переменные состояния описывают поведение систем в будущем, если известны текущее состояние, внешние воздействия и уравнения динамики системы.*

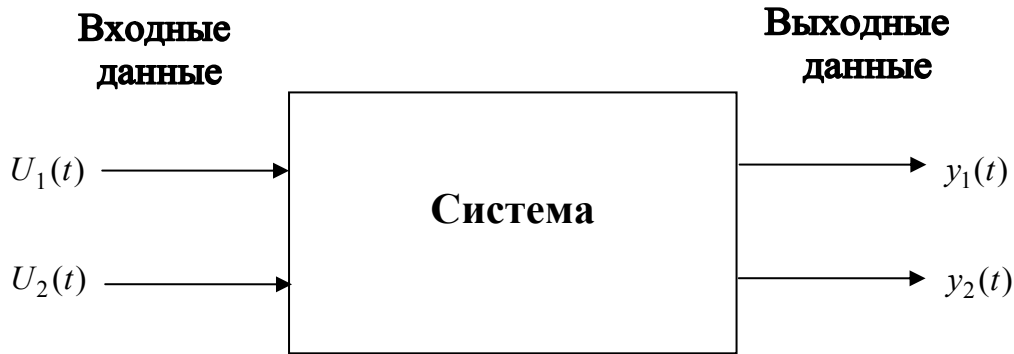


Рис. 8.1.1

Пример. Рассмотрим механическую систему «масса – пружина» с затуханием, изображенную на рис. 8.1.2.

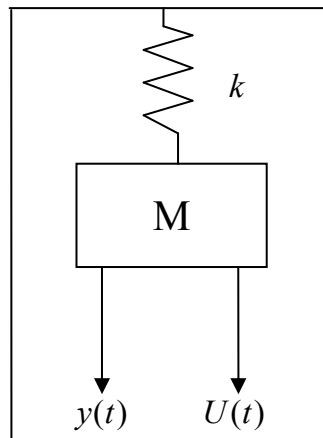


Рис. 8.1.2

Дифференциальное уравнение, описывающее поведение системы, обычно записывается в виде

$$M \frac{d^2 y(t)}{dt^2} + b \frac{dy(t)}{dt} + ky(t) = U(t). \quad (8.1.1)$$

Число переменных состояния, выбираемых для описания системы, должно быть по возможности минимальным, чтобы среди них не было излишних. Для данной системы вполне доста-

точно иметь две переменные состояния – положение и скорость движения массы:

$$\begin{aligned}x_1(t) &= y(t), \\x_2(t) &= \frac{dy(t)}{dt}.\end{aligned}\tag{8.1.2}$$

С учетом переменных состояния, уравнение (8.1.1) примет вид:

$$M \frac{dx_2(t)}{dt} + bx_2(t) + kx_1(t) = U(t).$$

Это уравнение можно представить в виде эквивалентной системы двух дифференциальных уравнений первого порядка:

$$\begin{aligned}\frac{dx_1(t)}{dt} &= x_2(t), \\ \frac{dx_2(t)}{dt} &= -\frac{k}{M}x_1(t) - \frac{b}{M}x_2(t) + \frac{1}{M}U(t).\end{aligned}\tag{8.1.3}$$

Уравнения (8.1.3) описывают поведение системы в терминах скорости изменения каждой переменной состояния.

Переменные состояния, описывающие систему, не являются единственными и всегда можно выбрать альтернативную комбинацию таких переменных. Например, для приведенной выше системы «масса – пружина» в качестве переменных состояния можно выбрать любые две линейно-независимые комбинации $x_1(t)$ и $x_2(t)$. На практике в качестве переменных состояния часто выбирают такие физические переменные, которые легко могут быть измерены.

Переменные состояния характеризуют *динамику* системы. Инженера в первую очередь интересуют физические системы, в которых переменными состояниями являются напряжения, токи, скорости, перемещения, давления, температуры и другие физические величины. Однако понятие состояния применимо к анализу не только физических, но также биологических, социальных и экономических систем. Для этих систем понятие состояния не ограничивается рамками представлений об энергии и подходит к переменным состояниям в более широком смысле, трактуя их как переменные любой природы, описывающие будущее поведение системы.

В общем случае систему обыкновенных дифференциальных уравнений, которая описывает систему управления в переменных состояниях, можно представить в виде:

$$\dot{x}(t) = f(t, x(t), u(t)), \quad x(t_0) = x_0, \quad (8.1.4)$$

где $x(t) = (x_1(t), x_2(t), \dots, x_n(t))^T$ – вектор состояния; $u(t) = (u_1(t), u_2(t), \dots, u_m(t))^T$ – вектор управляющих воздействий, $f(t, x(t), u(t))$ – вектор-функция указанных аргументов, задающая динамику системы, $x_0 = (x_{01}, \dots, x_{0n})^T$ – вектор начального состояния.

Если $x(t)$ и (или) $u(t)$ входят в $f(\cdot)$ нелинейно, то система (8.1.4) называется *нелинейной*, если систему (8.1.4) можно представить в виде:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)u(t), \quad x(t_0) = x_0, \quad (8.1.5)$$

где $\bar{A}(t)$ – матрица динамических свойств модели объекта, размерности $n \times n$, $\bar{B}(t)$ – матрица влияния управляющих воздействий, размерности $n \times m$, то такая система называется *линейной*. Если элементы матрицы $\bar{A}(t)$ и (или) $\bar{B}(t)$ зависят от времени, то система (8.1.5) является *нестационарной*, если ни один из элементов матриц $\bar{A}(t)$ и $\bar{B}(t)$ не зависит от времени, то система (8.1.5) является *стационарной*. Если система управления является нелинейной, то достаточно часто ее *линеаризуют* и представляют в виде (8.1.5).

Заметим, что систему управления в переменных состояниях можно представить и в дискретной форме:

$$x(k+1) = f(k, x(k), u(k)), \quad x(0) = x_0, \quad (8.1.6)$$

где $x(k) = (x_1(k), \dots, x_n(k))^T$ – вектор состояния на k -ом такте; $u(k) = (u_1(k), \dots, u_m(k))^T$ – вектор управляющих воздействий; $f(k, x(k), u(k))$ – вектор-функция дискретных аргументов, задающая динамику системы на k -ом такте; $x(0) = (x_{01}, \dots, x_{0n})^T$ – вектор начального состояния. Аналогично (8.1.4) система (8.1.6) может быть нелинейной и линейной, представимой в виде:

$$x(k+1) = A(k)x(k) + B(k)u(k), \quad x(0) = x_0. \quad (8.1.7)$$

Кроме того, системы (8.1.6), (8.1.7) могут быть как стационарными, так и нестационарными.

В дальнейшем, будем считать, что математическая модель, описывающая поведение управляемого объекта задана в виде (8.1.5) или (8.1.7). Использование линейных систем для описания моделей объектов обусловлено тем, что идеи и методы линейной теории автоматического управления с соответствующими оговорками широко используются и для других моделей объектов управления. Кроме того, математический аппарат матричной алгебры достаточно легко реализуется на ЭВМ.

8.2 Аналитическое конструирование оптимальных регуляторов

Синтез систем управления – одна из важнейших технических задач. Целью синтеза является создание системы, которая удовлетворяла бы требуемым показателям качества. Качество системы управления можно охарактеризовать, например, интегральными оценками, и тогда синтез системы управления должен быть основан на минимизации оценки качества. *Системы управления, в которых обеспечивается минимум соответствующей оценки качества, называются оптимальными системами управления.*

Свойства оптимального управления в значительной мере определяются выбранным критерием оптимизации. Для методов синтеза управления объектами, динамические свойства которых заданы в пространстве состояний, наиболее удобными являются интегральные показатели качества. При этом конкретная форма показателя качества тесно связана с методом синтеза. В общем случае такие показатели описываются функционалами вида:

$$J = V_z(x(T), T) + \int_{t_0}^T Q_1(x(t), t) dt + \int_{t_0}^T Q_2(u(t), t) dt, \quad (8.2.1)$$

определенными на всех возможных траекториях $x(t)$ в рассматриваемом пространстве состояний для всех $t \in [t_0, T]$, где Q_1 , Q_2 – заданные функции указанных аргументов, удовлетворяющие некоторым условиям. Первое слагаемое в (8.2.1) является терми-

нальным членом функционала. Оно определяет вклад в функционал конечного состояния $x(T)$. При этом рассматривается задача со свободным правым концом, когда отсутствует требование прохождения вектора состояния через заданную точку в момент T . Второе слагаемое в (8.2.1) представляет собой интегральную оценку качества переходного процесса объекта управления на интервале $[t_0, T]$. Третье слагаемое – интегральная оценка «расходов» сигнала управления на интервале $[t_0, T]$. Если функции V_z , Q_1 , Q_2 являются положительно определенными, а их единственные нулевые значения соответствуют состоянию объекта, требуемому по условию решаемой задачи, то оптимальность синтезируемого управления можно понимать в смысле достижения минимума функционала (8.2.1).

Положительная определенность подынтегральных функций в (8.2.1) допускает использование для разработки конкретных критериев оптимизации широкого класса функций Q_1 и Q_2 . В то же время практика решения задач управления динамическими объектами показывает, что в большинстве случаев требования к качеству переходных процессов могут быть заданы квадратичной формой функции $Q_1(x(t), t)$, т.е.

$$Q_1(x, t) = \frac{1}{2} x^T(t) C x(t), \quad (8.2.2)$$

где C – неотрицательно определенная весовая матрица.

Среди методов автоматического управления объектами, математические модели которых в пространстве состояний описываются линейными нестационарными дифференциальными уравнениями, выделяется *метод аналитического конструирования оптимальных регуляторов* (АКОР), так как линейно-квадратичная задача является единственной, при которой решение получается в общем виде в форме обратной связи.

Термин «аналитическое конструирование регуляторов» был введен А.М. Летовым и означает синтез оптимальных систем управления, основанный на минимизации функционала, т.е. на решении вариационной задачи. Практически одновременно эта же задача была решена Р.Калманом. Метод аналитического конструирования позволяет в аналитическом виде определять струк-

туру и параметры системы управления, достаточно полно учитывая при этом технические требования к качеству функционирования объекта.

Полнота общего решения обуславливается условиями, налагаемыми на динамические свойства управляемого объекта и на структуру оптимизируемого критерия, отражающего предъявляемые к управлению требования. Эти условия состоят в следующем:

1) уравнение, описывающее движение объекта, должно быть линейным относительно вектора управляющих воздействий;

2) область возможных значений управлений должна быть незамкнутой;

3) все возможные переходные функции объекта управления должны быть непрерывно-дифференцируемыми в рассматриваемом пространстве состояний;

4) минимизируемый функционал должен быть квадратичным относительно вектора управления, т.е.

$$Q_2(u(t), t) = \frac{1}{2} u^T(t) D u(t), \quad (8.2.3)$$

где D – некоторая положительно определенная матрица, характеризующая «свободу» выбора управления.

8.2.1 Оптимальное управление при минимизации классического квадратичного функционала

Согласно методу аналитического конструирования Летова-Калмана, оптимальное управление объектом (8.1.5), минимизирующее квадратичный функционал:

$$J = \frac{1}{2} x^T(T) E x(T) + \frac{1}{2} \int_{t_0}^T [x^T(t) C x(t) + u^T(t) D u(t)] dt, \quad (8.2.4)$$

имеет вид:

$$u(t) = -D^{-1} \bar{B}^T(t) \left(\frac{\partial V}{\partial x} \right)^T, \quad (8.2.5)$$

где функция $V = V(x, t)$ является решением нелинейного уравнения в частных производных

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} \bar{A}(\tau)x(\tau) - \frac{1}{2} \frac{\partial V}{\partial x} \bar{B}(\tau) D^{-1} \bar{B}^T(\tau) \left(\frac{\partial V}{\partial x} \right)^T = -\frac{1}{2} x^T(\tau) C x(\tau), \quad (8.2.6)$$

при граничном условии

$$V(T) = \frac{1}{2} x^T(T) E x(T). \quad (8.2.7)$$

В (8.2.4.) E – заданная положительно определенная матрица.

Решение задачи (8.2.6)–(8.2.7) ищется в виде квадратичной формы:

$$V = \frac{1}{2} x^T(t) S(t) x(t), \quad (8.2.8)$$

где $S(t)$ – симметрическая матрица. Тогда

$$u(t) = -D^{-1} \bar{B}^T(t) S(t) x(t), \quad (8.2.9)$$

а матрица $S(t)$ является решением нелинейного дифференциального уравнения Риккати с переменными коэффициентами:

$$\dot{S}(\tau) = -S(\tau) \bar{A}(\tau) - \bar{A}^T(\tau) S(\tau) + S(\tau) \bar{B}(\tau) D^{-1} \bar{B}^T(\tau) S(\tau) - C \quad (8.2.10)$$

при граничном условии

$$S(T) = E. \quad (8.2.11)$$

Таким образом, при синтезе оптимального управления методом аналитического конструирования Летова-Калмана необходимо для каждого момента $t \in [t_0, T]$ формирования управляющих воздействий решать в обратном времени уравнение (8.2.10) при условии (8.2.11). Теория решения матричного уравнения Риккати достаточно хорошо разработана. Однако при практическом применении необходимость решения в обратном времени матричного дифференциального уравнения Риккати делает затруднительным использование этого метода синтеза в процессе функционирования объекта из-за больших затрат времени на получение результата. Кроме того, для нестационарных систем редко удается указать обоснованные требования к состоянию объекта в фиксированный момент T , т.е. обоснованно назначить квадратичную форму $0.5x^T(T)Ex(T)$. Поэтому чаще всего рассматривают функционал (8.2.4) в виде:

$$J = \frac{1}{2} \int_{t_0}^T [x^T(t) C x(t) + u^T(t) D u(t)] dt. \quad (8.2.12)$$

При этом граничное условие для решения уравнения Риккати (8.2.11) принимает вид:

$$S(T) = 0. \quad (8.2.13)$$

В случае стационарной модели (8.1.5) ($\bar{A}(t) = \bar{A}$, $\bar{B}(t) = \bar{B}$) оптимальное управление, обеспечивающее минимум функционала:

$$J = \frac{1}{2} \int_{t_0}^{\infty} [x^T(t)Cx(t) + u^T(t)Du(t)] dt,$$

имеет вид:

$$u(t) = -D^{-1}\bar{B}(t)Sx(t),$$

где S – положительно определенная матрица, удовлетворяющая матричному алгебраическому уравнению Риккати:

$$\bar{A}^T S + S\bar{A} - S\bar{B}H\bar{B}^T S + C = 0.$$

Решение алгебраического уравнения Риккати заменяется определением с заданной точностью установившегося решения дифференциального уравнения Риккати:

$$\dot{S} = \bar{A}^T S + S\bar{A} - S\bar{B}H\bar{B}^T S + C$$

при нулевом начальном условии.

Синтез управляющих воздействий, который осуществляется в реальном времени в процессе функционирования объекта, называется *совмещенным синтезом*. В этом случае большое значение имеет временная задержка, вызванная затратами времени на формирование управляющего сигнала. Время формирования управления при АКОРе Летова-Калмана зависит от двух факторов: величины интервала оптимизации $[t_0, T]$ и времени, которое затрачивается на решение уравнения Риккати. Целенаправленное изменение каждой из этих величин приводит к уменьшению вычислительной задержки при формировании управляющего сигнала.

8.2.2 Оптимальное управление при минимизации функционала обобщенной работы

Термин «критерий обобщенной работы» был введен А.А. Красовским. В этом случае минимизируемый функционал задается в виде:

$$J = \frac{1}{2} x^T(T) E x(T) + \frac{1}{2} \int_{t_0}^T [x^T(t) C x(t) + u^T(t) D u(t) + u_{on}^T(t) D u_{on}(t)] dt, \quad (8.2.14)$$

где u_{on} – оптимальное управление, доставляющее минимум функционалу (8.2.14), которое имеет вид:

$$u(t) = u_{on}(t) = -D^{-1} \bar{B}^T(t) \left(\frac{\partial V}{\partial x} \right)^T, \quad (8.2.15)$$

а $V = V(x, t)$ является решением линейного уравнения в частных производных

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} \bar{A}(\tau) x(\tau) = -\frac{1}{2} x^T(\tau) C x(\tau) \quad (8.2.16)$$

при граничном условии

$$V(T) = \frac{1}{2} x^T(T) E x(T). \quad (8.2.17)$$

Если искать решение задачи (8.2.16)–(8.2.17) в виде квадратичной формы:

$$V = \frac{1}{2} x^T(t) L(t) x(t), \quad (8.2.18)$$

то

$$u(t) = u_{on}(t) = -D^{-1} \bar{B}^T(t) L(t) x(t), \quad (8.2.19)$$

где $L(t)$ является решением линейного матричного дифференциального уравнения Ляпунова

$$\begin{aligned} \dot{L}(\tau) &= -L(\tau) \bar{A}(\tau) - \bar{A}^T(\tau) L(\tau) - C, \\ L(T) &= E. \end{aligned} \quad (8.2.20)$$

Вычислительные затраты при решении уравнения Ляпунова в силу его линейности значительно сокращаются по сравнению с решением уравнения Риккати. Недостатком этого метода является то, что управление (8.2.15) является оптимальным при «свободном» движении объекта, т.е. при $u \equiv 0$. При этом понятно, что поведение объекта при нулевом управлении (нулевом положении органов управления), может очень сильно отличаться от реального управляемого поведения на интервале оптимизации и приводить в область пространства состояний, далекую от реально достигаемой.

Лучшие результаты можно получить, если «свободным» считать поведение объекта при фиксированном положении органов управления. При этом задача управления формулируется как *управление скоростью перемещения регулирующих органов*, а модель объекта представляется в виде:

$$\begin{aligned}\dot{x}(t) &= \bar{A}(t)x(t) + B(t)u(t), & x(t_0) &= x_0, \\ \dot{u}(t) &= v(t), & u(t_0) &= u_0,\end{aligned}\quad (8.2.21)$$

где $u(t)$ – вектор положения органов управления, а $v(t)$ – вектор управления, характеризующий скорость перемещения органов управления.

Заметим, что (8.2.21) можно записать в традиционной форме, как задачу управления положением органов управления, если ввести расширенный вектор состояния $\tilde{x}(t) = (x, u)^T$. Тогда (8.2.21) запишется в виде:

$$\dot{\tilde{x}}(t) = \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t)v(t), \quad \tilde{x}(t_0) = \tilde{x}_0, \quad (8.2.22)$$

где $\tilde{A}(t)$ и $\tilde{B}(t)$ – блочные матрицы вида:

$$\tilde{A}(t) = \begin{pmatrix} \bar{A}(t) & \bar{B}(t) \\ 0 & 0 \end{pmatrix}, \quad \tilde{B}(t) = \begin{pmatrix} 0 \\ I_m \end{pmatrix}. \quad (8.2.23)$$

Здесь 0 – нулевые матрицы соответствующих размерностей, I_m – единичная матрица порядка m .

Кроме того, учитывая сложность определения терминального слагаемого, функционал обобщенной работы для системы (8.2.22) запишем в виде:

$$J = \frac{1}{2} \int_{t_0}^T [\tilde{x}^T(t) \tilde{C} \tilde{x}(t) + v^T(t) D_1 v(t) + v_{on}^T(t) D_1 v_{on}(t)] dt, \quad (8.2.24)$$

где \tilde{C} – весовая матрица порядка $(n + m)$, которая имеет блочный вид:

$$\tilde{C} = \begin{pmatrix} C & 0 \\ 0 & D_2 \end{pmatrix}, \quad (8.2.25)$$

а управление $v_{on}(t)$ определяется следующим образом:

$$v_{on}(t) = -D_1^{-1} \tilde{B}^T \left(\frac{\partial V}{\partial \tilde{x}} \right)^T = D_1^{-1} \left(\frac{\partial V}{\partial u} \right)^T. \quad (8.2.26)$$

Здесь C, D_2 – неотрицательно определенные, а D_1 – положительно определенная весовые матрицы.

Минимизируемый функционал обобщенной работы для модели (8.2.21) запишется в виде:

$$J = \frac{1}{2} \int_{t_0}^T [x^T(t) C x(t) + u^T(t) D_2 u(t) + v^T(t) D_1 v(t) + v_{on}^T(t) D_1 v_{on}(t)] dt, \quad (8.2.27)$$

а модель, описывающая «свободное» движение объекта на интервале оптимизации, которую будем называть *прогнозирующей моделью*, есть

$$\begin{aligned} \dot{x}_M(\tau) &= \bar{A}(\tau)x_M(\tau) + \bar{B}(\tau)u_M(\tau), & x_M(t_0) &= x_0, \\ \dot{u}_M(\tau) &= 0, & u_M(t_0) &= u_0, \end{aligned} \quad (8.2.28)$$

где M – указывает на принадлежность прогнозирующей модели.

Тогда

$$v_{on}(t) = -D_1^{-1} \left(\frac{\partial V}{\partial u} \right)^T, \quad (8.2.29)$$

где функция $V = V(t, x, u)$ удовлетворяет уравнению:

$$\dot{V} = -\frac{1}{2} [x_M^T(\tau) C x_M(\tau) + u_M^T(\tau) D_2 u_M(\tau)], \quad V(T) = 0. \quad (8.2.30)$$

Заметим, что (8.2.30) справедливо на траектории движения модели (8.2.28).

Распишем в (8.2.30) полную производную функции V :

$$\frac{\partial V}{\partial \tau} + \frac{\partial V}{\partial x_M} \dot{x}_M(\tau) = -\frac{1}{2} [x_M^T(\tau) C x_M(\tau) + u_M^T(\tau) D_2 u_M(\tau)] \quad (8.2.31)$$

и обозначим

$$W_1 = \left(\frac{\partial V}{\partial x_M} \right)^T, \quad W_2 = \left(\frac{\partial V}{\partial u_M} \right)^T, \quad (8.2.32)$$

n -мерный и m -мерный векторы-столбцы.

Тогда на траектории движения модели (8.2.28) полные производные для W_1 и W_2 будут равны:

$$\begin{aligned}\dot{W}_1 &= \frac{\partial W_1}{\partial \tau} + \frac{\partial W_1}{\partial x_M} \dot{x}_M, \\ \dot{W}_2 &= \frac{\partial W_2}{\partial \tau} + \frac{\partial W_2}{\partial x_M} \dot{x}_M.\end{aligned}\tag{8.2.33}$$

Если продифференцировать (8.2.31) последовательно по x_M , u_M и подставить уравнения для модели (8.2.28), то, с учетом обозначений (8.2.32), получим систему обыкновенных дифференциальных уравнений:

$$\begin{aligned}\dot{W}_1(\tau) &= -\bar{A}^T(\tau)W_1(\tau) - C x_M(\tau), & W_1(T) &= 0, \\ \dot{W}_2 &= -\bar{B}^T(\tau)W_1(\tau) - D_2 u_M(\tau), & W_2(T) &= 0.\end{aligned}\tag{8.2.34}$$

Решая систему (8.2.34) в обратном времени, найдем управление в момент t , которое будет равно:

$$u_{on}(t) = -D_1^{-1}W_2(t).\tag{8.2.35}$$

8.2.3 Оптимальное управление при минимизации локального квадратичного критерия

Будем формировать управляющие воздействия по текущей информации об объекте. При этом минимизируемый функционал можно представить в виде:

$$J(k) = \frac{1}{2}[x^T(k+1) C x(k+1) + u^T(k) D u(k)],\tag{8.2.36}$$

где C – неотрицательно определенная, а D – положительно определенная весовые матрицы, k – соответствует дискретному моменту времени t_k , а модель объекта описать системой линейных разностных уравнений:

$$x(k+1) = A(k)x(k) + B(k)u(k), \quad x(0) = x_0.\tag{8.2.37}$$

Управление $u(k)$ будем определять из условия минимума функционала (8.2.36), т.е.

$$\frac{\partial J(k)}{\partial u(k)} = 0.\tag{8.2.38}$$

Запишем функционал (8.2.36) для модели (8.2.37) (для простоты записи аргумент k временно опустим)

$$\begin{aligned}
J &= \frac{1}{2}[(Ax + Bu)^T C (Ax + Bu) + u^T Du] = \\
&= \frac{1}{2}[x^T A^T C A x + x^T A^T C B u + u^T B^T C A x + u^T B^T C B u + u^T D u].
\end{aligned}$$

Воспользовавшись правилами дифференцирования билинейной формы $x^T A y$, т.е.

$$\frac{\partial x^T A y}{\partial x} = y^T A^T, \quad \frac{\partial x^T A y}{\partial y} = x^T A,$$

где x, y – векторы-столбцы, A – матрица соответствующего порядка, получим

$$\frac{\partial J}{\partial u} = x^T A^T C B + u^T B^T C B + u^T D = 0,$$

и

$$(B^T C B + D)u + B^T C A x = 0.$$

Тогда управление, полученное на основе минимума функционала (8.2.36), имеет вид:

$$u(k) = -(B^T(k)CB(k) + D)^{-1} B^T(k)CA(k)x(k). \quad (8.2.39)$$

8.2.4 Синтез следящей системы управления

Ранее рассматривались задачи формирования управляющих воздействий, когда целью управления являлось приведение системы в нулевое состояние. Существует большой круг задач, где выдвигается требование достижения объектом некоторого *заданного состояния* $x_z(t)$ с последующим удержанием состояния объекта в малой окрестности $x_z(t)$, причем задача оптимизации остается нетерминальной.

Имеются, вообще говоря, две возможности формализации требований к динамике переходных процессов управляемого движения:

1) $x_z(t)$ формируется почти мгновенно по результатам анализа решения навигационной задачи или положения органов управления;

2) используется эталонная модель движения объекта, возбуждаемая сигналами навигационного комплекса или датчиками органов управления.

В первом случае все требования сводятся к соответствующему подбору параметров функционала, во втором – параметры функционала подбираются из условия обеспечения наилучшего слежения за эталоном.

Будем считать, что $x_z(t)$ – вектор, компоненты которого заданы в каждый момент времени функционирования системы управления. В этом случае оптимизируемые функционалы (8.2.12) и (8.2.27) запишутся с помощью вектора рассогласования $(x(t) - x_z(t))$, а (8.2.36) – $(x(k+1) - x_z(k))$.

Управление при минимизации классического квадратичного функционала

$$J = \frac{1}{2} \int_{t_0}^T [(x(t) - x_z(t))^T C(x(t) - x_z(t)) + u^T(t) Du(t)] dt \quad (8.2.40)$$

для модели объекта (8.1.5) запишется в виде:

$$u(t) = -D^{-1} \bar{B}^T(t) S(t) (x(t) - x_z(t)). \quad (8.2.41)$$

Для функционала обобщенной работы:

$$J = \frac{1}{2} \int_{t_0}^T [(x(t) - x_z(t))^T C(x(t) - x_z(t)) + u^T(t) D_2 u(t) + v^T(t) D_1 v(t) + v_{on}^T(t) D_1 v_{on}(t)] dt \quad (8.2.42)$$

для модели объекта (8.2.21) получим:

$$v(t) = -D_1^{-1} W_2(t) \quad (8.2.43)$$

и

$$\begin{aligned} \dot{W}_1(\tau) &= -\bar{A}^T(\tau) W_1(\tau) - C(x(\tau) - x_z(\tau)), & W_1(T) &= 0, \\ \dot{W}_2(\tau) &= -\bar{B}^T(\tau) W_1(\tau) - D_2 u(\tau), & W_2(T) &= 0. \end{aligned} \quad (8.2.44)$$

При формировании локально-оптимального управления на основе минимизации функционала

$$J(k) = \frac{1}{2} [(x(k+1) - x_z(k))^T C(x(k+1) - x_z(k)) + u^T(k) Du(k)] \quad (8.2.45)$$

для модели объекта (8.2.37), получим:

$$u(k) = -(B^T(k)CB(k) + D)^{-1} B^T(k)C[A(k)x(k) - x_z(k)]. \quad (8.2.46)$$

8.3 Моделирование систем оптимального управления

Развитие цифровых информационно-измерительных комплексов привело к тому, что все чаще управление объектом в процессе его функционирования осуществляется с помощью компьютера. При этом центр тяжести процесса проектирования системы управления приходится на алгоритмическое и программное обеспечение.

8.3.1 Основные понятия цифровых систем управления

Включение ЭВМ в контур управления неизбежно создает особые условия его реализации, основными из которых являются: дискретность формируемого управления по времени и запаздывание подачи управляющего воздействия на приводы рулевых органов по отношению к моменту выдачи измерительной системой всей необходимой информации. Такие системы управления оперируют только с цифровой информацией и называются цифровыми системами управления.

Цифровую систему управления непрерывным объектом с замкнутой обратной связью схематично можно изобразить следующим образом (рис. 8.3.1):

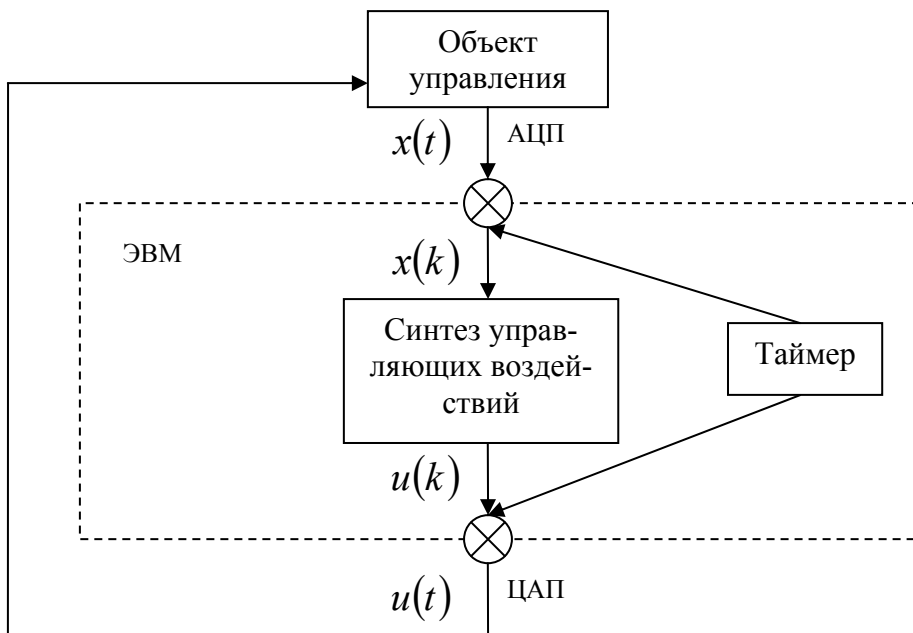


Рис. 8.3.1

Работа системы управления синхронизируется в ЭВМ таймером реального времени, который определяет момент поступления информации об объекте и момент воздействия управляющего сигнала на объект. При этом состояние объекта измеряется только в дискретные моменты времени. На входе в систему управления непрерывный вектор состояния $x(t)$ преобразуется в цифровую форму аналого-цифровым преобразователем (АЦП), а на выходе цифровой вектор управляющих воздействий $u(k)$ преобразуется в непрерывный сигнал $u(t)$ цифро-аналоговым преобразователем (ЦАП).

Момент времени, в который происходит преобразование непрерывной информации в дискретную, называется *моментом квантования* управляющего воздействия и обозначается k . Промежуток времени $\Delta t = t_{k+1} - t_k$ между двумя моментами квантования называется *периодом квантования*.

Квантование – неизбежный процесс в цифровых системах управления, обусловленный дискретной природой самих ЭВМ. Будем полагать, что для управления исходным непрерывным объектом используется цифровая система, формирующая кусочно-постоянный вектор управления $u(k)$ с моментом и периодом квантования, совпадающим с моментом и периодом поступления в ЭВМ информации об объекте. При этом непрерывность управляющего сигнала при воздействии на объект достигается путем сохранения его постоянной величины между формированиями. Таким образом, управление принадлежит к классу кусочно-постоянных непрерывных слева функций с фиксированным периодом квантования Δt . Если Δt достаточно мало, то цифровая система управления функционирует почти как непрерывная, так как потери информации за счет квантования будут незначительными. Но слишком малое время Δt может оказаться недостаточным для формирования управляющего сигнала, так как реализация систем управления в реальных условиях движения объекта характеризуется большой сложностью, при этом формирование управляющих воздействий должно осуществляться за время, пренебрежительно малое в сравнении со скоростью изменения внешней среды и самого объекта.

8.3.2 Моделирование поведения управляемого объекта

При проведении имитационных экспериментов для определения качества синтезируемого управления необходимо моделировать поведение управляемого объекта. Будем предполагать, что поведение объекта описывается системой обыкновенных линейных дифференциальных уравнений вида:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)u(t), \quad x(t_0) = x_0. \quad (8.3.1)$$

Моделирование будем осуществлять в предположении, что шаг дискретизации Δt совпадает с периодом квантования управляющего сигнала, а момент дискретизации k – с моментом приложения управляющих воздействий. При этом управление является кусочно-постоянным на каждом интервале выдачи управляющих воздействий, то есть

$$u(t) = u(t_k), \quad t_k \leq t < t_{k+1}. \quad (8.3.2)$$

Тогда можно ограничиться описанием поведения объекта в моменты квантования и записать дискретный аналог для системы (8.3.1) в виде:

$$x(t_{k+1}) = \Phi_{\bar{A}}(t_{k+1}, t_k)x(t_k) + \int_{t_k}^{t_{k+1}} \Phi_{\bar{A}}(t_{k+1}, \tau)\bar{B}(\tau)u(\tau)d\tau, \quad x(t_0) = x_0, \quad (8.3.3)$$

где $\Phi_{\bar{A}}(t_{k+1}, t_k)$ – фундаментальная матрица решений соответствующей однородной системы.

По свойству фундаментальной матрицы:

$$\Phi_{\bar{A}}(t_{k+1}, t_k) = \exp(\bar{A}(t_k)(t_{k+1} - t_k)). \quad (8.3.4)$$

Согласно свойству матричной экспоненты и, учитывая, что $t_{k+1} - t_k = \Delta t$, (8.3.4) можно представить в виде ряда:

$$\Phi_{\bar{A}}(t_{k+1}, t_k) = I_n + \sum_{i=1}^{\infty} \frac{\Delta t^i \bar{A}^i(t_k)}{i!}, \quad (8.3.5)$$

где I_n – единичная матрица порядка n .

Тогда, вычислив интеграл в (8.3.3) по формуле левых прямоугольников, дискретную систему (8.3.3) можно записать в разностной форме следующим образом:

$$x(k+1) = A(k)x(k) + B(k)u(k), \quad x(0) = x_0, \quad (8.3.6)$$

где

$$A(k) = I_n + \sum_{i=1}^{\infty} \frac{\Delta t^i \bar{A}^i(t_k)}{i!}, \quad (8.3.7)$$

$$B(k) = \sum_{i=1}^{\infty} \frac{\Delta t^i \bar{A}^{i-1}(t_k) \bar{B}(t_k)}{(i-1)!}, \quad (8.3.8)$$

k – соответствует моменту времени $t_k = t_0 + k\Delta t$.

Представление матричной экспоненты в виде ряда (8.3.5) позволяет достаточно легко регулировать точность построения соответствующей дискретной системы, которую можно определять точностью вычисления фундаментальной матрицы для следующей системы:

$$\dot{x}(t) = \bar{A}(t_0)x(t_0), \quad x(t_0) = x_0. \quad (8.3.9)$$

Дискретный аналог этой системы имеет вид:

$$x(k+1) = A(0)x(k), \quad x(0) = x_0, \quad (8.3.10)$$

где

$$A(0) = I_n + \sum_{i=1}^{\infty} \frac{\Delta t^i \bar{A}^i(t_0)}{i!}. \quad (8.3.11)$$

Если обозначить через $A_N(0)$ сумму N первых членов ряда (8.3.11), то матрица $A_N(0)$ аппроксимирует $A(0)$ с погрешностью порядка $O(\Delta t^N)$. При этом число слагаемых в $A_N(0)$ можно задавать заранее (вычисление с фиксированной точностью) или определять автоматически с помощью соотношения

$$\frac{\|A_N(0) - A_{N-1}(0)\|}{\|A_N(0)\|} \leq \varepsilon, \quad (8.3.12)$$

где ε выбирается, например, из условия обеспечения максимальной точности вычислений на ЭВМ конкретного типа.

Если модель объекта задана в виде:

$$\begin{aligned} \dot{x}(t) &= \bar{A}(t)x(t) + \bar{B}(t)u(t), \quad x(t_0) = x_0, \\ \dot{u}(t) &= v(t), \quad u(t_0) = u_0, \end{aligned} \quad (8.3.13)$$

где $u(t)$ – вектор положения органов управления, а $v(t)$ – вектор управления, то дискретная система для (8.3.13) получается, аналогично предыдущему, при введении расширенного вектора состояния $\tilde{x} = (x, u)^T$ и записи (8.3.13) в виде системы с блочными матрицами:

$$\dot{\tilde{x}} = \tilde{A}(t)\tilde{x}(t) + \tilde{B}(t)v(t), \quad \tilde{x}(t_0) = \tilde{x}_0, \quad (8.3.14)$$

где

$$\tilde{A}(t) = \left(\begin{array}{c|c} \overline{A}(t) & \overline{B}(t) \\ \hline 0 & 0 \end{array} \right), \quad \tilde{B}(t) = \left(\begin{array}{c} 0 \\ \hline I_m \end{array} \right). \quad (8.3.15)$$

Здесь 0 – нулевые матрицы соответствующей размерности, $\tilde{x}_0 = (x_0, u_0)^T$. Тогда система разностных уравнений для (8.3.13) примет вид:

$$x(k+1) = A(k)x(k) + B(k)u(k) + B_1(k)v(k), \quad x(0) = x_0, \quad (8.3.16)$$

$$u(k+1) = u(k) + \Delta t v(k), \quad u(0) = u_0,$$

где $A(k)$ и $B(k)$ определяются согласно (8.3.7) и (8.3.8) и

$$B_1(k) = \sum_{i=2}^{\infty} \frac{\Delta t^i \overline{A}^{i-2}(t_k) \overline{B}(t_k)}{(i-1)!}. \quad (8.3.17)$$

Полученные соотношения являются общими. Так, если при вычислении матриц отбросить все слагаемые, порядок малости которых превышает $O(\Delta t^4)$, то полученные соотношения будут соответствовать моделированию поведения управляемого объекта методом Рунге-Кутты. А если отбросить слагаемые, имеющие порядок малости выше $O(\Delta t)$, то применяется метод Эйлера.

При малом значении Δt для моделирования достаточно часто используется метод Эйлера. В этом случае получаются достаточно простые соотношения для вычисления матриц дискретной системы:

$$A(k) = I_n + \Delta t \overline{A}(t_k), \quad B(k) = \Delta t \overline{B}(t_k), \quad B_1(k) = 0, \quad (8.3.18)$$

и система (8.3.16) имеет вид:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k), \quad x(0) = x_0, \\ u(k+1) &= u(k) + \Delta t v(k), \quad u(0) = u_0. \end{aligned} \quad (8.3.19)$$

Моделирование поведения объекта в случае нестационарной модели (например, моделирование самолета на взлете) можно осуществлять следующим образом. На траектории движения объекта задаются значения матриц (или элементов этих матриц), характеризующих модель объекта в некоторые моменты времени T_i , $i = \overline{1, N_R}$. Нестационарная модель в каждый момент времени $t \in [t_0, T]$ строится по заданным характеристикам, используя ме-

тоды приближения данных: интерполирование, сплайн-функции, аппроксимацию по методу наименьших квадратов и т.д.

8.3.3 Синтез оптимального управления

Рассмотрим совмещенный синтез цифрового управления на основе оптимизации критериев классического квадратичного (8.2.40), обобщенной работы (8.2.42) и локального (8.2.45) при слежении за заданным состоянием $x_z(t)$. При этом, если решается задача стабилизации, то полагаем $x_z(t) \equiv 0$.

Кроме того, будем предполагать, что:

- 1) все временные переменные кратны Δt ;
- 2) моменты и периоды квантования одинаковы для всех координат объекта;
- 3) цифровая система формирует вектор управления с моментом и периодом квантования, которые совпадают с моментом и периодом поступления информации об объекте;
- 4) управления являются кусочно-постоянными функциями на каждом интервале выдачи управляющих воздействий

$$u(t) = u(k), \quad t_k \leq t < t_{k+1}.$$

Пусть в управляющем компьютере модель объекта представлена в виде:

$$x(k+1) = A(k)x(k) + B(k)u(k), \quad x(0) = x_0, \quad (8.3.20)$$

или

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k), \quad x(0) = x_0, \\ u(k+1) &= u(k) + \Delta t v(k), \quad u(0) = u_0, \end{aligned} \quad (8.3.21)$$

где

$$A(k) = I_n + \Delta t \bar{A}(t_k), \quad B(k) = \Delta t \bar{B}(t_k). \quad (8.3.22)$$

Форма представления модели объекта зависит от решаемой задачи управления: управление положением органов управления или управление скоростью отклонения органов управления.

Синтез управляющих воздействий в процессе функционирования объекта накладывает определенные ограничения на его реализацию. Основным ограничением является то, что при формировании управления в момент $t \in [t_0, T]$ известна информация о поведении объекта в предыдущие моменты времени и не из-

вестна в последующие. В связи с этим будем рассматривать алгоритмы синтеза управляющих воздействий, учитывающие эти ограничения и предназначенные для совмещенного синтеза.

Синтез управляющих воздействий по классическому квадратичному критерию

Если модель объекта (8.3.20) является стационарной, то управление, формируемое на k -ом такте (в момент $t_k = t_0 + k\Delta t$) на основе минимизации функционала (8.2.40), имеет вид:

$$u(k) = -D_d^{-1} B^T S(x(k) - x_z(t_k)), \quad (8.3.23)$$

где матрица S является решением уравнения Риккати, которое путем замены переменной можно записать в прямом времени:

$$\begin{aligned} \dot{S}(t) &= \bar{A}^T S(t) + S(t) \bar{A} - S(t) \bar{B} D^{-1} \bar{B}^T S(t) + C, \\ S(0) &= 0. \end{aligned} \quad (8.3.24)$$

При этом можно искать установившееся решение уравнения (8.3.24) по следующему итерационному алгоритму:

$$\begin{aligned} S(i+1) &= A^T S(i) + S(i) A - S(i) B D_d^{-1} B^T S(i) - S(i) + C_d, \\ S(0) &= 0, \end{aligned} \quad (8.3.25)$$

и, при выполнении условия:

$$\frac{\|S(i+1) - S(i)\|}{\|S(i+1)\|} \leq \varepsilon, \quad (8.3.26)$$

полагать $S = S(i+1)$. В (8.2.23)–(8.2.26) $D_d = \Delta t D$, $C_d = \Delta t C$, ε – заданная точность решения уравнения Риккати. Заметим, что уравнение (8.3.25) записано с помощью матриц дискретной системы, которая получена из непрерывной методом Эйлера.

Если модель объекта (8.3.20) является нестационарной, то запишем минимизируемый функционал с использованием *скользящего интервала оптимизации*, то есть:

$$J(t_k) = \frac{1}{2} \int_{t_k}^{t_k + l_p \Delta t} [(x(t) - x_z(t_k))^T C (x(t) - x_z(t_k)) + u^T(t) D u(t)] dt, \quad (8.3.27)$$

и поведение объекта на скользящем интервале оптимизации $[t_k, t_k + l_p \Delta t]$ описывать стационарной системой вида:

$$x_M(j+1) = A(k)x_M(j) + B(k)u(j), \quad x_M(j=k) = x(k), \\ j = k, k+1, \dots, k+l_p-1, \quad (8.3.28)$$

которую будем называть *прогнозирующей моделью*, а «М» указывает на принадлежность прогнозирующей модели.

Тогда управление, формируемое на k -ом такте, запишется в виде:

$$u(k) = -D_d^{-1} B(k) S_k (x(k) - x_z(t_k)), \quad (8.3.29)$$

где S_k – решение уравнения Риккати, которое в силу описания модели объекта на интервале оптимизации с помощью стационарной прогнозирующей модели можно записать в прямом времени и решать итерационным методом:

$$S(i+1) = A^T(k)S(i) + S(i)A(k) - \\ - S(i)B(k)D_d^{-1}B^T(k)S(i) - S(i) + C_d, \quad S(0) = 0. \quad (8.3.30)$$

При этом матрица S_k определяется следующим образом:

$$S_k = \begin{cases} S(i), & \text{если } \frac{\|S(i+1) - S(i)\|}{\|S(i+1)\|} \leq \varepsilon, \\ S(l_p), & \text{в противном случае.} \end{cases} \quad (8.3.31)$$

Синтез управляющих воздействий по квадратичному критерию обобщенной работы

Для модели объекта (8.3.21) функционал обобщенной работы запишем для скользящего интервала оптимизации $[t_k, t_k + l_p \Delta t]$:

$$J(t_k) = \frac{1}{2} \int_{t_k}^{t_k + l_p \Delta t} [(x(t) - x_z(t_k))^T C(x(t) - x_z(t_k)) + \\ + u^T(t)D_2u(t) + v^T(t)D_1v(t) + v_{on}^T(t)D_1v_{on}(t)]dt, \quad (8.3.32)$$

и будем описывать поведение объекта на интервале оптимизации с помощью стационарной прогнозирующей модели, характеризующей «свободное» движение объекта ($v(t) = 0$):

$$x_M(j+1) = A(k)x_M(j) + B(k)u(k), \quad x_M(j=k) = x(k), \\ j = k, k+1, \dots, k+l_p-1. \quad (8.3.33)$$

Тогда управление будет иметь вид:

$$v(k) = -D_1^{-1}W_2(k), \quad (8.3.34)$$

где $W_2(k)$ определяется решением в обратном времени системы обыкновенных дифференциальных уравнений (8.2.44) на интервале оптимизации $[t_k, t_k + l_p \Delta t]$. Так как решение этой системы зависит от значений состояния прогнозирующей модели на интервале $[t_k, t_k + l_p \Delta t]$, то сначала, решая в прямом времени уравнение (8.3.33), определяется состояние прогнозирующей модели в конечный момент интервала оптимизации $x_M(k + l_p)$, а затем $W_2(k)$ определяется решением в обратном времени системы разностных уравнений вида:

$$\begin{aligned} x'_M(k + l_p - (j + 1)) &= 2x'_M(k + l_p - j) - \\ &- A(k)x'_M(k + l_p - j) - B(k)u(k), \quad x'_M(k + l_p) = x_M(k + l_p), \\ W_1(k + l_p - (j + 1)) &= A^T(k)W_1(k + l_p - j) + \\ &+ \Delta t C(x'_M(k + l_p - j) - x_z(t_k)), \quad W_1(k + l_p) = 0, \\ W_2(k + l_p - (j + 1)) &= W_2(k + l_p - j) + B^T(k)W_1(k + l_p - j) + \\ &+ \Delta t D_2 u(k), \quad W_2(k + l_p) = 0, \end{aligned} \quad (8.3.35)$$

где первое уравнение описывает движение прогнозирующей модели (8.3.33) в обратном времени.

Синтез управляющих воздействий по локальному квадратичному критерию

Для модели объекта (8.3.20) локальный квадратичный критерий запишется следующим образом:

$$\begin{aligned} J(k) = \frac{1}{2}[(x(k + 1) - x_z(k))^T C(x(k + 1) - x_z(t_k)) + \\ + u^T(k)Du(k)]. \end{aligned} \quad (8.3.36)$$

Управление, формируемое на основе минимизации этого критерия, имеет вид:

$$u(k) = -(B^T(k)CB(k) + D)^{-1}B^T(k)C[A(k)x(k) - x_z(t_k)]. \quad (8.3.37)$$

Качество функционирования системы уравнения зависит от весовых матриц рассматриваемых критериев, определение которых осуществляется на этапе предварительного проектирования

путем коррекции элементов весовых матриц и анализа получаемых при этом переходных процессов и управлений.

Для выбора весовых матриц C и D локального критерия (8.3.36) можно воспользоваться методом случайного поиска или назначить их исходя из каких-либо конструктивных соображений с учетом свойств управляемого объекта. Достаточно хорошо работает следующая методика.

Запишем для (8.3.20) систему вида:

$$x(k+1) = \tilde{A}x(k) + \tilde{B}u(k), \quad x(0) = x_0, \quad (8.3.38)$$

где \tilde{A}, \tilde{B} – некоторые постоянные матрицы, например, $\tilde{A} = A(0)$, $\tilde{B} = B(0)$. Используя принцип равного вклада максимальных отклонений, выбираются весовые матрицы для суммарного критерия:

$$\sum_{k=0}^{\infty} [x^T(k+1)C_1x(k+1) + u^T(k)D_1u(k)] \quad (8.3.39)$$

в предположении, что требования к качеству функционирования системы управления для всех k задаются в виде неравенств для компонент векторов состояния и управления:

$$\begin{aligned} |x_i(k)| &\leq x_{\max_i}, \quad \overline{i = 1, n}, \\ |u_i(k)| &\leq u_{\max_i}, \quad \overline{i = 1, m}. \end{aligned}$$

Тогда полагают

$$\begin{aligned} C_1 &= \text{diag}\{x_{\max_1}^{-2}, \dots, x_{\max_n}^{-2}\}, \\ D_1 &= \text{diag}\{u_{\max_1}^{-2}, \dots, u_{\max_m}^{-2}\}, \end{aligned}$$

где $\text{diag}\{\dots\}$ – диагональная матрица соответствующей размерности с указанными элементами на главной диагонали. Весовые матрицы C и D локального критерия полагаются равными:

$$C = \tilde{A}^T C \tilde{A} - \tilde{A}^T C \tilde{B} (\tilde{B}^T C \tilde{B} + D_1)^{-1} \tilde{B}^T C \tilde{A} + C_1, \quad (8.3.40)$$

$$D = D_1. \quad (8.3.41)$$

Алгебраическое уравнение Риккати (8.3.40) можно решить по итерационной схеме:

$$\begin{aligned} C(i+1) &= [\tilde{A}^T C(i) \tilde{A} - \tilde{A}^T C(i) \tilde{B} (\tilde{B}^T C(i) \tilde{B} + D)^{-1} \tilde{B}^T C(i) \tilde{A} + \\ &+ C_1 - C(i)] \Delta\eta + C(i), \quad C(0) = C_1, \end{aligned} \quad (8.3.42)$$

где $\Delta\eta = \Delta t \eta$ ($\eta \geq 0$, целое) – ускоренное время и при выполнении условия:

$$\frac{\|C(i+1) - C(i)\|}{\|C(i+1)\|} \leq \varepsilon \quad (8.3.43)$$

матрица C полагается равной $C(i+1)$.

Структурная схема совмещенного синтеза цифрового управления объектом представлена на рис. 8.3.2.

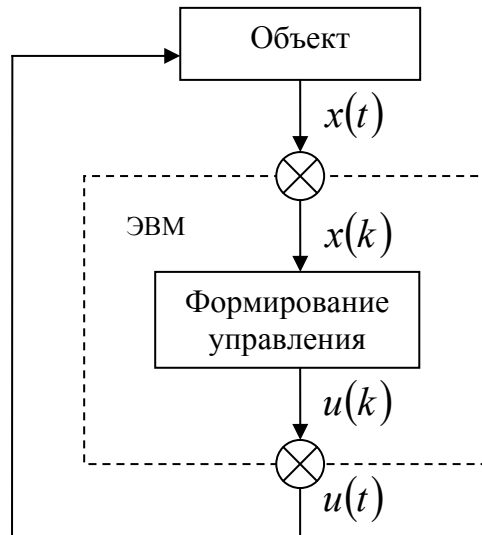


Рис. 8.3.2

8.4 Моделирование систем управления при случайных внешних воздействиях

При синтезе управляющих воздействий в реальных условиях информация о состоянии объекта поступает с измерительного комплекса, который достаточно часто содержит неполную информацию об объекте, искаженную ошибками измерений. Кроме того, при описании модели объекта необходимо учитывать влияние внешних возмущений. В связи с этим будем рассматривать алгоритмы синтеза управляющих воздействий, учитывающие условия их реализации.

8.4.1 Моделирование поведения объекта при наличии внешних возмущений

Будем предполагать, что внешние возмущения являются аддитивными и описываются вектором гауссовских шумов с задан-

ной матрицей влияния $\bar{F}(t)$. Тогда модель объекта можно записать в виде:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)u(t) + \bar{F}(t)q(t), \quad x(t_0) = x_0, \quad (8.4.1)$$

где $q(t)$ – l_1 -мерный вектор гауссовских шумов с заданными характеристиками:

$$\begin{aligned} M\{q(t)\} &= \bar{q}(t), \\ M\{(q(t) - \bar{q}(t))(q(\tau) - \bar{q}(\tau))^T\} &= Q(t)\delta(t - \tau), \end{aligned}$$

где $\delta(t - \tau)$ – дельта-функция Дирака, $\bar{F}(t)$ – матрица влияния внешних возмущений размерности $n \times l_1$.

Для моделирования поведения объекта строится разностный аналог уравнения (8.4.1) с помощью фундаментальных матриц соответствующей однородной системы и в предположении, что $q(t)$ является кусочно-постоянной функцией

$$q(t) = q(t_k), \quad t_k \leq t < t_{k+1}.$$

Тогда разностное уравнение, соответствующее (8.4.1), имеет вид:

$$x(k+1) = A(k)x(k) + B(k)u(k) + F(k)q(k), \quad x(0) = x_0, \quad (8.4.2)$$

где $A(k)$, $B(k)$ и $F(k)$ вычисляются следующим образом:

$$A(k) = I_n + \sum_{i=1}^{\infty} \frac{\Delta t^i \bar{A}^i(t_k)}{i!}, \quad (8.4.3)$$

$$B(k) = \sum_{i=1}^{\infty} \frac{\Delta t^i \bar{A}^{i-1}(t_k) \bar{B}(t_k)}{(i-1)!}, \quad (8.4.4)$$

$$F(k) = \sum_{i=1}^{\infty} \frac{\Delta t^i \bar{A}^{i-1}(t_k) \bar{F}(t_k)}{(i-1)!}. \quad (8.4.5)$$

В (8.4.2) $q(k)$ – последовательность гауссовских величин, при этом

$$\begin{aligned} M\{q(k)\} &= \bar{q}(k) = \bar{q}(t_k), \\ M\{(q(k) - \bar{q}(k))(q(j) - \bar{q}(j))^T\} &= Q(k)\delta_{kj}, \end{aligned}$$

$$Q(k) = \int_{t_k}^{t_{k+1}} \int_{t_k}^{t_{k+1}} M\{(q(t) - \bar{q}(t))(q(\tau) - \bar{q}(\tau))^T\} dt d\tau = Q(t_k)\Delta t,$$

где δ_{kj} – символ Кронекера.

Для того, чтобы внешние возмущения $q(t)$ и $q(k)$ имели одни и те же характеристики, матрицу $F(k)$ представим в виде:

$$F(k) = \sqrt{\Delta t} \sum_{i=1}^{\infty} \frac{\Delta t^{i-1} \bar{A}^{i-1}(t_k) \bar{F}(t_k)}{(i-1)!}. \quad (8.4.6)$$

Если для моделирования используется метод Эйлера, то

$$F(k) = \sqrt{\Delta t} \bar{F}(t_k),$$

а при использовании метода Рунге-Кутты в (8.4.6) в сумме остаются только четыре слагаемых.

Если рассматривается задача управления скоростью отклонения органов управления, то есть

$$\begin{aligned} \dot{x}(t) &= \bar{A}(t)x(t) + \bar{B}(t)u(t) + \bar{F}(t)q(t), \quad x(t_0) = x_0, \\ \dot{u}(t) &= v(t), \quad u(t_0) = u_0, \end{aligned} \quad (8.4.7)$$

то эквивалентная дискретная система имеет вид:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + B_1(k)v(k) + F(k)q(k), \quad x(0) = x_0, \\ u(k+1) &= u(k) + \Delta t v(k), \quad u(0) = u_0, \end{aligned} \quad (8.4.8)$$

где $A(k)$, $B(k)$, $F(k)$ вычисляются согласно (8.4.3), (8.4.4), (8.4.5), а

$$B_1(k) = \sum_{i=2}^{\infty} \frac{\Delta t^i \bar{A}^{i-2}(t_k) \bar{B}(t_k)}{(i-1)!}. \quad (8.4.9)$$

Если для построения дискретной системы используется метод Эйлера, то (8.4.7) имеет вид:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + F(k)q(k), \quad x(0) = x_0, \\ u(k+1) &= u(k) + \Delta t v(k), \quad u(0) = u_0. \end{aligned} \quad (8.4.10)$$

При использовании метода Эйлера для построения дискретных систем (8.4.2) и (8.4.10) матрицы $A(k)$, $B(k)$ и $F(k)$ имеют вид:

$$A(k) = I_n + \Delta t \bar{A}(t_k), \quad B(k) = \Delta t \bar{B}(t_k), \quad F(k) = \sqrt{\Delta t} \bar{F}(t_k). \quad (8.4.11)$$

8.4.2 Описание математической модели измерительного комплекса

Использование компьютеров для формирования управляющих воздействий приводит к тому, что измерения поступают в дискретные моменты времени t_1, t_2, \dots , а управления являются ку-

сочно-постоянными непрерывными слева функциями на каждом интервале выдачи управляющих воздействий, то есть

$$u(t) = u(k), \quad t_k \leq t < t_{k+1}, \quad t_{k+1} - t_k = \Delta t.$$

Будем предполагать, что математическая модель измерительного комплекса имеет вид:

$$y(k) = Hx(k) + r(k), \quad (8.4.12)$$

где $y(k)$ – l -мерный вектор измерений, H – матрица канала измерений размерности $l \times n$, состоящая из нулей и единиц, нулевые столбцы которой соответствуют неизмеряемым компонентам вектора состояния, $r(k)$ – дискретный гауссовский шум с характеристиками:

$$M\{r(k)\} = 0, \quad M\{r(k)r^T(j)\} = R_{\delta_{kj}},$$

$x(k)$ – состояние преобразованной в дискретную форму непрерывной системы, описывающей модель объекта.

8.4.3 Оценивание состояния модели объекта

Так как в реальных условиях управление объектом осуществляется по результатам измерений, а все измерения осуществляются с погрешностью, и, кроме того, в модели объекта могут существовать компоненты вектора состояния, недоступные измерению, то для формирования управляющих воздействий необходимо осуществлять оценивание вектора состояния. Так как информация о модели объекта поступает с измерительного комплекса в дискретные моменты времени, то будем строить оценку $\hat{x}(k)$ вектора состояния $x(k)$ по результатам текущих измерений $y(k)$.

Основные подходы к построению оценок векторов состояния состоят в следующем. Предположим, что для некоторой системы состояние $x(k)$ недоступно непосредственному измерению, а получены последовательные измерения $y(1), y(2), \dots, y(j)$. Обозначим оценку $\hat{x}(k)$, полученную на основе этих измерений, через $\hat{x}(k)$ и определим ее как n -мерную вектор-функцию измерений

$$\hat{x}(k/j) = \varphi_k \{y(i), \quad i = \overline{1, j}\}.$$

Задача построения оценки для состояния $x(k)$ представляет собой задачу определения функции φ_k некоторым рациональным и обоснованным способом.

Если $k > j$, то задача построения оценки называется *задачей предсказания*, если $k = j$, то такая задача называется *задачей фильтрации*, если $k < j$, то получаем *задачу сглаживания* или *интерполяции*.

Рассмотрим функцию ошибки оценки

$$\tilde{x}(k/j) = x(k) - \hat{x}(k/j)$$

и обозначим через $L = L[\tilde{x}(k/j)]$ *функцию потерь* или *штрафов*. Эта функция должна обладать следующими свойствами:

- 1) быть скалярной функцией;
- 2) $L(0) = 0$;
- 3) быть неубывающей функцией расстояния от начала координат;
- 4) быть симметричной относительно нуля.

Функция L , обладающая такими свойствами, называется *допустимой функцией потерь*.

Определим критерий качества оценивания как среднее значение функции потерь:

$$J[\tilde{x}(k/j)] = M\{L[\tilde{x}(k/j)]\}.$$

Говорят, что оценка $\hat{x}(k/j)$, минимизирующая $J[\tilde{x}(k/j)]$, является «*наилучшей*» или *оптимальной*. Заметим, что оптимальная оценка минимизирует среднее значение функции потерь, то есть является «*оптимальной в среднем*».

Пусть

$$L[\tilde{x}(k/j)] = \tilde{x}^T(k/j)\tilde{x}(k/j).$$

Соответствующий этой функции потерь критерий качества оценки называется *среднеквадратической ошибкой*, так как $M\{\tilde{x}^T(k/j)\tilde{x}(k/j)\}$ есть среднее значение квадрата евклидовой меры вектора ошибки.

Для построения оценки вектора состояния используются различные подходы. Будем строить рекуррентный алгоритм статистической обработки типа фильтра Калмана, осуществляющий оценивание состояния по результатам текущих измерений.

Пусть математические модели объекта и канала измерений имеют вид:

$$x(k+1) = A(k)x(k) + B(k)u(k) + F(k)q(k), \quad x(0) = x_0, \quad (8.4.13)$$

$$y(k+1) = Hx(k+1) + r(k), \quad (8.4.14)$$

где $q(k)$ и $r(k)$ – независимые гауссовские последовательности с характеристиками:

$$\begin{aligned} M\{q(k)\} &= \bar{q}(k), \quad M\{(q(k) - \bar{q}(k))(q(j) - \bar{q}(j))^T\} = Q\delta_{kj}, \\ M\{r(k)\} &= 0, \quad M\{r(k)r^T(j)\} = R\delta_{kj}, \end{aligned} \quad (8.4.15)$$

$$M\{q(k)r^T(j)\} = 0.$$

Кроме того, предполагается, что априорное распределение вектора x_0 является гауссовским:

$$M\{x_0\} = \bar{x}_0, \quad M\{(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)^T\} = P_{x_0}, \quad (8.4.16)$$

где P_{x_0} – дисперсионная матрица ошибок оценивания начального состояния.

Обозначим оценку фильтрации, полученную по результатам измерений $y(1), y(2), \dots, y(k)$ через $\hat{x}(k)$. При принятых предположениях задача оптимальной фильтрации может быть сформулирована следующим образом. Располагая последовательностью векторов измерений $y(1), y(2), \dots, y(k)$, моделью которых является система уравнений (8.4.13), (8.4.14), требуется определить оценку $\hat{x}(k)$ вектора состояния $x(k)$, удовлетворяющую требованиям не-смещенности

$$M\{\hat{x}(k)\} = M\{x(k)\} \quad (8.4.17)$$

при минимальной дисперсии ошибок оценки.

Обозначим

$$\varepsilon(k) = \hat{x}(k) - x(k) \quad (8.4.18)$$

ошибку оценки и

$$P(k) = M\{\varepsilon(k)\varepsilon^T(k)\} \quad (8.4.19)$$

дисперсионную матрицу ошибки оценки.

Тогда критерий оптимальности запишется в виде:

$$J(k) = M\{\varepsilon^T(k)\varepsilon(k)\} = M\{tr(\varepsilon(k)\varepsilon^T(k))\} = tr(P(k)), \quad (8.4.19)$$

где $tr(\cdot)$ – обозначает след матрицы.

Будем строить оптимальную оценку $\hat{x}(k+1)$ по текущему измерению $y(k+1)$ в виде:

$$\hat{x}(k+1) = \hat{x}(k/k+1) + K(k)[y(k+1) - H\hat{x}(k/k+1)], \quad (8.4.20)$$

где вектор

$$\hat{x}(k/k+1) = A(k)\hat{x}(k) + B(k)u(k) + F(k)\bar{q}(k), \quad \hat{x}(0) = \bar{x}_0 \quad (8.4.21)$$

является экстраполированной оценкой вектора состояния $x(k+1)$, $K(k)$ – матрица коэффициентов усиления фильтра.

Таким образом, задача построения оптимальной оценки дискретной системы (8.4.13) сводится к задаче определения такой матрицы $K(k)$, при которой оценка $\hat{x}(k+1)$ удовлетворяет условиям (8.4.17) при ограничениях (8.4.13)–(8.4.14), (8.4.20)–(8.4.21) и обеспечивает минимум критерия (8.4.19).

Заметим, что выражение (8.4.20) объединяет множество фильтров, различающихся законом изменения матриц коэффициентов усиления, и характеризует рекуррентную форму выработки оценок состояния.

Получим уравнение для вектора ошибки оценки $\varepsilon(k+1)$. Для этого подставим в (8.4.20) выражение (8.4.21) и вычтем (8.4.13) с учетом (8.4.14). Тогда (в дальнейшем для сокращения записи опустим аргумент k везде, кроме векторов ошибок и дисперсионных матриц):

$$\begin{aligned} \varepsilon(k+1) = (A - KHA)\varepsilon(k)(A - KHA)^T - \\ - (I - KH)F\bar{q} + (KH - I)Fq + Kr \end{aligned} \quad (8.4.22)$$

и дисперсионная матрица ошибки оценки $\varepsilon(k+1)$ будет равна:

$$\begin{aligned} P(k+1) = (A - KHA)P(k)(A - KHA)^T - \\ - (KH - I)FQF^T(KH - I)^T + KRK^T, \end{aligned} \quad (8.4.23)$$

где I – единичная матрица соответствующего порядка.

В соответствии с критерием оптимальности (8.4.19) матрица усиления фильтра K определяется из условия:

$$\frac{dJ(k)}{dK} = \frac{d \operatorname{tr} P(k+1)}{dK} = 0, \quad (8.4.24)$$

или

$$\begin{aligned} -2AP(k)A^T H^T + 2KHAP(k)A^T H^T - 2FQF^T H^T + \\ + 2KHFQF^T H^T + 2KR = 0. \end{aligned} \quad (8.4.25)$$

Из (8.4.25), после приведения подобных членов, получается:

$$\begin{aligned} KH(AP(k)A^T + FQF^T)H^T + KR = \\ = (AP(k)A^T + FQF^T)H^T. \end{aligned} \quad (8.4.26)$$

Обозначим

$$P(k+1/k) = AP(k)A^T + FQF^T \quad (8.4.27)$$

прогноз дисперсионной матрицы на один шаг вперед. Тогда выражение (8.4.26) с учетом (8.4.27) примет вид:

$$KHP(k+1/k)H^T + KR = P(k+1/k)H^T \quad (8.4.28)$$

и для определения матрицы K получается выражение:

$$K = P(k+1/k)H^T (HP(k+1/k)H^T + R)^{-1}. \quad (8.4.29)$$

Выражение для дисперсионной матрицы $P(k+1)$ можно упростить, если в (8.4.23) подставить (8.4.29) и учесть (8.4.27). Тогда получим

$$P(k+1) = (I - KH)P(k+1/k).$$

Таким образом, построение оценки $\hat{x}(k+1)$ по результатам измерений $y(k+1)$ осуществляется с помощью следующего рекуррентного алгоритма:

$$\begin{aligned} \hat{x}(k+1) &= \hat{x}(k+1/k) + K(k)[y(k+1) - H\hat{x}(k+1/k)], \\ \hat{x}(k+1/k) &= A(k)\hat{x}(k) + B(k)u(k) + F(k)\bar{q}(k), \hat{x}(0) = \bar{x}_0, \\ K(k) &= P(k+1/k)H^T [HP(k+1/k)H^T + R]^{-1}, \\ P(k+1/k) &= A(k)P(k)A^T(k) + F(k)QF^T(k), \\ P(k+1) &= [I - K(k)H]P(k+1/k), \\ P(0) &= P_{x_0}. \end{aligned} \quad (8.4.30)$$

8.4.4 Синтез управляющих воздействий по оценкам состояния

Для классического критерия оптимальности (8.3.24) и критерия обобщенной работы (8.3.30) доказаны теоремы разделения, согласно которым для линейного управляемого объекта, математическая модель которого описывается уравнениями типа (8.4.2) или (8.4.6), при линейном наблюдении с аддитивным белым гауссовским шумом, минимизация математического ожидания функ-

ционалов (8.3.24) или (8.3.30) приводит к системе, состоящей из фильтра Калмана (8.4.30) и алгоритма управления, структура которого совпадает с законом оптимального управления, построенным для детерминированной полностью известной модели объекта.

Таким образом, управление для модели объекта (8.4.1), синтезируемое на основе минимизации математического ожидания классического квадратичного функционала, будет формироваться по оценкам состояния, то есть вместо значений вектора состояния $x(k)$ в соответствующих формулах будет использоваться его оценка $\hat{x}(k)$. Аналогичная ситуация складывается при формировании управления на основе минимизации математического ожидания функционала обобщенной работы (8.3.22). В этом случае оценка состояния $\hat{x}(k)$ заменит вектор состояния в уравнении прогнозирующей модели, в которой будем использовать постоянные внешние возмущения $\bar{q}(k)$, равные математическому ожиданию шума в модели объекта в начальный момент интервала оптимизации:

$$\begin{aligned} x_M(j+1) &= A(k)x_M(j) + B(k)u(k) + F(k)\bar{q}(k), \\ x_M(j=k) &= \hat{x}(k), \quad j = k, k+1, \dots, k+l_p-1. \end{aligned} \quad (8.4.31)$$

Постоянные внешние возмущения $\bar{q}(k)$ должны учитываться и в уравнении, описывающем движение прогнозирующей модели в обратном времени, то есть первое уравнение в системе (8.3.35) должно иметь вид:

$$\begin{aligned} x'_M(k+l_p-(j+1)) &= 2x'_M(k+l_p-j) - \\ &- A(k)x'_M(k+l_p-j) - B(k)u(k) - F(k)\bar{q}(k), \\ x'_M(k+l_p) &= x_M(k+l_p). \end{aligned} \quad (8.4.32)$$

При формировании управления для модели объекта вида:

$$x(k+1) = A(k)x(k) + B(k)u(k) + F(k)q(k), \quad x(0) = x_0 \quad (8.4.33)$$

запишем математическое ожидание для локального критерия оптимизации, то есть

$$J(k) = \frac{1}{2} M \left\{ \left(x(k+1) - x_z(t_k) \right)^T C \left(x(k+1) - x_z(t_k) + u^T(k) D u(k) \right) \right\}. \quad (8.4.34)$$

Для того, чтобы найти оптимальное управление по критерию (8.4.34), запишем значение этого критерия, учитывая урав-

нение для модели объекта (8.4.33) и свойства операции $tr(\cdot)$ (для сокращения записи аргумент k опустим):

$$J(k) = \frac{1}{2} [tr(CFQF^T) + u^T (B^T CB + D)u + (Ax + Fq - x_z)^T CBu + u^T B^T C(Ax + F\bar{q} - x_z)]. \quad (8.4.35)$$

Оптимальное управление определим из условия:

$$\frac{dJ(k)}{du} = (B^T CB + D)u + B^T C(Ax + F\bar{q} - x_z) = 0. \quad (8.4.36)$$

Тогда получим:

$$u(k) = -(B^T(k)CB(k) + D)^{-1}B^T(k)C[A(k)x(k) + F(k)\bar{q}(k) - x_z(t_k)]. \quad (8.4.37)$$

Структурная схема совмещенного синтеза управляющих воздействий по оценкам состояния приведена на рис. 8.4.1.

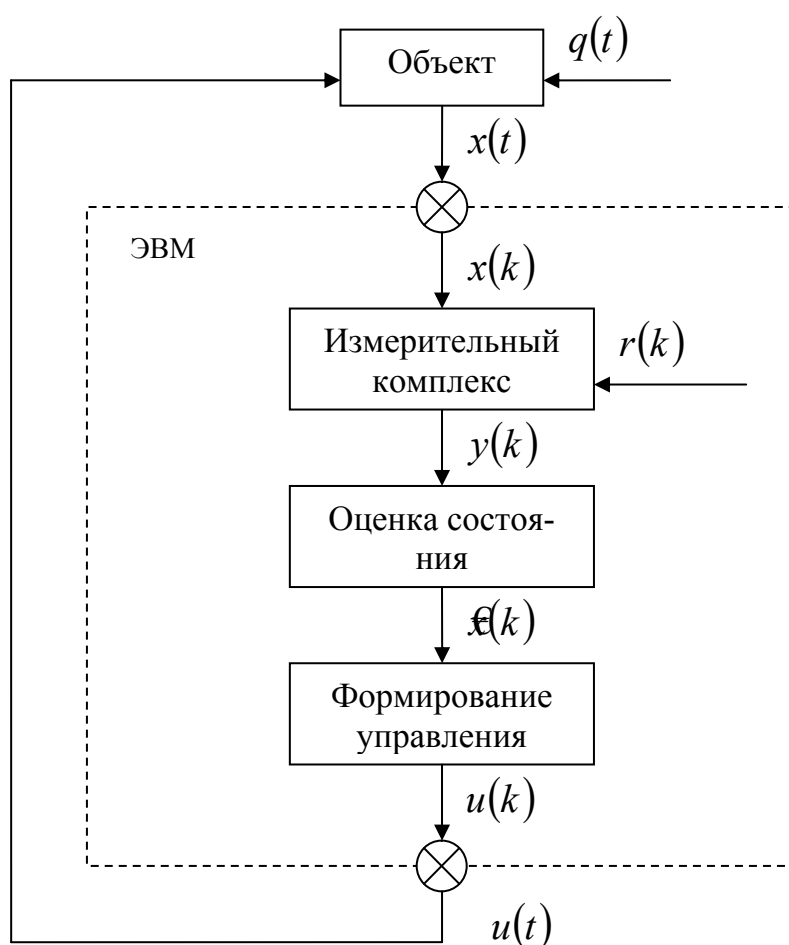


Рис. 8.4.1

8.5 Синтез адаптивной следящей системы

Процесс проектирования систем управления всегда предполагает наличие:

- 1) четко сформулированной цели управления;
- 2) априорной информации об объекте управления и о характере действующих на него возмущений.

Объем априорной информации при этом может быть различным и за редким исключением не является исчерпывающим. Однако в данном случае принципиальным является вопрос о достаточности или недостаточности имеющейся априорной информации об объекте для достижения сформулированной цели управления. Все системы управления, построенные с использованием априорной информации, достаточной для достижения поставленной цели, относятся к *неадаптивным* или *традиционным* системам управления, независимо от реализуемого принципа управления, наличия обратной связи, случайности или детерминированности возмущений, используемых вычислительных средств и т.д. Если же объем располагаемой априорной информации о свойствах объекта не может обеспечить достижения сформулированной цели управления, то речь идет об *адаптивных* системах управления.

Существует достаточно много вариантов различных определений адаптивных систем. Но основной смысл при этом остается неизменным. Одним из наиболее известных является определение Цыпкина Я.З.

Определение. Под адаптивными системами управления будем понимать такие системы, которые, используя информацию о внешних воздействиях, условиях работы и выходных величинах, изменяют структуру или параметры регулятора с целью обеспечения оптимального или заданного функционирования управляемой системы при изменяющихся условиях ее работы.

Достаточно известным является определение, данное Буковым В.Н.

Определение. Адаптивными называются такие системы управления, которые предназначены для функционирования в условиях априорной неопределенности и которые в процессе функционирования автоматически приспосабливаются к непред-

виденным изменениям свойств объекта управления и внешней среды.

Если изменяется структура регулятора, то адаптивная система управления называется *самоорганизующейся*, если структура остается неизменной, но изменяются параметры регулятора, то система управления называется *самонастраивающейся с параметрической адаптацией*.

8.5.1 Основные понятия адаптивных систем управления

Наиболее существенными являются следующие признаки деления принципов адаптации:

1. По уровню априорной неопределенности:

а) параметрическая адаптация, при которой априорная неопределенность заключается в недостаточном знании параметров (коэффициентов) управляемого объекта;

б) непараметрическая адаптация, при которой априорная неопределенность не связана непосредственно с какими-либо параметрами.

В обоих случаях неопределенность уменьшается на основе последовательных наблюдений входных и выходных данных в процессе управления.

2. По организации процесса адаптации:

а) поисковые, для которых характерны процессы итеративного движения к достижению требуемого качества управления;

б) беспоисковые, основанные на использовании некоторых необходимых (достаточных) условий требуемого качества управления.

У систем поисковой адаптации формируются специальными устройствами детерминированные или случайные пробные сигналы или создаются условия для возбуждения в объекте незатухающих колебаний, используемые как поисковые. Наличие пробных движений является основным недостатком поисковой адаптации, так как они не всегда допустимы по условиям функционирования объекта.

3. По целям организации адаптации:

а) системы со специальными свойствами, в результате функционирования которых управляемый процесс приобретает

некоторые обязательные свойства, в число которых могут быть включены устойчивость, чувствительность к каким-либо возмущениям или ошибкам априорной информации, заданное расположение корней характеристического уравнения и т.д.;

б) оптимальные системы, обеспечивающие минимизацию некоторых функционалов, отражающих качество управляемого движения.

Наиболее перспективными для создания цифровых систем управления являются оптимальные беспойсковые системы с параметрической адаптацией, в которых на основе полученной в процессе функционирования объекта информации о самом объекте и возмущающих воздействиях осуществляется автоматическая настройка параметров модели объекта. При этом используется «принцип разделения», согласно которому формирование адаптивного управления состоит из трех этапов:

1) выбор алгоритма управления в предположении, что параметры модели объекта известны точно (оптимальное управление);

2) оценивание неизвестных параметров (идентификация) и состояния (фильтрация) модели объекта;

3) формирование адаптивного управления посредством замены точных значений параметров и состояния на их оценки.

8.5.2 Одновременное оценивание состояния и параметров модели объекта

Для синтеза управлений в реальных условиях функционирования объекта необходимо учитывать как влияние внешних возмущений, так и наличие переменных во времени неизвестных параметров.

Пусть для формирования управляющих воздействий используется математическая модель объекта в виде:

$$x(k+1) = A(k, \theta(k))x(k) + B(k, \theta(k))u(k) + F(k)q(k), \quad x(0) = x_0, \quad (8.5.1)$$

где $\theta(k)$ – N -мерный вектор переменных во времени неизвестных параметров. При этом предполагается, что элементы матриц A и B линейно зависят от компонент вектора θ , что соответствует дискретизации по методу Эйлера, и априорные распределе-

ния векторов начальных условий $\theta(0)$ и x_0 являются гауссовскими:

$$\begin{aligned} M\{\theta(0)\} &= \bar{\theta}_0, \quad M\{(\theta(0) - \bar{\theta}_0)(\theta(0) - \bar{\theta}_0)^T\} = P_{\theta_0}, \\ M\{x_0\} &= \bar{x}_0, \quad M\{(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)^T\} = P_{x_0}. \end{aligned} \quad (8.5.2)$$

Здесь P_{θ_0} и P_{x_0} – ковариационные матрицы ошибок начальных условий векторов параметров и состояния модели объекта.

При совмещенном синтезе требуется одновременно оценивать состояние и параметры системы, описывающей поведение объекта в моменты квантования. Это обычно осуществляется с помощью рекуррентного алгоритма статистической обработки типа фильтра Калмана по результатам текущих измерений. При этом оценивается обобщенный вектор, формально объединяющий векторы состояния и параметров. Такой подход сопровождается резким возрастанием трудоемкости вычислений.

Алгоритмы оценивания параметров (идентификация) в реальном масштабе времени, используемые в адаптивных системах управления, строятся, как правило, на основе гипотезы квазистационарности характеристик объекта управления, то есть $\dot{\theta} = 0$. В соответствии с этой гипотезой, оцениваемые параметры или постоянны во времени, или изменяются с незначительной скоростью, пренебрежение которой практически не ухудшает оценок, получаемых на ограниченном временном интервале наблюдения.

Таким образом, можно записать дискретную модель изменения параметров в виде:

$$\theta(k+1) = \theta(k), \quad \theta(0) = \bar{\theta}_0. \quad (8.5.3)$$

Измерительный комплекс

$$y(k+1) = Hx(k+1) + r(k+1) \quad (8.5.4)$$

является общим для состояния и параметров модели объекта. Действительно

$$\begin{aligned} y(k+1) &= \\ &= HA(k, \theta(k))x(k) + HB(k, \theta(k))u(k) + HF(k)q(k) + r(k+1) = \\ &= H\Phi(x(k), u(k))\theta(k) + Hf(x(k), u(k)) + \tilde{q}(k), \end{aligned}$$

где матрица $\Phi(\cdot)$ размерности $N \times n$ и вектор-столбец $f(\cdot)$ с n элементами получаются в результате представления системы (8.5.1) следующим образом:

$$x(k+1) = \Phi(x(k), u(k))\theta(k) + f(x(k), u(k)) + F(k)q(k). \quad (8.5.5)$$

Канал измерений вектора параметров можно приближенно представить в виде:

$$y(k+1) = H\Phi(\mathbf{x}(k), u(k))\theta(k) + Hf(\mathbf{x}(k), u(k)) + \tilde{q}(k).$$

Будем строить оценку вектора параметров $\hat{\theta}(k+1)$ по текущему измерению $y(k+1)$ в виде:

$$\hat{\theta}(k+1) = \hat{\theta}(k) + L(k)[y(k+1) - H\Phi(\mathbf{x}(k), u(k)) - Hf(\mathbf{x}(k), u(k))], \quad (8.5.6)$$

где $L(k)$ – коэффициент усиления фильтра, выражение для которого можно получить аналогично тому, как был получен коэффициент усиления $K(k)$ фильтра для оценки состояния $\mathbf{x}(k+1)$.

Таким образом, оценивание состояния объекта и идентификацию его параметров можно осуществлять с помощью двух параллельно работающих дискретных фильтров Калмана. Рекуррентный алгоритм для оценки состояния будет иметь следующий вид:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k+1/k) + K(k)[y(k+1) - H\mathbf{x}(k+1/k)], \\ \mathbf{x}(k+1/k) &= A(k, \hat{\theta}(k))\mathbf{x}(k) + B(k, \hat{\theta}(k))u(k) + \\ &\quad + F(k)\bar{q}(k), \quad \mathbf{x}(0) = \bar{x}_0, \\ K(k) &= P_x(k+1/k)H^T[HP_x(k+1/k)H^T + R]^{-1}, \\ P_x(k+1/k) &= A(k, \hat{\theta}(k))P_x(k)A^T(k, \hat{\theta}(k)) + F(k)QF^T(k), \\ P_x(k+1) &= [I_n - K(k)H]P_x(k+1/k), \\ P_x(0) &= P_{x_0}. \end{aligned} \quad (8.5.7)$$

Идентификацию параметров будем осуществлять с помощью следующего фильтра Калмана:

$$\begin{aligned}
\hat{\theta}(k+1) &= \hat{\theta}(k) + L(k)[y(k+1) - H\Phi(\hat{\theta}(k), u(k))\hat{\theta}(k) - \\
&\quad - Hf(\hat{\theta}(k), u(k))], \quad \hat{\theta}(0) = \bar{\theta}_0, \\
L(k) &= P_\theta(k)\Phi^T(\hat{\theta}(k), u(k))M^{-1}, \\
M(k) &= H\Phi(\hat{\theta}(k), u(k))P_\theta(k)\Phi^T(\hat{\theta}(k), u(k))H^T + \\
&\quad + HF(K)QF^T(k)H^T + R, \\
P_\theta(k+1) &= [I_N - L(k)H\Phi(\hat{\theta}(k), u(k))]P_\theta(k), \\
P_\theta(0) &= P_{\theta_0}.
\end{aligned} \tag{8.5.8}$$

При постановке задачи идентификации параметров в большинстве случаев подразумевается, что приближение оценок параметров $\hat{\theta}$ к их истинным значениям θ обеспечивает приближение формируемого на основе оценок оптимального управления к искомому оптимальному управлению для реального объекта. Такое предположение подкрепляется теоремой разделения для предельной точности оценивания параметров и состояния объекта. Однако для случая ограниченной точности оценивания параметров эта теорема не дает конструктивных рекомендаций по организации совместного оценивания и оптимизации управления. В то же время опыт моделирования процессов адаптивного управления показывает не очень сильную связь между точностью оценок $\hat{\theta}$ и качеством поведения управляемого объекта. Поэтому необходимо использовать следующий подход к текущей идентификации, выполняемой при адаптивном управлении: идентификацию параметров осуществлять таким образом, чтобы в первую очередь достигался достаточно высокий уровень качества управления поведения объекта, и в меньшей степени преследовалась (или даже совсем не принималась во внимание) точность идентификации параметров объекта как таковая.

Структурная схема синтеза адаптивного управления представлена на рис. 8.5.1.

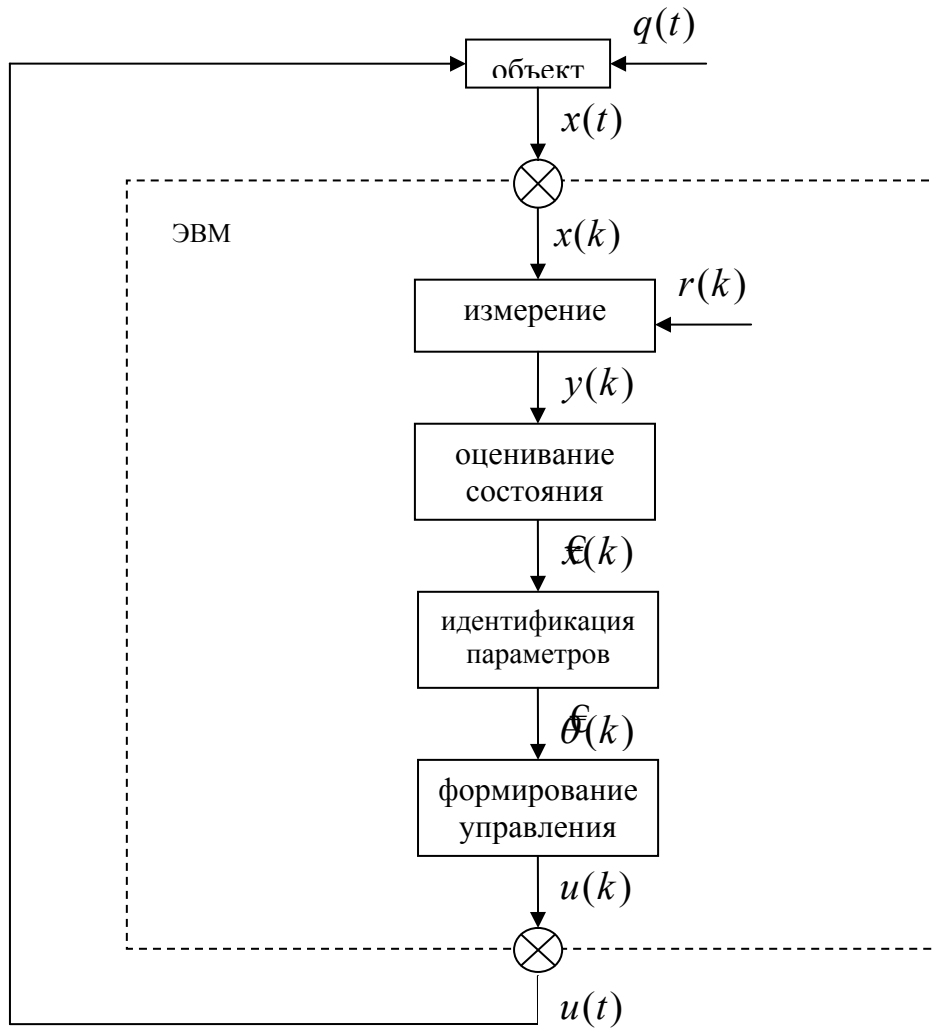


Рис. 8.5.1

8.6 Учет ограничений и запаздываний по управлению

При синтезе управлений на величину управляющего воздействия обычно накладываются ограничения. Достаточно часто эти ограничения накладываются в форме неравенств:

$$U \min_i \leq u_i(k) \leq U \max_i, \quad i = \overline{1, m}, \quad (8.6.1)$$

где $U \min_i, U \max_i$ – заданные величины (например, для автопилота в самолете $U \min_i, U \max_i$ определены по условию безопасности полета).

Тогда, учитывая ограничения, будем полагать:

$$\bar{u}_i(k) = \begin{cases} U \min_i, & \text{если } u_i(k) < U \min_i, \\ u_i(k), & \text{если } U \min_i \leq u_i(k) \leq U \max_i, \\ U \max_i, & \text{если } u_i(k) > U \max_i. \end{cases} \quad (8.6.2)$$

Включение ЭВМ в контур управления неизбежно создает особые условия его реализации, одним из основных является запаздывание подачи управляющего сигнала на приводы рулевых органов по отношению к моменту выдачи измерительной системой информации об объекте. Кроме того, запаздывание вызвано затратами времени на расчет оценок $\hat{x}(k)$ и $\hat{\theta}(k)$ и формирование управляющего сигнала $\bar{u}(k)$.

Управление с учетом запаздывания будем задавать уравнением:

$$\tilde{u}_i(k) = \bar{u}_i(k - l_z), \quad i = \overline{1, m}, \quad (8.6.3)$$

где $l_z \Delta t$ – время запаздывания.

8.7 Общая схема синтеза адаптивных систем управления

Проектирование любой системы управления должно осуществляться с учетом конкретных условий ее применения. В настоящей главе рассматривается проектирование системы адаптивного управления для совмещенного синтеза, т.е. для ситуации, когда управление формируется в процессе функционирования объекта. В этом случае большое значение приобретают алгоритмы и способы, позволяющие уменьшить вычислительную задержку, вызванную затратами времени на формирование управления. Одним из таких способов является введение дискретности решения задач оценивания состояния, идентификации параметров и формирования управляющих воздействий, т.е. $k_{оц}, k_{ид}, k_{упр}$. При этом $k_{оц} \geq k$, $k_{ид} \geq k$, $k_{упр} \geq k$ и $k_{оц} \geq k_{упр}$, $k_{ид} \geq k_{упр}$.

Тогда общая схема синтеза адаптивного управления будет иметь вид (рис. 8.7.1).

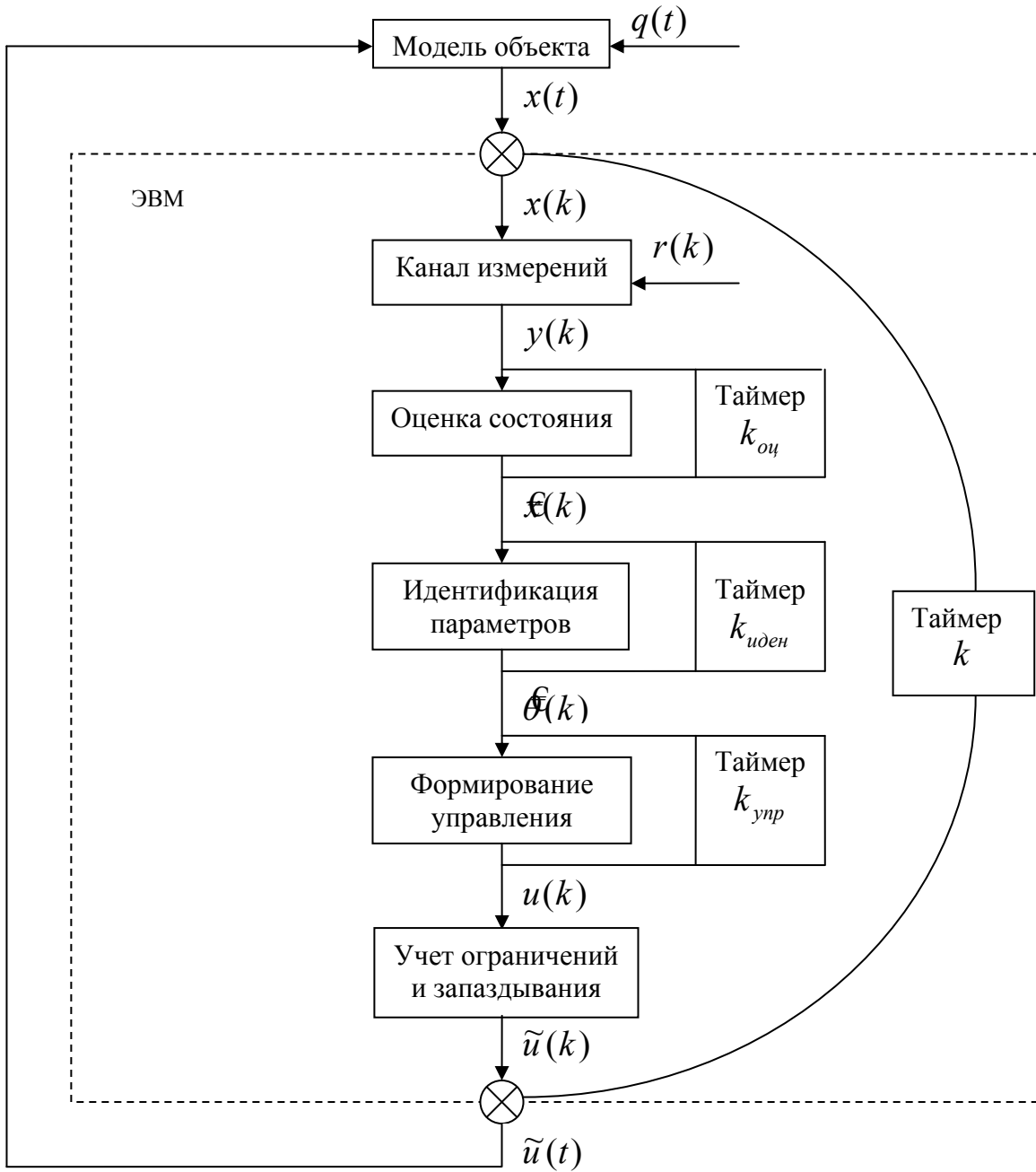


Рис. 8.7.1

При этом предполагается, что моменты и периоды квантования одинаковы для всех координат объекта и система управления формирует управляющее воздействие с моментом и периодом квантования, которые совпадают с моментом и периодом поступления информации об объекте. Кроме того, учет запаздывания осуществляется на время, кратное Δt .

Рассмотренные алгоритмы синтеза адаптивного управления все еще требуют достаточно высокой производительности управ-

ляющих ЭВМ, что ограничивает их применение в реальном времени функционирования объекта. Известны два подхода к преодолению этих трудностей.

Первый подход связан с построением для системы, описывающей движение объекта, системы пониженного порядка путем ее *агрегирования* и использования *централизованной* структуры управления. При этом исходная система заменяется при синтезе другой, меньшего порядка, которая отражает лишь некоторые существенные особенности исходной системы. Применение такой схемы управления приводит к субоптимальному функционированию реального объекта и является эффективным для слабо-связанных систем, или систем, имеющих быстрые и медленные составляющие.

Второй подход связан с разбиением системы управления на ряд подсистем, каждая из которых имеет меньшую размерность. Это приводит к построению распределенной *децентрализованной* или иерархической децентрализованной структуры управления. В последнем случае используются обычно два основных вида структур управления, основанных на многослойной и многоуровневой декомпозиции. При таком подходе трудоемким является построение структуры децентрализованного управления, что, в свою очередь, требует достаточно сложной программной реализации. Кроме того, существенное сокращение времени синтеза достигается, в основном, при параллельных вычислениях.

9 ПРИМЕРЫ ПОСТРОЕНИЯ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

К объектам управления, поведение которых описывается обыкновенными дифференциальными уравнениями, относятся все механические, электромеханические объекты, а также многие технологические процессы, подчиняющиеся законам механики и электричества. Кроме того, такими уравнениями можно описать и многие макро- и микроэкономические процессы.

При проектировании систем управления одной из основных задач, которую необходимо решить первой, – это построение математических моделей, описывающих управляемый объект или процесс. Качество функционирования системы управления будет существенно зависеть от того, насколько адекватной является построенная модель.

9.1 Построение математических моделей прямолинейного и вращательного движения

Основной закон механики заключается в том, что движение твердого тела состоит из движения его центра масс, которое подчиняется второму закону Ньютона:

$$m\ddot{x} = F, \quad (9.1.1)$$

и движения тела вокруг этого центра масс, которое описывается уравнением:

$$J\ddot{\varphi} = M, \quad (9.1.2)$$

где m – масса тела, J – момент инерции, F – силы, действующие на тело и приведенные к центру масс, M – момент сил, x – координаты центра масс, φ – углы положения тела.

9.1.1 Математическая модель прямолинейного движения

Примером прямолинейного движения может служить механическое колебательное звено, изображенное на рис. 9.1.1.

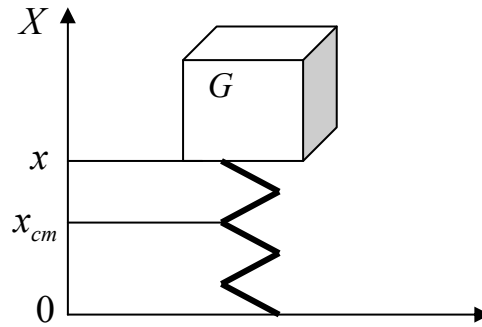


Рис. 9.1.1

Пусть тело G движется вдоль оси координат OX . На тело G действуют следующие силы:

1) вес

$$F_g = mg,$$

где m – масса тела, g – ускорение силы тяжести;

2) реакция пружины, которая пропорциональна отклонению размеров пружины от ненагруженного состояния, т.е.

$$F_{np} = -k_1 (x - x_0),$$

где x_0 – длина ненагруженной пружины, k_1 – коэффициент упругости ($k_1 > 0$);

3) реакция амортизатора, которая пропорциональна скорости движения штока, т.е.

$$F_{амор} = -k_2 \dot{x},$$

где k_2 – коэффициент ($k_2 > 0$), зависящий от вязкости жидкости, залитой в амортизатор.

Результирующая всех сил равна:

$$F = -mg - k_1(x - x_0) - k_2 \dot{x}. \quad (9.1.3)$$

Если ввести обозначения: $x_1 = x$, $x_2 = \dot{x}$, то уравнение (9.1.1) с учетом (9.1.3) можно переписать в виде системы уравнений, описывающей динамическую модель рассматриваемого процесса:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -g - \frac{k_1}{m}(x_1 - x_0) - \frac{k_2}{m}x_2. \end{aligned} \quad (9.1.4)$$

Стационарное решение системы можно найти, если приравнять правые части нулю. В результате получается *стационарная модель*:

$$x_{1cm} = x_0 - \frac{mg}{k_1},$$

$$x_{2cm} = 0.$$

Очевидно, что значение x_{1cm} должно быть положительным. Поэтому коэффициент упругости k_1 должен быть больше значения $\frac{mg}{x_0}$.

Если ввести разность $\Delta x_1 = x_1 - x_{1cm}$, которая характеризует отклонение координаты $x(t)$ от стационарного значения x_{1cm} , то согласно (9.1.4), эта разность удовлетворяет системе уравнений:

$$\Delta \dot{x}_1 = \Delta x_2,$$

$$\Delta \dot{x}_2 = -\frac{k_1}{m} \Delta x_1 - \frac{k_2}{m} \Delta x_2. \quad (9.1.5)$$

Система (9.1.5) является *моделью прямолинейного движения в отклонениях*.

9.1.2 Математическая модель вращательного движения

Многие технологические процессы связаны с вращением механических систем. Довольно часто задача сводится к необходимости подавать электрический ток нужной силы на вход электродвигателя, на валу которого насажена некоторая масса m . При этом необходимо задавать такую силу тока, чтобы обеспечить заданное вращательное движение массы. Примерами таких ситуаций может быть суточное слежение наземных антенн за спутниками связи, равномерное движение наматывающих барабанов в бумагоделательных машинах и т.п.

Математически вращательное движение массы описывается уравнением (9.1.2). Если обозначить через x_1 угол φ , а через x_2 – угловую скорость $\omega = \dot{\varphi}$, то это уравнение можно переписать в виде:

$$\begin{aligned}\dot{x}_1 &= x_2, \\ \dot{x}_2 &= \frac{M}{x}.\end{aligned}\tag{9.1.6}$$

Момент сил M складывается из следующих составляющих:

- 1) $k_1 u$ – сила вращения, создаваемая электродвигателем, на вход которого поступает ток силой u ;
- 2) $k_2 x_2$ – сила сухого трения, пропорциональная скорости вращения и обратная ей по знаку;
- 3) $f(t)$ – сила внешних нагрузок.

В результате уравнения вращательного движения, описывающие *динамическую модель* рассматриваемого процесса, примут вид:

$$\begin{aligned}\dot{x}_1 &= x_2, \\ \dot{x}_2 &= -\frac{k_2}{J} x_2 + \frac{k_1}{J} u + \frac{f(t)}{J}.\end{aligned}\tag{9.1.7}$$

Здесь k_1 и k_2 некоторые положительные коэффициенты.

9.2 Макроэкономическая модель динамики фондов производственного накопления и потребления

Пусть: $x_1(t)$ – фонд производственного накопления, $x_2(t)$ – фонд потребления.

Уравнение баланса имеет вид:

$$x_1(t) = b_1 \dot{x}_1(t) + b_2 \dot{x}_2(t),\tag{9.2.1}$$

где b_1 и b_2 – коэффициенты приростной капиталоемкости фондов накопления и потребления соответственно.

Пусть

$$L(t) = L_0 e^{\gamma t}$$

динамика изменения численности населения, L_0 – численность населения в начальный момент времени, γ – темп роста населения.

Тогда уравнения, характеризующие фонд потребления на душу населения и изменение душевого фонда потребления, будут соответственно равны:

$$\frac{x_2(t)}{L(t)} = \frac{x_2(t)}{L_0} e^{-\gamma t}$$

$$\frac{d}{dt} \left[\frac{x_2(t)}{L(t)} e^{-\gamma t} \right] = (\dot{x}_2(t) - \gamma x_2(t)) \frac{e^{-\gamma t}}{L_0}.$$

Если в качестве управления использовать скорость роста душевого фонда потребления

$$u(t) = \dot{x}_2(t) - \gamma x_2(t),$$

то

$$x_2(t) = \gamma x_2(t) + u(t). \quad (9.2.2)$$

Подставляя (9.2.2) в (9.1.1), получим:

$$x_1(t) = b_1 \dot{x}_1(t) + b_2 (\gamma x_2(t) + u(t)).$$

Таким образом, динамическая модель изменения фондов накопления и потребления описывается системой обыкновенных дифференциальных уравнений вида:

$$\dot{x}_1(t) = \frac{1}{b_1} x_1(t) - \gamma \frac{b_2}{b_1} x_2(t) - \frac{b_2}{b_1} u(t), \quad x_1(t_0) = x_1^{(0)}, \quad (9.2.3)$$

$$\dot{x}_2(t) = \gamma x_2(t) + u(t), \quad x_2(t_0) = x_2^{(0)}.$$

Заметим, что модель (9.2.3) является линейной по состоянию и управлению.

Если обозначить $x(t) = (x_1(t), x_2(t))^T$, $x^{(0)} = (x_1^{(0)}, x_2^{(0)})^T$, то система (9.2.3) в матричной форме запишется следующим образом:

$$\dot{x}(t) = \bar{A} x(t) + \bar{B} u(t), \quad x(t_0) = x^{(0)}, \quad (9.2.4)$$

где

$$\bar{A} = \begin{pmatrix} \frac{1}{b_1} & -\gamma \frac{b_2}{b_1} \\ 0 & \gamma \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} -\frac{b_2}{b_1} \\ 1 \end{pmatrix}.$$

9.3 Построение математических моделей производства, хранения и сбыта товара повседневного спроса

Введем следующие обозначения:

X – общее количество единиц товара, выпущенных предприятием;

u – темп производства;
 Z – количество товара на рынке;
 V – количество товара у потребителя (еще не потребленного);

W – доход;

Y – потенциальный спрос (полное количество товара, способное мгновенно удовлетворить спрос в условиях отсутствия ажиотажного спроса);

c – коэффициент, характеризующий превышение цены над себестоимостью ($c > 1$, так как себестоимость считается равной единице);

k_1 – коэффициент, характеризующий темп потребления;

k_2 – коэффициент, характеризующий плату за хранение товара;

$n(c)$ – коэффициент, характеризующий скорость продажи товара, например, $n(c) = n_0 e^{1-c}$ отражает распределение по достатку.

Параметры Y , k_1 , k_2 считаются постоянными. Величины u и c назначаются производителем и могут быть как постоянными, так и переменными. Переменные X , Z , V , W – функции времени, которые могут изменяться в соответствии с законами рынка.

Статическая модель описывает один цикл производства и сбыта: выработанный за расчетную единицу времени товар поступает на рынок и продается, в результате чего получается определенный доход. Статическая модель строится, исходя из предположения, что в равновесном состоянии система характеризуется двумя соотношениями баланса:

1) поступившее на рынок количество товара равно проданному;

2) сколько товара продано, столько и потреблено.

Эти соотношения с учетом введенных обозначений запишутся следующим образом: за единицу времени прирост товара на рынке составляет u единиц; количество товара на рынке – Z единиц; у потребителя – V единиц; в единицу времени продается $n(c)(Y - V)Z$ единиц товара, где $(Y - V)$ текущий спрос. При этом скорость продажи пропорциональна спросу и количеству товара

на рынке. Тогда статическая модель процесса производства, хранения и сбыта товара запишется в виде:

$$\begin{aligned} u - n(c)(Y - V)Z &= 0, \\ n(c)(Y - V)Z - k_1 V &= 0, \\ W &= c n(c)(Y - V)Z - \frac{u}{c} - k_2 Z. \end{aligned} \quad (9.3.1)$$

Последнее уравнение показывает, что доход в единицу времени складывается из выручки от продаж $n(c)(Y - V)Z$ единиц товара, расходов на производство и затрат на хранение.

Динамическая модель строится следующим образом. Так как за время dt прирост произведенного товара равен $u(t)dt$, а продается $n(c)(Y - V(t))Z(t)dt$ единиц товара, то скорость прироста товара на рынке равна:

$$\frac{dZ(t)}{dt} = u(t) - n(c)(Y - V(t))Z(t).$$

Прирост товара у потребителя за время dt равен количеству проданного товара минус количество потребленного, т.е. скорость прироста товара у потребителя описывается уравнением:

$$\frac{dV(t)}{dt} = n(c)(Y - V(t))Z(t) - k_1 V(t).$$

Таким образом, динамическая модель, описывающая процесс производства, хранения и сбыта, имеет вид:

$$\begin{aligned} \frac{dX(t)}{dt} &= u(t), \\ \frac{dZ(t)}{dt} &= u(t) - n(c)(Y - V(t))Z(t), \\ \frac{dV(t)}{dt} &= n(c)(Y - V(t))Z(t) - k_1 V(t), \\ \frac{dW(t)}{dt} &= c n(c)(Y - V(t))Z(t) - \frac{u(t)}{c} - k_2 Z(t). \end{aligned} \quad (9.3.2)$$

Заметим, что статическая модель (9.3.1) может быть получена из динамической (9.3.2) путем приравнивания нулю правых частей в (9.3.2). Поэтому статическая модель может рассматриваться как частный случай динамической, отражающей состояние динамического равновесия.

Перепишем динамическую модель (9.3.2) в терминах теории автоматического управления.

Пусть $x(t) = (x_1(t), x_2(t), x_3(t), x_4(t))^T$ – вектор состояния, где $x_1(t) = X$, $x_2(t) = Z$, $x_3(t) = V$, $x_4(t) = W$; $u(t)$ – управление, задающее темп производства. Тогда система (9.3.2) запишется следующим образом:

$$\dot{x}(t) = f(t, x(t), u(t)), \quad (9.3.3)$$

где $f(t, x(t), u(t))$ – вектор-функция вида:

$$f(x(t), u(t)) = \begin{pmatrix} u(t) \\ u(t) - n(c)Y x_2(t) + n(c)x_2(t)x_3(t) \\ n(c)Y x_2(t) - n(c)x_2(t)x_3(t) - k_1 x_3(t) \\ c n(c)Y x_2(t) - c n(c)x_2(t)x_3(t) - \frac{u(t)}{c} - k_2 x_2(t) \end{pmatrix}. \quad (9.3.4)$$

Система (9.3.3) является нелинейной *нестационарной детерминированной моделью* процесса производства, сбыта и хранения товара.

Для того чтобы получить *линейную* систему, (9.3.3) запишем:

$$\dot{x}(t) = \left. \frac{\partial f(\cdot)}{\partial x} \right|_{x=x^H(t)} x(t) + \left. \frac{\partial f(\cdot)}{\partial u} \right|_{u=u^H(t)} u(t) \quad (9.3.5)$$

или

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)u(t), \quad (9.3.6)$$

где $x^H(t)$, $u^H(t)$ – номинальные траектория и управление (например, характеристики, соответствующие плановым, расчетным, прогнозируемым показателям);

$$\bar{A}(t) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -n(c)Y + n(c)x_3^H(t) & n(c)x_2^H(t) & 0 \\ 0 & n(c)Y - n(c)x_3^H(t) & -n(c)x_2^H(t) - k_1 & 0 \\ 0 & c(n(c)Y - n(c))x_3^H(t) - k_2 & -c n(c)x_2^H(t) & 0 \end{pmatrix}, \quad (9.3.7)$$

$$\bar{B}(t) = \begin{pmatrix} 1 & 1 & 0 & -\frac{1}{c} \end{pmatrix}^T. \quad (9.3.8)$$

Для учета колебаний рынка, поведения покупателей, погрешности линеаризации и т.д., добавим в систему (9.3.6) вектор гауссовских шумов $q(t)$ с матрицей влияния $\bar{F}(t)$, компоненты которой, например, можно определить для конкретного предприятия экспериментально. Тогда математическая модель, описывающая процесс производства, сбыта и хранения товара, запишется в виде системы *линейных нестационарных стохастических дифференциальных уравнений*:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)u(t) + \bar{F}(t)q(t), \quad (9.3.9)$$

где $M\{q(t)\} = \bar{q}(t)$, $M\{(q(t) - \bar{q}(t))(q(t) - \bar{q}(t))^T\} = Q(t)\delta(t - \tau)$, $\delta(t - \tau)$ – δ -функция Дирака.

В терминах теории автоматического управления система (9.3.9) сформулирована для решения задачи управления положением рулей (в данном случае темпом производства). Можно решать задачу управления скоростью отклонения рулей. В данном случае это будет задача управления изменением темпа производства, математическая модель которой может быть описана следующей системой линейных нестационарных стохастических дифференциальных уравнений:

$$\begin{aligned} \dot{x}(t) &= \bar{A}(t)x(t) + \bar{B}(t)u(t) + \bar{F}(t)q(t), \\ \dot{i}(t) &= v(t), \end{aligned} \quad (9.3.10)$$

где $x(t) = (Z, V, W)^T \in R^3$ – вектор состояния, $u(t)$ – темп производства, $q(t)$ – вектор внешних возмущений, $v(t)$ – управление, задающее изменение темпа производства;

$$\bar{A}(t) = \begin{pmatrix} -n(c)Y + n(c)x_3^H(t) & n(c)x_2^H(t) & 0 \\ n(c)Y - n(c)x_3^H(t) & -n(c)x_2^H(t) - k_1 & 0 \\ c(n(c)Y - n(c))x_3^H(t) - k_2 & -cn(c)x_2^H(t) & 0 \end{pmatrix},$$

$$\bar{B}(t) = \begin{pmatrix} 1 & 0 & -\frac{1}{c} \end{pmatrix}^T,$$

$\bar{F}(t)$ – матрица соответствующей размерности, описывающая влияние случайных факторов в модели (9.3.10).

10 ПРОГРАММНЫЕ СРЕДСТВА МОДЕЛИРОВАНИЯ

Выделим следующие основные требования к программным системам, предназначенным для моделирования:

- наличие математических функций;
- точность вычислений;
- обработка ошибок;
- возможность пополнения библиотек;
- система отображения;
- система разработки интерфейса;
- возможности коммуникации с другими программами.

Учитывая вышеперечисленные требования, рассмотрим различные системы моделирования как специализированные системы, предназначенные для математических расчетов, так и универсальные, широко применяемые инструментальные системы. Рассмотрение программных средств моделирования начнем именно с универсальных систем.

Прежде чем перейти к описанию систем программирования, построенных на базе языков C/C++, следует понять разницу между языком программирования и средой разработки. Язык – это некоторый формальный способ описания задачи, решаемой машиной. Транслятор языка – это программа, которая переводит текст задачи с заданного языка на язык машины. А система программирования – это ряд программ и системных библиотек, которые призваны облегчить программисту разработку программ на этом языке. В состав систем программирования, помимо транслятора языка (иногда даже нескольких), обычно включается интегрированная среда разработки, различные системные библиотеки и т.д. Например, есть язык C++, есть множество различных реализаций компилятора этого языка, в том числе реализация от фирмы Microsoft, и есть система программирования Microsoft Visual C++, которая строится на этом компиляторе и большом количестве дополнительных библиотек и инструментальных средств.

10.1 Языки программирования

10.1.1 Язык *Fortran*

Язык Fortran (от англ. **F**ormula **T**ranslation) – один из первых языков программирования, появившийся еще в 50-х годах. В то время было разработано множество библиотек, в том числе математических, ориентированных на этот язык. Несмотря на то, что язык оказался не слишком удобным в использовании, он довольно прочно занял свою нишу для разработки математических программ, в первую очередь для моделирования. С тех пор язык претерпел несколько редакций (последняя – в 95-м году) и существует большое множество различных версий компилятора этого языка. Среди современных реализаций для персональных компьютеров следует отметить Microsoft Visual Fortran.

Как следует из названия языка, основное его предназначение – перевод формул. И с этой работой он справляется достаточно успешно. К недостаткам этого языка следует отнести слабые возможности по описанию абстрактных типов данных, неудобную организацию циклов и других языковых конструкций.

10.1.2 Язык *C*

Язык C, появившийся в первой половине 70-х годов прошлого столетия в лаборатории Bell Labs компании AT&T [Керниган&Ритчи], является в настоящее время де-факто стандартом для разработки программ системного назначения малого и среднего размеров. В настоящее время реализация этого языка есть практически для всех вычислительных систем от микроконтроллеров до сверхбольших вычислительных машин. Понятие «среднего размера» весьма относительно и, как правило, соответствует проектам от 10 до 100 тысяч строк исходного текста. Под «системными» понимаются программы, предназначенные, в основном, для нужд операционной системы (в том числе и сама операционная система является системной программой). Однако этот язык вполне подходит и для разработки прикладных программ, в том числе связанных с математическими вычислениями. Изю-

минка языка С – в его относительной близости к вычислительной системе, что позволяет разрабатывать высокоэффективный код (с минимальными накладными расходами, а соответственно, максимальный по скорости выполнения и минимальный по расходу памяти).

Язык С обладает набором из 50 операций (как целочисленных, так и вещественных), распределённых по 15 уровням приоритетов. Среди математических операций языка собраны все операции, которые могут выполнить центральный процессор и математический сопроцессор. Часть операций применима как к целым типам данных, так и к вещественным.

Язык не обладает встроенным понятием модуля, но, как правило, среда разработки (или по-другому – окружение языка) позволяет ввести это понятие искусственно и использовать как большое количество стандартных библиотек, так и свои собственные разработанные библиотеки. Более того, язык не обладает собственными средствами ввода/вывода, все эти средства реализуются внешними библиотеками, что позволяет разрабатывать более гибкий исходный код программ, применимый для различного операционного окружения языка и исполняемой программы.

Недостатки языка являются обратной стороной его достоинств. К ним можно отнести отсутствие встроенных типов данных высокой степени абстракции (например, «строка», часто применимого для организации диалога с пользователем), отсутствие средств диагностики (контроль выхода за пределы массива, переполнения стека и числа и т.д.). Такой контроль можно обеспечить лишь, связавшись на низком уровне с операционной системой, при условии, что операционная система поддерживает такой контроль.

Таким образом язык С хорошо подходит для разработки низкоуровневых системных библиотек и для математического моделирования.

10.1.3 Язык C++

Язык C++, разработанный в первой половине 80-х годов, как объектно-ориентированное (ОО) развитие языка С, Бьярном

Страуструпом, занимает несколько иную нишу. В отличие от языка С, программы, скомпилированные с языка С++, обладают, как правило, немного худшей производительностью и большим объёмом исполняемого кода. Основная ниша данного языка – разработка больших (обычно прикладных) программных систем сравнительно большой группой разработчиков, распределённой в пространстве и/или времени. Основное преимущество данного языка – языковая поддержка 3-х базисных конструкций ОО программирования: инкапсуляция, полиморфизм и наследование.

Под *инкапсуляцией* понимается разделение реализации объекта (модуля) и его интерфейса. Как правило, скрываются данные, а в интерфейсе остаются только методы их обработки. Поэтому инкапсуляцию иногда трактуют как объединение данных и методов их обработки в единой языковой конструкции. При решении задач моделирования это позволяет создавать типы данных высокой степени абстракции. Такие, например, как вектора, матрицы, генераторы, фильтры и так далее. Фактически, любое понятие предметной области (ПО) можно рассматривать как объект с точки зрения ОО программирования и строить соответствующую ему модель с детерминированным набором свойств и поведением. В языке С++ для поддержки инкапсуляции вводят понятия «объект» и «класс», который представляет собой обобщенное описание объекта.

Язык С++ позволяет, используя еще одну языковую конструкцию – *наследование*, создавать обобщенные (базовые) классы, описывая в них общую составляющую всех дочерних классов. Это позволяет упростить процесс доработки программ, который теперь можно сводить не к переделке всего разработанного ранее класса. Для изменения набора свойств и/или расширения функциональности можно описать новый класс, наследованный от созданного ранее.

Еще одной принципиальной с точки зрения ОО программирования возможностью языка является *полиморфизм*. Это понятие определяется как возможность указания разной семантической нагрузки для конструкций с одинаковым синтаксисом. Например, пусть есть классы «вектор» и «матрица». Оба этих класса имеют метод «абсолютное значение», который имеет схожий синтаксис (на входе объект, на выходе вещественное число), но

внутренняя реализация этих методов принципиально различна. Тем не менее, вполне вероятно, что некоторая функция может работать с любым типом данных, если тот обладает методом вычисления абсолютного значения. Можно разработать такой метод для всех типов данных, с которыми будет работать Ваша программа, но при этом возникнет большая степень дублированности кода. С другой стороны, можно разработать некоторый базовый класс, в котором перечислены все основные свойства и методы большой группы классов. При этом часть методов могут быть описаны, как виртуальные, что позволяет давать этим методам разную реализацию в дочерних классах.

Еще одной возможностью языка C++ является возможность описания шаблонов функций и классов (так называемое параметризованное описание). Параметром такого описания является, как правило, тип (или набор типов), с которым работает данная функция или класс. Это позволяет создавать более высокоэффективные (по сравнению с наследованием) программы.

Кроме того, язык C++, один из немногих, позволяет перегружать смысл своих операций для абстрактных типов данных. Таким образом, можно получить набор типов (классов), максимально приближенный к аналогичным объектам в математике, например, класс матрица или вектор. Совместное использование всех языковых конструкций позволило создать весьма эффективную, но в то же самое время очень гибкую стандартную библиотеку шаблонов (standard template library – STL), включающую все основные способы хранения данных и базовые алгоритмы обработки этих данных. Существует построенная на этой библиотеке, например, библиотека MTL (matrix template library – библиотека шаблонов матрицы), которая позволяет достаточно эффективно обрабатывать матрицы произвольной размерности с произвольным типом хранимых данных.

К сожалению, богатая история развития языка C++ привела к появлению огромного количества разночтений и, фактически, различных диалектов этого языка. Усилия консорциума разработчиков привели к тому, что в 1998 г. был утвержден стандарт этого языка. Однако далеко не все существующие компиляторы языка C++ совместимы с этим стандартом.

10.1.4 Языки Pascal и Object Pascal

Процедурный язык Pascal был разработан Николаусом Виртом, исключительно, как лабораторный язык. Этот язык отлично подходил для обучения студентов программированию, но не предполагался использоваться, как серьезный язык для разработки прикладных программ. Однако некоторые фирмы (например, Borland) смогли сделать из этого языка серьезную систему разработки. Позже Николаус Вирт разработал языки Modula, Modula-2, Modula-3 и язык Oberon. Лучшие идеи, реализованные в этих языках, были использованы в коммерческих разработках, которые называются языком Pascal.

Стандартных операций в языке Pascal несколько меньше, чем, например, в языке C, но их вполне достаточно для того, чтобы покрыть все основные потребности при разработке математических программ. В коммерческих реализациях Pascal присутствует понятие модуля, что позволяет разрабатывать для него библиотеки и дает шанс найти библиотеку, нужную для конкретной задачи.

Позже был разработан язык Object Pascal, который добавил к языку Pascal возможности объектно-ориентированности. Объектная модель языка Object Pascal беднее, чем C++. Например, отсутствует возможность перегрузки смысла операций и нет шаблонов, что несколько уменьшает применимость данного языка.

10.2 Системы разработки программного обеспечения

10.2.1 Microsoft Visual C++ и MFC

Microsoft Visual C++ является составной частью пакета Microsoft Visual Studio. Как уже было сказано, система разработки ПО Microsoft Visual C++ базируется на компиляторе языка C++ от фирмы Microsoft, который, к сожалению, очень далек от стандарта языка. Тем не менее, Visual C++ является де-факто стандартом при разработке программного обеспечения для ОС Windows, в том числе и программ моделирования. Ограниченность персональных компьютеров не позволяет решать на них серьезные задачи моделирования, что, впрочем, не умаляет роли этих компь-

ютеров и системы Visual C++ для разработки программ-интерфейсов.

Именно для разработки интерфейсов с пользователем эта система программирования предоставляет максимальные возможности. Можно назвать богатую по возможностям библиотеку MFC, есть средства доступа к базам данных ADO, средства по созданию и использованию ActiveX компонент – ATL и многое другое.

Однако есть также и масса недостатков. Главный недостаток – не полная совместимость компилятора со стандартом языка. Наибольшие отличия наблюдаются в реализации шаблонов, что не позволяет эффективно использовать STL, MTL и ряд других известных шаблонных библиотек.

10.2.2 Borland Delphi

В основе этой системы программирования лежит сильно модифицированный фирмой Borland язык Object Pascal. Язык изменен настолько сильно, что правильнее его называть языком Delphi. Первая версия этой среды разработки была ориентирована на ОС Windows 3.x и в то время стала прорывом в области разработок программ с интерфейсом пользователя. Основная ниша этой среды разработки – разработка клиентских программ для баз данных.

Для этого в системе разработки программ предусмотрены богатые возможности по разработке интерфейса и несколько различных вариантов доступа к базам данных.

10.2.3 Borland C++Builder

Эта система разработки распространена значительно меньше предыдущей. Однако она имеет ряд принципиальных преимуществ, которые позволяют о ней говорить как о реально используемой платформе. В частности, компилятор C++, на базе которого построена эта система, очень близок к стандарту языка (вообще говоря, начиная с 5-й версии, Borland C++Builder получил сертификат соответствия стандарту, хотя на самом деле небольшие отличия есть). Все специализированные средства языка

(которых нет в стандарте) являются расширениями и не противоречат стандарту.

В основу положена та же библиотека, которая лежит в основе системы Delphi. И соответственно, эта система обладает всеми теми же возможностями, добавляя возможности, присущие стандарту языка C++.

К недостаткам следует отнести сравнительно большой объем исполняемого кода и несколько худшие (по сравнению с Visual C++) скоростные показатели для программ графического интерфейса.

10.3 Специализированные математические пакеты

В последние двадцать лет возникло и получило бурное развитие новое фундаментальное научное направление – *компьютерная математика*, которая зародилась на стыке математики и информатики. Первыми серьезными средствами для автоматизированного выполнения массовых научно-технических расчетов были программируемые микрокалькуляторы. Предвестниками систем компьютерной математики стали специализированные программы для математических численных расчетов, работающие в среде Microsoft MS-DOS. Это – Eureka, Mercury, первые версии системы MathCAD и MatLAB под операционную систему MS-DOS. Вслед за этим на основе достижений компьютерной математики появились новейшие программы символьной математики или компьютерной алгебры. Среди них особенно большую известность получили системы MathCAD под Windows, Derive, Mathematica и Maple. Созданные для проведения символьных (аналитических) преобразований математических выражений, эти системы были в поразительно короткое время доведены до уровня, позволяющего резко облегчить, а подчас и заменить труд самой почитаемой научной элиты мира – математиков-теоретиков и аналитиков.

В последнее время появившиеся тенденции к объединению передовых разработчиков математических систем привели к тому, что лидеры в данной области, такие, как Maple, MathCAD, Mathematica, MatLAB, интегрировались и теперь стоят приблизи-

тельно на одной ступени развития. Различия между возможностями этих систем практически стерлись и сохранились лишь в каких-то базовых элементах и визуализации программных документов.

10.3.1 Система Maple

Системы компьютерной математики класса Maple были созданы корпорацией Waterloo Maple, как система компьютерной алгебры с расширенными возможностями в области символьных вычислений. Система содержит средства для выполнения быстрых численных расчетов, лежащих в основе математического моделирования различных явлений окружающего мира, систем и устройств различного назначения. Все это сочетается с новейшими и весьма эффективными средствами визуализации вычислений.

Maple – типичная интегрированная система. Она объединяет в себе мощный язык программирования (он же язык для интерактивного общения с системой), редактор для подготовки и редактирования документов и программ, многооконный пользовательский интерфейс с возможностью работы в диалоговом режиме, справочную систему с тысячей примеров, ядро алгоритмов и правил преобразования математических выражений, численный и символьный процессоры, систему диагностики, библиотеки встроенных и дополнительных функций, пакеты функций сторонних производителей и поддержку некоторых других языков программирования и программ. Ко всем этим средствам имеется полный доступ прямо из окна системы. Она реализована на больших ЭВМ, ПК класса IBM PC, Macintosh и др.

Ядро системы Maple используется целым рядом систем компьютерной математики, например, таких, как MatLAB и MathCAD.

10.3.2 Система Mathematica

Система Mathematica разработана фирмой Wolfram Research. Основная идея разработчиков – объединить все известные поня-

тия и методы математики в единую универсальную систему, способную функционировать на любой вычислительной платформе. Эта система дает возможность решать большое количество достаточно сложных задач, не вдаваясь в тонкости программирования, что привело к её широкому использованию в таких науках, как физика, биология, экономика и т.д.

Система Mathematica состоит из двух частей – ядра, которое, собственно, и проводит вычисления, выполняя заданные команды, и интерфейсного процессора, фактически задающего внешнее оформление и характер взаимодействия с пользователем и программой. Пользователь записывает все выкладки в основном рабочем документе программы – notebook («записная книжка»). Внешний вид рабочего документа на экране монитора в большей или меньшей степени зависит от типа платформы (Windows, Macintosh, Unix) и определяется интерфейсным процессором, своим для каждой платформы.

Notebook – полностью интерактивный документ, содержащий текст, таблицы, графики, вычисления и другие элементы. Документы могут быть представлены на экране с различным полиграфическим исполнением, могут включать в себя всевозможные математические или специальные символы. Однако внутреннее представление документов, с которыми работает ядро программы, всегда – не форматируемый текст в виде печатаемых (printable) 7-битовых ASCII-символов. Это позволяет легко переписывать документы с одной платформы на другую.

Язык программирования Mathematica рассматривается как единый универсальный язык программирования и математики. Он является беспрецедентно гибким и интуитивным языком, содержит уникальную комбинацию математических и вычислительных обозначений. Основная унифицирующая идея языка – рассматривать любой объект как символьное выражение (это касается не только привычных структур данных, но и графиков, звуков и даже ячеек, и даже самого документа). Это делает легким добавление к системе Mathematica новых конструкций языка и изменения интерфейса программы.

Язык включает в себя представление широко известных развитых методов программирования компьютерной науки и добав-

ляет множество новых. На языке программирования можно всегда написать программу в наиболее естественном виде, так как Mathematica включает в себя множество парадигм программирования: процедурное программирование; основанное на операциях со списками (list-based); основанное на операциях со строками (string-based); функциональное; объектно-ориентированное; программирование, задающее правила преобразования выражений («правила переписывания термов»).

В виде встроенных функций реализовано большое множество высокоинтеллектуальных математических операций. В состав Mathematica входят стандартные дополнения, включающие в себя подпрограммы (пакеты), которые значительно расширяют функциональные возможности системы в таких областях, как алгебра, аналитические и численные расчеты, графика, дискретная математика, геометрия, теория чисел, статистика и др.

10.3.3 Система MatLAB

Лидером в области численных и матричных расчетов, а также реализации техники имитационного и ситуационного моделирования является система MatLAB с ее многочисленными пакетами расширения. Однако в области аналитических вычислений она сильно уступает таким системам, как Maple и Mathematica.

Система MatLAB (сокращенно от MATrix LABoratory – Матричная лаборатория) разработана фирмой Math Works. Это интерактивная система, ориентированная в первую очередь на обработку массивов данных, базируется на матричных математических операциях. Даже одиночное число MatLAB рассматривает как матрицу, что позволило существенно повысить скорость выполнения вычислений.

С точки зрения пользователя MatLAB представляет собой богатейшую библиотеку функций. Для облегчения поиска библиотека функций разбита на разделы. Те из функций, которые носят более общий характер и используются наиболее часто, входят в состав ядра MatLAB. Те функции, которые являются специфическими для конкретной области, включены в состав соответствующих специализированных разделов. Эти разделы назы-

ваются *MatLAB Toolboxes (Инструменты)*. Каждый из них имеет свое собственное название, отражающее его предназначение. Полная комплектация системы MatLAB содержит более 30 инструментальных приложений. В их число входят как достаточно стандартные для математических пакетов средства, так и нетрадиционные: средства цифровой обработки изображений, поиска решений на основе нечеткой логики, аппарат построения и анализа нейронных сетей, средства финансового анализа и др. Кроме того, имеются средства взаимодействия с офисными продуктами фирмы Microsoft – MS Word и MS Excel.

К несомненным достоинствам системы MatLAB следует отнести тот факт, что она базируется на современных математических подходах, имеющих непосредственный выход в практику проектирования технических систем, например таких, как теория робастности и др.

Особое место среди инструментальных приложений занимает система визуального моделирования SIMULINK. В определенном смысле SIMULINK можно рассматривать как самостоятельный продукт фирмы Math Works, однако он работает только при наличии ядра MatLAB и использует многие функции, входящие в его состав.

MatLAB является платформно-независимой системой, так как может работать под управлением нескольких операционных систем: Windows, UNIX, MacOS. При этом технология моделирования с помощью SIMULINK остается неизменной.

10.3.4 Система MathCAD

Система MathCAD разработана фирмой MathSoft. Она существенно отличается от аналогичных прикладных систем, таких, как MatLAB, Mathematica, Maple, тем, что является единственной прикладной системой, в которой описания математических задач и их решений задаются с помощью обычных в математике символов, формул и операторов, а документ MathCAD выглядит как страницы учебника или научной статьи.

MathCAD – это популярная система компьютерной математики, предназначенная для решения математических задач в са-

мых разных областях науки, техники и образования. Ранние версии MathCAD вообще не имели средств обычного программирования, а имели лишь средства визуально-ориентированного программирования в виде шаблонов математических операций, из которых составлялись математические выражения. Возможность задания программных модулей появилась, начиная с версии MathCAD PLUS 6.0. Программные модули, в сущности, являются функциями, но описанными с применением программных средств.

Последние версии MathCAD допускают применение внешних расширений: электронных книг, пакетов расширений и библиотек. Внешние библиотеки чаще всего являются справочниками. Пакеты расширений и сопровождающие их электронные книги имеют различное назначение и ориентированы, как правило, на выполнение узкоспециальных вычислений. Пакеты расширений предоставляют следующие возможности: осуществлять вейвлет преобразования, анализ временных рядов, финансовые расчеты, обработку сигналов и изображений, расчеты по прикладной статистике, механике, дифференциальным уравнениям и т.д. Существенными дополнениями к системе являются пакеты расширения графических возможностей, численных методов.

В MathCAD возможна интеграция с текстовым процессором Word, электронными таблицами Excel, графической системой Axum, пакетом научной и инженерной графики Visio, с пакетом SmartSketch LE, который позволяет включать в состав документов MathCAD довольно сложные конструкторские чертежи, с пакетом VisSim – моделирующей программой, в которой объекты задаются блоками, между которыми указываются дополнительные связи.

Привычный вид математических формул, встроенный язык программирования, широкие возможности в отображении графической информации, представление текстовых комментариев с использованием любых символов, доступных в Windows, возможность проводить расчеты любой сложности и готовить документы высокого качества – все это характеризует систему MathCAD.

10.3.5 Система STATISTICA

Программная система STATISTICA разработана фирмой Copyright StatSoft. Это современный пакет для статистического анализа данных, в котором реализованы все новейшие компьютерные и математические методы анализа данных. Это – описательные статистики; анализ многомерных таблиц; многомерная и нелинейная регрессии; подгонка распределений; дискриминантный, кластерный, факторный, дисперсионный и ковариационный анализы; структурные модели; прогнозирование временных рядов; непараметрическая статистика; анализы надежности предпочтений, Монте-Карло, выживаемости и т.д. Система STATISTICA подходит для применения в любой области: маркетинге, финансах, страховании, бизнесе, промышленности, медицине и т.д.

10.3.6 Система EXCEL

Табличный процессор EXCEL входит в самый популярный пакет автоматизации офисной деятельности Microsoft Office. EXCEL – одна из самых мощных и гибких систем обработки электронных таблиц. Эта система может работать не только с двумерными, но и с трехмерными таблицами, представленными листами с двумерными таблицами.

EXCEL широко используется для подготовки прекрасно иллюстрированных финансово-экономических и других документов. Этот процессор содержит сотни математических и экономических функций, что позволяет решать множество задач в области естественных и технических наук.

Возможности системы EXCEL можно использовать для анализа внешних данных, представленных в Microsoft FoxPro, Access, Paradox, dBASE, SQL Server, а также базы данных сторонних производителей, поддерживающих технологию OLE (Object Linking and Embedding – связывания и внедрения объектов).

В системе EXCEL можно создавать макросы и приложения с помощью Visual Basic for Applications (VBA). Visual Basic for

Applications – это среда разработки приложений на базе программ, входящих в пакет Microsoft Office. Одно из главных достоинств языка Visual Basic for Applications заключается в том, что созданные средствами EXCEL VBA-макросы можно без труда использовать в других программах фирмы Microsoft.

ЛИТЕРАТУРА

1. Андреев Ю.Н. Управление конечномерными линейными объектами. – М.: Наука, 1976. – 424 с.
2. Брайсон А., Хо Ю-Ши. Прикладная теория оптимального управления. – М.: Мир, 1972. – 544 с.
3. Браммер К., Зиффлинг Г. Фильтр Калмана-Бьюси. – М.: Наука, 1982. – 200 с.
4. Буков В.Н. Адаптивные прогнозирующие системы управления полетом. – М.: Наука, 1987. – 232 с.
5. Горский А.А., Колпакова И.Г., Локшин Б.Я. Динамическая модель процесса производства, хранения и сбыта товара повседневного спроса // Изв. РАН Теория и системы управления. – 1998. – №1. – С. 144–148.
6. Дорф Р., Бишоп Р. Современные системы управления. – М.: Лаборатория базовых знаний, 2002. – 832 с.
7. Красовский А.А., Буков В.Н., Шендрик В.С. Универсальные алгоритмы оптимального управления непрерывными процессами. – М.: Наука, 1977. – 272 с.
8. Крылов В.И., Бобков В.В., Монастырный П.И. Вычислительные методы. Том I. – М.: Наука, 1976. – 304 с.
9. Крылов В.И., Бобков В.В., Монастырный П.И. Вычислительные методы. Том II. – М.: Наука, 1976. – 400 с.
10. Летов А.М. Аналитическое конструирование регуляторов. – //Автоматика и телемеханика. – 1960. – №1. – С. 436–441; 1960. – №5. – С.561–568; 1960. – №6. – С. 661–665; 1960 – №4. – С. 425–435; 1962. – №11. – С. 1405–1413.
11. Медич Дж. Статистически оптимальные линейные оценки и управление. – М.: Энергия, 1973. – 440 с.
12. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. – Томск: МП «РАСКО», 1991. – 272 с.
13. Острем К., Виттенмарк Б. Системы управления с ЭВМ. – М.: Мир, 1987. – 480 с.

14. Решетникова Г.Н. Синтез и моделирование дискретных адаптивных систем (гос. Регистр. №50880000594) //Алгоритмы и программы: Информационный бюллетень. – 1989. – №1. – С. 10.

15. Решетникова Г.Н., Смагин В.И.. Адаптивное управление по локальным и квазилокальным критериям: Учебное пособие по курсу «Адаптивные системы». – Томск: ТГУ, 1993. – 27 с.

16. Смагин В.И., Параев Ю.И. Синтез следящих систем управления по квадратичным критериям. – Томск: Изд-во Том. ун-та, 1996. – 171 с.

17. Советов Б.Я., Яковлев С.А. Моделирование систем: Учебник для вузов. – 3-е изд., перераб. и доп. – М.: Высш. шк., 2001. – 343 с.

18. Справочник по теории автоматического управления / Под ред. А.А.Красовского. – М.: Наука, 1987. – 712 с.