Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

В. П. Шестакович

ОСНОВЫ ЧИСЛЕННЫХ МЕТОДОВ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники для специальностей, закрепленных за УМО, в качестве учебно-методического пособия по дисциплине «Основы алгоритмизации и программирования»

Минск БГУИР 2012

УДК 519.6(076)
ББК 22.19я73
Ш51

Р е ц е н з е н т ы:
кафедра технологий программирования
Белорусского государственного университета
(протокол №11 от 15.06.2012 г.);

заведующий кафедрой программного обеспечения
вычислительной техники и автоматизированных систем
Белорусского национального технического университета,
кандидат технических наук, доцент Н. Н. Гурский;

профессор кафедры ВМиП учреждения образования
«Белорусский государственный университет информатики
и радиоэлектроники», доктор физико-математических наук С. В. Колосов

Шестакович, В. П.

Ш51 Основы численных методов : учеб.-метод. пособие / В. П. Шестакович. – Минск : БГУИР, 2012. – 68 с. : ил.

ISBN 978-985-488-789-0.

В пособии даны краткие теоретические сведения по основам численных методов, рассмотрены алгоритмы их реализации. В конце каждой темы приведены индивидуальные задания и контрольные вопросы.

УДК 519.6(076)
ББК 22.19я73

ISBN 978-985-488-789-0

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Широкое применение компьютеров является характерной чертой современной жизни. В настоящее время наряду с широким использованием компьютеров для поиска, обработки и передачи информации компьютеры все более широко используются в области компьютерного проектирования и управления физическими, химическими и экономическими процессами. Любая задача компьютерного проектирования состоит из трех основных этапов – математического моделирования, анализа и оптимизации. Математическое моделирование позволяет заменить натурный эксперимент вычислительным, что резко сокращает время разработки новой продукции и затраты на изготовление опытных образцов, позволяет оптимизировать конструкцию устройств, усовершенствовать технологию, автоматизировать процессы управления. Любая математическая модель представляет собой совокупность типовых математических задач – решение систем линейных алгебраических уравнений, вычисление интегралов, нахождение корней нелинейных уравнений, аппроксимация функций и т. д. При построении математических моделей динамических процессов (процессов перехода физических систем из одного состояния в другое, бесконечно близкое) возникает необходимость решения дифференциальных уравнений. Примерами таких процессов могут служить явления, возникающие в электрических цепях, поведение системы взаимодействующих частиц во внешних полях, явления химической кинетики, описания электромагнитных полей и волн в направляемых системах и приборах СВЧ и т. д. Однако точные методы решения типовых математических задач, позволяющие выразить решение в виде элементарных или специальных функций, существуют только для узкого класса задач. В силу этого важное значение приобретают численные методы решения, ориентированные на широкий класс встречающихся в вычислительной практике задач. Все это и послужило причиной того, что составной частью дисциплины «Основы алгоритмизации и программирования» явился раздел, содержащий описание численных методов решения наиболее часто встречающихся типовых математических задач и методов оптимизации.

ТЕМА 1. АЛГОРИТМЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Цель работы: изучить основные методы и алгоритмы решения систем линейных алгебраических уравнений (СЛАУ).

1.1. Основные понятия и определения

Выделяют четыре основные задачи линейной алгебры: решение СЛАУ, вычисление определителя матрицы, нахождение обратной матрицы, определение собственных значений и собственных векторов матрицы.

Задача отыскания решения СЛАУ с n неизвестными – одна из наиболее часто встречающихся в практике вычислительных задач, т. к. большинство методов решения сложных задач основано на сведении их к решению некоторой последовательности СЛАУ.

Обычно СЛАУ записывается в виде

$$\sum_{j=1}^{n} a_{ij} x_j = b_i;\ 1 \le i \le n \qquad \text{или в матричном виде } A\vec{x} = \vec{b},$$

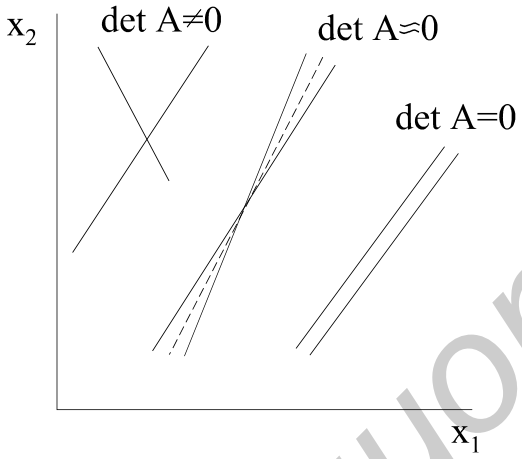

Рис. 1.1. Решения СЛАУ

$$A = \begin{bmatrix} a_{11}a_{12}...a_{1n} \\ a_{21}a_{22}...a_{2n} \\ ... \\ a_{n1}a_{n2}...a_{nn} \end{bmatrix};\quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix};\ \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{bmatrix}. \quad (1.1)$$

Здесь A и $\vec{b}$ заданы, требуется найти $\vec{x}$, удовлетворяющий (1.1).

Известно, что если определитель матрицы $|A| \neq 0$, то СЛАУ имеет единственное решение. В противном случае решение либо отсутствует (если $\vec{b} \neq 0$), либо имеется бесконечное множество решений (если $\vec{b} = 0$) (рис. 1.1). При решении систем, кроме условия $\det|A| \neq 0$, важно, чтобы задача была ***корректной,*** т. е. чтобы при малых погрешностях правой части $\Delta\vec{b}$ и (или) коэффициентов Δa_{ij} погрешность решения $\Delta\vec{x}$ также оставалась малой. Признаком некорректности, или плохой обусловленности, является близость к нулю определителя матрицы.

Плохо обусловленная система двух уравнений геометрически соответствует почти параллельным прямым (см. рис. 1.1). Точка пересечения таких прямых (решение) при малейшей погрешности коэффициентов резко сдвигается. Обусловленность (корректность) СЛАУ характеризуется числом $\chi = \|A\| \cdot \|A^{-1}\| \ge 1$. Чем дальше χ от 1, тем хуже обусловлена система. Обычно при $\chi > 10^3$ система некорректна и требует специальных методов решения –

методов регуляризации. Приведенные ниже методы применимы только для корректных систем.

Методы решения СЛАУ делятся на прямые и итерационные.

Прямые методы дают в принципе точное решение (если не учитывать ошибок округления) за конечное число арифметических операций. Для хорошо обусловленных СЛАУ небольшого порядка $n \le 200$ применяются практически только прямые методы.

Наибольшее распространение среди прямых методов получили **метод Гаусса** для СЛАУ общего вида, его модификация для трехдиагональной матрицы – **метод прогонки** и **метод квадратного корня** для СЛАУ с симметричной матрицей.

Итерационные методы основаны на построении сходящейся к точному решению $\vec{x}*$ рекуррентной последовательности векторов $(\vec{x}^o, \vec{x}^1, \vec{x}^2, ..., \vec{x}^k \xrightarrow[k\to\infty]{} \vec{x}^*$. Итерации выполняют до тех пор, пока норма разности $\delta_k = \left\| \overset{\to k}{x} - \overset{\to k-1}{x} \right\| = \max_i \left| x_i^k - x_i^{k-1} \right| \le \varepsilon$. ($\varepsilon$ – заданная малая величина).

Итерационные методы, как правило, используются для систем большого порядка n>100, а также для решения плохо обусловленных систем. Многообразие итерационных методов решения СЛАУ объясняется возможностью большого выбора рекуррентных последовательностей, определяющих метод. Наибольшее распространение среди итерационных методов получили одношаговые методы **простой итерации** и **Зейделя** с использованием релаксации.

Для контроля полезно найти невязку полученного решения $\vec{x}$:

$$\Delta = \max_{1\le k\le n} \left| b_k - \sum_{i=1}^{n} a_{ki} x_i \right|;$$

если Δ велико, то это указывает на грубую ошибку в расчете.

Ниже приведено описание алгоритмов указанных методов решения СЛАУ.

1.2. Прямые методы решения СЛАУ

1.2.1. Метод Гаусса

Метод основан на приведении с помощью преобразований, не меняющих решение, исходной СЛАУ (1.1) с произвольной матрицей к СЛАУ с верхней треугольной матрицей вида

$$\begin{aligned} a'_{11}x_1 + a'_{12}x_2 + ... + a'_{1n}x_n &= b'_1, \\ a'_{22}x_2 + ... + a'_{2n}x_n &= b'_2, \\ &..... \\ a'_{nn}x_n &= b'_n. \end{aligned} \qquad (1.2)$$

Этап приведения к системе с треугольной матрицей называется ***прямым ходом метода Гаусса*.**

Решение системы с верхней треугольной матрицей (1.2), как легко видеть, находится по формулам, называемым ***обратным ходом метода Гаусса*:**

$$x_n = b'_n / a'_{nn};\ x_k = \frac{1}{a'_{kk}}\left[b'_k - \sum_{i=k+1}^{n} a'_{ki} x_i \right],\ k = n-1,\ n-2,\ ...,\ 1. \qquad (1.3)$$

Прямой ход метода Гаусса осуществляется следующим образом: вычтем из каждого *m*-го уравнения *(m*=2, 3,..,*n)* первое уравнение, умноженное на a_{m1}/a_{11}, и вместо *m*-го уравнения подставим полученное. В результате в матрице системы исключаются все коэффициенты 1-го столбца ниже диагонального. Затем, используя 2-е полученное уравнение, аналогично исключим элементы второго столбца *(m*=3, 4,..,*n)* ниже диагонального и т. д. Такое исключение называется ***циклом метода Гаусса***. Проделывая последовательно эту операцию с расположенными ниже *k*-го уравнениями *(k*=1, 2 ,..., *n*–1*)*, приходим к системе вида (1.2). При указанных операциях решение СЛАУ не изменяется.

На каждом *k*-м шаге преобразований прямого хода элементы матриц изменяются по ***формулам прямого хода метода Гаусса***:

$$\begin{aligned} a_{mi} &= a_{mi} - a_{ki}\frac{a_{mk}}{a_{kk}}, \qquad k = 1,...,\ n-1,\quad i = k,...,\ n; \\ b_m &= b_m - b_k\frac{a_{mk}}{a_{kk}}\ , \qquad m = k+1,...,\ n. \end{aligned} \qquad (1.4)$$

Элементы a_{kk} называются главными. Заметим, что если в ходе расчетов по данному алгоритму на главной диагонали окажется нулевой элемент $a_{kk} = 0$, то произойдет переполнение разрядной сетки компьютера. Для того чтобы избежать этого, следует каждый цикл по *k* начинать с перестановки строк: среди элементов *k*-го столбца a_{mk}, $k \le m \le n$ находят номер *p* главного, т. е. наибольшего по модулю, и меняют местами строки *k* и *p*. Такой выбор главного элемента значительно повышает устойчивость алгоритма к ошибкам округления, т. к. в формулах (1.4) при этом производится умножение на числа a_{mk}/a_{kk}, меньшие единицы, и ошибка, возникшая ранее, уменьшается.

1.2.2. Метод прогонки

Многие задачи (например решение дифференциальных уравнений 2-го порядка) приводят к необходимости решения СЛАУ с трехдиагональной матрицей:

$$\begin{vmatrix} q_1 & r_1 & 0 & 0 & .. & 0 & 0 & 0 \\ p_2 & q_2 & r_2 & 0 & .. & 0 & 0 & 0 \\ 0 & p_3 & q_3 & r_3 & .. & 0 & 0 & 0 \\ .. & .. & .. & .. & .. & .. & .. & .. \\ 0 & 0 & 0 & 0 & .. & p_{n-1} & q_{n-1} & r_{n-1} \\ 0 & 0 & 0 & 0 & .. & 0 & p_n & q_n \end{vmatrix} \times \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ .. \\ x_{n-1} \\ x_n \end{vmatrix} = \begin{vmatrix} d_1 \\ d_2 \\ d_3 \\ .. \\ d_{n-1} \\ d_n \end{vmatrix}, \qquad (1.5)$$

или коротко эту систему записывают в виде

$$\begin{aligned} & q_1 x_1 + r_1 x_2 = d_1; \\ & p_i x_{i-1} + q_i x_i + r_i x_{i+1} = d_i, \quad 2 \le i \le n-1; \\ & p_n x_{n-1} + q_n x_n = d_n. \end{aligned} \tag{1.6}$$

В этом случае расчетные формулы метода Гаусса значительно упрощаются. После исключения поддиагональных элементов в результате прямого хода метода Гаусса и последующего деления каждого уравнения на диагональный элемент систему (1.5) можно привести к виду

$$\begin{vmatrix} 1 & -\xi_1 & 0 & .. & 0 & .. & 0 & 0 \\ 0 & 1 & -\xi_2 & .. & 0 & .. & 0 & 0 \\ .. & .. & .. & .. & .. & .. & .. & .. \\ 0 & 0 & 0 & .. & 0 & .. & 1 & -\xi_{n-1} \\ 0 & 0 & 0 & .. & 0 & .. & 0 & 1 \end{vmatrix} \times \begin{vmatrix} x_1 \\ x_2 \\ .. \\ x_{n-1} \\ x_n \end{vmatrix} = \begin{vmatrix} \eta_1 \\ \eta_2 \\ .. \\ \eta_{n-1} \\ \eta_n \end{vmatrix}. \tag{1.7}$$

При этом ***формулы прямого хода*** для вычисления ξ_i, η_i имеют вид

$$\begin{aligned} & \xi_1 = -r_1 / q_1; \qquad \eta_1 = d_1 / q_1; \\ & \xi_i = -r_i / (q_i + p_i \xi_{i-1}); \quad \eta_i = (d_i - p_i \eta_{i-1}) / (q_i + p_i \xi_{i-1}); \\ & i = 2, 3, ..., n-1. \end{aligned} \tag{1.8}$$

Когда такое преобразование (прямой ход) сделано, ***формулы обратного хода*** метода Гаусса получаются в виде

$$\begin{aligned} & x_n = (d_n - p_n \eta_{n-1}) / (q_n + p_n \xi_{n-1}); \\ & x_i = \xi_i x_{i+1} + \eta_i; \\ & i = n-1, \ n-2, \ ..., \ 1. \end{aligned} \tag{1.9}$$

Расчетные формулы (1.8), (1.9) получили название ***метод прогонки***. Достаточным условием того, что в формулах метода прогонки не произойдет деления на нуль и расчет будет устойчив относительно погрешностей округления, является выполнение неравенства $|q_i| \ge |p_i| + |r_i|$ (хотя бы для одного i должно быть строгое неравенство).

1.2.3. Метод квадратного корня

Предназначен для решения СЛАУ с симметричной матрицей. Этот метод основан на представлении такой матрицы в виде произведения трех матриц: $A = S^T \cdot D \cdot S$, где D – диагональная с элементами $d_i = \pm 1$; S – верхняя треугольная ($s_{ik} = 0$, если $i > k$, причем $s_{ii} > 0$); S^T – транспонированная нижняя треугольная. Матрицу S можно по аналогии с числами трактовать как корень квадратный из матрицы A, отсюда и название метода.

Если *S* и *D* известны, то решение исходной системы $A\cdot\vec{x}=S^T\cdot D\cdot S\cdot\vec{x}=\vec{b}$ сводится к последовательному решению трех систем – двух треугольных и одной диагональной:

$$S^T\cdot\vec{z}=\vec{b};\ \ D\vec{y}=\vec{z};\ \ \vec{S}\vec{x}=\vec{y}\,, \tag{1.10}$$

где $\vec{z}=DS\vec{x},\ \ \vec{y}=S\vec{x}$.

Решение систем (1.10) ввиду треугольности матрицы *S* осуществляется по формулам, аналогичным обратному ходу метода Гаусса:

$$y_1=b_1/s_{11}d_1;\quad y_i=(b_i-\sum_{k=1}^{i-1}d_k y_k s_{ki})/s_{ii}d_i;\ \ i=2,\ 3\ ,...,\ n;$$

$$x_n=y_n/s_{nn};\quad x_i=(y_i-\sum_{k=i+1}^{n}s_{ik}x_k)/s_{ii};\ \ i=n-1,\ n-2,\ ...,\ 1.$$

Нахождение элементов матрицы *S* (извлечение корня из *A*) осуществляется по рекуррентным формулам:

$$\begin{aligned}
&d_k=sign(a_{kk}-\sum_{i=1}^{k-1}d_i\left|s_{ik}\right|^2);\\
&s_{kk}=\sqrt{\left|a_{kk}-\sum_{i=1}^{k-1}d_i\left|s_{ik}\right|^2\right|};\\
&k=1,2,\ldots,n;\\
&s_{kj}=(a_{kj}-\sum_{i=1}^{k-1}d_i s_{ik}s_{ij})/(s_{kk}d_k);\\
&j=k+1,\ k+2,\ \ldots,\ n.
\end{aligned} \tag{1.11}$$

В этих формулах сначала полагают *k=1* и последовательно вычисляют $d_1=sign(a_{11});\ \ s_{11}=\sqrt{|a_{11}|}$ и все элементы первой строки матрицы $S\ \ (s_{1j},j>1)$, затем полагают *k*=2, вычисляют s_{22} и вторую строку s_{1j} для *j*>2 и т. д.

Метод квадратного корня почти вдвое эффективнее метода Гаусса, т. к. полезно использует симметричность матрицы.

Функция **sign(x)** возвращает –1 для всех *x*<0 и +1 для всех *x*>.

1.3. Итерационные методы решения СЛАУ

1.3.1. Метод простой итерации

В соответствии с общей идеей итерационных методов исходная система (1.1) должна быть приведена к виду, разрешенному относительно $\vec{x}$:

$$\vec{x}=G\vec{x}+\vec{c}=\varphi(\vec{x})\,, \tag{1.12}$$

где *G* – матрица; $\vec{c}$ – столбец свободных членов.

При этом решение (1.12) должно совпадать с решением (1.1). Затем строится рекуррентная последовательность первого порядка в виде

$$\vec{x}^k = \varphi(\vec{x}^{k-1}) = G\vec{x}^{k-1} + \vec{c}, \quad k = 1,\ 2,\ \ldots .$$

Для начала вычислений задается некоторое начальное приближение $\vec{x}^0$ (например, $x_1^0 = 1,\ \ldots,\ x_n^0 = 1$), для окончания – некоторое малое ε.

Получаемая последовательность будет сходиться к точному решению, если норма матрицы $\|G\| < 1$.

Привести исходную систему к виду (1.12) можно различными способами, например

$$\vec{x} = \vec{x} + \alpha(A\vec{x} - \vec{b}) = (E + \alpha A)\vec{x} - \alpha\vec{b} = G\vec{x} + \vec{c},$$

где E – единичная матрица; α – некоторый параметр, подбирая который, можно добиться, чтобы $\|G\| = \|E + \alpha A\| < 1$.

В частном случае, если исходная матрица A имеет преобладающую главную диагональ, т. е. $|a_{ii}| > \sum\limits_{\substack{k=1 \\ k \neq i}}^{n} |a_{ik}|$, то преобразование (1.1) к (1.12) можно осуществить просто, решая каждое i-е уравнение относительно x_i. В результате получим следующую рекуррентную формулу:

$$x_i^k = \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{k-1} \right) / a_{ii} = \sum_{j=1}^{n} g_{ij} x_j^{k-1} + c_i; \qquad (1.13)$$

$$g_{ij} = -a_{ij} / a_{ii}; \quad g_{ii} = 0; \quad c_i = b_i / a_{ii}.$$

1.3.2. Метод Зейделя

Метод Зейделя является модификацией метода простой итерации. Суть его состоит в том, что при вычислении очередного приближения $x_i^k (2 \le i \le n)$ в формуле (1.13) используются вместо $x_1^{k-1}, \ldots, x_{i-1}^{k-1}$ уже вычисленные ранее $x_1^k, \ldots, x_{i-1}^k$, т. е. (1.13) преобразуется к виду

$$x_i^k = \sum_{j=1}^{i-1} g_{ij} x_j^k + \sum_{j=i+1}^{n} g_{ij} x_j^{k-1} + c_i. \qquad (1.14)$$

Такое усовершенствование позволяет ускорить сходимость итераций почти в два раза. Кроме того, данный метод может быть реализован на компьютере без привлечения дополнительного массива, т. к. полученное новое x_i^k сразу засылается на место старого.

1.3.3. Понятие релаксации

Методы простой итерации и Зейделя сходятся примерно так же, как геометрическая прогрессия со знаменателем $\|G\|$. Если норма матрицы G близка к 1, то сходимость очень медленная. Для ускорения сходимости используется метод релаксации. Суть его в том, что полученное по методу простой итерации или Зейделя очередное значение x_i^k пересчитывается по формуле

$$x_i^k = \omega x_i^k + (1-\omega) x_i^{k-1}, \tag{1.15}$$

где $0 < \omega \le 2$ –- параметр релаксации.

Если $\omega < 1$ *– нижняя релаксация,* если $\omega > 1$ *– верхняя релаксация.* Параметр ω подбирают так, чтобы сходимость метода достигалась за минимальное число итераций.

1.4. Индивидуальные задания

Составить программу решения СЛАУ n-го порядка одним из методов согласно варианту. Вычисления оформить в виде подпрограммы, помещенной в библиотечный модуль.

1. Метод Гаусса.
2. Метод квадратного корня.
3. Метод прогонки.
4. Метод простой итерации.
5. Метод Зейделя.

Схемы алгоритмов некоторых методов решения систем линейных алгебраических уравнений приведены ниже (рис. 1.2, 1.3, 1.4).

1.5. Контрольные вопросы

1. Что понимается под корректностью СЛАУ?
2. Решите по методу Гаусса заданную систему из трех уравнений.
3. В чем суть метода квадратного корня?
4. Когда используются методы прогонки и квадратного корня?
5. Решите заданную систему трех уравнений методом простой итерации и методом Зейделя.
6. Для чего нужна релаксация? Ее суть.

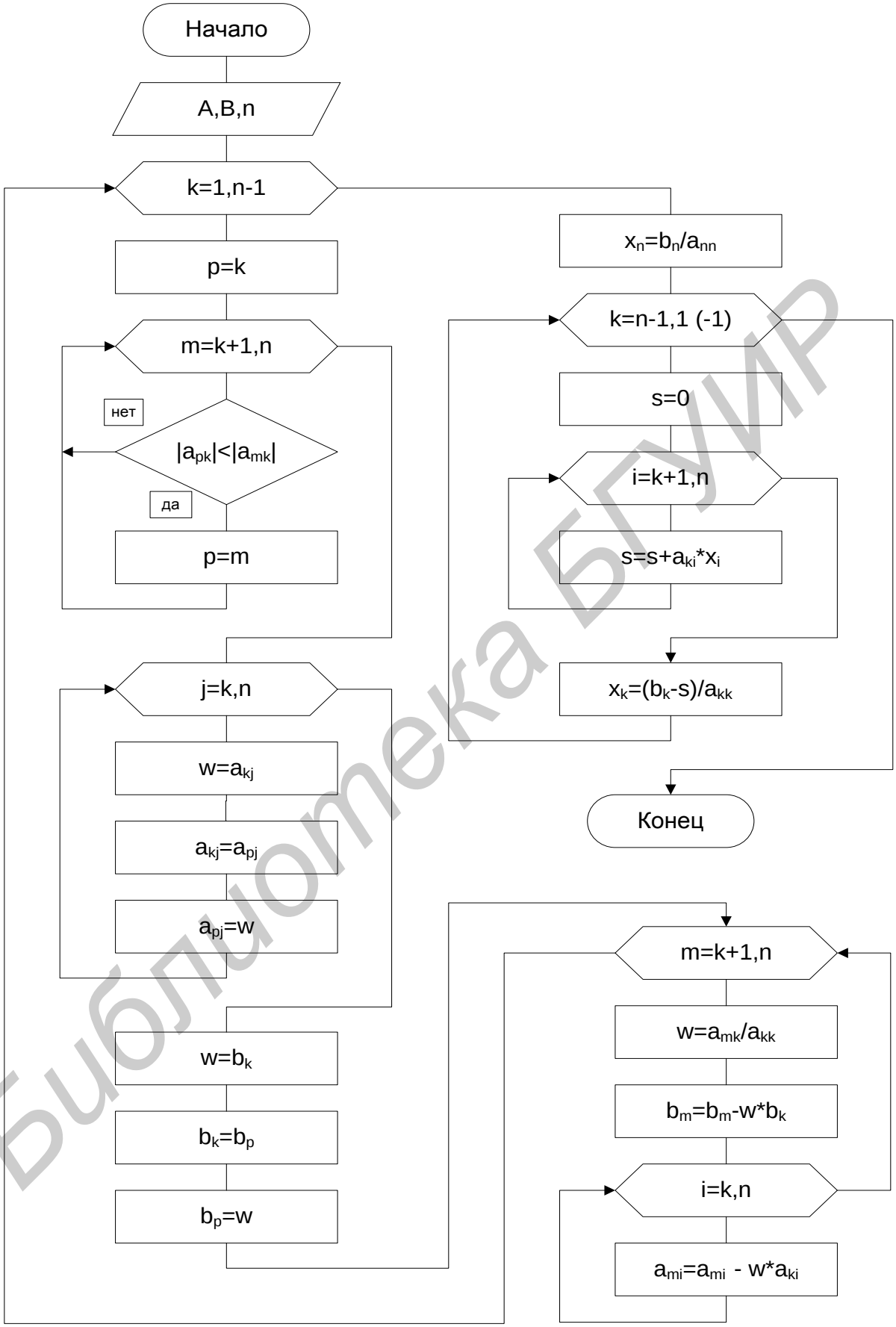

Рис. 1.2. Схема алгоритма метода Гаусса

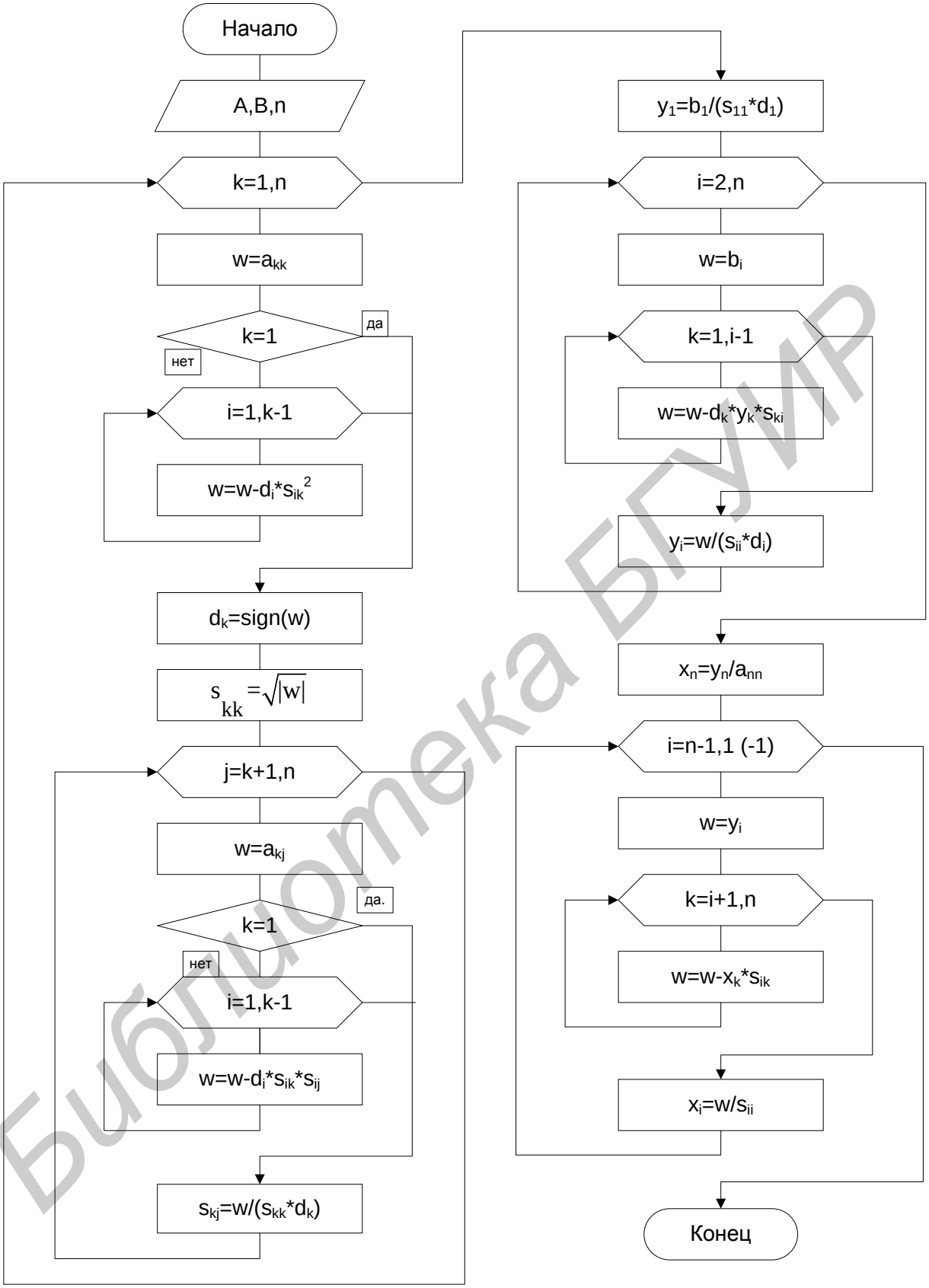

Рис. 1.3. Схема алгоритма метода квадратного корня

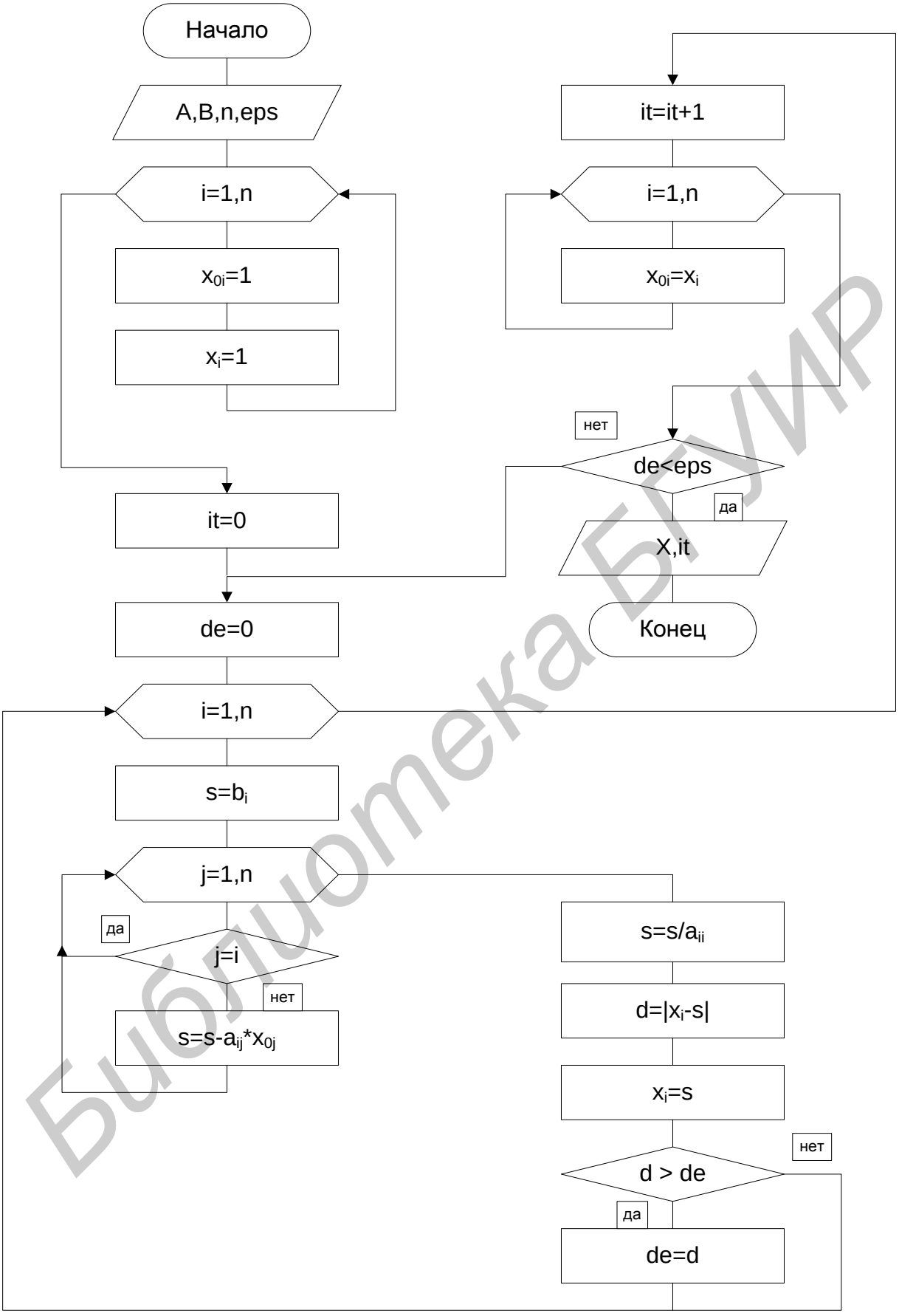

Рис. 1.4. Схема алгоритма метода простой итерации

ТЕМА 2. АППРОКСИМАЦИЯ ФУНКЦИЙ

Цель работы: изучить основные методы и алгоритмы аппроксимации функций.

2.1. Зачем нужна аппроксимация функций

В окружающем нас мире все взаимосвязано, поэтому одной из наиболее часто встречающихся задач является установление характера зависимости между различными величинами, что позволяет по значению одной величины определить значение другой. Математической моделью зависимости одной величины от другой является понятие функции *y=f(x)*.

В практике расчетов, связанных с обработкой экспериментальных данных, вычислением *f(x),* разработкой вычислительных методов, встречаются следующие две ситуации:

1. Как установить вид функции *y=f(x),* если она неизвестна? Предполагается при этом, что задана таблица ее значений x_i, y_i , $i = 1,...,m$, которая получена либо из экспериментальных измерений, либо из сложных расчетов.

2. Как упростить вычисление известной функции *f(x)* или же ее характеристик ($f'(x)$, $\max f(x)$), если *f(x)* слишком сложная?

Ответы на эти вопросы даются теорией аппроксимации функций, ***основная задача*** которой состоит в нахождении функции $y=\varphi(x)$, близкой (т. е. аппроксимирующей) в некотором нормированном пространстве к исходной функции *y=f(x).* Функцию $\varphi(x)$ при этом выбирают такой, чтобы она была максимально удобной для последующих расчетов.

Основной подход к решению этой задачи заключается в том, что $\varphi(x)$ выбирается зависящей от нескольких свободных параметров $\vec{c} = (c_1, c_2, ..., c_n)$, т. е. $y = \varphi(x) = \varphi(x, c_1, ..., c_n) = \varphi(x, \vec{c})$, значения которых подбираются из некоторого условия близости *f(x)* и $\varphi(x)$.

Обоснование способов нахождения удачного вида функциональной зависимости $\varphi(x, \vec{c})$ и подбора параметров $\vec{c}$ составляет задачу ***теории аппроксимации функций.***

В зависимости от способа подбора параметров $\vec{c}$ получают различные ***методы аппроксимации***, среди которых наибольшее распространение получили ***интерполяция*** и ***среднеквадратичное приближение,*** частным случаем которого является ***метод наименьших квадратов.***

Наиболее простой, хорошо изученной и нашедшей широкое применение в настоящее время является ***линейная аппроксимация,*** при которой выбирают функцию $\varphi(x, \vec{c})$, линейно зависящую от параметров $\vec{c}$, т. е. в виде обобщенного многочлена:

$$\varphi(x, \vec{c}) = c_1\varphi_1(x) + ... + c_n\varphi_n(x) = \sum_{k=1}^{n} c_k\varphi_k(x), \qquad (2.1)$$

где $\{\varphi_1(k), ..., \varphi_n(x)\}$ – известная система линейно независимых (базисных) функций. В качестве $\varphi_k\ x$ в принципе могут быть выбраны любые элементарные функции: например, тригонометрические, экспоненты, логарифмические или комбинации таких функций. Важно, чтобы система базисных функций была *полной*, т. е. обеспечивающей аппроксимацию *f(x)* многочленом (2.1) с заданной точностью при $n \to \infty$.

Приведем хорошо известные и часто используемые системы. При интерполяции обычно используется система линейно независимых функций $\varphi_k\ x = x^{k-1}$. Для среднеквадратичной аппроксимации удобнее в качестве $\varphi_k\ x$ брать ортогональные на интервале [–1, 1] многочлены Лежандра:

$$\varphi_1\ x = 1;\ \varphi_2\ x = x;\ \varphi_{k+1}\ x = \left[\ 2k+1\ x\varphi_k\ x - k\varphi_{k-1}\ x\ \right],\ k = 2,3, ..., n\ ;$$

$$\int_{-1}^{1} \varphi_k\ x \cdot \varphi_l\ x\ dx = 0;\ k \neq l\,.$$

Заметим, что если функция $f(x)$ задана на отрезке *[a, b]*, то при использовании этой системы необходимо предварительно осуществить преобразование координат $x' = \left(x - \frac{b+a}{2} \right) \frac{2}{b-a}$, приводящее интервал $a \le x \le b$ к интервалу $-1 \le x' \le 1$.

Для аппроксимации периодических функций используют ортогональную на *[a, b]* систему тригонометрических функций $\left\{ \varphi_k\ x = \cos(2k\pi \frac{x-a}{b-a}),\right.$ $\left. \psi_k\ x = \sin(2k\pi \frac{x-a}{b-a}) \right\}$. В этом случае обобщенный многочлен (20.1) записывается в виде $y = \sum_{k=1}^{n} c_k \varphi_k(x) + d_k \psi_k(x)$.

2.2. Что такое интерполяция

Интерполяция является одним из способов аппроксимации функций. Суть ее состоит в следующем. В области значений x, представляющей некоторый интервал [*a*, *b*], где функции *f(x)* и $\varphi(x)$ должны быть близки, выбирают упорядоченную систему точек (узлов) $x_1 < x_2 < ... < x_n$, (обозначим $\vec{x} = (x_1, ..., x_n)$), число которых равно количеству искомых параметров $c_1, c_2, ..., c_n$. Далее параметры $\vec{c}$ подбирают такими, чтобы функция $\varphi(x, \vec{c})$ совпадала с *f(x)* в этих узлах, $\varphi\ x_i, \vec{c} = f\ x_i\ ,\ i = 1... n$, для чего решают полученную систему n алгебраических, в общем случае нелинейных, уравнений.

В случае линейной аппроксимации (2.1) система для нахождения коэффициентов $\vec{c}$ линейна и имеет следующий вид:

$$\sum_{k=1}^{n} c_k \varphi_k(x_i) = f_i; \;\; i = 1,\ 2,\ ...,\ n; \;\; f_i = f(x_i). \qquad (2.2)$$

Система базисных функций $\varphi_k\ x$, используемых для интерполяции, должна быть **чебышевской**, т. е. такой, чтобы определитель матрицы системы (2.2) был отличен от нуля и, следовательно, задача интерполяции имела единственное решение.

Для большинства практически важных приложений при интерполяции наиболее удобны обычные алгебраические многочлены, ибо они легко обрабатываются.

Интерполяционным многочленом называют алгебраический многочлен степени *n*–1, совпадающий с аппроксимируемой функцией в выбранных *n* точках.

Общий вид алгебраического многочлена

$$\varphi(x,\ \vec{c}) = P_{n-1}(x) = c_1 + c_2 x + c_3 x^2 + ... + c_n x^{n-1} = \sum_{k=1}^{n} a_k x^{k-1}. \qquad (2.3)$$

Матрица системы (2.2) в этом случае имеет вид

$$G = \begin{bmatrix} 1 & x_1 & .. & x_1^{n-1} \\ 1 & x_2 & .. & x_2^{n-1} \\ .. & .. & .. & .. \\ 1 & x_n & .. & x_n^{n-1} \end{bmatrix}; \qquad |G| = \prod_{n \ge k > m \ge 0} x_k - x_m \qquad (2.4)$$

и ее определитель (это определитель Вандермонда) отличен от нуля, если точки x_i разные. Поэтому задача (2.2) имеет единственное решение, т. е. для заданной системы различных точек существует единственный интерполяционный многочлен.

Погрешность аппроксимации функции *f(x)* интерполяционным многочленом степени *n*–1, построенным по *n* точкам, можно оценить, если известна ее производная порядка *n*, по формуле

$$\varepsilon = \| f(x) - P_{n-1}(x) \|_C \le \sqrt{\frac{2}{(n-1)\pi}} \left\| \frac{d^n f(x)}{dx^n} \right\|_C (h/2)^n,\ h = \max_i |x_i - x_{i-1}|. \qquad (2.5)$$

Из (2.5) следует, что при $h \to 0$ порядок погрешности *p* при интерполяции алгебраическим многочленом равен количеству выбранных узлов *p*=*n*. Величина ε может быть сделана малой как за счет увеличения *n*, так и уменьшения *h*. В практических расчетах используют, как правило, многочлены невысокого порядка в связи с тем, что с ростом *n* резко возрастает погрешность вычисления самого многочлена из-за погрешностей округления.

2.3. Многочлены и способы интерполяции

Один и тот же многочлен можно записать по-разному, например $1 - 2x + x^2 = (x-1)^2$. Поэтому в зависимости от решаемых задач применяют раз-

личные виды представления интерполяционного многочлена и способы интерполяции.

Наряду с общим представлением (2.3) наиболее часто в приложениях используют интерполяционные многочлены в форме Лагранжа и Ньютона. Их особенность в том, что не надо находить параметры $\vec{c}$, т. к. многочлены в этой форме прямо записаны через значения таблицы $(x_i,\ y_i),\ i=1\ \ldots\ n$.

2.3.1. Интерполяционный многочлен Ньютона

Аппроксимация в виде (2.1), дополненная системой (2.2), называется *аппроксимацией общего вида.* Недостатком такого вида аппроксимации является необходимость решения системы линейных алгебраических уравнений (2.2) для определения коэффициентов $\vec{c}=(c_1,\ c_2,\ \ldots,\ c_n)$. Ньютоном была предложена форма записи многочлена (2.3) в виде, не требующем решения системы линейных алгебраических уравнений:

$$N_{n-1}(x_T)=f_1+\sum_{k=1}^{n-1}(x_T-x_1)(x_T-x_2)\ldots(x_T-x_k)\Delta_k, \qquad (2.6)$$

где x_T – текущая точка, в которой надо вычислить значение многочлена; Δ_k – разделенные разности порядка *k*, которые вычисляются по следующим рекуррентным формулам:

$$\Delta_1 f(x_i x_{i+1})=\frac{f_i-f_{i+1}}{x_i-x_{i+1}};\quad \Delta_2 f(x_i x_{i+1} x_{i+2})=\frac{\Delta_i f(x_i x_{i+1})-\Delta_1 f(x_{i+1} x_{i+2})}{x_i-x_{i+2}}\ldots .$$

2.3.2. Линейная и квадратичная интерполяция

Иногда при интерполяции по заданной таблице из *m>3* точек применяют квадратичную *n=3* или линейную *n=2* интерполяцию. В этом случае для приближенного вычисления значения функции *f(x)* в текущей точке x_T находят в таблице ближайший к этой точке *i*-узел из общей таблицы, строят интерполяционный многочлен Ньютона первой или второй степени по формулам

$$N_1(x_T)=f_{i-1}+(x_T-x_{i-1})\frac{f_i-f_{i-1}}{x_i-x_{i-1}};\quad x_{i-1}\le x_T\le x_i; \qquad (2.7)$$

$$N_2(x_T)=N_1(x_T)+(x_T-x_{i-1})(x_T-x_i)\frac{\left(\frac{f_{i-1}-f_i}{x_{i-1}-x_i}\right)-\left(\frac{f_i-f_{i+1}}{x_i-x_{i+1}}\right)}{x_{i-1}-x_{i+1}};\quad x_{i-1}\le x_T\le x_{i+1}$$

и за значение *f(x)* принимают $N_1(x)$ (*линейная интерполяция*) или $N_2(x)$ (*квадратичная интерполяция*).

2.3.3. Интерполяционный многочлен Лагранжа

Лагранжем была предложена своя форма записи многочлена (2.3) в виде, не требующем решения системы линейных алгебраических уравнений:

$$L_{n-1}(x_T) = \sum_{k=1}^{n} f_k \prod_{\substack{i=1 \\ i \neq k}}^{n} \frac{x_T - x_i}{x_k - x_i} \ . \qquad (2.8)$$

Произведение $\prod_{\substack{i=1 \\ i \neq k}}^{n} \frac{x_T - x_i}{x_k - x_i}$ выбрано так, что во всех узлах, кроме *k-го*, оно обращается в нуль, а в *k-м* узле оно равно единице:

$$\prod_{\substack{i=1 \\ i \neq k}}^{n} \frac{x_T - x_i}{x_k - x_i} = \begin{cases} 1, & при \quad x_T = x_k \\ 0, & при \quad x_T \neq x_k. \end{cases}$$

Поэтому из выражения (2.8) видно, что $L_{n-1}(x_i) = f_i$.

2.3.4. Интерполяция общего вида

Следует отметить, что ввиду громоздкости многочлены Ньютона и Лагранжа уступают по эффективности расчета многочлену общего вида (2.3), если предварительно найдены коэффициенты $\vec{c}$.

Поэтому когда требуется производить многократные вычисления многочлена, построенного по одной таблице, оказывается выгодно вначале один раз найти коэффициенты $\vec{c}$ (решив систему линейных алгебраических уравнений) и затем использовать формулу (2.3). Коэффициенты $\vec{c}$ находят прямым решением системы (2.2) с матрицей (2.4), затем вычисляют его значения по экономно программируемой формуле (алгоритм Горнера)

$$P_{n-1}\ x\ = c_1 + x(c_2 + \ldots\ x(c_{n-2} + x(c_{n-1} + xc_n)\ \ldots)\,. \qquad (2.9)$$

2.4. Среднеквадратичная аппроксимация

Суть среднеквадратичной аппроксимации заключается в том, что параметры $\vec{c}$ функции $\varphi\ x,\vec{c}$ подбираются такими, чтобы обеспечить минимум квадрата расстояния между функциями *f(x)* и $\varphi\ \ x,\vec{c}$, т. е. из условия

$$\min_{c_1,\ldots,c_n} \left\| f(x) - \varphi\ \ x,\vec{c}\ \right\|_{L_2}. \qquad (2.10)$$

В случае линейной аппроксимации (2.1) задача (2.10) сводится к решению СЛАУ для нахождения необходимых коэффициентов $\vec{c}$:

$$\sum_{k=1}^{n} \varphi_i \varphi_k\ _{L_2} \cdot c_k = \ f, \varphi_i\ _{L_2}; \quad i = 1,\ldots,n, \qquad (2.11)$$

где $\varphi_i \varphi_k\ _{L_2}$, $f, \varphi_i\ _{L_2}$ – скалярные произведения в L_2.

Матрица системы (2.11) симметричная, и ее следует решать методом квадратного корня.

Особенно просто эта задача решается, если выбрана ***ортогональная система функций*** $\varphi_k\ \ x$, т. е. такая, что

$$\varphi_i \varphi_k = \begin{cases} 0, & i \neq k, \\ \|\varphi_k\|^2, & i = k. \end{cases}$$

Тогда матрица СЛАУ (2.13) диагональная и параметры $\vec{c}$ находятся по формуле

$$c_k = \frac{f,\ \varphi_k}{\|\varphi_k\|^2}.$$

В этом случае представление (2.1) называется ***обобщенным рядом Фурье***, а c_k называются коэффициентами Фурье.

2.4.1. Метод наименьших квадратов (МНК)

МНК является частным случаем среднеквадратичной аппроксимации. При использовании МНК аналогично задаче интерполяции в области значений *x*, представляющей некоторый интервал [*a, b*], где функции *f(x)* и *φ(x)* должны быть близки, выбирают систему различных точек (узлов) $x_1, ..., x_m$, число которых обычно больше, чем количество искомых параметров $c_1, \ldots, c_n$, $m \geq n$. Далее, потребовав, чтобы сумма квадратов невязок во всех узлах была минимальна:

$$\min_{\vec{c}} \sum_{j=1}^{m} [f(x_j) - \varphi(x_j, \vec{c})]^2 = \min_{\vec{c}} \sum_{j=1}^{m} \delta_j^2 = \min_{\vec{c}} \delta(\vec{c}), \tag{2.12}$$

находим из этого условия параметры $c_1, ..., c_n$.

В общем случае эта задача сложная и требует применения численных методов оптимизации. Однако в случае линейной аппроксимации (2.1), составляя условия минимума суммы квадратов невязок во всех точках $\delta\ \vec{c}$ (в точке минимума все частные производные должны быть равны нулю):

$$\frac{\partial \delta(c_1, c_2, ..., c_n)}{\partial c_i} = 0, \quad i = 1, 2, ..., n, \tag{2.13}$$

получаем систему *n* линейных уравнений относительно *n* неизвестных $c_1, ..., c_n$ следующего вида:

$$\sum_{k=1}^{n} (\vec{\varphi}_i\ \vec{\varphi}_k) c_k = (\vec{f},\ \vec{\varphi}_i), \quad i = 1, ..., n \text{ или } G\vec{c} = \vec{b}, \tag{2.14}$$

где $\vec{\varphi}_i = \varphi_i(x_1), \varphi_i(x_2), \ldots, \varphi_i(x_m)$, $\vec{f} = f_1, \ldots, f_m$ – векторы-столбцы функций.

Элементы матрицы *G* и вектора $\vec{b}$ в (2.14) определяются выражениями

$$\left.\begin{aligned} g_{ik} &= (\vec{\varphi}_i\ \ \vec{\varphi}_k) = \sum_{j=1}^{m} \varphi_i(x_j) \varphi_k(x_j), \\ b_i &= (\vec{f},\ \vec{\varphi}_i) = \sum_{j=1}^{m} f_j\, \varphi_i(x_j). \end{aligned}\right\}$$ скалярные произведения векторов.

Система (2.14) имеет симметричную матрицу *G* и решается методом квадратного корня.

При аппроксимации по МНК обычно применяют такие функции φ_i , которые используют особенности решаемой задачи и удобны для последующей обработки.

2.5. Индивидуальные задания

Составить программу аппроксимации функции *f*(*x*) соответствующим методом согласно варианту из табл. 2.1. Во всех вариантах требуется аппроксимировать заданную исходную функцию *f(x)* многочленом на интервале [*a, b*]. Задается количество неизвестных параметров *n*, вид аппроксимации и количество точек *m*, в которых задана функция. Таблица исходной функции $y_i=f(x_i)$ вычисляется в точках $x_i = a + (i-1)(b-a)/(m-1),\ i=1,\ m.$ Используя полученную таблицу $(x_i,\ y_i)$, требуется вычислить значения функций $f(x_j)$, $\varphi(x_j, \vec{c})$ и погрешность $d(x_j) = f(x_j) - \varphi(x_j, \vec{c})$ в точках $x_j = a + j*(b-a)/20;\ \ j=0,\ 20.$

Таблица 2.1

Функция f(x)	*a*	*b*	*m*	*n*	*Вид аппроксимации*
1. $4x - 7\sin(x)$	–2	3	11	3	Метод наименьших квадратов
2. $x^2 - 10\sin^2(x)$	0	3	4	4	Ньютон
3. $\ln(x) - 5\cos(x)$	1	8	4	4	Лагранж
4. $e^x / x^3 - \sin^3(x)$	4	7	4	4	Общего вида
5. $\ln(x) - 5\sin^2(x)$	3	6	11	2	Линейная
6. $\sin^2(x) - x/5$	0	4	11	3	Квадратичная

Схемы алгоритмов некоторых методов аппроксимации приведены ниже (рис. 2.1, 2.2).

2.6. Контрольные вопросы

1. Как ставится задача линейной аппроксимации функций?

2. Что такое интерполяция, ее геометрическая интерпретация?

3. Напишите интерполяционный многочлен Ньютона 2-го порядка.

4. Напишите интерполяционный многочлен Лагранжа 2-го порядка.

5. Как осуществляется аппроксимация по методу наименьших квадратов и его геометрическая интерпретация?

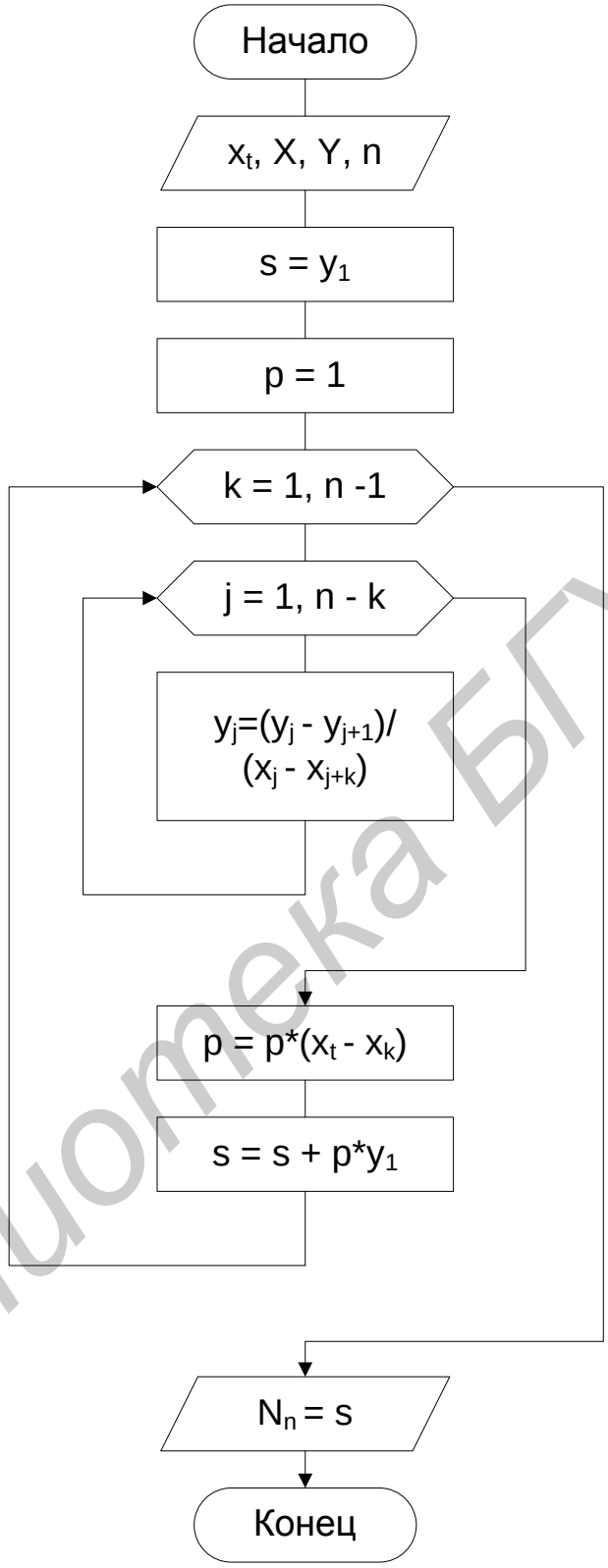

Рис. 2.1. Схема алгоритма расчета интерполяционного многочлена Ньютона

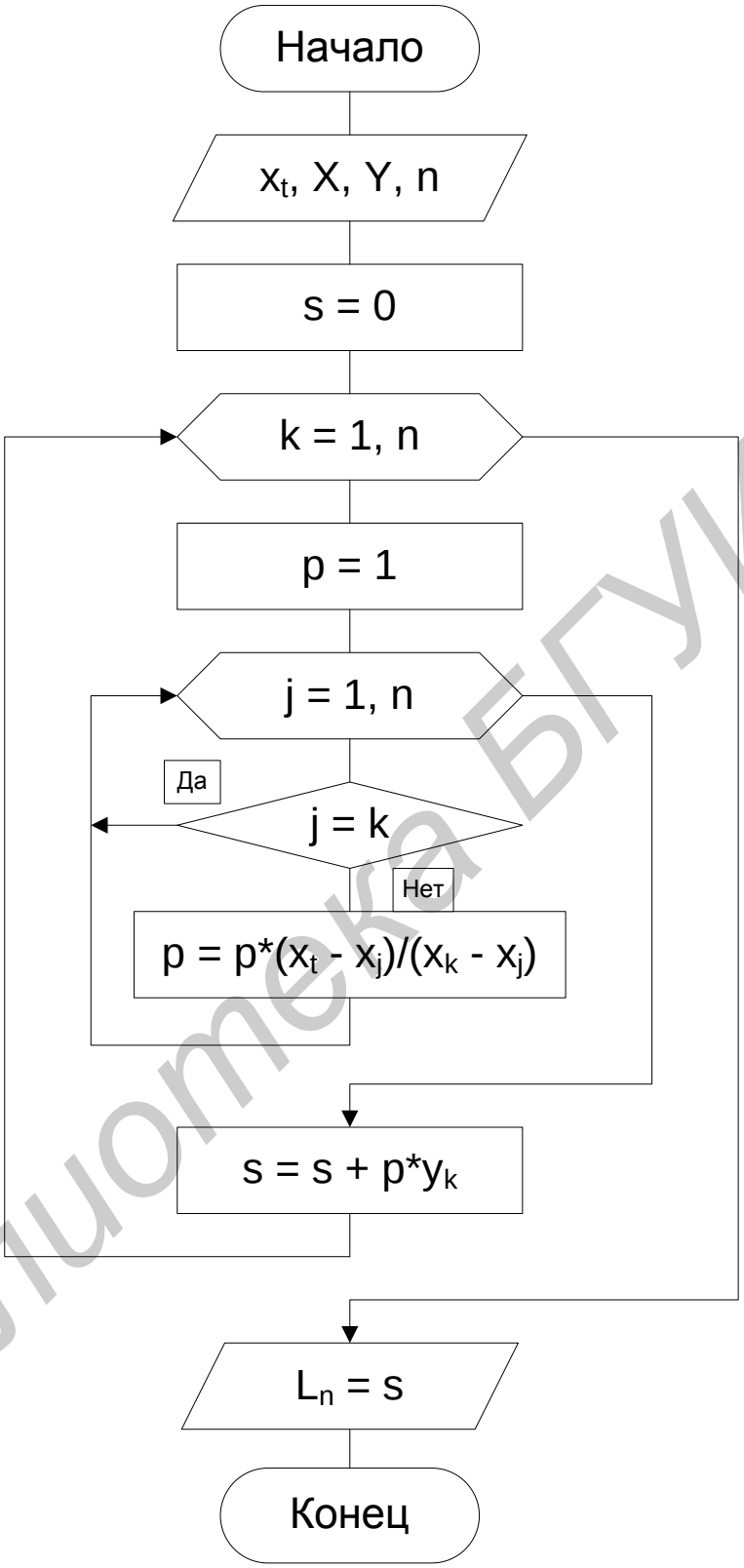

Рис. 2.2. Схема алгоритма расчета интерполяционного многочлена Лагранжа

ТЕМА 3. ВЫЧИСЛЕНИЕ ПРОИЗВОДНЫХ И ИНТЕГРАЛОВ

Цель работы: изучить основные методы и алгоритмы вычисления производных и интегралов.

3.1. Формулы численного дифференцирования

Формулы для расчета производной $d^m f / dx^m$ в точке x получаются следующим образом. Берется несколько близких к x узлов $x_1, x_2, ..., x_n$ ($n \geq m+1$), называемых ***шаблоном*** (точка x может быть одним из узлов). Вычисляются значения $f_i = f(x_i)$ в узлах шаблона, строится интерполяционный многочлен Ньютона и после взятия производной от этого многочлена $d^m P_{n-1} / dx^m$ получается выражение приближенного значения производной (формула численного дифференцирования) через значения функции в узлах шаблона

$$d^m f / dx^m \approx \Lambda_m^n[f] = d^m P_{n-1} / dx^m .$$

При $n=m+1$ формула численного дифференцирования не зависит от положения точки x внутри шаблона. Так как m-я производная от полинома m-й степени $P_m(x)$ есть константа, такие формулы называют ***простейшими формулами*** численного дифференцирования.

Анализ оценки погрешности вычисления производной

$$\varepsilon = \max_{x_1 < x < x_n} \left| \frac{d^m f}{dx^m} - \Lambda_m^n[f] \right| \leq \frac{\max_x \left| f^{(m)}(x) \right|}{(n-m)} \max_i \left| x - x_i \right| \leq C h^{n-m}. \qquad (3.1)$$

$$h = \max \left| x_i - x_{i-1} \right|; \; C = const, \; n \geq m+1$$

показывает, что погрешность минимальна для значения x в центре шаблона и возрастает на краях. Поэтому узлы шаблона, если это возможно, выбираются симметрично относительно x. Заметим, что порядок погрешности при $h \to 0$ равен $n-m \geq 1$ и для повышения точности можно либо увеличивать n, либо уменьшать h (последнее более предпочтительно).

Приведем несколько важных формул для равномерного шаблона:

$$\frac{df}{dx} \approx \frac{dP_1}{dx} = \Lambda_1^2[f(x)] = \frac{f_2 - f_1}{h}; \quad x_1 \leq x \leq x_2 . \qquad (3.2)$$

Простейшая формула (3.2) имеет второй порядок погрешности в центре интервала и первый по краям.

$$\frac{df}{dx} \approx \frac{dP_2}{dx} = \Lambda_1^3[f(x)] = \frac{f_2 - f_1}{h} + (2x - x_1 - x_2) \frac{f_1 - 2f_2 + f_3}{2h^2}. \qquad (3.3)$$

Эта формула имеет второй порядок погрешности во всем интервале $x_1 \leq x \leq x_3$ и часто используется для вычисления производной в крайних точках таблицы, где нет возможности выбрать симметричное расположение узлов. Заметим, что из (3.3) получается три важные формулы второго порядка точности:

$$\frac{df(x_2)}{dx} = \Lambda_1^3[f(x_2)] = \frac{f_3 - f_1}{2h}; \qquad (3.4)$$

$$\frac{df(x_1)}{dx} = \Lambda_1^3[f(x_1)] = -\frac{3f_1 - 4f_2 + f_3}{2h}; \tag{3.5}$$

$$\frac{df(x_3)}{dx} = \Lambda_1^3[f(x_3)] = \frac{f_1 - 4f_2 + 3f_3}{2h}. \tag{3.6}$$

Для вычисления второй производной часто используют следующую простейшую формулу:

$$\frac{d^2 f}{dx^2} \approx \frac{d^2 P_2}{dx^2} = \Lambda_2[f(x)] = \frac{f_1 - 2f_2 + f_3}{h^2}; \qquad x_1 \le x \le x_3, \tag{3.7}$$

которая имеет второй порядок погрешности в центральной точке x_2.

3.2. Формулы численного интегрирования

Формулы для вычисления интеграла $U = \int_a^b f(x)dx$ получают следующим образом. Область интегрирования [*a, b*] разбивают на *m* малых отрезков с шагом $h = (b-a)/m$. Значение интеграла по всей области равно сумме интегралов на отрезках $\int_a^b f(x)\,dx = \sum_{i=1}^{m} \int_{x_{i-1}}^{x_i} f(x)\,dx$, где $x_i = a + ih$. Выбирают на каждом отрезке x_{i-1}, x_i 1 – 5 узлов и строят для каждого отрезка интерполяционный многочлен соответствующего порядка. Вычисляют интеграл от этого многочлена на отрезке. В результате получают выражение интеграла (формулу численного интегрирования) через значения подынтегральной функции в выбранной системе точек. Такие выражения называют ***квадратурными формулами.***

3.2.1. Формула средних

Формула средних получается, если на каждом *i*-м отрезке x_{i-1}, x_i взять один центральный узел $x_{i-1/2} = x_i - h/2$, соответствующий середине отрезка. Функция на каждом отрезке аппроксимируется многочленом нулевой степени (константой) $P_0(x) = f(x_{i-1/2})$. В этом случае получим

$$\int_a^b f(x)dx \approx \sum_{i=1}^{m} \int_{x_{i-1}}^{x_i} P_0(x)dx = h\sum_{i=1}^{m} f_{i-1/2} = \Sigma_{\text{ср}} f\,. \tag{3.8}$$

Погрешность формулы средних имеет второй порядок по *h:*

$$\varepsilon_{\text{ср}} = \max\left|\int_a^b f(x)dx - \Sigma_{\text{ср}} f\right| \le \left|\frac{h^2}{24}\int_a^b f''(x)dx\right|. \tag{3.9}$$

3.2.2. Формула трапеций

Формула трапеций получается при аппроксимации функции *f(x)* на каждом отрезке x_{i-1}, x_i интерполяционным многочленом первого порядка, т. е. прямой, проходящей через точки (x_{i-1}, f_{i-1}), (x_i, f_i). Площадь криволинейной фигуры заменяется площадью трапеции с основаниями f_{i-1}, f_i и высотой *h*:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{m} \int_{x_{i-1}}^{x_i} P_1(x)dx = h\sum_{i=1}^{m} \frac{f_{i-1}+f_i}{2} = h\left[\frac{f_0+f_m}{2} + \sum_{i=1}^{m-1} f_i\right] = \Sigma_{mp} f. \quad (3.10)$$

Погрешность формулы трапеций в два раза больше, чем погрешность формулы средних:

$$\varepsilon_{mp} = \max\left|\int_a^b f(x)dx - \Sigma_{mp} f\right| \le \left|-\frac{h^2}{12}\int_a^b f''(x)dx\right|. \quad (3.11)$$

3.2.3. Формула Симпсона

Формула Симпсона получается при аппроксимации функции *f*(*x*) на каждом отрезке x_{i-1}, x_i интерполяционным многочленом второго порядка (параболой) с узлами x_{i-1}, $x_{i-1/2}$, x_i. После интегрирования параболы получаем

$$\int_a^b f(x)dx \approx \sum_{i=1}^{m} \int_{x_{i-1}}^{x_i} P_2(x)dx = \frac{h}{6}\sum_{i=1}^{m}(f_{i-1} + 4f_{i-1/2} + f_i) = \Sigma_{cu} f. \quad (3.12)$$

После приведения подобных членов формула (3.12) приобретает удобный для программирования вид:

$$\sum_{cu} f = \frac{h}{3}\cdot\left[\frac{f_0+f_m}{2} + 2\sum_{i=1}^{m} f_{i-1/2} + \sum_{i=1}^{m-1} f_i\right].$$

Погрешность формулы Симпсона имеет четвертый порядок по *h*:

$$\varepsilon_{cu} = \max\left|\int_a^b f(x)dx - \Sigma_{cu} f\right| \le \left|\frac{h^4}{2880}\int_a^b f^{(4)}(x)dx\right|. \quad (3.13)$$

3.2.4. Формулы Гаусса

При построении предыдущих формул в качестве узлов интерполяционного многочлена выбирались середины и (или) концы интервала разбиения. При этом оказывается, что увеличение количества узлов не всегда приводит к уменьшению погрешности (сравни формулы средних и трапеций), т. е. за счет удачного расположения узлов можно значительно увеличить точность. ***Суть методов Гаусса*** с *n* узлами состоит в таком расположении этих *n* узлов интерполяционного многочлена на отрезке x_{i-1}, x_i, при котором достигается минимум погрешности квадратурной формулы. Детальный анализ показывает, что узлами, удовлетворяющими такому условию, являются нули ортогонального многочлена Лежандра *n-й* степени. Так, для $n=1$ один узел должен

быть выбран в центре. Следовательно, метод средних является **методом Гаусса с одним узлом**.

Для n=2 узлы на отрезке x_{i-1}, x_i должны быть выбраны следующим образом:

$$x_i^{1,2} = x_{i-1/2} \pm \frac{h}{2} \cdot 0,5\,773\,502\,692$$

и соответствующая **формула Гаусса с двумя узлами** имеет вид

$$\int_a^b f(x)dx \approx \frac{h}{2}\sum_{i=1}^{m}[f(x_i^1)+f(x_i^2)]. \qquad (3.14)$$

Порядок погрешности этой формулы при $h \to 0$ – четвертый, т.е. такой же, как у метода Симпсона, хотя используется только два узла!

Для n=3 узлы на отрезке x_{i-1}, x_i выбираются следующим образом:

$$x_i^0 = x_{i-1/2}, \qquad x_i^{1,2} = x_i^0 \pm \frac{h}{2} \cdot 0,7\,745\,966\,692$$

и соответствующая **формула Гаусса с тремя узлами** имеет вид

$$\int_a^b f(x)dx \approx \frac{h}{18}\sum_{i=1}^{m}\left[5f(x_i^1)+8f(x_i^0)+5f(x_i^2)\right]. \qquad (3.15)$$

Порядок погрешности этой формулы при $h \to 0$ – шестой, т. е. значительно выше, чем у формулы Симпсона практически при одинаковых затратах на вычисления. Следует отметить, что формулы Гаусса особенно широко применяются для вычисления несобственных интегралов специального вида, когда подынтегральная функция имеет достаточно высокие производные.

3.3. Индивидуальные задания

Составить программу для вычисления определенного интеграла соответствующим методом согласно варианту из табл. 3.1. Для каждого варианта задается интервал интегрирования [*a, b*], подынтегральная функция *f*(*x*) и указывается метод вычисления интеграла. Составить программу вычисления интеграла указанным методом и первой и второй производных функции *f*(*x*) по формулам численного дифференцирования. Вывести таблицу значений функции, ее точных и приближенных первой и второй производных f, f', $\Lambda_1^3 f$, f'', $\Lambda_2^3 f$ в точках $x_j = a + j*(b-a)/10; \quad j = 0...10$.

Таблица 3.1

Функция f(x)	*Интервал*		*Метод интегрирования*	*Значение* $\int f(x)dx$
	a	*b*		
1. $4x-7\sin(x)$	–2	3	Средних	5,983
2. $x^2-10\sin^2(x)$	0	3	Трапеций	–6,699
3. $\ln(x)-5\cos(x)$	1	8	Симпсона	8,896
4. $e^x/x^3-\sin^3(x)$	4	7	Гаусса с 2-мя узлами	6,118
5. $\sqrt{x}-\cos^2(x)$	5	8	Гаусса с 3-мя узлами	6,067

3.4. Контрольные вопросы

1. На чем основаны приближенные формулы численного дифференцирования ?

2. Дайте геометрическую интерпретацию методов средних, трапеций, Симпсона. Какой порядок погрешности они имеют?

3. В чем суть методов Гаусса? Какой порядок погрешности имеют 2- и 3-узловые методы Гаусса?

ТЕМА 4. МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Цель работы: изучить основные методы и алгоритмы нахождения простых корней нелинейных уравнений.

4.1. Как решаются нелинейные уравнения

Математической моделью многих физических процессов является функциональная зависимость $y=f(x)$. Поэтому задачи исследования различных свойств функции $f(x)$ часто возникают в инженерных расчетах. Одной из таких задач является нахождение значений x, при которых функция $f(x)$ обращается в нуль, т. е. решение уравнения

$$f(x)=0. \qquad (4.1)$$

Точное решение удается получить в исключительных случаях, и обычно для нахождения корней уравнения применяются численные методы. Решение уравнения (4.1) при этом осуществляется в два этапа:

1. Приближенное определение местоположения, характер и выбор интересующего нас корня.

2. Уточнение выбранного корня с заданной точностью ε.

На рис. 4.1 представлены три типа корней:

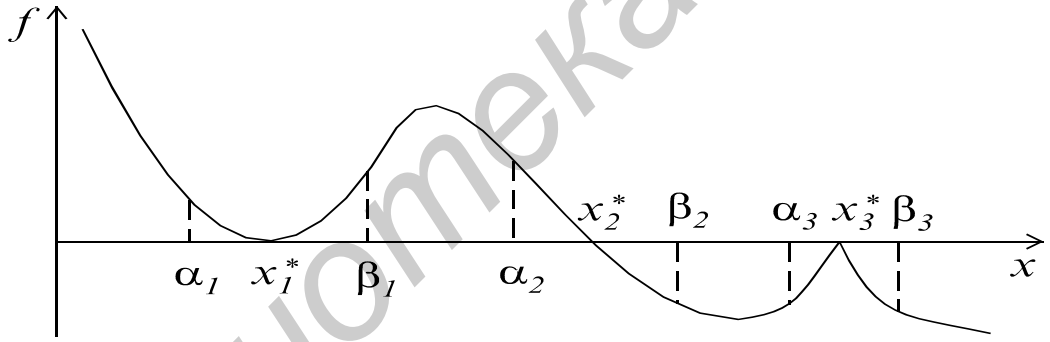

Рис. 4.1. Типы корней

а) кратный корень: $f'(x_1^*)=0, \quad f(\alpha_1)\cdot f(\beta_1)>0;$

б) простой корень: $f'(x_2^*)\neq 0, \quad f(\alpha_2)\cdot f(\beta_2)<0;$

в) вырожденный корень: $f'(x_3^*)$ не существует, $f(\alpha_3)\cdot f(\beta_3)>0.$

Как видно из рис. 4.1, в случаях «а» и «в» значение корня совпадает с точкой экстремума функции, и для нахождения таких корней рекомендуется использовать методы поиска минимума функции, описанные в теме 5.

На втором этапе вычисление значения корня с заданной точностью осуществляется одним из итерационных методов. При этом в соответствии с общей методологией *m-шагового итерационного метода*, на интервале $[\alpha,\beta]$, где находится интересующий нас корень x, выбирается m начальных значений x_0,

$x_1, \ldots, x_{m-1}$ (обычно $x_0=\alpha$, $x_{m-1}=\beta$), после чего последовательно находятся члены ($x_m, x_{m+1}, \ldots, x_{n-1}, x_n$) рекуррентной последовательности *порядка m* по правилу $x_k = \varphi(x_{k-1}, \ldots, x_{k-m})$ до тех пор, пока $|x_n - x_{n-1}| < \varepsilon$. Последнее значение x_n выбирается в качестве приближенного значения корня ($x^* \approx x_n$), найденного с погрешностью ε .

Многообразие методов определяется возможностью большого выбора законов φ. Наиболее часто используемые на практике методы описаны ниже.

4.2. Итерационные методы уточнения корней

4.2.1. Метод простой итерации

Очень часто в практике вычислений встречается ситуация, когда уравнение (4.1) записано в виде, разрешенном относительно *x*:

$$x = \varphi(x). \qquad (4.2)$$

Заметим, что переход от записи уравнения (4.1) к эквивалентной записи (4.2) можно сделать многими способами, например, положив

$$\varphi(x) = x + \psi(x) f(x), \qquad (4.3)$$

где $\psi(x)$ – произвольная, непрерывная, знакопостоянная функция (часто достаточно выбрать $\psi=const$).

В этом случае корни уравнения (4.2) являются также корнями (4.1) и наоборот. Исходя из записи (4.2), члены рекуррентной последовательности в методе простой итерации вычисляются по закону

$$x_k = \varphi(x_{k-1}), \quad k = 1,2,\ldots . \qquad (4.4)$$

Метод является одношаговым, и для начала вычислений достаточно знать одно начальное приближение $x_0 = \alpha$ или $x_0 = \beta$ или $x_0 = (\alpha + \beta)/2$.

Условием сходимости метода простой итерации, если $\varphi(x)$ дифференцируема, является выполнение неравенства $|\varphi'(\xi)| < 1$ для любого

$$\xi \in [\alpha, \beta], \quad x^* \in [\alpha, \beta]. \qquad (4.5)$$

Максимальный интервал [α, β], для которого выполняется неравенство (4.5), называется *областью сходимости*. При выполнении условия (4.5) метод сходится, если начальное приближение x_0 выбрано из области сходимости. При этом *скорость сходимости погрешности* $\varepsilon_k = |x^* - x_k|$ к нулю вблизи корня приблизительно такая же, как у геометрической прогрессии $\varepsilon_k \approx \varepsilon_{k-1} q$ со знаменателем $q \cong |\varphi'(x^*)|$, т. е. чем меньше *q*, тем быстрее сходимость, и наоборот. Поэтому при переходе от (4.1) к (4.2) функцию $\psi(x)$ в (4.3) выбирают так, чтобы выполнялось условие сходимости (4.5) для как можно большей области [α, β] и с наименьшим q. Удачный выбор этих условий гарантирует эффективность расчетов.

4.2.2. Метод Ньютона

Этот метод является модификацией метода простой итерации и часто называется *методом касательных*. Если $f(x)$ имеет непрерывную производную, тогда, выбрав в (4.3) $\psi(x)=1/f'(x)$, получаем эквивалентное уравнение $x=x-f(x)/f'(x)=\varphi(x)$, в котором $q=\varphi'(x^*)\equiv 0$. Поэтому скорость сходимости рекуррентной последовательности метода Ньютона

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})} = \varphi(x_{k-1}) \tag{4.6}$$

вблизи корня очень большая, погрешность очередного приближения примерно равна квадрату погрешности предыдущего $\varepsilon_k \cong |\varphi''(x^*)|\,\varepsilon_{k-1}^2$.

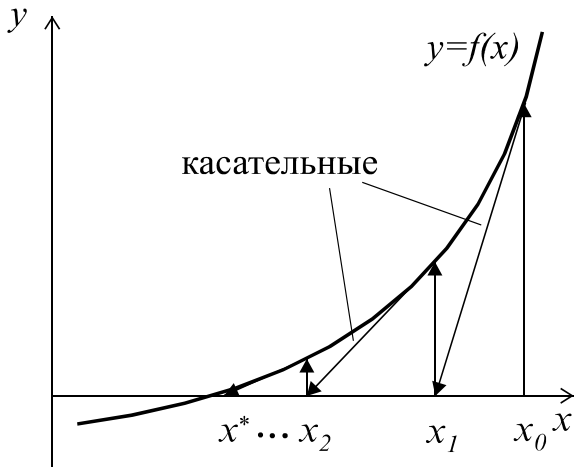

Рис. 4.2. Геометрическая интерпретация метода Ньютона

Из (4.6) видно, что этот метод одношаговый *(m=1)* и для начала вычислений требуется задать одно начальное приближение x_0 из *области сходимости*, определяемой неравенством $|f(x)\cdot f''(x)|<[f'(x)]^2$. Метод Ньютона получил также второе название *метод касательных* благодаря геометрической иллюстрации его сходимости, представленной на рис. 4.2. Этот метод позволяет находить как простые, так и кратные корни. Основной его недостаток – малая область сходимости и необходимость вычисления производной $f'(x)$.

4.2.3. Метод секущих

Данный метод является модификацией метода Ньютона, позволяющей избавиться от явного вычисления производной путем ее замены приближенной формулой (4.2). Это эквивалентно тому, что вместо касательной на рис. 4.2 проводится секущая. Тогда вместо процесса (4.6) получаем

$$x_k = x_{k-1} - \frac{f(x_{k-1})\,h}{f(x_{k-1}) - f(x_{k-1}-h)} = \varphi(x_{k-1}), \tag{4.7}$$

где h – некоторый малый параметр метода, который подбирается из условия наиболее точного вычисления производной.

Метод одношаговый *(m=1)*, и его условие сходимости при правильном выборе h такое же, как у метода Ньютона.

4.2.4. Метод Вегстейна

Этот метод является модификацией предыдущего метода секущих. В нем предлагается при расчете приближенного значения производной по разностной формуле использовать вместо точки $x_{k-1}-h$ в (4.7) точку x_{k-2}, полученную на предыдущей итерации. Расчетная формула метода Вегстейна:

$$x_k = x_{k-1} - \frac{f(x_{k-1})(x_{k-1}-x_{k-2})}{f(x_{k-1})-f(x_{k-2})} = \varphi(x_{k-1}, x_{k-2}). \qquad (4.8)$$

Метод является двухшаговым (m=2), и для начала вычислений требуется задать два начальных приближения x_0, x_1. Лучше всего $x_0=\alpha$, $x_1=\beta$. Метод Вегстейна сходится медленнее метода секущих, однако требует в два раза меньшего числа вычислений *f(x)* и за счет этого оказывается более эффективным.

Этот метод иногда называется улучшенным методом простой итерации и в применении к записи уравнения в форме (4.2) имеет вид

$$x_k = x_{k-1} - \frac{x_{k-1}-\varphi(x_{k-1})}{1-\dfrac{\varphi(x_{k-1})-\varphi(x_{k-2})}{x_{k-1}-x_{k-2}}}. \qquad (4.9)$$

4.2.5. Метод парабол

Предыдущие три метода (Ньютона, секущих, Вегстейна) фактически основаны на том, что исходная функция *f*(*x*) аппроксимируется линейной зависимостью вблизи корня и в качестве следующего приближения выбирается точка пересечения аппроксимирующей прямой с осью абсцисс. Ясно, что аппроксимация будет лучше, если вместо линейной зависимости использовать квадратичную. На этом и основан один из самых эффективных методов – ***метод парабол***. Суть его в следующем: в окрестности корня задаются три начальные точки x_0, x_1, x_2, ($f(x_0)\cdot f(x_2)<0$, $x_1=(x_0+x_2)/2$), в этих точках рассчитываются три значения функции $f(x)$: f_0, f_1, f_2 и строится интерполяционный многочлен второго порядка, который удобно записать в форме

$$P_2(x) = a(x-x_2)^2 + b(x-x_2) + c = az^2 + bz + c. \qquad (4.10)$$

Коэффициенты этого многочлена вычисляются по формулам:

$$z=x-x_2;\ \ z_0=x_0-x_2;\ \ z_1=x_1-x_2;\ \ c=f_2;$$

$$a = \frac{(f_0-f_2)/z_0-(f_1-f_2)/z_1}{z_0-z_1};\quad b=(f_0-f_2)/z_0 - a\,z_0. \qquad (4.11)$$

Полином (4.10) имеет два корня:

$$z_{1,2} = (-b \pm \sqrt{b^2-4ac})/(2a),$$

из которых выбирается наименьший по модулю z_m и рассчитывается следующая точка $x_m = x_2 + z_m$. Если выполняется условие $|x_m - x_1| < \varepsilon$, то за значение

корня принимается x_m, в противном случае одна из крайних точек заменяется на точку x_m: если $x_m < x_1$ *то* $x_0 = x_m$, $f_0 = f(x_m)$, иначе $x_2 = x_m$, $f_2 = f(x_m)$ и процесс повторяется.

4.2.6. Метод деления отрезка пополам

Все вышеописанные методы могут работать, если функция $f(x)$ является непрерывной и дифференцируемой вблизи искомого корня. В противном случае они не гарантируют получение решения. Для разрывных функций, а также если не требуется быстрая сходимость, для нахождения *простого корня* на интервале $[\alpha, \beta]$ применяют надежный метод деления отрезка пополам. Его алгоритм основан на построении рекуррентной последовательности по следующему закону: в качестве начального приближения выбираются границы интервала, на котором имеется один простой корень $x_0 = \alpha$, $x_1 = \beta$, далее находится середина интервала $x = (x_0 + x_1)/2$, после чего отбрасывается половина интервала, не содержащая корня. Очередная точка x выбирается как середина нового, в два раза меньшего интервала. В результате получается следующий алгоритм метода деления отрезка пополам:

1. Вычисляем $f_0 = f(x_0)$, $f_1 = f(x_1)$.
2. Вычисляем $x = x_0 + x_1 / 2$, $f_m = f(x)$.
3. Если $f_0 \cdot f_m > 0$, то $x_0 = x$, $f_0 = f_m$, иначе $x_1 = x$, $f_1 = f_m$.
4. Если $|x_1 - x_0| > \varepsilon$, то повторять с п. 2.
5. Вычисляем корень $x^* = (x_0 + x_1)/2$.

За одно вычисление функции интервал уменьшается вдвое, т. е. скорость сходимости невелика, однако метод устойчив к ошибкам округления и всегда сходится.

4.3. Индивидуальные задания

Разработать программу нахождения всех простых корней функции $f(x)$ в указанном интервале $[a, b]$ соответствующим методом согласно варианту табл. 4.1.

Таблица 4.1

f(x)	*Интервал*		*Метод*
	a	*b*	
1. $4x-7\sin(x)$	–2	2	Деления пополам
2. $x^2-10\sin^2(x)+2$	–1	3	Ньютона
3. $\ln(x)-5\cos(x)$	1	8	Секущих
4. $e^x/x^3-\sin^3(x)-2$	4	7	Вегстейна
5. $\sqrt{x}-\cos^2(x)-2$	4	8	Парабол
6. $4*x-\cos(x)$	–1	4	Простой итерации

4.4. Контрольные вопросы

1. Как решается задача нахождения корней?
2. В чем суть метода простой итерации и его условие сходимости?
3. Дайте геометрическую интерпретацию метода Ньютона.
4. В чем отличие метода Вегстейна от метода секущих?
5. Дайте геометрическую интерпретацию метода парабол.

ТЕМА 5. МЕТОДЫ ОПТИМИЗАЦИИ

Цель работы: изучить основные методы и алгоритмы нахождения минимума функции одной переменной.

5.1. Постановка задач оптимизации, их классификация

Трудно назвать область деятельности, где не возникали бы задачи оптимизационного характера. Это, например, задачи определения наиболее эффективного режима работы различных систем, задачи организации производства, дающего наибольшую возможную прибыль при заданных ограниченных ресурсах или ограничении на количество товара, которое может поглотить рынок, и т. д.

Постановка каждой задачи оптимизации включает в себя моделирование рассматриваемой ситуации с целью получения математической функции, которую необходимо минимизировать, а также определения ограничений, если таковые существуют и выбора подходящей процедуры для осуществления минимизации функции.

Задача оптимизации заключается в выборе среди элементов множества X *(множества допустимых решений*) такого решения, которое было бы с определенной точки зрения наиболее предпочтительным. В дальнейшем понятие решения отождествляется с вектором (точкой) n-мерного евклидова пространства R^n. В соответствии с этим допустимое множество X представляет собой некоторое подмножество пространства R^n, т. е. $X \subset R^n$, а *целевая функция* (а также *критерий качества* или *критерий оптимальности*) $f(\vec{x})$ – это функция n переменных $x_1, x_2, ..., x_n$. Сравнение решений по предпочтительности осуществляется с помощью целевой функции $f(\vec{x})$, которую формулируют таким образом, чтобы наиболее предпочтительному решению $\vec{x}^{(0)}$ соответствовал минимум целевой функции:

$$f(\vec{x}^{(0)}) = \min_{\vec{x} \in X} f(\vec{x}). \qquad (5.1)$$

При этом решение $\vec{x}^{(0)}$ называют *оптимальным* (точнее говоря, *минимальным*) , а значение $f(\vec{x}^{(0)})$ – *оптимумом (минимумом*).

Существует два вида минимумов *: локальный и глобальный*. Говорят, что точка $\vec{x}^{(0)} \in X$ доставляет функции $f(\vec{x})$ на множестве X локальный минимум, если существует такая окрестность $U_\varepsilon(\vec{x}^{(0)})$ $(\varepsilon > 0)$ точки $\vec{x}^{(0)}$, что неравенство $f(\vec{x}^{(0)}) \le f(\vec{x})$ справедливо для всех $\vec{x} \in X \cap U_\varepsilon(\vec{x}^{(0)})$. Глобальный минимум функции $f(\vec{x})$ доставляет точка $\vec{x}^{(0)} \in X$, для которой записанное выше неравенство выполняется при всех $\vec{x} \in X$.

Если множество допустимых значений $X = R^n$, то говорят о *задаче минимизации без ограничений*. В этом случае нужно найти такую точку $\vec{x}^{(0)}$, чтобы неравенство $f(\vec{x}^{(0)}) \le f(\vec{x})$ выполнялось для всех точек пространства R^n без ограничения. Задачу минимизации без ограничений называют также *задачей безусловной минимизации*. При этом для характеристики точки минимума и самого минимума добавляют прилагательное «безусловный».

Если $X \neq R^n$, то имеет место *задача минимизации с ограничениями*. В этом случае также говорят о *задаче условной минимизации*, о *точках условного минимума* и об *условном минимуме.*

Если допустимое множество X задано в виде

$$X = \{\vec{x} \in R^n \mid g_j(\vec{x}) \le 0,\ j = 1,2,...,k;\ g_j(\vec{x}) = 0,\ j = k+1,...,m\}, \qquad (5.2)$$

где все функции $g_j(\vec{x})$ определены на R^n, то говорят о *задаче математического программирования.* Среди задач этого класса различают *задачи с ограничениями типа неравенств*, когда множество X имеет вид (5.2) и $m = k$; *задачи с ограничениями типа равенств,* когда в (5.2) неравенства отсутствуют ($k = 0$), и *задачи со смешанными ограничениями,* когда в задании множества X встречаются как равенства, так и неравенства. Следует отметить, что ограничение типа равенства $g(\vec{x}) = 0$ всегда можно заменить двумя ограничениями типа неравенства $g(\vec{x}) = 0 \rightarrow g(\vec{x}) \le 0,\ g(\vec{x}) \ge 0$. С другой стороны, ограничение-неравенство всегда можно заменить эквивалентным ему ограничением-равенством, введя дополнительную переменную x_{n+1} $g(\vec{x}) \le 0 \rightarrow g(\vec{x}) + x_{n+1}^2 = 0$.

5.2. Методы нахождения минимума функции одной переменной

Задача нахождения минимума функции одной переменной *min f(x)* нередко возникает в практических приложениях. Кроме того, многие методы решения задачи минимизации функции многих переменных сводятся к многократному поиску одномерного минимума. Поэтому разработка новых, более эффективных одномерных методов оптимизации продолжается и сейчас, несмотря на кажущуюся простоту задачи.

Примечание. *В дальнейшем, если не будет особо оговорено, под минимумом функции будет подразумеваться локальный минимум.*

Нахождение минимума функции осуществляется в два этапа:

1. Приближенное определение местоположения минимума.

2. Вычисление точки минимума $x_{\min}$ с заданной точностью ε одним из нижеприведенных методов.

На первом этапе, задав некоторую начальную точку x_0, спускаются с заданным шагом h в направлении уменьшения функции и устанавливают интервал длиной $2h$, на котором находится минимум, из условия

$f(x_m - h) > f(x_m) < f(x_m + h)$. Для функции, изображенной на рис. 5.1, если $A < x_0 < x_g$, будет выделен интервал $[a,b]$ с локальным минимумом x_{min1}, а если

$x_g < x_0 < B$ – с глобальным минимумом x_{min2}, т. е. тот, в области «притяжения» которого оказалась начальная точка x_0.

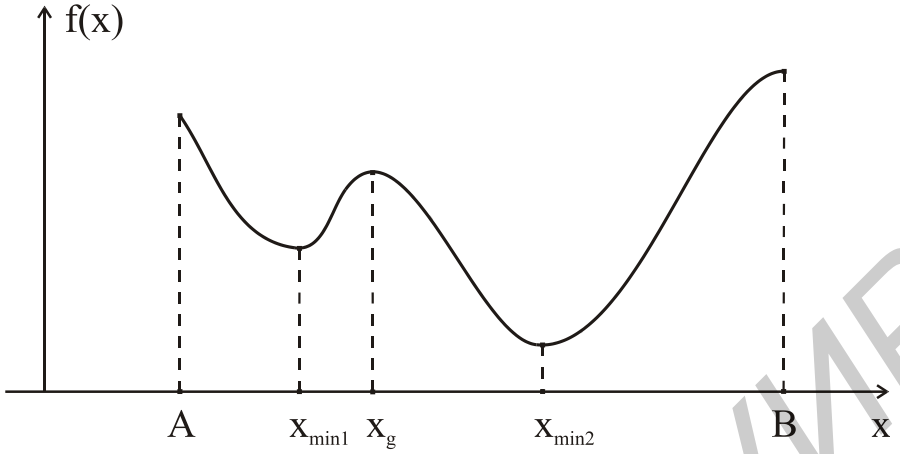

Рис. 5.1. Локальный и глобальный минимумы

Если на отрезке [*a,b*] функция *f(x)* унимодальна, т. е. она имеет на этом отрезке единственную точку минимума x_{min} и слева от этой точки является строго убывающей, а справа – строго возрастающей, то для вычисления точки минимума с заданной точностью могут использоваться нижеприведенные методы.

5.2.1. Метод деления отрезка пополам

Задаются интервал *[a, b]* и погрешность ε.

1. Вычисляется середина интервала [*a, b*]: $x = (a+b)/2$.
2. Отбрасывается половина интервала, не содержащая минимум:
   Если $f(x-\varepsilon) > f(x+\varepsilon)$, то $a = x$, иначе $b = x$.
3. Если $|b - a| > 2\varepsilon$, то повторяем с п. 1.
4. Вычисляем $x_{min} = (a+b)/2$, $f_{min} = f(x_{min})$.

Этот метод прост в реализации, позволяет находить минимум разрывной функции, однако, требует большого числа вычислений функции для обеспечения заданной точности.

5.2.2. Метод золотого сечения

Золотое сечение – это такое деление отрезка [*a, b*] на две неравные части, при котором отношение большего отрезка ко всему интервалу равно отношению меньшего отрезка к большему. При этом имеет место следующее соотношение:

$$(b - x_1)/(b - a) = (x_1 - a)/(b - x_1) = 1 - \xi \cong 0,618, \quad \xi = (3 - \sqrt{5})/2 \cong 0,382.$$

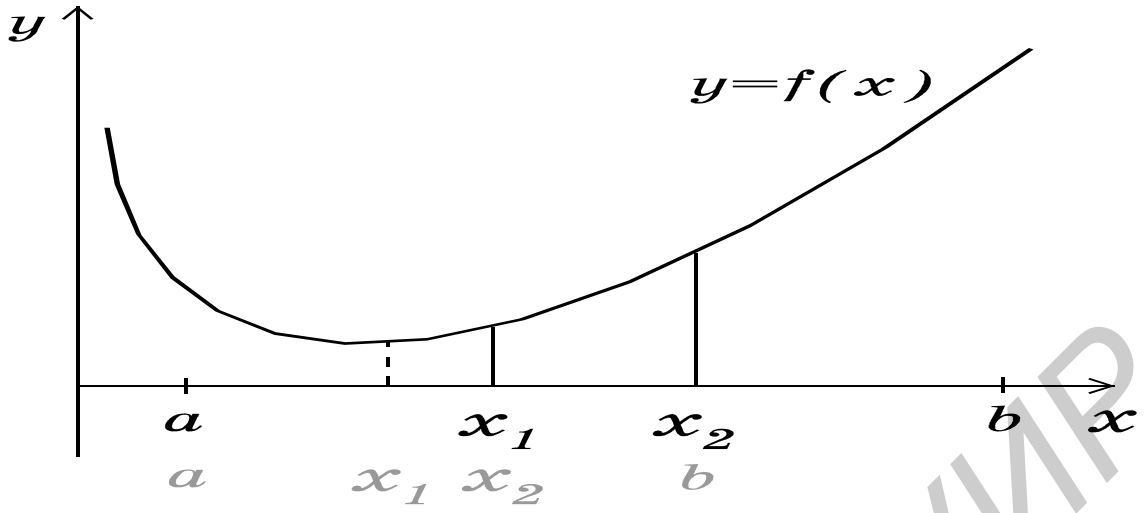

Рис. 5.2. Иллюстрация метода золотого сечения

О точке, которая расположена на расстоянии ξ длины от одного из концов отрезка, говорят, что она осуществляет *золотое сечение* данного отрезка. Каждый отрезок имеет две такие точки, расположенные симметрично относительно середины.

Алгоритм поиска минимума аналогичен вышеописанному методу деления пополам и отличается тем, что вначале точки x_1 и x_2 выбираются так, чтобы они осуществляли золотое сечение отрезка, и вычисляются значения функции в этих точках:

$$x_1 = a + \xi(b-a), \quad x_2 = b - \xi(b-a), \quad f_1 = f(x_1), \quad f_2 = f(x_2).$$

В последующем после сокращения интервала путем отбрасывания неблагоприятной крайней точки на оставшемся *отрезке уже имеется точка, делящая его в золотом отношении* (точка x_1 на рис. 5.2), известно и значение функции в этой точке. Остается лишь выбрать ей симметричную и вычислить значение функции в этой точке для того, чтобы вновь решить, какую из крайних точек отбросить.

Алгоритм метода

Задаются a, b и погрешность ε.

1. Вычисляются две точки золотого сечения:
$x_1 = a + \xi(b-a), \quad x_2 = b - \xi(b-a), \quad f_1 = f(x_1), \quad f_2 = f(x_2)$.
2. Если $f_1 > f_2$, то $a = x_1, \quad x_1 = x_2, \quad f_1 = f_2, \quad x_2 = b - \xi(b-a), \quad f_2 = f(x_2)$,
иначе $b = x_2, \quad x_2 = x_1, \quad f_2 = f_1, \quad x_1 = a + \xi(b-a), \quad f_1 = f(x_1)$.
3. Если $|b-a| > 2\varepsilon$, то повторить с п. 2.
4. Вычисляется $x_{\min} = (a+b)/2, \quad f_{\min} = f(x_{\min})$.

При одинаковом количестве вычислений функции отрезок, на котором находится $x_{\min}$, уменьшается быстрее, чем в методе деления пополам.

5.2.3. Метод Фибоначчи

На практике количество вычислений значений функции часто бывает ограничено некоторым числом n (тем самым ограничено и число шагов вычислений по методу золотого сечения; оно не превышает n–1). Метод Фибоначчи отличается от метода золотого сечения лишь выбором первых двух симметричных точек и формул их пересчета и гарантирует более точное приближение к точке $x_{\min}$ за n –1 шаг, чем метод золотого сечения за то же количество шагов.

Согласно методу Фибоначчи, на нулевом шаге первые две симметричные точки вычисляют по формулам

$$x_1^0 = a_0 + F_n (b_0 - a_0) / F_{n+2} ,$$
$$x_2^0 = b_0 - F_n (b_0 - a_0) / F_{n+2} = a_0 + F_{n+1} (b_0 - a_0) / F_{n+2} ,$$

где F_n, F_{n+1}, F_{n+2} – числа Фибоначчи, определяемые рекуррентной формулой

$$F_k = F_{k-1} + F_{k-2} , \quad k=3, 4, \ldots ; \qquad F_1 = F_2 = 1.$$

Запишем первые десять чисел Фибоначчи: $F_1 = 1$, $F_2 = 1$, $F_3 = 2$, $F_4 = 3$, $F_5 = 5$, $F_6 = 8$, $F_7 = 13$, $F_8 = 21$, $F_9 = 34$, $F_{10} = 55$.

В последующем после сокращения интервала путем отбрасывания неблагоприятной крайней точки одна из точек пересчитывается по одной из соответствующих формул:

$$x_1^k = a_k + F_{n-k} (b_0 - a_0) / F_{n+2} ,$$
$$x_2^k = a_k + F_{n-k+1} (b_0 - a_0) / F_{n+2} ,$$

Выполняется n – 1 шаг, при k = 1, 2, ..., n – 1.

На последнем шаге две симметричные точки сливаются в одну, которая и принимается за точку минимума:

$$x_{\min} = x_1^{n-1}.$$

Погрешность вычисления точки минимума не превышает $(b_0 - a_0)/(2F_{n+2})$, т. е. за три вычисления функции (n=2) получают точку минимума с погрешностью, не превышающей 1/6 первоначального интервала, пять вычислений (n=4) – 1/16, девять (n=8) – 1/110.

Так как $\lim\limits_{n \to \infty} F_n / F_{n+2} = (3-\sqrt{5})/2$, то при достаточно больших n вычисления по методу Фибоначчи и золотого сечения начинаются практически из одной и той же пары симметричных точек.

Алгоритм метода

Задаются a, b и число n.

1. Вычисляются $d = (b–a) / F_{n+2}$ и две симметричные точки:

$$x_1 = a + F_n d, \quad x_2 = b - F_n d, \quad f_1 = f(x_1), \quad f_2 = f(x_2) .$$

2. Если $f_1 > f_2$, то $a = x_1$, $x_1 = x_2$, $f_1 = f_2$, $x_2 = a + F_{n-k+1} d$, $f_2 = f (x_2)$,
иначе $b = x_2$, $x_2 = x_1$, $f_2 = f_1$, $x_1 = a + F_{n-k} d$, $f_1 = f (x_1)$,
п. 2 повторяется n–1 раз, при k = 1, 2, ..., n–1.

3. Вычисляется $x_{\min} = x_1$, $f_{\min} = f(x_{\min})$.

5.2.4. Метод последовательного перебора

Этот метод не требует предварительного определения местоположения точки минимума. Идея метода состоит в том, что, спускаясь из точки x_0 с заданным шагом h в направлении уменьшения функции, устанавливают интервал длиной $2h$, на котором находится минимум, который затем последовательно уточняют, повторяя спуск с последней точки, уменьшив шаг и изменив его знак, пока не будет достигнута заданная точность (некое подобие затухающего маятника).

Алгоритм метода

Задаются x_0, начальный шаг h, *(h>0)* и погрешность ε.

1. Вычисляем $f_0 = f(x_0)$
2. Определяем направление убывания функции.

Если $f(x_0 + \varepsilon) > f_0$, то $h = -h$.

3. Из точки x_0 делается шаг $x_1 = x_0 + h$ и вычисляется $f_1 = f(x_1)$.
4. Если $f_1 < f_0$, то $x_0 = x_1$, $f_0 = f_1$, и повторить с п. 3.

5. В точке x_1 функция оказалась большей, чем в x_0, следовательно, перешагнули точку минимума. Организуем спуск в обратном направлении с меньшим шагом, например $h = -h/4$ или $h = -h/10$.

6. Если $|h| > \varepsilon$, то повторить с п. 3.

7. $x_{\min} = x_0$, $f_{\min} = f_0$.

Скорость сходимости данного метода существенно зависит от удачного выбора начального приближения x_0 и шага h. Шаг h следует выбирать как половину оценки расстояния от x_0 до предполагаемого минимума $x_{\min}$.

5.2.5. Метод квадратичной параболы

Для нахождения точки минимума с заданной точностью задают 3 точки x_1, x_2, x_3 для которых выполняются условия $f(x_2)<f(x_1)$ *и* $f(x_2)<f(x_3)$. На этом интервале функцию аппроксимируют квадратичной параболой, минимум которой известен. Суть метода в следующем.

Для заданных трех точек x_1, x_2, x_3 вычисляются значения функции в них f_1, f_2, f_3. Через эти точки проводится квадратичная парабола:

$$p(x - x_3)^2 + q(x - x_3) + r = pz^2 + qz + r,$$

$$z = x - x_3, \quad z_1 = x_1 - x_3, \quad z_2 = x_2 - x_3, \quad r = f_3, \tag{5.3}$$

$$p = \frac{(f_1 - f_3)/z_1 - (f_2 - f_3)/z_2}{z_1 - z_2}, \qquad q = (f_1 - f_3)/z_1 - pz_1.$$

Парабола имеет минимум в точке $z_m = -q/(2p)$. Следовательно, можно аппроксимировать положение минимума функции значением $x_m = x_3 + z_m$ и, если точность не достигнута, следующий спуск производить, используя эту новую точку и две предыдущие, отбросив одну наихудшую точку. Получается последовательность x_{m1}, x_{m2}, x_{m3}, ... , сходящаяся к точке $x_{\min}$.

Алгоритм метода

1. Задаются точки x_1, x_2, x_3 и точность нахождения минимума ε.
2. Вычисляем $f_1 = f(x_1)$, $f_2 = f(x_2)$, $f_3 = f(x_3)$.
3. Вычисляем z_1, z_2, p, q, z_m по вышеприведенным формулам (5.3).
4. Вычисляем точку минимума параболы $x_m = x_3 + z_m$, $f_m = f(x_m)$.
5. Если $|x_m - x_2| < \varepsilon$, то $x_{\min} = x_m$, иначе переименовываем точки, отбрасывая наихудшую точку: если $x_m < x_2$ то $x_3 = x_2$, $f_3 = f_2$, $x_2 = x_m$, $f_2 = f_m$, иначе $x_1 = x_2$, $f_1 = f_2$, $x_2 = x_m$, $f_2 = f_m$ и повторяем с п. 3.

Данный метод сходится очень быстро и является одним из наилучших методов спуска.

5.2.6. Метод кубической параболы

Данный метод аналогичен предыдущему, но за счет использования аппроксимации кубической параболой имеет более высокую сходимость, *если функция допускает простое вычисление производной*. При его использовании выбираются две точки x_1 и x_2 ($f'(x_1) < 0$, $f'(x_2) > 0$), вычисляются значения функции f_1, f_2 и ее производной $D_1 = f'(x_1)$, $D_2 = f'(x_2)$. Затем через эти точки проводится кубическая парабола, коэффициенты которой определяются таким образом, чтобы совпадали значения функции и производных в точках x_1 и x_2:

$$p(x - x_2)^3 + q(x - x_2)^2 + r(x - x_2) + s = pz^3 + qz^2 + rz + s = P(z),$$
$$z = x - x_2, \quad z_1 = x_1 - x_2,$$
$$P(z_1) = f_1, \quad P'(z_1) = D_1, \quad P(0) = f_2, \quad P'(0) = D_2.$$

Как нетрудно убедиться, коэффициенты параболы вычисляются по следующим формулам:

$$s = f_2, \quad r = D_2,$$
$$p = (D_1 + D_2 - 2(f_1 - f_2)/z_1)/z_1^2,$$
$$q = ((D_1 - D_2)/z_1 - 3pz_1)/2.$$

Известно, что кубическая парабола имеет минимум в точке

$$z_m = (-q + \sqrt{q^2 - 3pr})/(3p).$$

Поэтому приближенное положение минимума можно получить по формуле $x_m = x_2 + z_m$ и, если точность не достигнута, заменить одну из крайних точек точкой x_m и снова повторить процесс.

Алгоритм метода

1. Задаются точки x_1, x_2, ($f'(x_1) < 0, \quad f'(x_2) > 0$) и точность ε.
2. Вычисляем $f_1 = f(x_1),\ f_2 = f(x_2),\ D_1 = f'(x_1), D_2 = f'(x_2),\ x_p = (x_1 + x_2)/2.$
3. Вычисляем z_1, p, q, r, z_m, x_m по вышеприведенным формулам.
4. Если $|x_m - x_p| < \varepsilon$, то $x_{\min} = x_{\text{m}}$.

В противном случае заменяем одну из крайних точек точкой x_{m}:

если $f'(x_m) < 0$, то $x_1 = x_m, \quad f_1 = f(x_m),\ D_1 = f'(x_m),$

иначе $x_2 = x_m, \quad f_2 = f(x_m),\ D_2 = f'(x_m),$ запоминаем полученную точку $x_p = x_m$ и снова повторяем с п. 3.

5.3. Методы нахождения минимума функции нескольких переменных

5.3.1. Классификация методов

Сущность всех методов нахождения безусловного минимума функции n переменных состоит в построении последовательности точек $\vec{x}^0, \vec{x}^1, \vec{x}^2, ..., \vec{x}^k, ...$ монотонно уменьшающих значение целевой функции $f(\vec{x})$:

$$f(\vec{x}^0) \geq f(\vec{x}^1) \geq f(\vec{x}^2) \geq ... \geq f(\vec{x}^k) \geq ...$$

Такие методы называют *методами спуска*. Общая схема этих методов следующая. Пусть на k-й итерации имеется точка $\vec{x}^k$. Выбирается направление спуска $\vec{p}^k \in R^n$, длина шага вдоль этого направления α k > 0 и находится минимум αkmin вдоль этого направления (рис 5.3). После чего вычисляют следующую точку последовательности по формуле $\vec{x}^{k+1} = \vec{x}^k + \alpha^k_{\min} \vec{p}^k$.

Примечание. *Согласно последней формуле величина продвижения из точки* $\vec{x}^k$ *в* $\vec{x}^{k+1}$ *1 зависит как от* α^k, *так и от* $\vec{p}^k$. *Однако* α^k *традиционно называют длиной шага.*

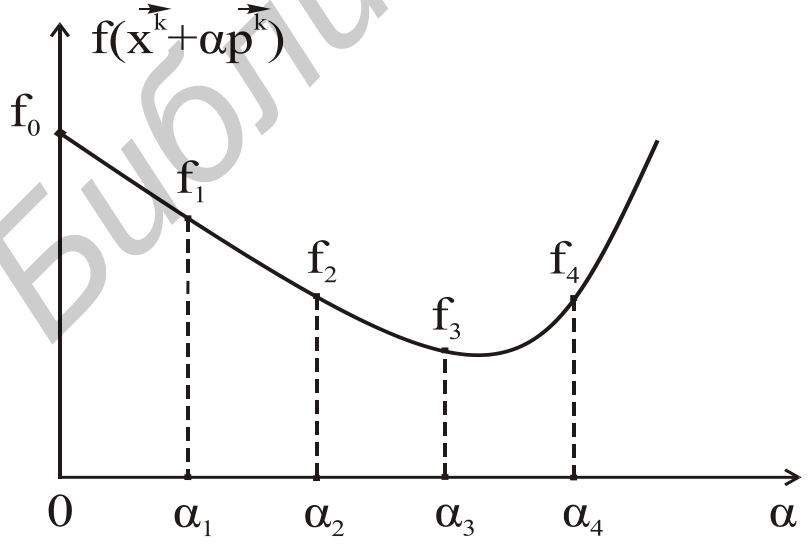

Рис. 5.3. Иллюстрация спуска по направлению

Формально различные методы спуска отличаются друг от друга способом выбора шага α^k и вектора $\vec{p}^k$. Различают *методы с постоянным шагом,* когда начальный шаг на *k*-й итерации α^k постоянен и равен α_0, и *методы с автоматическим выбором шага,* когда величина начального шага на (*k+1)*-й итерации выбирается с учетом величины продвижения на *k*-й итерации, например $\alpha^{k+1} = \alpha^k_{\min}/2$. Если для определения α^k и $\vec{p}^k$ требуется вычислять только значения целевой функции, то соответствующие методы называются *методами нулевого порядка*. *Методы первого порядка* требуют кроме того вычисления первых производных целевой функции. Если же метод предполагает использование и вторых производных, то его называют *методом второго порядка* и т. д. С помощью методов нулевого порядка можно решать задачи более широкого класса, чем с помощью методов первого или второго порядков. Однако методы нулевого порядка, как правило, требуют большего числа вычислений для достижения заданной точности, поскольку использование только значений целевой функции не позволяет достаточно точно определять направление на точку минимума. Методы первого и второго порядков обладают, как правило, более высокой скоростью сходимости, однако необходимость вычисления производных от целевой функции затрудняет решение задачи оптимизации. В ряде случаев они не могут быть получены в виде аналитических выражений, а вычисление производных численными методами осуществляется с ошибками, которые могут ограничить применение таких методов. Кроме того, на практике встречаются задачи, решение которых возможно лишь с помощью методов нулевого порядка, например задачи минимизации функций с разрывными первыми производными.

5.4. Методы нулевого порядка

5.4.1. Метод покоординатного спуска

В этом методе направление спуска выбирают параллельным координатным осям. Задается начальная точка $\boldsymbol{x}_0=\{x_1^0, x_2^0, ..., x_n^0\}$ в *n*-мерном пространстве. Фиксируются все координаты, кроме первой, и решается задача одномерной минимизации для координаты x_1 одним из рассмотренных ранее методов. В результате мы переходим к точке $\boldsymbol{x}_1=\{x_1^1, x_2^0, ..., x_n^0\}$, в которой функция $f(\boldsymbol{x})$ принимает наименьшее значение по координате x_1 при фиксированных остальных координатах. Затем фиксируются все координаты, кроме второй, и решается задача одномерной минимизации для координаты x_2 и т. д. После решения задачи минимизации для всех *n* переменных проверяется условие

$$||\boldsymbol{x}_n - \boldsymbol{x}_0|| < \delta \quad \text{или} \quad |f(\boldsymbol{x}_n) - f(\boldsymbol{x}_0)| < \varepsilon,$$

где δ и ε – некоторые положительные числа, характеризующие точность решения задачи минимизации.

При невыполнении этого условия вновь проводится цикл минимизаций для всех *n* переменных и т. д. Вычисления продолжаются до тех пор, пока не будет выполнено условие

$$||\boldsymbol{x}_n^{k+1} - \boldsymbol{x}_0^k|| < \delta \quad \text{или} \quad |f(\boldsymbol{x}_n^{k+1}) - f(\boldsymbol{x}_0^k)| < \varepsilon.$$

5.4.2. Метод Хука – Дживса

Метод состоит из последовательности итераций, каждая из которых содержит перемещения двух типов*:* *исследующий поиск* вокруг базисной точки, за которым в случае успеха следует *поиск по образцу*.

1. Задается начальная точка $\boldsymbol{x}_0 = (x_1^0, x_2^0, ..., x_n^0)$, вектор приращений для каждой координаты $\boldsymbol{d} = (d_1, d_2, ..., d_n)$ и точность решения задачи минимизации ε.

2. Вычисляется целевая функция $f_0=f(\boldsymbol{x}_0)$.

3. Запоминается базисная точка $\boldsymbol{x}_1=\boldsymbol{x}_0$, $f_1=f_0$.

4. Задаем $m=0$.

5. Для каждой k-й координаты по очереди вычисляются: $x_t=x_1+d_k\varepsilon_k$, $f_t=f(x_t)$, где ε_k – единичный вектор в направлении оси x_k. Если $f_t < f_1$, то полагаем $x_1=x_t$, $f_1=f_t$, и переходим к следующей координате, иначе вычисляем $x_t=x_1-d_k\varepsilon_k$, $f_t=f(x_t)$. Если $f_t < f_1$, то полагаем $x_1=x_t$, $f_1=f_t$, и переходим к следующей координате, иначе значение m увеличиваем на единицу и переходим к следующей координате. После выполнения таких вычислений для всех координат ($k=1,2,...,n$), переходим к следующему пункту.

6. Если $m=n$, т. е. уменьшение функции не было достигнуто ни по одной координате, то вектор приращений $\boldsymbol{d}$ уменьшают (например в два, четыре или десять раз) и повторяют вычисления с 4-го пункта. Вычисления повторяются до тех пор, пока не будет достигнуто уменьшение функции. Если все компоненты вектора приращений $\boldsymbol{d}$ станут меньше ε ($d_k < \varepsilon$, для всех $k=1,2,...,n$), а уменьшение функции не достигнуто, то базисная точка $\boldsymbol{x}_0$ является точкой минимума целевой функции $f(\boldsymbol{x})$ и вычисления прекращаются.

7. При поиске по образцу используется информация, полученная в процессе исследующего поиска. Разумно двигаться из полученной точки $\boldsymbol{x}_1$ в направлении $\boldsymbol{p}=\boldsymbol{x}_1-\boldsymbol{x}_0$, поскольку поиск в этом направлении уже привел к уменьшению значения целевой функции. Поэтому на этапе поиска по образцу решается задача одномерной минимизации целевой функции $f(\boldsymbol{x}_1+\alpha\boldsymbol{p})$ относительно шага α одним из рассмотренных ранее методов. После получения значения α_{min} вычисляется новая базисная точка $\boldsymbol{x}_0=\boldsymbol{x}_1+\alpha_{min}\boldsymbol{p}$ и вычисления повторяются со 2-го пункта.

5.4.3. Метод Нелдера – Мида

Метод Нелдера – Мида является развитием симплексного метода Спендли, Хекста и Химсворда. Геометрическая фигура, порожденная $n+1$ точкой в n-мерном пространстве, называется симплексом, а сами точки называются вершинами симплекса. Следовательно, в двумерном пространстве симплексом является треугольник, в трехмерном пространстве – тетраэдр. Если вершины равноудалены друг от друга, симплекс называется правильным. Идея метода состоит в сравнении значений целевой функции в $n+1$ вершинах симплекса и перемещении симплекса в направлении оптимальной точки с помощью итерационной процедуры. В результате последовательных итераций симплекс модифицируется, двигаясь к точке минимума и сжимаясь вокруг неё. Симплекс преоб-

разуется с помощью операций отражения, растяжения, редукции и сжатия. При рассмотрении этих операций будем использовать следующие обозначения:

f_i – значение целевой функции в *i-й* вершине, т. е. $f_i = f(\boldsymbol{x}^i)$;

m – номер вершины, соответствующей наибольшему значению $f(\boldsymbol{x})$, т. е. $f_m = max\{f_i\}$, $i=1,2,...,n+1$;

s – номер, соответствующий второй по величине вершине после наибольшей, т. е. $f_s = \max\{f_i\}$, $i=1,2,...,n+1$; $i \neq m$;

l – номер вершины с наименьшим значением $f(\boldsymbol{x})$, т. е. $f_l = \min\{f_i\}$, $i=1,2,...,n+1$;

$\boldsymbol{x}^0$ – центр тяжести симплекса, образованного всеми вершинами, кроме $\boldsymbol{x}^m$:

$$\boldsymbol{x}^0 = \frac{1}{n}\sum_{\substack{i=1 \\ i\neq m}}^{n+1} \boldsymbol{x}^i .$$

Преобразование симплекса начинается с операции отражения.

Отражение. Так как $\boldsymbol{x}^m$ – вершина, соответствующая максимальному значению целевой функции, то представляется разумным сравнить значения целевой функции в точках $\boldsymbol{x}^m$ и $\boldsymbol{x}^r$, где $\boldsymbol{x}^r$ получена из $\boldsymbol{x}^m$ отражением относительно противоположной грани гиперплоскости симплекса. Если f_r меньше f_m, то строим новый симплекс, заменяя точку $\boldsymbol{x}^m$ на $\boldsymbol{x}^r$. Процесс отражения проиллюстрирован на рис. 5.4 для двумерного симплекса. Отраженная точка получается как $\boldsymbol{x}^r = \boldsymbol{x}^0 + \alpha(\boldsymbol{x}^0 - \boldsymbol{x}^m)$, где $\alpha > 0$ – коэффициент отражения. Точка $\boldsymbol{x}^r$ лежит на прямой проходящей через точки $\boldsymbol{x}^m$ и $\boldsymbol{x}^0$ с другой стороны от $\boldsymbol{x}^0$. Вычисляется значение целевой функции в отраженной точке $\boldsymbol{x}^r$ и сравнивается со значением целевой функции в точках $\boldsymbol{x}^l$, $\boldsymbol{x}^s$, $\boldsymbol{x}^m$. Возможны следующие случаи, согласно которым далее следуют соответствующие операции:

$f_l < f_r \leq f_s$ – отражение в новом симплексе;

$f_r < f_l$ – растяжение;

$f_s < f_r < f_m$ – редукция;

$f_r > f_m$ – сжатие;

В первом случае точка $\boldsymbol{x}^r$ является лучшей точкой по сравнению с точками $\boldsymbol{x}^m$ и $\boldsymbol{x}^s$, поэтому точка $\boldsymbol{x}^m$ отбрасывается и заменяется на $\boldsymbol{x}^r$, строится новый симплекс и операция отражения повторяется.

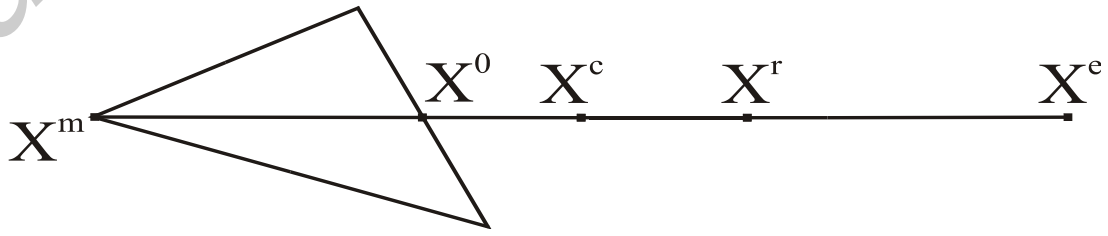

Рис. 5.4. Процесс отражения

Растяжение. Если процедура отражения дает точку $\boldsymbol{x}^r$, для которой $f_r < f_l$, т. е. минимальную точку, то можно ожидать, что значение функции уменьшится еще более при движении по прямой, соединяющей точки $\boldsymbol{x}^0$ и $\boldsymbol{x}^r$. Эта гипотеза проверяется в процедуре растяжения в этом направлении $\boldsymbol{x}^e = \boldsymbol{x}^0 + \gamma(\boldsymbol{x}^r - \boldsymbol{x}^0)$, где $\gamma>1$ – коэффициент растяжения. Вычисляем $f_e = f(\boldsymbol{x}^e)$. Если $f_e < f_r$, то заменяем точку $\boldsymbol{x}^m$ на $\boldsymbol{x}^e$, в противном случае ($f_e > f_r$) растяжение оказалось неудачным и точку $\boldsymbol{x}^m$ заменяем на $\boldsymbol{x}^r$, после чего строится новый симплекс и операция отражения повторяется.

Редукция. Если в процессе отражения получилась точка $\boldsymbol{x}^r$, такая, что $f_s < f_r < f_m$, то отражение дает лишь незначительное улучшение. В этом случае выполняют редукцию в направлении, соединяющем точки $\boldsymbol{x}^0$ и $\boldsymbol{x}^r$, чтобы проверить, не перешагнули ли лучшую точку. Для редукции рассчитываем точку $\boldsymbol{x}^c = \boldsymbol{x}^0 + \beta\,(\,\boldsymbol{x}^r - \boldsymbol{x}^0\,)$, где β – коэффициент редукции $(0<\beta<1)$. Вычисляем $f_c = f(\boldsymbol{x}^c)$. Если $f_c < f_r$, то точку $\boldsymbol{x}^m$ заменяем на $\boldsymbol{x}^c$, в противном случае точка $\boldsymbol{x}^m$ заменяется на $\boldsymbol{x}^r$. Строится новый симплекс и операция отражения повторяется.

Сжатие. К операции сжатия прибегают, когда отражение дает полностью неудовлетворительный результат ($f_r > f_m$). Остается предположить, что минимум, вероятно, лежит внутри симплекса. Поэтому симплекс сжимается в два раза вокруг вершины с минимальным значением f_l путем пересчета всех вершин по формулам $\boldsymbol{x}^i = (\boldsymbol{x}^i + \boldsymbol{x}^l)/2, \quad i=1,2,...,n+1,$ после чего операция отражения повторяется. Вычисления прекращаются, если среднеквадратичное отклонение δ целевой функции в $n+1$ вершинах текущего симплекса меньше заданного малого значения ε, т. е.

$$\delta = \left[\sum_{i=1}^{n+1} f_i - f_{cp}\ ^2 \ / \ n+1 \right]^{1/2} < \varepsilon, \quad \text{где} \quad f_{\text{ср}} = \frac{1}{1+n} \sum_{i=1}^{n+1} f_i$$

Если коэффициенты отражения, редукции и растяжения равны соответственно $\alpha=1$, $\beta=0{,}5$, $\gamma=2$, то симплексный метод носит название метода Нелдера – Мида.

5.5. Методы первого порядка

Как известно, направление градиента $\vec{g} = \nabla f(\vec{x}) = \left(\frac{df}{dx_1}, \frac{df}{dx_2}, ..., \frac{df}{dx_n} \right)$ является направлением наискорейшего возрастания функции. Следовательно, противоположное направление $-\vec{g}$ является направлением наискорейшего убывания функции. Это свойство антиградиента лежит в основе градиентных методов первого порядка. При этом направление наискорейшего убывания в данной точке не всегда оказывается наилучшим для спуска к точке минимума, поэтому для повышения эффективности вводят различные поправки. При выборе очередного направления используют накопленную информацию о функции из

предыдущих спусков. Множество возможностей введения таких поправок определяет многообразие различных методов первого порядка.

5.5.1. Метод наискорейшего спуска

В этом методе на каждой k-й итерации решается задача одномерной минимизации целевой функции $f(\boldsymbol{x}_k+\alpha\boldsymbol{p}_k)$ относительно величины шага α в направлении $\boldsymbol{p}_k=-\nabla f(\boldsymbol{x}_k)$. Новая точка вычисляется по формуле

$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\alpha_{\min}p_k$.

Процесс прекращается, когда выполняется одно из условий:

$||\boldsymbol{x}_{k+1}-\boldsymbol{x}_k||<\varepsilon$ или $||\boldsymbol{p}_{k+1}||<\varepsilon$.

5.5.2. Метод сопряженных градиентов Флетчера – Ривса

Широкое распространение получили градиентные методы, основанные на так называемых сопряженных направлениях. Общая схема рассматриваемого класса методов может быть представлена в следующем виде:

$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\alpha_{k\,min}\boldsymbol{p}_k$, где $\boldsymbol{p}_k=-\nabla f(\boldsymbol{x}_k)+\beta_k\boldsymbol{p}_{k-1}$, $k=1,2,\ldots$; $\boldsymbol{p}_0=-\nabla f(\boldsymbol{x}_0)$

Здесь, как и ранее, на каждой k-й итерации решается задача одномерной минимизации целевой функции относительно величины шага α, однако, направление поиска $\boldsymbol{p}_k$ выбирается с учетом направлений на предыдущих итерациях. Различные способы выбора параметра β_k приводят к различным модификациям градиентных методов. Очевидно, что при $\beta_k=0$ схема вырождается в метод наискорейшего спуска. В частности, чтобы обеспечить хорошую сходимость рассматриваемых методов, β_k необходимо выбирать таким образом, чтобы i направляющих векторов $(i \le n)$ на последующих шагах процесса минимизации были линейно независимыми. Этому условию удовлетворяют сопряженные направления. Два вектора $\boldsymbol{v}$ и $\boldsymbol{u}$ являются сопряженными относительно положительно определенной матрицы A, если имеет место соотношение $\boldsymbol{v}^TA\boldsymbol{u}=0$. Всегда имеется хотя бы одно взаимно независимое множество сопряженных векторов, ибо такое множество образуют собственные векторы матрицы A. Метод сопряженных направлений заключается в выборе соответствующим образом β_k и образовании такой последовательности направляющих векторов $\boldsymbol{p}_0$, $\boldsymbol{p}_1,\ldots,\boldsymbol{p}_{n-1}$, чтобы эти векторы были сопряженными относительно матрицы Гессе. В методе Флетчера – Ривса β_k вычисляется по формуле

$$\beta_k=\frac{g^T(x^k)\,g(x^k)}{g^T(x^{k-1})\,g(x^{k-1})}=\sum_{i=1}^{n}\left(\frac{df}{dx_i^k}\right)^2\Bigg/\sum_{i=1}^{n}\left(\frac{df}{dx_i^{k-1}}\right)^2$$

Алгоритм метода.

1. Задается начальная точка $\boldsymbol{x}_0$, точность решения задачи минимизации ε.
2. Вычисляем градиент целевой функции $\boldsymbol{g}_0=\nabla f(\boldsymbol{x}_0)$.
3. Вычисляем вектор направления поиска $\boldsymbol{p}_0=-\boldsymbol{g}_0$.
4. Решаем задачу одномерной минимизации целевой функции $f(\boldsymbol{x}_0+\alpha\boldsymbol{p}_0)$ относительно шага α одним из ранее рассмотренных методов.

5. Вычисляем новую точку $\boldsymbol{x}_1=\boldsymbol{x}_0+\alpha_{min}\boldsymbol{p}_0$. Для $k=1,2,\ldots, n-1$ последовательно повторяем пп. 6–10.

6. Вычисляем градиент целевой функции $\boldsymbol{g}_k=\nabla f(\boldsymbol{x}_k)$.

7. Вычисляем β_k

$$\beta_k=\frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$$

8. Вычисляем вектор направления поиска:

$\boldsymbol{p}_k=-\boldsymbol{g}_k+\beta_k\boldsymbol{p}_{k-1}$

9. Решаем задачу одномерной минимизации целевой функции $f(\boldsymbol{x}_k+\alpha\boldsymbol{p}_k)$ относительно α.

10. Вычисляем новую точку $\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\alpha_{k\,min}\boldsymbol{p}_k$

11. Проверяем условие сходимости $||\boldsymbol{x}_n-\boldsymbol{x}_0|| < \varepsilon$ или $||\boldsymbol{g}_n|| < \varepsilon$.

12. Если заданная точность решения не достигнута, то переходим к п. 2 с заменой $\boldsymbol{x}_0$ на $\boldsymbol{x}_n$.

5.6. Методы второго порядка

5.6.1. Обобщенный метод Ньютона – Рафсона

Эти методы основаны на квадратичной аппроксимации целевой функции и как следствие для своей реализации требуют вычисления производных первого и второго порядков. Одним из таких методов является обобщенный метод Ньютона – Рафсона, в котором на каждой k-й итерации производится одномерный поиск минимума целевой функции $f(\boldsymbol{x}_k+\alpha\boldsymbol{p}_k)$ относительно шага α в направлении $\boldsymbol{p}_k=-H_{k-1}\boldsymbol{g}_k$, где H_{k-1} – матрица, обратная матрице Гессе (матрицы вторых производных) в точке $\boldsymbol{x}_k$, а $\boldsymbol{g}_k$ – градиент целевой функции в той же точке. Одним из преимуществ этого метода по сравнению с градиентными методами состоит в том, что он не реагирует на «овражный» характер минимизируемой функции. Градиентные методы по существу используют линейную аппроксимацию целевой функции и поэтому менее точно определяют направление на точку минимума. Поэтому метод Ньютона – Рафсона позволяет достичь заданной точности за меньшее число итераций, чем градиентные методы. Однако каждая итерация метода Ньютона – Рафсона связана с вычислением матрицы вторых производных H_k и последующим её обращением, что требует большего объема вычислений по сравнению с одной итерацией градиентного метода. Кроме того, в ряде случаев производные не могут быть получены в виде аналитических выражений, а вычисление их численными методами осуществляется с ошибками, которые зачастую сводят на нет преимущества этого метода. Поэтому широкое распространение получила группа методов, называемая методами переменной метрики или квазиньютоновскими, базирующихся на аппроксимации обратной матрицы Гессе на основе только первых производных. В этих методах вектор направления $\boldsymbol{p}_k$ рассчитывается по формуле $\boldsymbol{p}_k=-A_k\boldsymbol{g}_k$, где матрица A_k, называемая иногда матрицей направлений, представляет собой ап-

проксимацию матрицы H_{k-1}. Один из методов переменной метрики, полученный Дэвидоном и модифицированный Флэтчером и Пауэллом, получил название метода Дэвидона – Флэтчера – Пауэлла (ДФП).

5.7. Методы переменной метрики

5.7.1. Метод Дэвидона – Флэтчера – Пауэлла

Основной трудностью реализации обобщенного метода Ньютона – Рафсона является необходимость вычисления матрицы вторых производных H (матрицы Гессе) и ее обращения на каждой итерации. В методе ДФП эта трудность преодолевается благодаря использованию определенного приближения A_k для матрицы, обратной гессиану H. Матрица A_k – положительно определенная симметричная матрица, которая обновляется на каждой итерации. В пределе матрица A_k становится равной обратному гессиану H_k^{-1}. Начальное значение матрицы A_0 принимается равным единичной матрице E. На каждой итерации текущее значение матрицы направлений пересчитывается в соответствии с рекуррентным соотношением

$$A_{k+1} = A_k + \frac{\alpha^k v_k v_k^T}{v_k^T u_k} - \frac{A_k u_k u_k^T A_k}{u_k^T A_k u_k}$$

где $\boldsymbol{v}_k = \alpha_k \boldsymbol{p}_k$, $\boldsymbol{u}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$.

Алгоритм метода.

1. Задается начальная точка $\boldsymbol{x}_0$ и погрешность ε.
2. Задается начальное значение матрицы $A_0 = E$ (единичная матрица).

На каждой *k-й* итерации ($\kappa = 0, 1, 2, \ldots$) вычисляется:

3. Градиент целевой функции:

$$g_k = \nabla f(x^k) = \left(\frac{df}{dx_1^k}, \frac{df}{dx_2^k}, \ldots, \frac{df}{dx_n^k} \right);$$

4. Вектор направления поиска $\boldsymbol{p}_k = -A_k \boldsymbol{g}_k$.
5. Находится α_k в результате решения задачи одномерной минимизации целевой функции $f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)$ одним из ранее рассмотренных методов.
6. Вычисляется вектор перемещения в пространстве n переменных:

$\boldsymbol{v}_k = \alpha_k \boldsymbol{p}_k$.

7. Вычисляется новая точка $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_k$.
8. Вычисляется целевая функция $f_{k+1} = f(\boldsymbol{x}_{\kappa+1})$ и градиент $\boldsymbol{g}_{k+1} = \nabla f(\boldsymbol{x}_{k+1})$ во вновь полученной точке $\boldsymbol{x}_{k+1}$.
9. Завершить процедуру поиска, если $||\boldsymbol{g}_{k+1}|| < \varepsilon$ или $||\boldsymbol{v}_k|| < \varepsilon$.

В противном случае:

10. Вычислить $\boldsymbol{u}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$.
11. Пересчитать матрицу A_k:

$$A_{k+1} = A_k + \frac{\alpha^k v_k v_k^T}{v_k^T u_k} - \frac{A_k u_k u_k^T A_k}{u_k^T A_k u_k}.$$

12. Увеличить κ на единицу и вернуться к п. 3.

5.8. Индивидуальные задания

Составить программу нахождения минимума функции *n* переменных соответствующим методом согласно номеру варианта табл. 5.1. Для методов, в которых на каждой итерации проводится одномерная минимизация по текущему направлению, в третьей графе таблицы указан метод уточнения минимума. Проверить программу на одной из тестовых функций, указанных ниже:

$f(\vec{x}) = (x_1 - 1)^2 + 5(x_2 + 2)^2 + (x_3 - 2)^2$;

$f(\vec{x}) = 100(x_2 + x_1^2)^2 + (1 - x_1)^2$.

Таблица 5.1

Метод	*Метод одномерной минимизации по направлению*
1. Покоординатного спуска	Фибоначчи
2. Хука – Дживса	Золотого сечения
3. Нелдера – Мида	–
4. Наискорейшего спуска	Деления пополам
5. Флетчера – Ривса	Квадратичной параболы
6. Давидона – Флетчера – Пауэлла	Кубической параболы
7. Покоординатного спуска	Золотого сечения
8. Хука – Дживса	Фибоначчи
9. Нелдера – Мида	–
10. Наискорейшего спуска	Квадратичной параболы
11. Флетчера – Ривса	Деления пополам
12. Давидона – Флетчера – Пауэлла	Фибоначчи
13. Покоординатного спуска	Золотого сечения
14. Хука – Дживса	Деления пополам
15. Нелдера – Мида	–

5.9. Контрольные вопросы

1. Что такое условный и безусловный минимумы, в чем их отличие?
2. Что такое локальный и глобальный минимумы, в чем их отличие?
3. В чем суть метода последовательного перебора?
4. Чем отличается метод золотого сечения от метода Фибоначчи?
5. Дайте геометрическую интерпретацию методов квадратичной и кубической парабол.

ТЕМА 6. РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Цель работы: изучить основные методы и алгоритмы решения задачи Коши и краевой задачи.

6.1. Задачи для обыкновенных дифференциальных уравнений

Обыкновенными дифференциальными уравнениями можно описать поведение системы взаимодействующих частиц во внешних полях, процессы в электрических цепях, закономерности химической кинетики и многие другие явления. Поэтому решение обыкновенных дифференциальных уравнений занимает одно из важнейших мест среди прикладных задач физики, электроники, экономики, химии и техники.

Конкретная прикладная задача может приводить к дифференциальному уравнению любого порядка или к системе таких уравнений. Известно, что произвольную систему дифференциальных уравнений любого порядка можно привести к некоторой эквивалентной системе уравнений первого порядка. Среди таких систем выделим класс систем, разрешенных относительно производной неизвестных функций:

$$\begin{cases} \dfrac{du_1(x)}{dx} = f_1(x,\ u_1,\ ...,\ u_m), \\ \quad . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \\ \dfrac{du_m(x)}{dx} = f_m(x,\ u_1,\ ...,\ u_m). \end{cases} \tag{6.1}$$

Обычно требуется найти решение системы $\vec{u}(x) = u_1(x),\ ...,\ u_m(x)$ для значений x из заданного интервала $a \le x \le b$.

Известно, что система (6.1) имеет бесконечное множество решений, семейство которых в общем случае зависит от m произвольных параметров $\vec{c} = c_1,\ ...,\ c_m$ и может быть записано в виде $\vec{u} = \vec{u}(x,\ \vec{c})$. Для определения значений этих параметров, т. е. для выделения одного нужного решения, надо наложить дополнительно m условий на функции $\vec{u} = u_1,\ ...,\ u_m$. В зависимости от способа постановки дополнительных условий можно выделить два основных типа задач, наиболее часто встречающихся на практике:

краевая (граничная) задача, когда часть условий задается на границе a (при $x=a$), остальные условия – на границе b (при $x=b$). Обычно это значения искомых функций на границах;

задача Коши (задача с начальными условиями), когда все условия заданы в начале отрезка в виде

$$u_1(a) = u_1^0;\ ...;\ u_m(a) = u_m^0. \tag{6.2}$$

При изложении методов решения задачи Коши воспользуемся компактной записью задач (6.1), (6.2) в векторной форме:

$$\frac{d\vec{u}}{dx} = \vec{f}(x,\vec{u}); \quad \vec{u}(a) = \vec{u}^0. \tag{6.3}$$

Требуется найти $\vec{u}(x)$ для $a \le x \le b$.

6.2. Основные положения метода сеток для решения задачи Коши

Чаще всего задача (6.3) решается методом сеток.

Суть метода сеток состоит в следующем.

В области интегрирования выбирается упорядоченная система точек $a = x_0 < x_1 < x_2 < ... < x_n = b,$ называемая *сеткой*. Точки x_i называют *узлами*, а $h_k = x_k - x_{k-1}$ – *шагом сетки*. Если $h_k = h = (b-a)/n$, сетка называется *равномерной*. Для *неравномерной* сетки обозначим $h = \max\limits_k h_k$. Для упрощения в дальнейшем будем считать сетку равномерной. Решение $\vec{u}(x)$ ищется в виде таблицы значений в узлах выбранной сетки $\vec{u}^k = \vec{u}(x_k)$, для чего дифференциальное уравнение заменяется системой алгебраических уравнений, связывающих между собой значения искомой функции в соседних узлах. Такая система называется *конечно-разностной схемой*. Имеется несколько распространенных способов получения конечно-разностных схем. Приведем здесь один из самых универсальных – *интегроинтерполяционный метод*. Согласно этому методу для получения конечно-разностной схемы проинтегрируем уравнение (6.3) на каждом интервале $x_{k-1},\ x_k$ для k=1, 2,...,n:

$$\int\limits_{x_{k-1}}^{x_k} \frac{d\vec{u}}{dx} dx = \vec{u}^k - \vec{u}^{k-1} = \int\limits_{x_{k-1}}^{x_k} \vec{f}(x,\ \vec{u}(x))dx.$$

Переписав последнюю формулу в виде

$$\vec{u}^k = \vec{u}^{k-1} + \int\limits_{x_{k-1}}^{x_k} \vec{f}(x,\ \vec{u}(x))dx \tag{6.4}$$

получаем, что значение искомой функции в k-*м* узле определяется через значение в предшествующем узле с поправкой, выраженной в форме интеграла. Аппроксимируя интеграл одной из квадратурных формул, получаем те или иные формулы относительно *приближенных* значений искомой функции, которые в отличие от точных обозначим $\vec{y}^k \approx \vec{u}^k$. Структура конечно-разностной схемы для задачи Коши такова, что она устанавливает закон рекуррентной последовательности $\vec{y}^k = \varphi\ \vec{y}^{k-1}$ для искомого решения $\vec{y}^0,\ \vec{y}^1,\ \vec{y}^2,\ ...,\ \vec{y}^n$. Поэтому, используя начальное условие задачи (6.2) и задавая $\vec{y}^0 = \vec{u}^0$, затем по рекуррентным формулам последовательно находят все $y^k,\ k = 1,\ ...,\ n.$ При замене интеграла приближенной квадратурной формулой вносится *погрешность аппроксимации* дифференциального уравнения разностным. Говорят, что разностная схема *аппроксимирует* исходную дифференциальную задачу с поряд-

ком p, если при $h \to 0$ погрешность аппроксимации $\psi(h) \le Ch^p$, $C - \text{const}$. Чем больший *порядок аппроксимации* p, тем выше *точность решения*:

$$\varepsilon(h) = \|\vec{y} - \vec{u}\| = \max_k \left|\vec{y}^k - \vec{u}^k\right|. \qquad (6.5)$$

Основная теорема теории метода сеток утверждает, что если схема устойчива, то при $h \to 0$ погрешность решения $\varepsilon(h)$ стремится к нулю с тем же порядком, что и погрешность аппроксимации:

$$\varepsilon\ h\ \le C_0 \psi\ h\ \le C_0 \cdot C \cdot h^p,$$

где C_0 – константа устойчивости.

Неустойчивость обычно проявляется в том, что с уменьшением h решение $\vec{y}^k \to \infty$ при возрастании k, что легко устанавливается экспериментально с помощью просчета на последовательности сеток с уменьшающимся шагом h, $h/2$, $h/4$… Если при этом $\vec{y}^k \to \infty$, то метод неустойчив. Таким образом, если имеется аппроксимация и схема устойчива, то, выбрав достаточно малый шаг h, можно получить решение с заданной точностью, при этом затраты на вычисления резко уменьшаются с увеличением порядка аппроксимации p, т. е. при большем p можно достичь той же точности, используя более крупный шаг h. Большое разнообразие методов обусловлено возможностью по-разному выбирать узлы и квадратурные формулы для аппроксимации интеграла в (6.4).

6.2.1. Явная схема 1-го порядка (метод Эйлера)

Вычисляя интеграл в (6.4) по формуле левых прямоугольников, получим

$$\vec{y}^k = \vec{y}^{k-1} + h \cdot f(x_{k-1}, \vec{y}^{k-1}), \quad k = 1,\ 2,\ ...,\ n. \qquad (6.6)$$

Задавая $\vec{y}^0 = \vec{u}^0$, с помощью (6.6) легко получить все последующие значения $\vec{y}^k$, $k = 1,\ 2,\ ...,\ n$, т. к. формула явно разрешается относительно $\vec{y}^k$. Погрешность аппроксимации $\psi(h)$ и соответственно точность $\varepsilon(h)$ имеют первый порядок в силу того, что формула левых прямоугольников на интервале $x_{k-1},\ x_k$ имеет погрешность первого порядка, а схема устойчива.

6.2.2. Неявная схема 1-го порядка

Вычисляя интеграл в (6.4) по формуле правых прямоугольников, получим

$$\vec{y}^k = \vec{y}^{k-1} + h \cdot \vec{f}(x_k, \vec{y}^k), \quad k = 1,\ 2,\ ...,\ n. \qquad (6.7)$$

Эта схема явно не разрешена относительно $\vec{y}^k$, поэтому для получения $\vec{y}^k$ требуется использовать итерационную процедуру решения уравнения (6.7):

$$\vec{y}^{k,s} = \vec{y}^{k-1} + h \cdot \vec{f}(x_k,\ \vec{y}^{k,s-1}); \quad s = 1,\ 2,\ ... - \text{ номер итерации.}$$

За начальное приближение можно взять значение $\vec{y}^{k,0} = \vec{y}^{k-1}$ из предыдущего узла. Обычно, если h выбрано удачно, достаточно сделать 2–3 итерации для до-

стижения заданной погрешности $\left\| y^{k,s} - y^{k,s-1} \right\| < \varepsilon$. Эффективность неявной схемы заключается в том, что у нее константа устойчивости C_0 значительно меньше, чем у явной схемы.

6.2.3. Неявная схема 2-го порядка

Вычисляя интеграл в (6.4) по формуле трапеций, получим

$$\vec{y}^{k} = \vec{y}^{k-1} + \frac{h}{2} \cdot \left[\vec{f}(x_{k-1},\ \vec{y}^{k-1}) + \vec{f}(x_k,\ \vec{y}^{k}) \right]. \tag{6.8}$$

Так как формула трапеций имеет второй порядок точности, то и погрешность метода имеет второй порядок. Схема (6.8) явно не разрешена относительно $\vec{y}^{k}$, поэтому требуется итерационная процедура:

$$\vec{y}^{k,s} = \vec{y}^{k-1} + \frac{h}{2} \cdot \left[\vec{f}(x_{k-1},\ \vec{y}^{k-1}) + \vec{f}(x_k, \vec{y}^{k,s-1}) \right], \quad s = 1,\ 2,\ ...,\ \vec{y}^{k,0} = \vec{y}^{k-1}.$$

6.2.4. Схема предиктор-корректор (Рунге – Кутта) 2-го порядка

Вычисляя интеграл в (6.4) по формуле средних прямоугольников, получим

$$\vec{y}^{k} = \vec{y}^{k-1} + h \cdot \vec{f}(x_{k-1/2},\ \vec{y}^{k-1/2}). \tag{6.9}$$

Уравнение разрешено явно относительно $\vec{y}^{k}$, однако в правой части присутствует неизвестное значение $\vec{y}^{k-1/2}$ в середине отрезка $[x_{k-1},\ x_k]$. Для решения этого уравнения используют следующий способ. Вначале по явной схеме (6.6) рассчитывают $\vec{y}^{k-1/2}$ (предиктор):

$$\vec{y}^{k-1/2} = \vec{y}^{k-1} + \frac{h}{2} \cdot \vec{f}(x_{k-1},\ \vec{y}^{k-1}).$$

После этого рассчитывают $\vec{y}^{k}$ по (6.9) (корректор). В результате схема оказывается явной и имеет второй порядок.

6.2.5. Схема Рунге – Кутта 4-го порядка

Вычисляя интеграл в (6.4) по формуле Симпсона, получим

$$\vec{y}^{k} = \vec{y}^{k-1} + \frac{h}{6} \cdot [\vec{f}(x_{k-1},\ \vec{y}^{k-1}) + 4\vec{f}(x_{k-1/2},\ \vec{y}^{k-1/2}) + \vec{f}(x_k,\ \vec{y}^{k})]. \tag{6.10}$$

Ввиду того, что формула Симпсона имеет четвертый порядок, погрешность метода тоже имеет четвертый порядок. Можно по-разному реализовать расчет неявного по $\vec{y}^{k}$ уравнения (6.10), однако наибольшее распространение получил следующий способ. Вычисляют предиктор:

$$\vec{y}^{k-1/2,1} = \vec{y}^{k-1} + \frac{h}{2}(\vec{f}(x_{k-1},\ \vec{y}^{k-1});$$

$$\vec{y}^{k-1/2,2} = \vec{y}^{k-1} + \frac{h}{2}\vec{f}(x_{k-1/2},\ \vec{y}^{k-1/2,1});$$

$$\vec{y}^{k,1} = \vec{y}^{k-1} + h\vec{f}(x_{k-1/2},\ \vec{y}^{k-1/2,2}),$$

затем корректор по формуле

$$\vec{y}^{k} = \vec{y}^{k-1} + \frac{h}{6}[f(x_{k-1},\ \vec{y}^{k-1}) + 2\vec{f}(x_{k-1/2,}\ \vec{y}^{k-1/2,1}) +$$

$$+2\vec{f}(x_{k-1/2,}\ \vec{y}^{k-1/2,2}) + \vec{f}(x_k, \vec{y}^{k,1})].$$

6.3. Многошаговые схемы Адамса

При построении всех предыдущих схем для вычисления интеграла в правой части (6.4) использовались лишь точки в диапазоне одного шага $[x_{k-1},\ x_k]$. Поэтому при реализации таких схем для вычисления следующего значения $\vec{y}^{k}$ требуется знать только одно предыдущее значение $\vec{y}^{k-1}$. Такие схемы называют *одношаговыми*. Мы, однако, видели, что для повышения точности при переходе от узла x_{k-1} к узлу x_k приходилось использовать и значения функции $\vec{f}(x_{k-1/2,}\ \vec{y}^{k-1/2})$ внутри интервала $[x_{k-1},\ x_k]$. Схемы, в которых применяют эту методику (пп. 6.2.4, 6.2.5), называют *схемами с дробными шагами*. В этих схемах повышение точности достигается за счет дополнительных затрат на вычисление функции $\vec{f}(x, \vec{y}\ )$ в промежуточных точках интервала $[x_{k-1},\ x_k]$.

Идея методов Адамса заключается в том, чтобы для повышения точности использовать уже вычисленные на предыдущих шагах значения $\vec{y}^{k-1}$, $\vec{y}^{k-2}$, $\vec{y}^{k-3}$, ... в ***нескольких*** предыдущих узлах. Заменим в (6.4) подынтегральную функцию интерполяционным многочленом Ньютона вида

$$f(x) \approx f(x_{k-1}) + (x - x_{k-1})\frac{f(x_{k-1}) - f(x_{k-2})}{h} +$$

$$+(x - x_{k-1})(x - x_{k-2})\frac{f(x_{k-1}) - 2f(x_{k-2}) + f(x_{k-3})}{2h^2} + \ \ldots\ .$$

После интегрирования на интервале $[x_{k-1},\ x_k]$ получим *явную экстраполяционную схему* Адамса. *Экстраполяцией* называется получение значений интерполяционного многочлена в точках, выходящих за крайние узлы сетки. В нашем случае интегрирование производится на интервале $[x_{k-1},\ x_k]$, а полином строится по узлам x_{k-1}, x_{k-2}, x_{k-3}. Порядок аппроксимации схемы в этом случае определяется количеством использованных при построении полинома узлов

(например, если используются x_{k-1}, x_k, то схема второго порядка). Если в (6.4) подынтегральную функцию заменим многочленом Ньютона вида

$$f(x) \approx f(x_k) + (x - x_k)\frac{f(x_k) - f(x_{k-1})}{h} +$$

$$+(x - x_k)(x - x_{k-1})\frac{f(x_k) - 2f(x_{k-1}) + f(x_{k-2})}{2h^2} + \dots,$$

то после интегрирования получим *неявную интерполяционную схему Адамса*.

6.3.1. Явная экстраполяционная схема Адамса 2-го порядка

Заменив в (6.4) подынтегральную функцию интерполяционным многочленом Ньютона вида

$$f(x) \approx f(x_{k-1}) + (x - x_{k-1})\frac{f(x_{k-1}) - f(x_{k-2})}{h},$$

получим формулу

$$\vec{y}^k = \vec{y}^{k-1} + \frac{h}{2}\cdot[3\vec{f}(x_{k-1},\ \vec{y}^{k-1}) - \vec{f}(x_{k-2},\ \vec{y}^{k-2})]. \qquad (6.11)$$

Схема двухшаговая, поэтому для начала расчетов необходимо, сделав один шаг, найти $\vec{y}^1$ по методу Рунге – Кутта 2-го порядка, после чего $\vec{y}^2$, $\vec{y}^3$, ... вычислять по (6.11).

6.3.2. Явная экстраполяционная схема Адамса 3-го порядка

Заменив в (6.4) подынтегральную функцию интерполяционным многочленом Ньютона вида

$$f(x) \approx f(x_{k-1}) + (x - x_{k-1})\frac{f(x_{k-1}) - f(x_{k-2})}{h} +$$

$$+(x - x_{k-1})(x - x_{k-2})\frac{f(x_{k-1}) - 2f(x_{k-2}) + f(x_{k-3})}{2h^2},$$

получим формулу

$$\vec{y}^k = \vec{y}^{k-1} + \frac{h}{12}\cdot[23\vec{f}(x_{k-1},\ \vec{y}^{k-1}) - 16\vec{f}(x_{k-2},\ \vec{y}^{k-2}) + 5\vec{f}(x_{k-3}, \vec{y}^{k-3}). \qquad (6.12)$$

Схема трехшаговая, поэтому для начала расчетов необходимо, сделав два шага, найти $\vec{y}^1$, $\vec{y}^2$ по методу Рунге – Кутта 4-го порядка, после чего $\vec{y}^3$, $\vec{y}^4$, ... вычислить по (6.12).

6.3.3. Неявная схема Адамса 3-го порядка

Заменив в (6.4) подынтегральную функцию интерполяционным многочленом Ньютона вида

$$f(x) \approx f(x_k) + (x - x_k)\frac{f(x_k) - f(x_{k-1})}{h} +$$
$$+(x - x_k)(x - x_{k-1})\frac{f(x_k) - 2f(x_{k-1}) + f(x_{k-2})}{2h^2},$$

получим формулу

$$\vec{y}^k = \vec{y}^{k-1} + \frac{h}{12}\cdot[5\vec{f}(x_k,\ \vec{y}^k) + 8\vec{f}(x_{k-1},\ \vec{y}^{k-1}) - \vec{f}(x_{k-2},\ \vec{y}^{k-2})]. \qquad (6.13)$$

Так как схема двухшаговая, то для начала расчетов необходимо, сделав один шаг, найти $\vec{y}^1$ по методу Рунге – Кутта 4-го порядка, после чего $\vec{y}^2$, $\vec{y}^3$, ... вычисляются по (6.13). Эта формула явно не разрешена относительно $\vec{y}^k$, поэтому для получения $\vec{y}^k$ требуется использовать итерационную процедуру решения уравнения (6.13):

$$\vec{y}^{k,s} = \vec{y}^{k-1} + \frac{h}{12}\cdot\left[5\vec{f}(x_k,\ \vec{y}^{k,s-1}) + 8\vec{f}(x_{k-1},\ \vec{y}^{k-1}) - \vec{f}(x_{k-2},\ \vec{y}^{k-2})\right].$$

Значение $\vec{y}^{k,0}$ следует расчитать по формуле (6.11):

$$\vec{y}^{k,0} = \vec{y}^{k-1} + \frac{h}{2}\cdot\left[3\vec{f}(x_{k-1},\ \vec{y}^{k-1}) - \vec{f}(x_{k-2},\ \vec{y}^{k-2})\right].$$

6.4. Краевая (граничная) задача

Рассмотрим граничную задачу для линейного дифференциального уравнения второго порядка с переменными коэффициентами

$$y'' + p(x)\,y' + q(x)\,y = f(x). \qquad (6.14)$$

на отрезке $[a, b]$ с граничными условиями общего вида:

$$\alpha_1 y(a) + \beta_1 y'(a) = A,$$
$$\alpha_2 y(b) + \beta_2 y'(b) = B. \qquad (6.15)$$

В тех случаях, когда невозможно получить решение этой задачи аналитическим методом, используются приближенные или численные методы.

6.4.1. Суть приближенных методов

Выбирается система линейно-независимых дважды дифференцируемых функций $\{\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)\}$, при этом функция $\varphi_0(x)$ должна удовлетворять граничным условиям (6.15), а функции $\varphi_1(x)$, $\varphi_2(x)$, …, $\varphi_n(x)$ – соответствующим однородным граничным условиям. Искомое решение представляется в виде линейной комбинации базисных функций:

$$y(x) = \varphi_0(x) + \sum_{k=1}^{n} c_k \varphi_k(x). \qquad (6.16)$$

Подставим выражение (6.16) в (6.14) и найдем невязку левой и правой частей уравнения (6.14):

$$\delta(x, c_1, c_2, \ldots, c_n) = \varphi_0''(x) + p(x)\,\varphi_0'(x) + q(x)\,\varphi_0(x) - f(x) +$$
$$+ \sum_{k=1}^{n} c_k\,[\,\varphi_k''(x) + p(x)\,\varphi_k'(x) + q(x)\,\varphi_k(x)\,]. \qquad (6.17)$$

Коэффициенты $c_1, c_2, \ldots, c_n$ подбирают так, чтобы невязка (6.17) была минимальна. Способ определения этих коэффициентов и характеризует тот или иной метод. В ***методе коллокаций*** выбирают n точек $x_k \in [a,b]$, $k=1, 2,\ldots,n$, называемых точками коллокации, невязки в которых приравниваются нулю. Решив полученную систему линейных алгебраических уравнений, получают значения искомых коэффициентов $c_1, c_2, \ldots, c_n$. ***Метод наименьших квадратов*** основан на минимизации суммы квадратов невязок в заданной системе точек $x_k \in [a,b]$, $k=1, 2,\ldots,m;\ m>n$. Из условия равенства нулю частных производных от суммы квадратов невязок по искомым коэффициентам c_k, $k=1, 2, \ldots, n$ также получают систему линейных алгебраических уравнений. В основе ***метода Галеркина*** лежит требование ортогональности базисных функций $\varphi_1(x), \varphi_2(x), \ldots, \varphi_n(x)$ к невязке (6.17), которое выражается в виде:

$$\int_a^b \delta\,(x, c_1, c_2, \ldots, c_n)\,\varphi_k(x)dx = 0,\ k=1, 2,\ldots,n.$$

Из этого условия также получается система алгебраических уравнений. Решив полученную систему линейных алгебраических уравнений, получают значения искомых коэффициентов $c_1, c_2, \ldots, c_n$. В этом методе нет необходимости выбора семейства узловых точек.

6.5. Численные методы решения краевых задач

6.5.1. Метод стрельбы

Сущность метода стрельбы заключается в сведении решения граничной задачи (6.14) – (6.15) к многократному решению задачи Коши. Введя замену переменных $y_1(x)=dy/dx$, заменим дифференциальное уравнение второго порядка системой двух дифференциальных уравнений первого порядка

$$dy/dx = y_1, \qquad (6.18)$$
$$dy_1/dx = f(x) - p(x)y_1 - q(x)y$$

с граничными условиями общего вида:

$$\alpha_1 y(a) + \beta_1 y_1(a) = A, \qquad (6.19)$$
$$\alpha_2 y(b) + \beta_2 y_1(b) = B.$$

Задавшись произвольным начальным условием для $y(x)$:

$$y(a) = y^0, \qquad (6.20)$$

из первого уравнения (6.19) получаем начальное условие для $y_1(x)$:

$$y_1(a) = (A - \alpha_1 y^0)/\beta_1. \qquad (6.21)$$

Система уравнений (6.18) с начальными условиями (6.20), (6.21) представляет собой задачу Коши, которая решается одним из ранее рассмотренных методов. Получив в результате решения задачи Коши значения $y(b)$, $y_1(b)$ на правом конце отрезка $[a, b]$, проверяют, выполнилось ли второе условие (6.19), которое может быть представлено в виде $F(y^0) = \alpha_2\, y(b) + \beta_2\, y_1(b) - B = 0$. Таким образом,

граничная задача (6.14) – (6.15) в итоге сводится к нахождению корня уравнения $F(y^0)=0$, для вычисления правой части которого необходимо решить задачу Коши (6.18), (6.20), (6.21). Описанный алгоритм называется методом стрельбы, поскольку в нем как бы проводится «пристрелка» по углу наклона интегральной кривой в начальной точке.

6.5.2. Метод конечных разностей

Сущность метода в том, что он сводит решение граничной задачи для дифференциального уравнения к решению системы линейных алгебраических уравнений относительно значений искомой функции на заданном множестве точек. Это достигается путем замены производных, входящих в дифференциальное уравнение, их конечно-разностными аппроксимациями. Рассмотрим применение этого метода для решения дифференциального уравнения (6.14) с граничными условиями (6.15). Выберем на отрезке $[a,b]$ систему равноотстоящих точек $x_k=a+kh; \quad k=0, 1, 2, \ldots, n; \quad h=(b-a)/n$. Решение граничной задачи (6.14) – (6.15) сведем к вычислению значений искомой функции $y(x)$ в узловых точках x_k. Обозначим $y_k = y(x_k)$. Заменим производные, входящие в (6.14), их конечно-разностными аппроксимациями

$$y'(x_k)=(y_{k+1}-y_{k-1})/(2h),$$
$$y''(x_k)=(y_{k+1}-2y_k+y_{k-1})/h^2.$$

и запишем уравнение (6.14) во внутренних узловых точках x_k, $k=1, 2, \ldots, n-1$:

$$(y_{k+1}-2y_k+y_{k-1})/h^2+p_k(y_{k+1}-y_{k-1})/(2h)+q_k y_k=f_k, \qquad (6.22)$$
$$k=1, 2, \ldots, n-1.$$

где введены обозначения $p_k=p(x_k), \quad q_k=q(x_k), \quad f_k=f(x_k)$.

Граничные условия также должны представляться в разностном виде путем аппроксимации производных $y'(a)$, $y'(b)$ помощью конечно-разностных соотношений. Если использовать односторонние разности, при которых производные аппроксимируются с первым порядком точности, то разностные граничные условия в точках x_0 и x_n принимают вид

$$\alpha_1 y_0+\beta_1(y_1-y_0)/h=A,$$
$$\alpha_2 y_n+\beta_2(y_n-y_{n-1})/h=B.$$

Однако предпочтительнее аппроксимировать первые производные со вторым порядком точности с помощью следующих соотношений:

$$y'(x_0)=(-3y_0+4y_1-y_2)/(2h),$$
$$y'(x_n)=(3y_n-4y_{n-1}+y_{n-2})/(2h).$$

В этом случае граничные условия примут вид

$$(\alpha_1-3\beta_1/2h)\, y_0+\beta_1(4y_1-y_2)/(2h)=A, \qquad (6.23)$$
$$(\alpha_2+3\beta_2/2h)y_n+\beta_2(-4y_{n-1}+y_{n-2})/(2h)=B.$$

Выражения (6.22), (6.23) образуют систему линейных алгебраических уравнений $(n+1)$-*го* порядка, решив которую, получают решение граничной задачи (6.14), (6.15) в виде значений искомой функции $y(x)$ в узловых точках $x_0, x_1, , x_n$.

6.6. Индивидуальные задания

Составить программу для решения задачи Коши соответствующим методом согласно варианту табл. 6.1. С помощью этой программы решить задачу для системы двух уравнений в соответствии с вариантом.

$$\frac{du_1}{dx} = f_1(x, u_1, u_2), \quad u_1(a) = u_1^0,$$

$$\frac{du_2}{dx} = f_2(x, u_1, u_2), \quad u_2(a) = u_2^0.$$

$$a \le x \le b.$$

Точное решение для всех вариантов: $u_1 = 2x$, $u_2 = e^x$.

Таблица 6.1

$f_1(x, u_1, u_2)$	$f_2(x, u_1, u_2)$	$[a,b]$	$u_1(a)$	$u_2(a)$	*Метод*
1. $u_1 / x - u_2 / e^x + 1$	$u_1/(2x) + u_2 - 1$	[1,2]	2	$e^{(1)}$	Явная схема 1-го порядка
2. $u_1 / 2x + u_2 / e^x$	$u_1 + u_2 - 2x$	[1, 2]	2	$e^{(1)}$	Неявная схема 1-го порядка
3. $u_1 + 2u_2 / e^x - 2x$	$u_1/(2x) - 1 + u_2$	[2, 3]	4	$e^{(2)}$	Неявная схема 2-го порядка
4. $(u_1 \cdot e^x)/(x \cdot u_2)$	$2u_1 + u_2 - 4x$	[1, 4]	2	$e^{(1)}$	Рунге – Кутта 2-го порядка
5. $u_1 / x + u_2 - e^x$	$2x \cdot u_2 / u_1$	[2, 4]	4	$e^{(2)}$	Рунге – Кутта 4-го порядка
6. $u_1 \cdot u_2 /(e^x \cdot x)$	$2x/u_1 + u_2 - 1$	[1, 3]	2	$e^{(1)}$	Явная схема Адамса 2-го порядка
7. $u_1 / 2x + u_2 / e^x$	$u_1 \cdot u_2 / 2x$	[2, 3]	4	$e^{(2)}$	Явная схема Адамса 3-го порядка
8. $u_1 / x + u_2 - e^x$	$2x/u_1 + u_2^2 / e^x - 1$	[1, 4]	2	$e^{(1)}$	Неявная схема Адамса 3-го порядка

Схемы алгоритмов некоторых методов решения задачи Коши приведены ниже (рис. 6.1 – 6.4).

6.7. Контрольные вопросы

1. Как формулируется задача Коши?
2. В чем суть метода сеток?
3. В чем отличие явных схем от неявных?
4. В чем отличие методов Адамса от методов Рунге – Кутта?

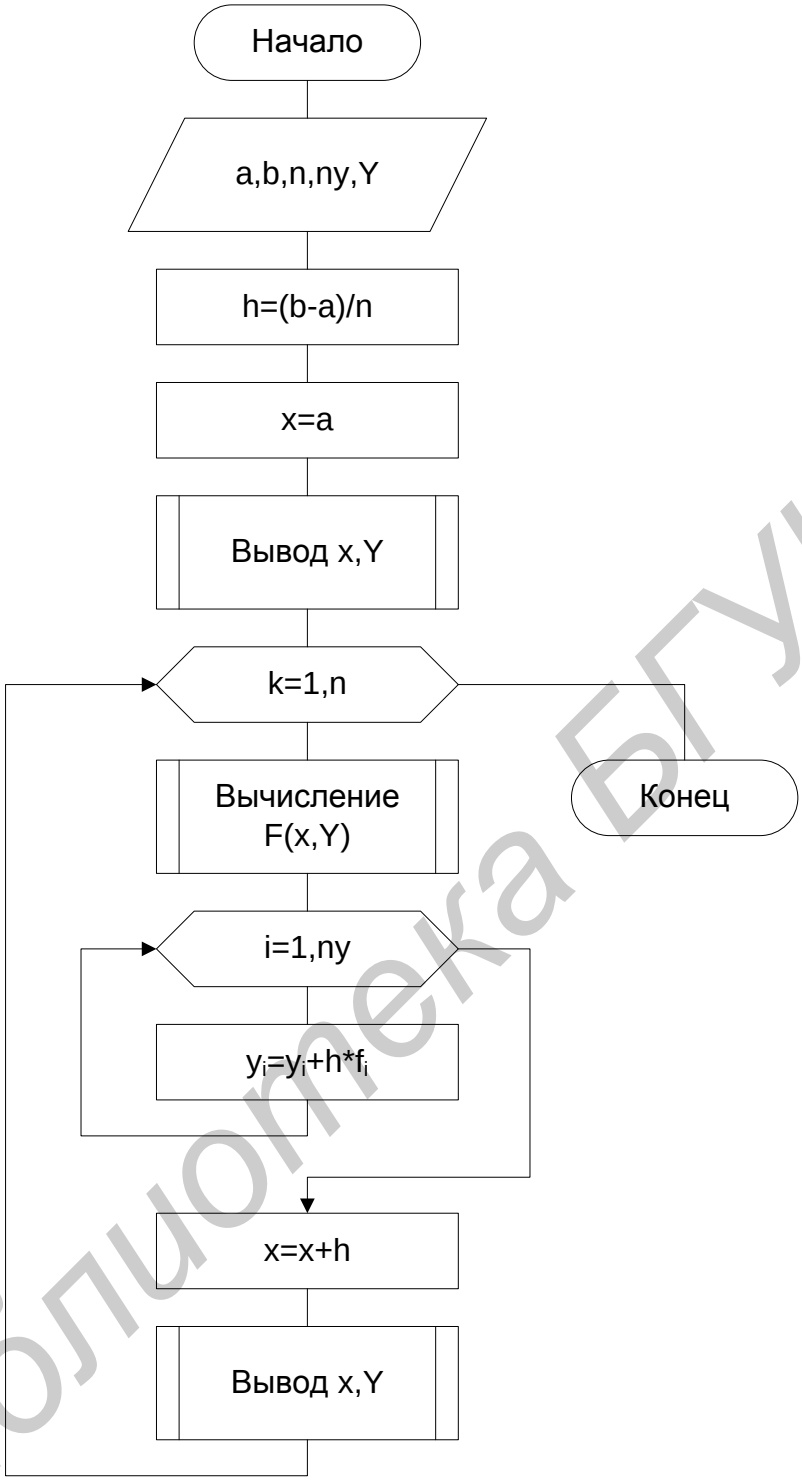

Рис. 6.1. Схема алгоритма метода Эйлера

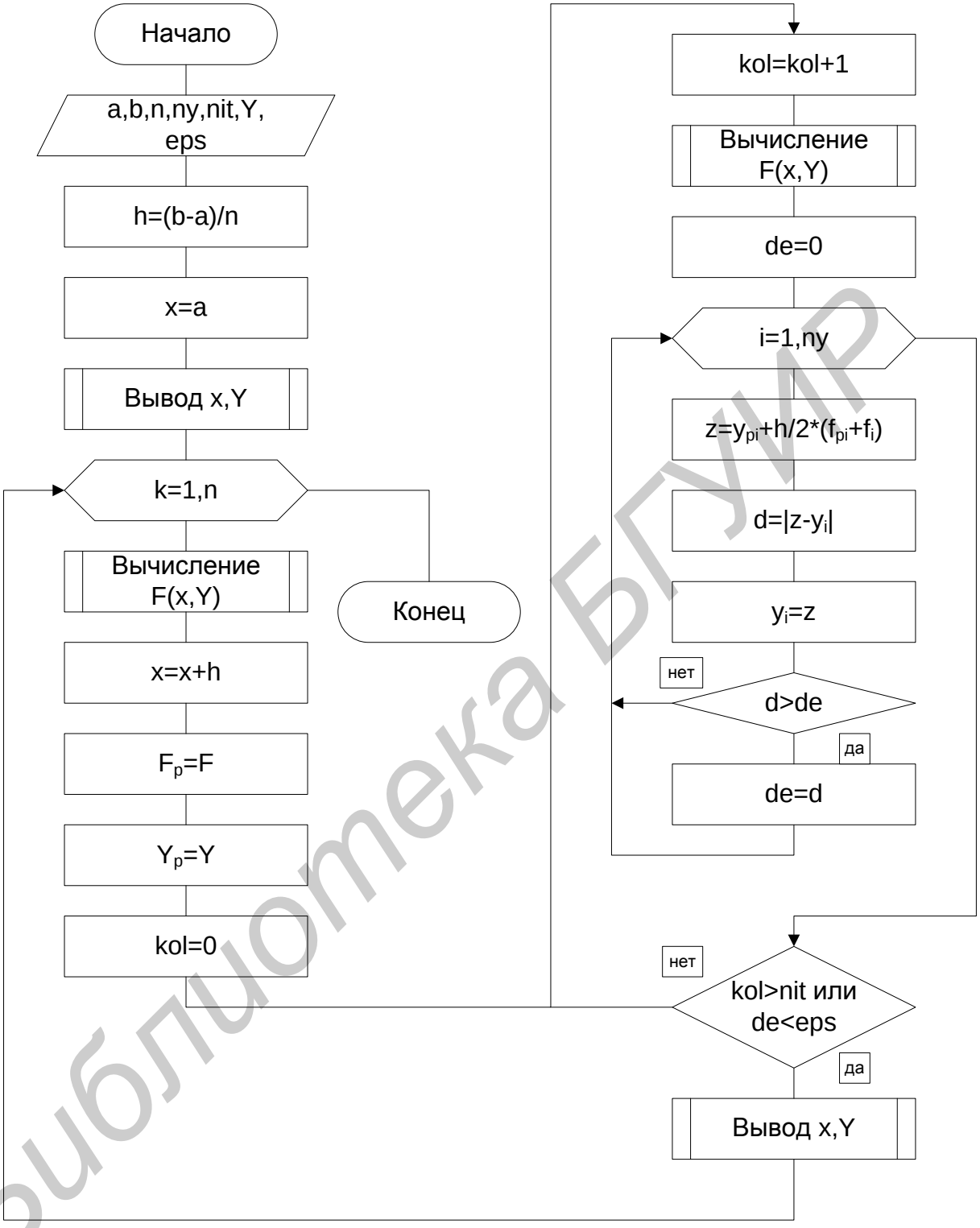

Рис. 6.2. Схема алгоритма неявного метода 2-го порядка

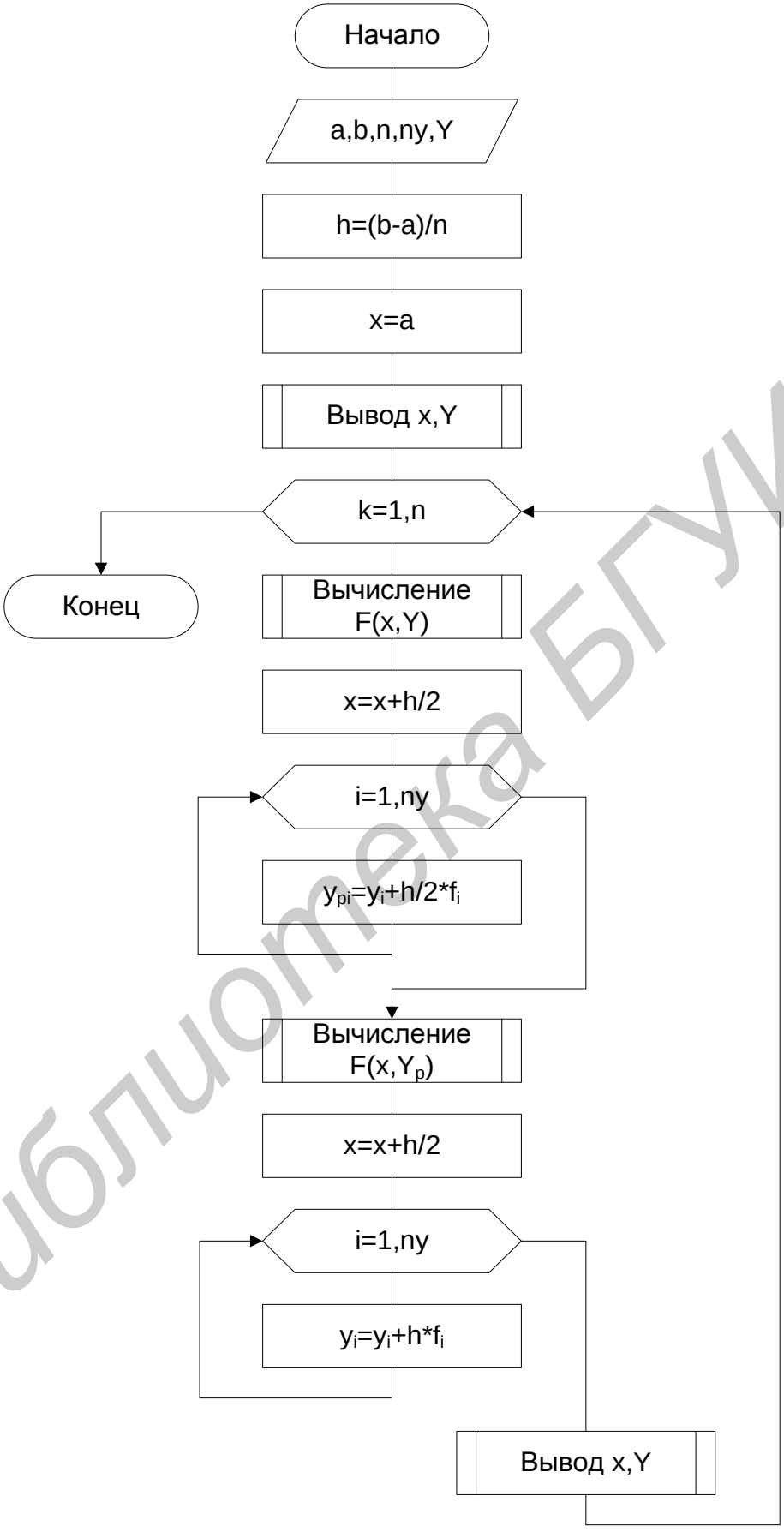

Рис. 6.3. Схема алгоритма метода Рунге – Кутта 2-го порядка

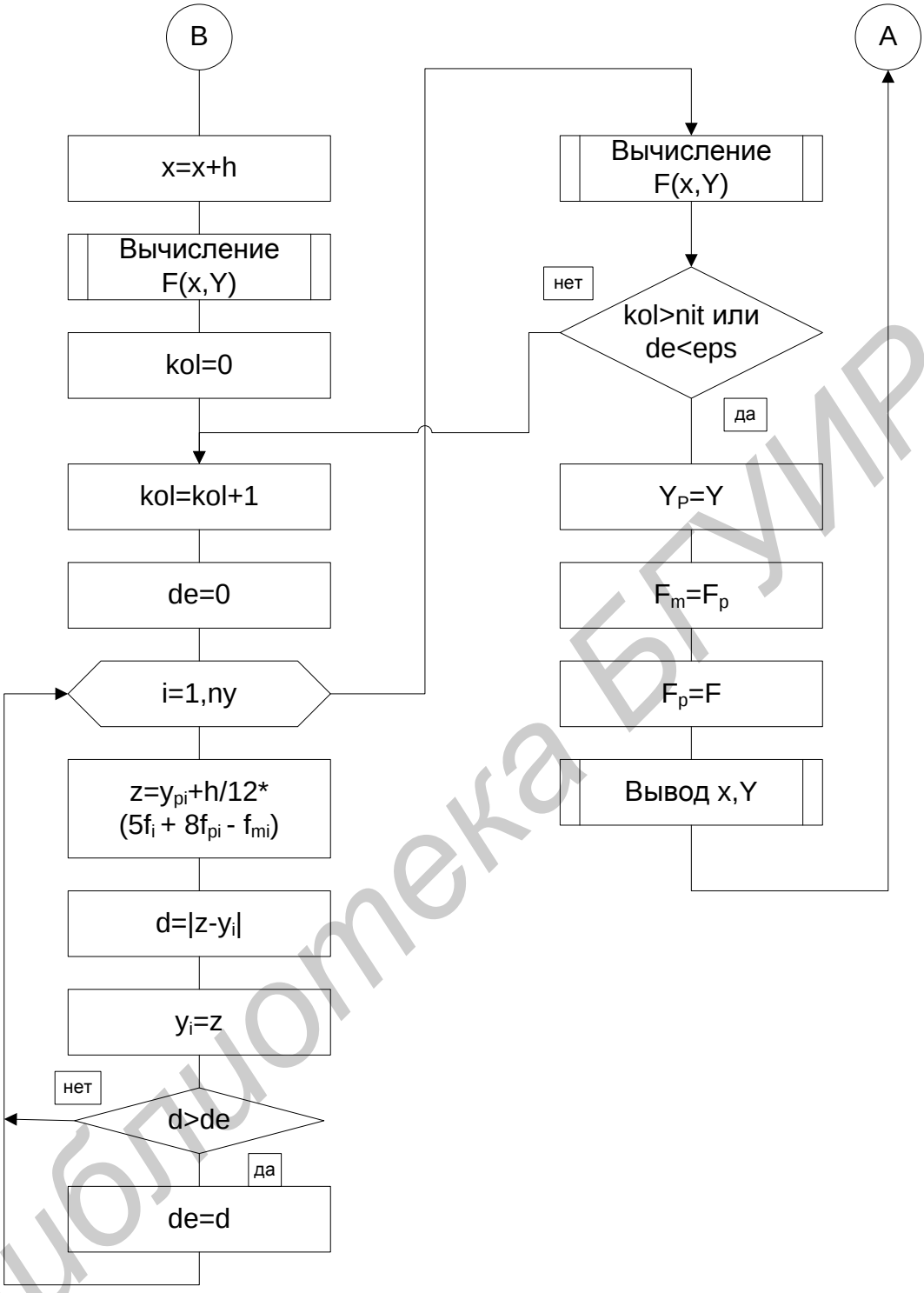

Рис. 6.4. Схема алгоритма неявного метода Адамса 3-го порядка

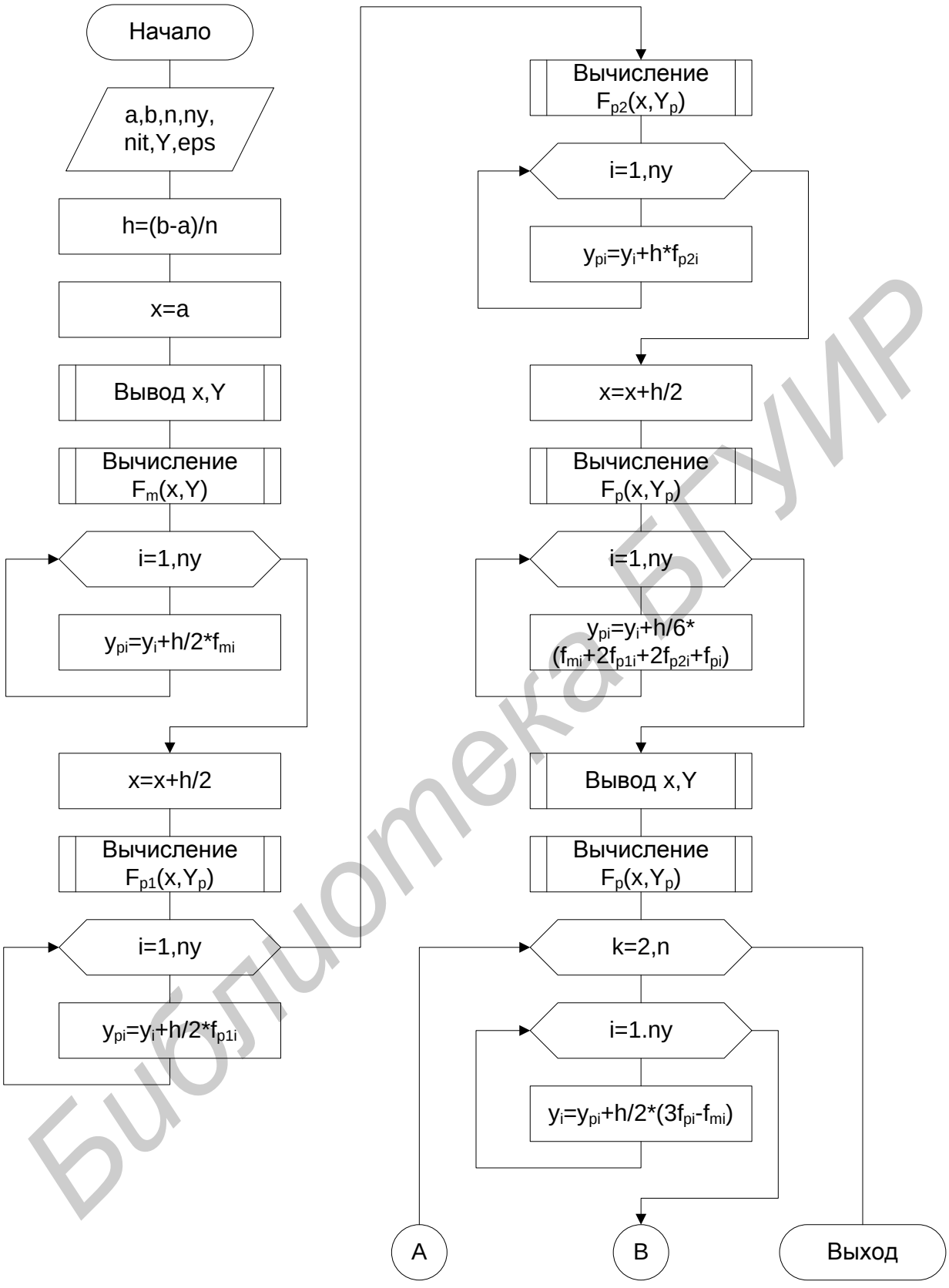

Рис. 6.4. Окончание (начало см. с. 65)

ЛИТЕРАТУРА

1. Калиткин, Н. Н. Численные методы / Н. Н. Калиткин. – М. : Наука, 1978.

2. Бахвалов, Н. С. Численные методы / Н. С. Бахвалов. – М. : Наука, 1975.

3. Волков, Е. А. Численные методы / Е. А. Волков. – М. : Наука, 1982.

4. Крылов, В. И. Вычислительные методы высшей математики / В. И. Крылов, В. В. Бобков, П. И. Монастырный. – Минск : Выш. шк., 1972. – Т. 1.

5. Крылов, В. И. Вычислительные методы высшей математики / В. И. Крылов, В. В. Бобков, П. И. Монастырный. – Минск : Выш. шк., 1975. – Т. 2.

6. Машинные методы математических вычислений / Дж. Форсайт [и др.]. – М. : Мир, 1980.

7. Марчук, Г. И. Введение в проекционно-сеточные методы / Г. И. Марчук, В. И. Агошков. – М. : Наука, 1981.

8. Марчук, Г. И. Методы вычислительной математики / Г. И. Марчук. – М. : Наука, 1980.

9. Шуп, Т. Решение инженерных задач на ЭВМ / Т. Шуп. – М. : Мир, 1982.

10. Самарский, А. А. Введение в численные методы / А. А. Самарский. – М. : Наука, 1982.

11. Березин, И. С. Методы вычислений / И. С. Березин, Н. П. Жидков. – М. : Физматгиз, 1962. – Т. 1.

12. Березин, И. С. Методы вычислений / И. С. Березин, Н. П. Жидков. – М. : Физматгиз, 1970. – Т. 2.

13. Банди, Б. Методы оптимизации. Вводный курс / Б. Банди. – М. : Мир, 1989.

14. Копченова, Н. В. Вычислительная математика в примерах и задачах : учеб. пособие / Н. В. Копченова, И. А. Марон. – М. : Наука, 1972.

15. Синицын, А. К. Алгоритмы вычислительной математики : учеб.-метод. пособие по курсу «Основы алгоритмизации и программирования» / А. К. Синицын, А. А. Навроцкий. – Минск : БГУИР, 2007.

Св. план 2011, поз. 39

Учебное издание

Шестакович Вячеслав Павлович

ОСНОВЫ ЧИСЛЕННЫХ МЕТОДОВ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *Н. В. Гриневич*
Корректор *Е. Н. Батурчик*

Подписано в печать Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. Уч.-изд. л. 3,8. Тираж 150 экз. Заказ 319.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6