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

Кафедра «Вычислительные методы и программирование»

А. К. Синицын, А. А. Навроцкий

АЛГОРИТМЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

Учебно-методическое пособие по курсу
«Основы алгоритмизации и программирования»

Минск 2007

УДК 519.6 (075.8)
ББК 22.193 я 73
С 38

Синицын, А. К.

С 38 Алгоритмы вычислительной математики : учебно-метод. пособие по курсу «Основы алгоритмизации и программирования» / А. К. Синицын, А. А. Навроцкий. – Минск : БГУИР, 2007. – 80 с. : ил.
ISBN 978-985-488-112-6

В пособии изложены классические методы решения основных задач вычислительной математики.

Предназначено для студентов всех специальностей и всех форм обучения БГУИР.

УДК 519.6 (075.8)
ББК 22.193 я 73

Авторы выражают благодарность преподавателям кафедры ВМиП С. В. Колосову, А. А. Бурцеву, В. А. Новикову, В. В. Соловьеву, Т. М. Кривоносовой, принимавшим в разное время участие в обсуждении данного пособия.

ISBN 978-985-488-112-6

СОДЕРЖАНИЕ

ТЕМА 1. МАТЕМАТИЧЕСКИЕ МОДЕЛИ И ЧИСЛЕННЫЕ МЕТОДЫ

1.1. КАК ИСПОЛЬЗОВАТЬ КОМПЬЮТЕР?

После того как получены начальные навыки работы с компьютером, возникает естественный вопрос: что программировать?

Следует четко представлять два уровня использования ЭВМ.

Пользователи готовых программ. В каждой области деятельности уже накоплен большой набор готовых программ для решения наиболее часто встречающихся задач. Возможности современных ПЭВМ позволяют организовать сервис так, что пользователю требуется лишь отвечать на запросы и задавать исходные данные решаемой задачи на привычном специалисту языке. При этом, когда вы нажимаете ту или иную клавишу, бывает совершенно не обязательно владеть навыками программирования и знать, что происходит внутри ЭВМ. Конечно, знание этих вопросов дает уверенность в получаемых результатах и понимание ограничений, налагаемых используемыми методами.

Разработчики пакетов программ. Обычно, если вы придумываете новую методику или сталкиваетесь с новой малоизученной проблемой, оказывается, что отсутствуют программы, позволяющие ее решить. А это всегда случается, если вы работаете творчески и желаете создать что-то новое. Вот здесь и возникает проблема разработки новых пакетов программ.

В настоящее время на первое место по объему вышли задачи поиска, обработки и передачи информации. С этими задачами сталкивается любая организация и большинство людей: наличие билетов в кассе, бухгалтерский учет, обработка и верстка текстов, компьютерные игры, биржевые операции и др. Большой объем составляют программы сервиса компьютера – операционные системы, языки программирования, компьютерные игры, базы данных. Эти программы ввиду значительного тиража представляют коммерческий интерес и приносят быструю ощутимую прибыль, так как облегчают нашу жизнь, позволяют организовать досуг, быстро получить требуемую информацию.

В начале эры ЭВМ основные усилия были направлены на разработку программ ***моделирования, оптимизации и управления физическими процессами.*** Эти задачи решаются, как правило, узкими специалистами. Тираж программ относительно небольшой, и поэтому коммерческого интереса они не представляют. Окупаются они за счет уменьшения затрат на натурное моделирование, оптимизации конструкций. Именно эти уникальные программы являются двигателем прогресса, позволяют совершенствовать технологию, с их помощью открываются новые явления, автоматизируются процессы управления.

В настоящее время объем таких задач и соответственно программ возрастает, хотя их доля резко уменьшилась. Их решение основано на формулировке

математической постановки задачи, разработке вычислительного алгоритма ее решения на ЭВМ и его программирования.

1.2. КАК ИССЛЕДУЮТСЯ ФИЗИЧЕСКИЕ ЯВЛЕНИЯ И РЕШАЮТСЯ ЗАДАЧИ?

Имеется два способа решения инженерных и физических задач: экспериментальный и теоретический.

Экспериментальный метод предполагает создание экспериментальной установки и выполнение натурных экспериментов с целью выбора оптимальных параметров. Как правило, такие исследования связаны с большими материальными затратами, иногда в принципе невозможны (например если это эксперименты с атомной электростанцией).

Теоретический метод, или математическое моделирование, опирается на знание фундаментальных законов природы, используя которые, строят математическую модель исследуемого явления и, решая математические задачи, производят выбор оптимальных параметров исследуемой конструкции.

Математическая модель – это описание исследуемого явления с помощью математических символов и операций над ними.

Математическая постановка задачи предполагает описание математической модели и указание цели ее исследования. Для одной и той же модели могут быть сформулированы различные задачи. Например, наиболее часто встречающейся моделью является понятие функциональной зависимости $y=f(x)$. Для нее могут быть поставлены различные задачи, например: 1) найти $max\ f(x)$; 2) найти x, при котором $f(x)=0$, и др.

Оказывается, наш мир устроен на удивление однообразно с точки зрения математических моделей. Практически все окружающие нас явления описываются сравнительно небольшим количеством математических моделей, умещающихся в стандартный справочник. Этот справочник называется «Курс высшей математики». В нем также приведены все представляющие интерес постановки задач, доказано существование и методы нахождения решений.

Решить задачу – это значит указать алгоритм (т.е. строгую последовательность действий) для получения требуемого результата из исходных данных.

Методы (алгоритмы) решения математических задач можно разделить на точные, приближенные и численные.

К точным методам относятся алгоритмы, позволяющие за конечное число действий получить в принципе (если нет ошибок округления) точное решение. Обычно оно получается в виде формулы или конечного вычислительного алгоритма.

Приближенные – это методы, позволяющие за счет некоторых допущений свести решение исходной задачи к задаче, имеющей точное решение.

Численные методы предполагают разработку ***вычислительного алгоритма***, т.е. конечной, строгой последовательности арифметических и логических действий, обеспечивающих получение решения с заданной

контролируемой погрешностью. Изначально ЭВМ «умеет» выполнять только арифметические и логические операции. Но по мере того как она «учится», в ее память закладывают вычислительные алгоритмы решения все более сложных задач. Поэтому ***вычислительным*** можно считать алгоритм, последовательность действий которого «умеет» выполнять ЭВМ.

1.3. КАК ОЦЕНИВАЕТСЯ ПОГРЕШНОСТЬ ВЫЧИСЛЕНИЙ?

Большое значение при вычислениях придается анализу погрешностей.

Погрешность обычно оценивают одним числом ε, характеризующим близость между точным и приближенным значением некоторой величины.

Близость мы привыкли оценивать расстоянием между объектами. Если рассматриваемая величина есть число a и $\tilde{a}$ – его приближенное значение, то погрешность есть просто модуль разности $\varepsilon = |a - \tilde{a}|$ или расстояние между точками a и $\tilde{a}$ на числовой оси. Чем ближе $\tilde{a}$ к a, тем меньше ε. А как оценить близость между двумя функциями $f(x)$ и $g(x)$ (векторами $\vec{x}, \vec{y}$, матрицами A, B)? Что такое «близко» для этих объектов?

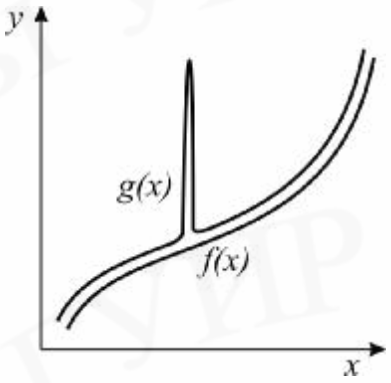


Рис. 1.1

Оказывается, в зависимости от выбранного класса функций (непрерывные; разрывные, но интегрируемые; дифференцируемые) близость может трактоваться по-разному. Например, две функции $f(x)$ и $g(x)$ на рис.1.1 в одних инженерных задачах могут счи-таться близкими, а в других – нет. Ввиду этого должны быть предложены математические модели рас-стояний, различные для функций (матриц векторов) из разных классов. Обычно при выполнении расчетов имеют дело с объектами, являющимися элементами некоторого нормированного пространства.

Нормированным пространством называется такое множество элементов (здесь в качестве элементов рассматриваются функции, векторы или матрицы), в котором каждому элементу поставлено в соответствие число $\|X\|$ – (норма X), удовлетворяющее следующим аксиомам:

1. $\|X\| \geq 0$, норма – положительное число.
2. $\|X\| = 0$, только при $X = \varnothing$ ($\varnothing$– нулевой элемент).
3. $\|\alpha X\| = |\alpha| * \|X\|$, α – число (может быть комплексным).
4. $\|X_1 \pm X_2\| \leq \|X_1\| + \|X_2\|$ – неравенство треугольника.

Расстояние между элементами $r(X_1, X_2) = \|X_1 - X_2\|$ определяется через норму. В зависимости от класса рассматриваемых объектов норма, а следовательно, и расстояние, вводится по-разному.

Наиболее часто используются следующие нормированные пространства:

$C[a,b]$ – ***множество непрерывных функций*** $\{f(x), g(x), h(x), \ldots\}$, определенных на интервале $[a, b]$.

Норма и расстояние в *C[a, b]* определяются по формулам

$$\|f\|_c = \max_{x\in[a,b]} |f(x)|; \quad r_c(f,g) = \max_{x\in[a,b]} |f(x) - g(x)|;$$

$L_2[a, b]$ – ***множество функций, интегрируемых с квадратом***, для которых $\int_a^b f^2(x)dx < \infty$. В $L_2[a, b]$ имеются и разрывные функции.

Норма и расстояние:

$$\|f\|_{l_2} = \sqrt{\int_a^b f^2(x)dx}\ ; \ r_{L_2}(f,g) = \sqrt{\int_a^b f(x)g(x)dx}.$$

В $L_2[a, b]$ вводится ***скалярное произведение*** $(f,\ q) = \int_a^b f(x)\cdot g(x)dx,$, причем $(f, f) = \|f\|_{L_2}^2$.

Две функции из L_2 называются ***ортогональными***, если $(f, g) = 0$.

Заметим, что функции f и g на рис. 1.1 будут «близкими» в пространстве L_2 и «далекими» в пространстве C.

В множестве векторов $\{\vec{x}, \vec{y},...\}$, $\vec{x} = (x_1 x_2 ... x_n)$ нормы также можно ввести по-разному: $\|\vec{x}\|_{L_2} = \sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}$; $\|\vec{x}\|_c = \max_i |x_i|$.

В множестве квадратных матриц наиболее употребительны следующие нормы (λ_i – собственные значения матрицы A):

$$\|A\|_c = \max_i \left(\sum_{j=i}^n |a_{ij}| \right); \quad \|A\|_E = \sqrt{\sum_{ij=1}^n |a_{ij}|^2}; \quad \|A\|_\lambda = \sqrt{\max_i \lambda_i} \quad .$$

Абсолютная погрешность ΔX между точным X и приближенным $\hat{X}$ значениями некоторого элемента определяется через норму разности $\Delta X = \|X - \hat{X}\|$.

Относительная погрешность δX определяется как отношение абсолютной погрешности к норме элемента $\delta X = \Delta X / \|\hat{X}\|$.

1.4. ОТКУДА ВОЗНИКАЮТ ПОГРЕШНОСТИ РАСЧЕТОВ?

Есть четыре источника погрешности результата, о которых следует помнить при выполнении расчетов.

1. Неточность математической модели. Любая модель является определенной идеализацией рассматриваемого физического явления и описывает лишь основные факторы, существенные при решении конкретной технической задачи. Уточнение модели за счет введения описания дополнительных факторов обычно приводит к ее усложнению и, как следствие, к трудности использования. Поэтому необходим определенный компромисс.

Выбор удачного компромисса – это творческий процесс, требующий большого опыта и инженерной интуиции.

2. Погрешность исходных данных. Исходные данные обычно получаются из измерений, либо, наоборот, по этим данным затем делается устройство. В каждом случае имеется так называемая неустранимая погрешность между исходными данными, участвующими в расчетах, и теми, которые реализуются. В результате этого получаемое решение также будет отличаться от реализуемого в устройстве. В зависимости от того, как ошибки исходных данных отражаются на результате, задачи разделяют на два класса: ***корректные*** и ***некорректные***. Задача называется ***корректной***, если малые ошибки исходных данных приводят к пропорционально малым ошибкам решения. Наоборот, если малые ошибки исходных данных приводят к большим ошибкам в результатах, задача называется ***некорректной***. Чаще всего некорректность математической задачи является признаком неправильной математической постановки. Для решения некорректных, но правильных с физической точки зрения задач разрабатываются специальные методы.

3. Погрешность метода. При построении вычислительного алгоритма обычно точное решение представляется в виде бесконечного предела последовательности арифметических и логических действий. При ограничении лишь конечным числом вычислений вносится погрешность, контролируемая некоторыми параметрами метода. Получение зависимости погрешности решения от параметров вычислительного метода является одной из основных задач вычислительной математики.

Например, чтобы вычислить значение функции $y=e^{-x}$ при $x>0$, предлагается в качестве вычислительного метода $M(x)$ взять n первых членов ряда $e^{-x} \cong M(x) = 1 - x + \frac{x^2}{2} - ...(-1)^n \cdot \frac{x^n}{n!}$. Как видно, при расчете $M(x)$ используются только арифметические действия. Параметром метода здесь является n, погрешность метода при этом оценивается последним отброшенным членом $\varepsilon_n = \left\| M(x) - e^{-x} \right\|_c < \frac{x^{n+1}}{(n+1)!}$ и при $n \to \infty$, $\varepsilon_n \to 0$.

Однако не всегда можно так просто найти зависимость погрешности от параметра метода, поэтому часто используют асимптотические оценки. Обычно при уменьшении некоторого параметра h метода погрешность решения e_h стремится к нулю, т.е. $\varepsilon_h \to 0$ при $h \to 0$, например в вышеприведенном примере $h = 1/n$. В этом случае, если выполняется оценка $\varepsilon_h < Ch^p$, где C = const и не зависит от h, считается, что ***порядок погрешности*** равен p, и обозначается коротко $\varepsilon_h \approx O(h^p)$.

4. Ошибки округлений. Все расчеты на ЭВМ производятся с конечным числом значащих цифр, определяемым объемом ячеек памяти. Поэтому при вычислении, например 1/3 = 0.3333...3..., если округление производится на

седьмом знаке, то вносится ошибка $e \approx 10^{-8}$. Когда вычислений много, то такие ошибки могут накапливаться и, наоборот, компенсироваться (положительные и отрицательные). В зависимости от реакции на погрешность округлений вычислительные методы разделяются на ***устойчивые*** и ***неустойчивые***.

Метод ***устойчив***, если в процессе вычислений ошибки округлений не накапливаются, в противном случае метод неустойчив. Неустойчивость обычно устанавливается путем проведения прямых расчетов с различными значениями параметра метода. При увеличении количества вычислений по неустойчивому методу ошибки быстро нарастают, что приводит к переполнению ЭВМ. Одной из задач вычислительной математики является установление условий устойчивости и разработка рекомендаций по созданию устойчивых методов.

1.5. ИТЕРАЦИОННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ

Символически решаемую задачу можно записать в виде

$$A(X) = b. \qquad (1.1)$$

Здесь A – заданный оператор; X, b – элементы некоторых нормированных пространств, причем b задано и требуется найти X.

Обозначим через X^* точное решение поставленной задачи (X^* может быть числом, вектором, функцией).

Одним из наиболее часто используемых вычислительных методов является ***метод итераций*** и различные его модификации.

Итерационные методы основаны на построении сходящейся к точному решению X^* бесконечной рекуррентной последовательности $X^0, X^1, ..., X^k \underset{k\to\infty}{\to} X^*$ элементов той же природы, что и X^* (числа, векторы, функции).

Последовательность называется ***рекуррентной*** порядка m, если каждый следующий ее член выражается через m предыдущих по некоторому правилу:

$$X^k = \varphi(X^{k-1}, X^{k-2}, ... X^{k-m}). \qquad (1.2)$$

Соответствующий итерационный метод называется ***m-шаговым***. Для реализации m-шагового метода требуется задать m первых членов, $\{X^0, X^1, ..., X^{m-1}\}$, называемых ***начальным приближением***. Зная начальное приближение, по формуле (1.2) последовательно находят $X^m, X^{m+1}, ... X^k, ...$ Процесс получения следующего k-го члена через предыдущие называется k-й ***итерацией***. Итерации выполняются до тех пор, пока очередной член X^k не будет удовлетворять заданной точности, т.е. $\|X^k - X^*\| < \delta$. Ввиду того, что точное решение X^* заранее неизвестно, обычно сходимость метода определяют по близости двух последних членов, т.е. расчеты производят до тех пор, пока не выполнится условие $\|X^k - X^{k-1}\| < \varepsilon$, где ε – некоторая заданная малая величина. Из теории сходящихся последовательностей известно, что δ и ε связаны пропорциональной зависимостью, поэтому если взять ε с некоторым

запасом (обычно достаточно ε = δ/10), то требуемая точность будет достигнута. В качестве искомого решения берут последний член последовательности X^k, при котором выполнилось указанное неравенство.

Итерационный метод для решения задачи (1.1) строится следующим образом.

Преобразуем задачу (1.1) к виду, разрешенному относительно неизвестного X:

$$X = \varphi(X). \tag{1.3}$$

При этом точное решение (1.1) X^* является и решением (1.3). Заметим, что привести задачу (1.1) к виду (1.3), не изменяя решения, можно различными способами, например: $X = \alpha[A(X) - b] + X = \varphi(X)$, где α – произвольный параметр, который в дальнейшем подбирается из условия сходимости итераций.

Используем выражение (1.3) в качестве рекуррентной формулы (m=1):

$$X^k = j(X^{k-1}). \tag{1.4}$$

Задав одно X^0 (начальное приближение), последовательно находим X^1, X^2, ..., X^k. Если полученная таким образом последовательность сходится к некоторому конечному пределу, то этот предел совпадает с точным решением X^*.

Условие сходимости последовательности, задаваемой рекуррентной формулой (1.4), определяется выполнением неравенства

$$\|dj / dX\| = \|G\| < 1. \tag{1.5}$$

Чем ближе $\|G\|$ к нулю, тем быстрее сходится рекуррентная последовательность к точному решению и, следовательно, для получения решения с заданной точностью требуется меньше итераций.

1.6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое математическая модель и ее погрешность?
2. Что такое математическая постановка задачи?
3. Что такое численный метод, чем он отличается от других методов?
4. Какие нормированные пространства вы знаете?
5. Как оценить погрешность метода в нормированном пространстве?
6. Что понимается под корректностью задачи?
7. Что понимается под устойчивостью (неустойчивостью) метода?
8. Что такое итерационный метод, как определяется его погрешность?

ТЕМА 2. РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ)

2.1. 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, так как большинство методов решения сложных задач основано на сведении их к решению некоторой последовательности СЛАУ.

Обычно СЛАУ записывается в виде

$$\sum_{j=1}^{n} a_{i,j}\, x_j = b_i \;;\; 1 \le i \le n \quad \text{или коротко}$$

$$A\vec{x} = \vec{b}, \qquad A = \begin{bmatrix} a_{1,1}\, a_{1,2} \ldots a_{1,n} \\ a_{2,1}\, a_{2,2} \ldots a_{2,n} \\ \ldots\ldots\ldots\ldots\ldots \\ a_{n,1}\, a_{n,2} \ldots a_{n,n} \end{bmatrix}; \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}; \; \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix}. \tag{2.1}$$

Здесь A и $\vec{b}$ заданы, требуется найти $\vec{x}^*$, удовлетворяющий (2.1).

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

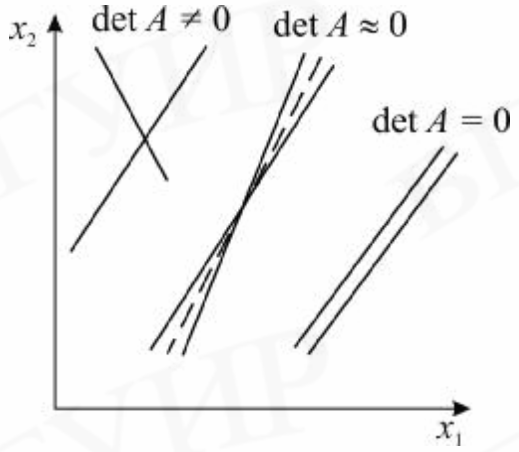


Рис. 2.1

Плохо обусловленная система двух уравнений геометрически соответствует почти параллельным прямым (рис. 2.1). 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 $n \le 10^3 - 10^4$ 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 (см. подразд. 1.5) $\vec{x}^0, \vec{x}^1, \vec{x}^2, ..., \vec{x}^k \xrightarrow[k\to\infty]{} \vec{x}^*$. Итерации выполняют до тех пор, пока норма разности $\delta_k = \left\| \vec{x}^k - \vec{x}^{k-1} \right\|_c = \max_i \left| x_i^k - x_i^{k-1} \right| \le \varepsilon$. Последнее $\vec{x}^k$ берут в качестве приближенного решения.

Итерационные методы выгодны для систем большого порядка $n > 1000$, 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 $\vec{x}^k$:

$$\Delta = \max_{1 \le i \le n} \left| b_i - \sum_{j=1}^{n} a_{i,j} x_j^k \right|;$$

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

Ниже приведено описание алгоритмов указанных методов решения СЛАУ.

2.2. ПРЯМЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ

Метод Гаусса (MG)

Метод основан на приведении с помощью преобразований, не меняющих решение, исходной СЛАУ (2.1) с произвольной матрицей к СЛАУ с верхней треугольной матрицей вида

$$\begin{aligned} a'_{1,1} x_1 + a'_{1,2} x_2 + ... + a'_{1,n} x_n &= b'_1 \\ a'_{2,2} x_2 + ... + a'_{2,n} x_n &= b'_2 \\ ..... & \\ a'_{n,n} x_n &= b'_n \end{aligned} \qquad (2.2)$$

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

Решение системы с верхней треугольной матрицей (2.2), как легко видеть, находится по формулам, называемым ***обратным ходом метода Гаусса*:**

$$x_n = b'_n / a'_{n,n}; \quad x_k = \frac{1}{a'_{k,k}}\left[b'_k - \sum_{i=k+1}^{n} a'_{k,i}\, x_i \right], \quad k = n-1, \ n-2, \ ..., \ 1. \qquad (2.3)$$

Прямой ход метода Гаусса осуществляется следующим образом: вычтем из каждого *m*-го уравнения *(m = 2 ... n)* первое уравнение, умноженное на $a_{m,1} / a_{1,1}$, и вместо *m*-го уравнения оставим полученное. В результате в матрице системы исключаются все коэффициенты первого столбца ниже диагонального. Затем, используя второе полученное уравнение, аналогично исключим элементы второго столбца *(m = 3 ... 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 (2.2). При указанных операциях решение СЛАУ не изменяется.

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

$$a_{m,i} = a_{m,i} - a_{k,i} \frac{a_{m,k}}{a_{k,k}}, \qquad k = 1, \ n-1, \quad i = k, \ n;$$

$$b_m = b_m - b_k \frac{a_{m,k}}{a_{k,k}}, \qquad m = k+1, \ n. \qquad (2.4)$$

Элементы $a_{k,k}$ называются *главными*. Заметим, что если в ходе расчетов по данному алгоритму на главной диагонали окажется нулевой элемент $a_{k,k} = 0$, то произойдет сбой в ЭВМ. Для того чтобы этого избежать, следует каждый цикл по *k* начинать с перестановки строк: среди элементов *k*-го столбца $a_{m,k}$, $k \le m \le n$ 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, так как в формулах (2.4) при этом производится умножение на числа $a_{m,k} / a_{k,k}$, меньшие единицы, и ошибка, возникшая ранее, уменьшается.

Схема алгоритма с выбором главного элемента приведена на рис. 2.2.

Проиллюстрируем метод Гаусса на решении СЛАУ 3-го порядка:

$$\left.\begin{aligned} 2x_1 + \ x_2 \ - \ x_3 &= 1 \\ 4x_1 + 6x_2 + 2x_3 &= 6 \\ 6x_1 + 5x_2 + 8x_3 &= 14 \end{aligned}\right\| \begin{aligned} &\\ &-I \times 2 \\ &-I \times 3. \end{aligned}$$

Первый цикл: вычтем из второго уравнения первое, умноженное на $a_{2,1} / a_{1,1} = 2$, а из третьего – первое, умноженное на $a_{3,1} / a_{1,1} = 3$; получим

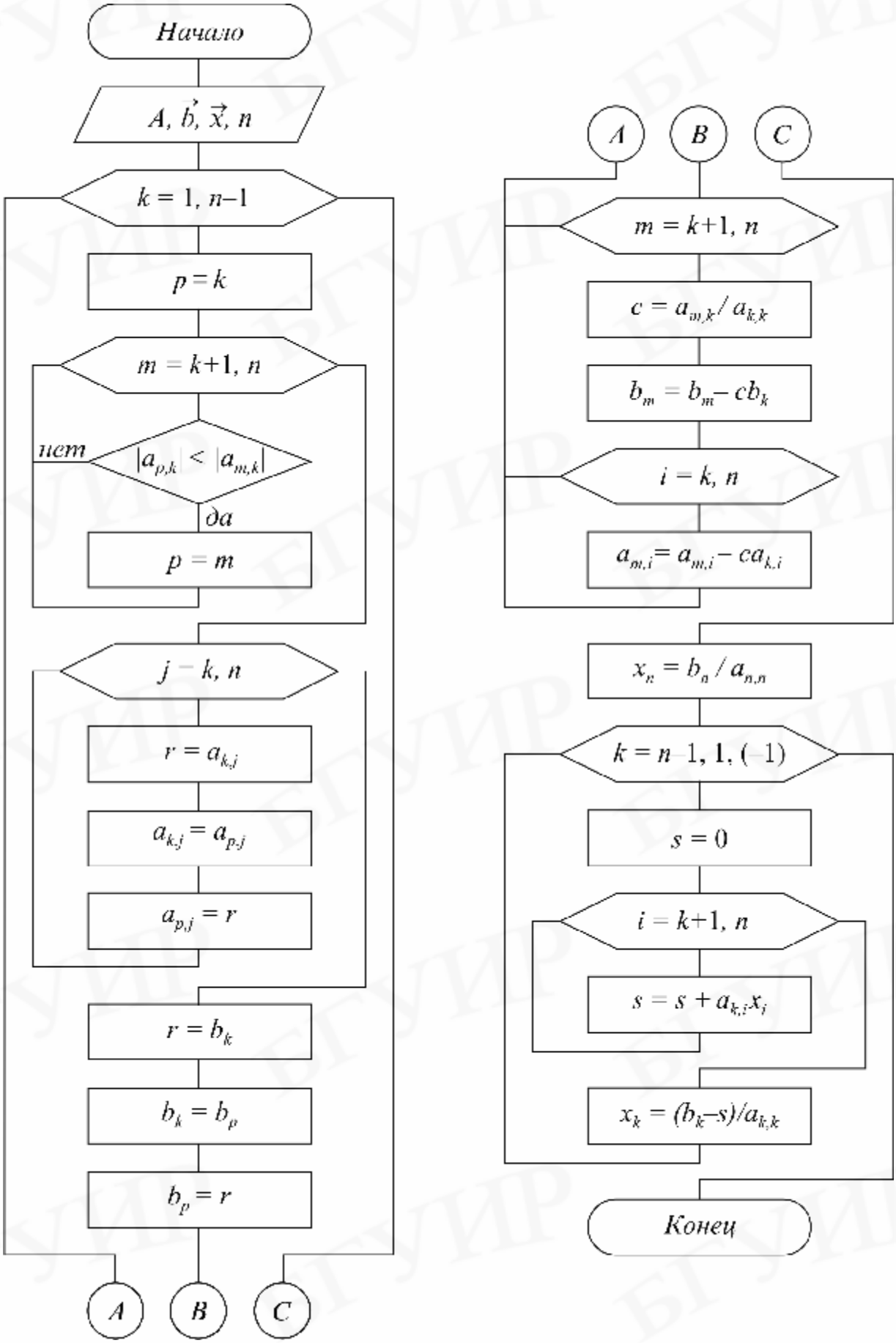

Начало
A, b, x, n
k = 1, n–1
p = k
m = k+1, n
нет
|a_p,k| < |a_m,k|
да
p = m
j = k, n
r = a_k,j
a_k,j = a_p,j
a_p,j = r
r = b_k
b_k = b_p
b_p = r
A
B
C
A
B
C
m = k+1, n
c = a_m,k / a_k,k
b_m = b_m – cb_k
i = k, n
a_m,i = a_m,i – ca_k,i
x_n = b_n / a_n,n
k = n–1, 1, (–1)
s = 0
i = k+1, n
s = s + a_k,i x_i
x_k = (b_k–s)/a_k,k
Конец

Рис. 2.2

$$\begin{array}{l} 2x_1 + x_2 - x_3 = 1 \\ 0 \quad 4x_2 + 4x_3 = 4 \\ 0 \quad 2x_2 + 11x_3 = 11 \end{array} \Bigg\| \; -II \times 0,5.$$

Второй цикл: вычтем из третьего уравнения второе, умноженное на $a_{3,2}/a_{2,2} = 0{,}5$; получим систему с треугольной матрицей вида (2.2):

$$\begin{array}{l} 2x_1 + x_2 - x_3 = 1 \\ 4x_2 + 4x_3 = 4 \\ 0 \quad 9x_3 = 9. \end{array}$$

Обратный ход: из последнего уравнения находим $x_3 = 1$, подставляя во второе, находим $x_2 = 0$, подставляя в первое, находим $x_1 = 1$.

Таким образом, получен вектор решения $\mathbf{r}_* \; \overset{*}{x} = (x_1^*,\ x_2^*,\ x_3^*) = (1,\ 0,\ 1)$.

Метод прогонки (МР)

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

$$\begin{vmatrix} q_1 & r_1 & 0 & 0 & \ldots & 0 & 0 & 0 \\ p_2 & q_2 & r_2 & 0 & \ldots & 0 & 0 & 0 \\ 0 & p_3 & q_3 & r_3 & \ldots & 0 & 0 & 0 \\ .. & .. & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & p_n & q_n & r_n \\ 0 & 0 & 0 & 0 & \ldots & 0 & p_{n1} & q_{n1} \end{vmatrix} \times \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ \ldots \\ x_n \\ x_{n1} \end{vmatrix} = \begin{vmatrix} d_1 \\ d_2 \\ d_3 \\ \ldots \\ d_n \\ d_{n1} \end{vmatrix}. \qquad (2.5)$$

или коротко эту систему записывают в виде

$$\begin{array}{l} q_1 x_1 + r_1 x_2 = d_1 \\ p_i x_{i-1} + q_i x_i + r_i x_{i+1} = d_i \\ p_{n1} x_n + q_{n1} x_{n1} = d_{n1}, \\ n1 = n+1,\ 2 \le i \le n. \end{array} \qquad (2.6)$$

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 (2.5) можно привести к виду

$$\begin{vmatrix} 1 & -\xi_1 & 0 & \ldots & 0 & \ldots & 0 & 0 \\ 0 & 1 & -\xi_2 & \ldots & 0 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 & \ldots & 1 & -\xi_n \\ 0 & 0 & 0 & \ldots & 0 & \ldots & 0 & 1 \end{vmatrix} \times \begin{vmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \\ x_{n1} \end{vmatrix} = \begin{vmatrix} \eta_1 \\ \eta_2 \\ \ldots \\ \eta_n \\ \eta_{n1} \end{vmatrix}. \qquad (2.7)$$

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

$$\xi_1 = -r_1 / q_1; \qquad \eta_1 = d_1 / q_1;$$
$$\xi_i = -r_i /(q_i + p_i\xi_{i-1}); \quad \eta_i = (d_i - p_i\eta_{i-1})/(q_i + p_i\xi_{i-1}); \qquad (2.8)$$
$$i = 2 \ \ldots \ n.$$

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

$$x_{n1} = (d_{n1} - p_{n1}\eta_n)/(q_{n1} + p_{n1}\xi_n);$$
$$x_i = \xi_i x_{i+1} + \eta_i \ ; \qquad (2.9)$$
$$i = n, \ n-1, \ \ldots, \ 1.$$

Расчетные формулы (2.8), (2.9) получили название «метод прогонки». Достаточным условием того, что в формулах метода прогонки не произойдет деления на ноль и расчет будет устойчив относительно погрешностей округления, является выполнение неравенства $|q_i| \geq |p_i| + |r_i|$ (хотя бы для одного i должно быть строгое неравенство).

Схема алгоритма метода прогонки представлена на рис. 2.3.

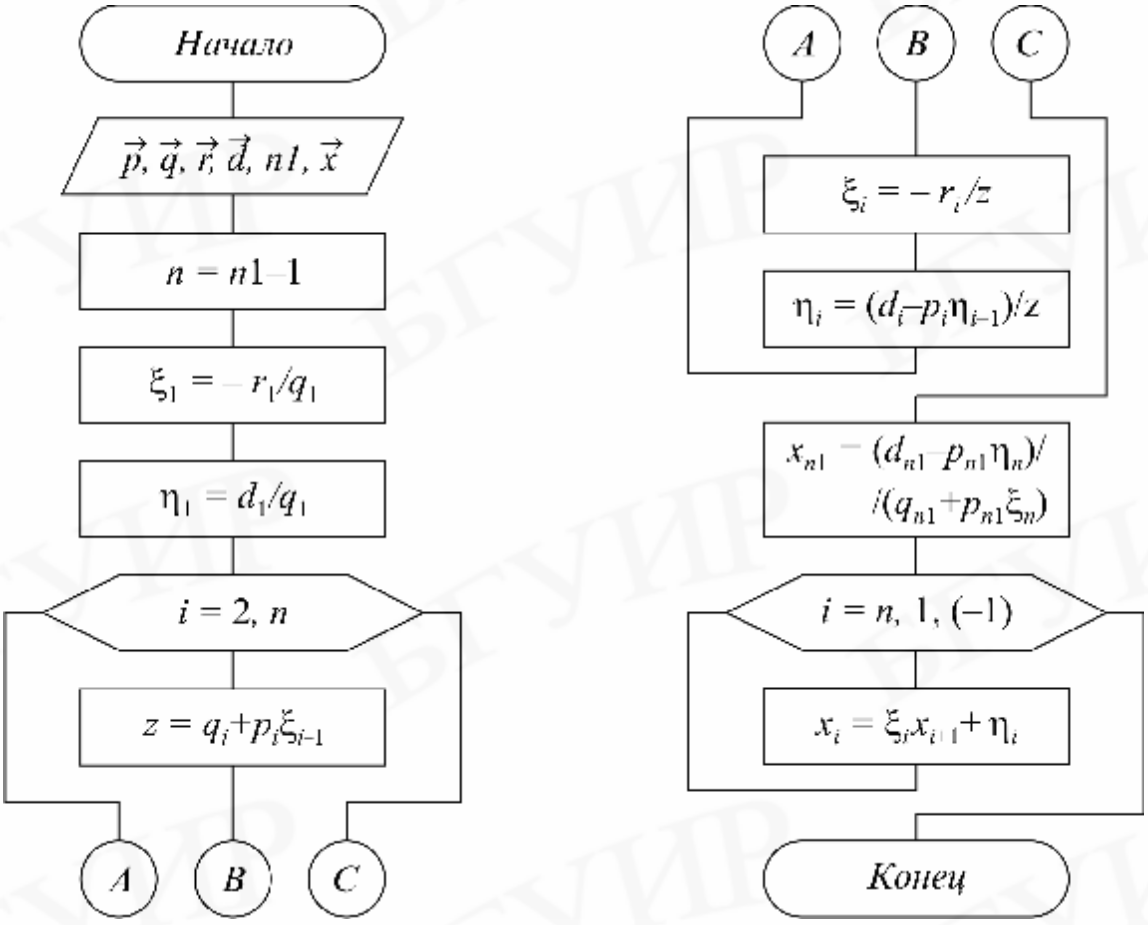


Рис. 2.3

Метод квадратного корня (MQ)

Метод 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=±1; S – верхняя треугольная ($s_{i,k} = 0$, если $i>k$, причем $s_{i,i} > 0$); 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:

$$S^T \cdot \vec{z} = \vec{b}; \quad D\vec{y} = \vec{z}; \quad S\vec{x} = \vec{y}. \tag{2.10}$$

Здесь $\vec{z} = D S \vec{x}$, $\vec{y} = S \vec{x}$.

Решение систем (2.10) ввиду треугольности матрицы S осуществляется по формулам, аналогичным обратному ходу метода Гаусса:

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

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

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

$$d_k = \text{sign}(a_{k,k} - \sum_{i=1}^{k-1} d_i \left| s_{i,k} \right|^2);$$

$$s_{k,k} = \sqrt{\left| a_{k,k} - \sum_{i=1}^{k-1} d_i \left| s_{i,k} \right|^2 \right|}; \tag{2.11}$$

$$k = 1, 2, ..., n;$$

$$s_{k,j} = (a_{k,j} - \sum_{i=1}^{k-1} d_i s_{i,k} s_{i,j}) / (s_{k,k} d_k);$$

$$j = k+1, k+2, ..., n.$$

В этих формулах сначала полагаем $k = 1$ и последовательно вычисляем $d_1 = \text{sign}(a_{1,1})$; $s_{1,1} = \sqrt{\left| a_{1,1} \right|}$ и все элементы первой строки матрицы $S(s_{1,j}, j>1)$, затем полагаем k=2, вычисляем $s_{2,2}$ и вторую строку $s_{1,j}$ для j>2 и т.д.

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

Схема алгоритма метода квадратного корня представлена на рис. 2.4. Функция **sign(x)** возвращает –1 для всех x<0 и +1 для всех x>0.

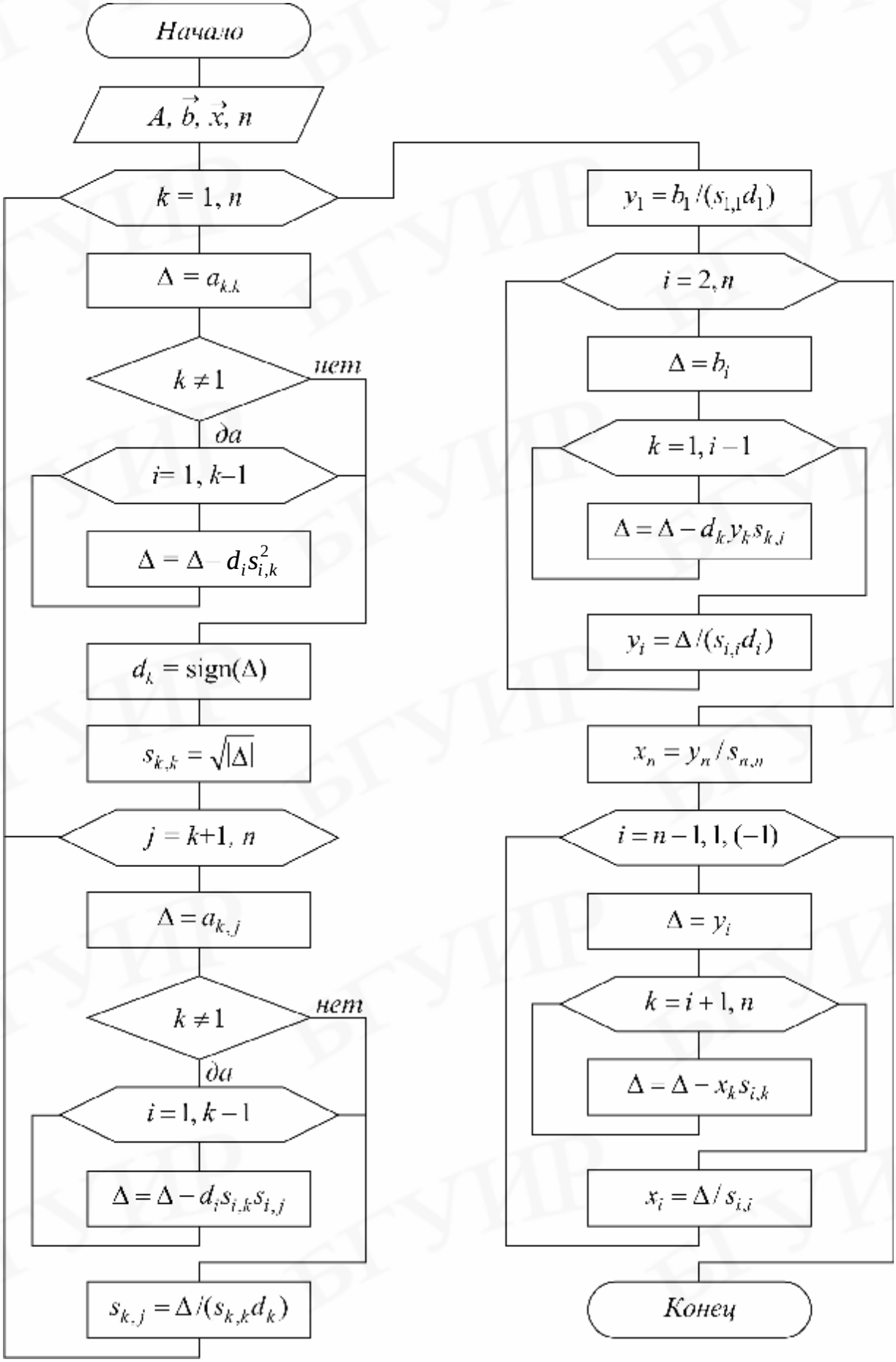

Начало
A, b, x, n
k = 1, n
Δ = a_{k,k}
k ≠ 1
нет
да
i = 1, k–1
Δ = Δ – d_i s²_{i,k}
d_k = sign(Δ)
s_{k,k} = √|Δ|
j = k+1, n
Δ = a_{k,j}
k ≠ 1
нет
да
i = 1, k – 1
Δ = Δ – d_i s_{i,k} s_{i,j}
s_{k,j} = Δ/(s_{k,k} d_k)
y_1 = b_1/(s_{1,1} d_1)
i = 2, n
Δ = b_i
k = 1, i – 1
Δ = Δ – d_k y_k s_{k,i}
y_i = Δ/(s_{i,i} d_i)
x_n = y_n / s_{n,n}
i = n – 1, 1, (–1)
Δ = y_i
k = i + 1, n
Δ = Δ – x_k s_{i,k}
x_i = Δ / s_{i,i}
Конец

Рис. 2.4

Проиллюстрируем метод квадратного корня, решая систему трех уравнений:

$$\begin{cases} x_1 + x_2 + x_3 = 3 \\ x_1 + 2x_2 + 2x_3 = 5 \\ x_1 + 2x_2 + 3x_3 = 6 \end{cases} \quad A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \quad \vec{b} = \begin{bmatrix} 3 \\ 5 \\ 6 \end{bmatrix}. \tag{2.12}$$

Нетрудно проверить, что матрица A есть произведение двух треугольных матриц (здесь $d_i = 1$):

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = S^T \cdot S .$$

Исходную систему запишем в виде

$$S^T \cdot S \cdot \vec{x} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 6 \end{bmatrix}.$$

Обозначим

$$S \cdot \vec{x} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}.$$

Тогда для вектора $\vec{y}$ получим систему $S^T \vec{y} = \vec{b}_i$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 6 \end{bmatrix} \Rightarrow y_1 = 3, \ y_2 = 2, \ y_3 = 1.$$

Зная $\vec{y}$, решаем систему $S\vec{x} = \vec{y}$:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \Rightarrow x_3 = 1, \ x_2 = 1, \ x_1 = 1.$$

2.3. ИТЕРАЦИОННЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ

Метод простой итерации (MI)

В соответствии с общей идеей итерационных методов (см. подразд. 1.5) исходная система (2.1) должна быть приведена к виду, разрешенному относительно $\vec{x}$:

$$\vec{x} = G\vec{x} + \vec{c} = j(\vec{x}), \tag{2.13}$$

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

При этом решение (2.13) должно совпадать с решением (2.1). Затем строится рекуррентная последовательность первого порядка в виде

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

Для начала вычислений задается некоторое начальное приближение $\mathbf{x}^0$ (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 $\|G\| < 1$ (см. подразд. 1.5).

Привести исходную систему к виду (2.13) можно различными способами, например:

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

Здесь 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. $|a_{i,i}| > \sum_{\substack{k=1 \\ k \neq i}}^{n} |a_{i,k}|$, то преобразование (2.1) к (2.13) можно осуществить просто, решая каждое i-е уравнение относительно x_i. В результате получим следующую рекуррентную формулу:

$$x_i^k = -\frac{1}{a_{i,i}}\left[\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j} x_j^{k-1} - b_i\right] = \sum_{j=1}^{n} g_{i,j} x_j^{k-1} + c_i, \tag{2.14}$$

$$g_{i,j} = -a_{i,j} / a_{i,i}; \quad g_{i,i} = 0; \quad c_i = b_i / a_{i,i}.$$

Схема алгоритма представлена на рис. 2.5.

Приведем пример решения методом простой итерации следующей системы трех уравнений:

$$\begin{cases} 4x_1 - x_2 - x_3 = 2; \\ x_1 + 5x_2 - 2x_3 = 4; \\ x_1 + x_2 + 4x_3 = 6. \end{cases} \tag{2.15}$$

Преобразуем ее к виду (2.13), для чего из первого уравнения выразим x_1, из второго x_2 и из третьего x_3 и получим рекуррентную формулу

$$\begin{cases} x_1 = (2 + x_2 + x_3)/4 \\ x_2 = (4 - x_1 + 2x_3)/5 \\ x_3 = (6 - x_1 - x_2)/4 \end{cases} \Rightarrow \begin{cases} x_1^k = (2 + x_2^{k-1} + x_3^{k-1})/4; \\ x_2^k = (4 - x_1^{k-1} + 2x_3^{k-1})/5; \\ x_3^k = (6 - x_1^{k-1} - x_2^{k-1})/4. \end{cases} \tag{2.16}$$

Зададим начальное приближение $\mathbf{x}^0 = (x_1^0, x_2^0, x_3^0) = (0, 0, 0)$, подставим в правую часть (k = 1), получим $\mathbf{x}^1$: $(x_1^1 = 0.5, \quad x_2^1 = 0.8, \quad x_3^1 = 1.5)$.

Подставляя полученные значения снова в (2.16) (k = 2), получим $\mathbf{x}^2$:

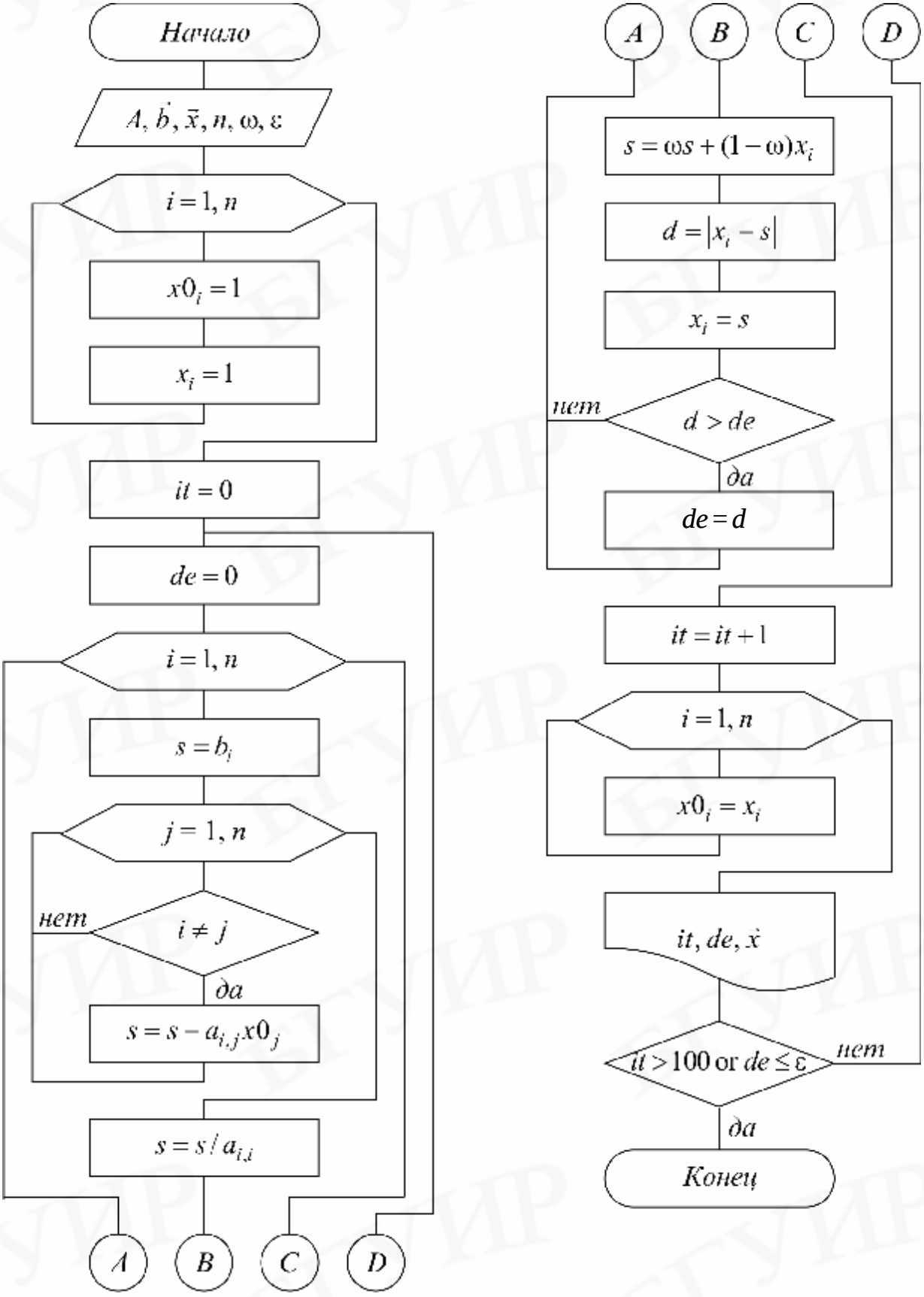

Начало
A, b, x̄, n, ω, ε
i = 1, n
x0_i = 1
x_i = 1
it = 0
de = 0
i = 1, n
s = b_i
j = 1, n
нет
i ≠ j
да
s = s − a_{i,j} x0_j
s = s / a_{i,i}
A
B
C
D
A
B
C
D
s = ωs + (1 − ω)x_i
d = |x_i − s|
x_i = s
нет
d > de
да
de = d
it = it + 1
i = 1, n
x0_i = x_i
it, de, x̄
it > 100 or de ≤ ε
нет
да
Конец

Рис. 2.5

$$x_1^2 = (2 + 0.8 + 1.5)/4 = 1.075;$$
$$x_2^2 = (4 - 0.5 + 2 \cdot 1.5)/5 = 1.3;$$
$$x_3^2 = (6 - 0.5 - 0.8)/4 = 1.175;$$

$\overline{\mathbf{x}}^2 = (1.075;\ 1.3;\ 1.175)$.

Вычислим погрешность на второй итерации:

$$\delta_2 = \left\| \overline{\mathbf{x}}^2 - \overline{\mathbf{x}}^1 \right\|_C = \max_{1 \le i \le 3} \left| x_i^2 - x_i^1 \right| = \max(0.575;\ 0.5;\ 0.325) = 0.575.$$

Если $\delta > \varepsilon$, то процесс продолжаем.

Метод Зейделя (MZ)

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 (2.14) используются вместо x_1^{k-1}, ..., x_{i-1}^{k-1} уже вычисленные ранее x_1^k, ..., x_{i-1}^k, т.е. (2.14) преобразуется к виду

$$x_i^k = \sum_{j=1}^{i-1} g_{i,j} x_j^k + \sum_{j=i+1}^{n} g_{i,j} x_j^{k-1} + c_i, \quad i = 1, \ldots, n. \qquad (2.17)$$

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

Схема алгоритма аналогична схеме метода простой итерации (см. рис. 2.5), если $x0_j$ заменить на x_j и убрать строки $x0_i = 1$, $x0_i = x_i$.

Для иллюстрации решим систему (2.15) методом Зейделя. Разрешая ее, относительно (x_1, x_2, x_3), аналогично (2.16) получаем рекуррентную формулу (2.17) вида

$$x_1^k = (2 + x_2^{k-1} + x_3^{k-1})/4;$$
$$x_2^k = (4 - x_1^k + 2x_3^{k-1})/5; \qquad (2.18)$$
$$x_3^k = (6 - x_1^k - x_2^k)/4.$$

Зададим, как и в методе простой итерации, начальное условие $\overline{\mathbf{x}}^0 = (0,\ 0,\ 0)$, подставим в первое уравнение, получаем $x_1^1 = 0.5$. При вычислении x_2^1 это значение сразу используем: $x_2^1 = (4 - 0.5 + 0)/5 = 0.7$, аналогично $x_3^1 = (6 - 0.5 - 0.7)/4 = 1.2$ получаем $\overline{\mathbf{x}}^1 = (0.5;\ 0.7;\ 1.2)$. Для второй итерации:

$$x_1^2 = (2 + 0.5 + 1.2)/4 = 0.925;$$
$$x_2^2 = (4 - 0.925 + 2 \cdot 1.2)/5 = 1.085;$$
$$x_3^2 = (6 - 0.925 - 1.085)/4 = 0.998;$$

$$\mathbf{r}_{x^2} = (0.925;\ 1.085;\ 0.998).$$

Погрешность после двух итераций $\delta = \left\| \mathbf{r}_{x^2} - \mathbf{r}_{x^1} \right\| = 0.425$ меньше, чем в методе простой итерации.

2.4. 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

$$x_i^k = \omega x_i^k + (1-\omega) x_i^{k-1}. \qquad (2.19)$$

Здесь $0 < \omega \le 2$ – параметр релаксации.

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

Схема алгоритма на рис. 2.5 выполнена с использованием релаксации.

2.5. ВАРИАНТЫ ЗАДАНИЙ

Составить программу решения СЛАУ порядка n и решить систему линейных уравнений пятого порядка с трехдиагональной симметричной матрицей вида

$$\begin{vmatrix} q & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & q \end{vmatrix} \times \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{vmatrix} = \begin{vmatrix} 0 \\ d \\ d \\ d \\ 0 \end{vmatrix}.$$

Значения q и d заданы в табл. 2.1.

В вариантах, использующих прямые методы вычислить невязку (см. подразд. 2.1).

В вариантах, использующих итерационные методы, построить график зависимости количества итераций *it*, необходимых для достижения заданной точности, от параметра релаксации ω и установить, при каком значении ω число итераций минимально. Параметр релаксации менять от 0.2 до 2 с шагом 0.2.

Таблица 2.1

№ вар.	*Метод*	*d*	*q*	*e*
1	MG	–1	–4.25	

2	MI	1	–3.17	10^{-3}
3	MQ	–2	–3.23	
4	MZ	2	–2.57	10^{-4}
5	MP	–3	–4.67	
6	MI	3	–2.23	10^{-4}
7	MG	–4	–2.75	
8	MI	4	–2.25	10^{-4}
9	MQ	–5	–2.85	
10	MZ	5	–2.24	10^{-5}
11	MP	–6	–3.83	
12	MZ	6	–2.17	10^{-4}
13	MG	–7	–2.86	
14	MI	7	–3.14	10^{-3}
15	MQ	–8	–4.88	

2.6. 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.

ТЕМА 3. АППРОКСИМАЦИЯ ФУНКЦИЙ

3.1. 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;*** наибольшее распространение среди них 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:

$$\varphi(x, \vec{c}) = c_1\varphi_1(x) + ... + c_n\varphi_n(x) = \sum_{k=1}^{n} c_k\varphi_k(x). \qquad (3.1)$$

Здесь $\{\varphi_1(x), ..., \varphi_n(x)\}$ – 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 (3.1) 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:

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

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

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 (3.1) записывается в виде $y = \sum_{k=1}^{n} c_k\varphi_k(x) + d_k\psi_k(x)$.

3.2. 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 и j 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, $\varphi(x_i, \vec{c}) = f(x_i),\ i = 1...\ n$ (рис. 3.1), для чего решают полученную систему из n алгебраических в общем случае нелинейных уравнений.

В случае линейной аппроксимации (3.1) система для нахождения коэффициентов $\vec{c}$ линейна и имеет следующий вид:

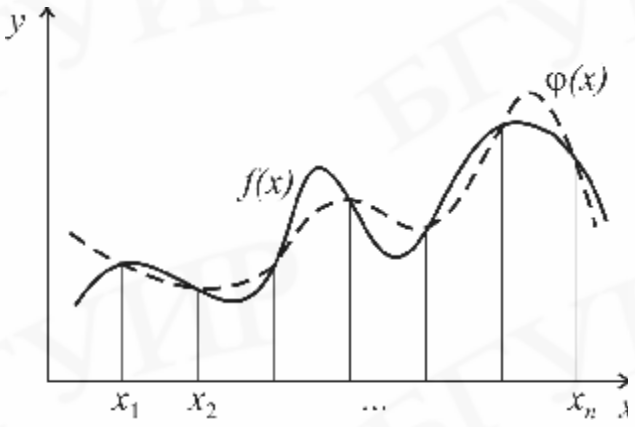


Рис. 3.1

$$\sum_{k=1}^{n} c_k \varphi_k(x_i) = y_i;$$

$$i = 1,\ 2,\ ...,\ n;\quad y_i = f(x_i). \tag{3.2}$$

Система базисных функций $\{\varphi_k(x)\}$, используемых для интерполяции, должна быть **чебышевской**, т.е. такой, чтобы определитель матрицы системы (3.2) 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

$$\varphi(x,\ \mathbf{r}\ c) = P_{n-1}(x) = c_1 + c_2 x + c_3 x^2 + ... + c_n x^{n-1} = \sum_{k=1}^{n} a_k x^{k-1}. \tag{3.3}$$

Матрица системы (3.2) в этом случае имеет вид

$$G = \begin{bmatrix} 1 & x_1 & .. & x_1^{n-1} \\ 1 & x_2 & .. & x_2^{n-1} \\ .. & .. & .. & .. \\ 1 & x_n & .. & x_n^{n-1} \end{bmatrix};\quad |G| = \prod_{n \ge k > m \ge 0} (x_k - x_m), \tag{3.4}$$

и ее определитель (это определитель Вандермонда) отличен от нуля, если точки x_i разные. Поэтому задача (3.2) 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

$$\varepsilon = \| f(x) - P_{n-1}(x) \|_C \le \sqrt{\frac{2}{(n-1)\pi}} \left\| \frac{d^n f(x)}{dx^n} \right\|_C \left(\frac{h}{2}\right)^n,\quad h = \max_i |x_i - x_{i-1}|. \tag{3.5}$$

Из (3.5) следует, что при $h \to 0$ порядок погрешности p (см. подразд. 1.4) 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 ($n \le 6$), в связи с тем что с ростом n резко возрастает погрешность вычисления самого многочлена из-за погрешностей округления.

3.3. КАКИЕ БЫВАЮТ МНОГОЧЛЕНЫ И СПОСОБЫ ИНТЕРПОЛЯЦИИ?

Один и тот же многочлен можно записать по-разному, например $P_1(x) = 1 - 2x + x^2 = (x-1)^2$. Поэтому в зависимости от решаемых задач применяют различные виды представления интерполяционного многочлена и соответственно способы интерполяции.

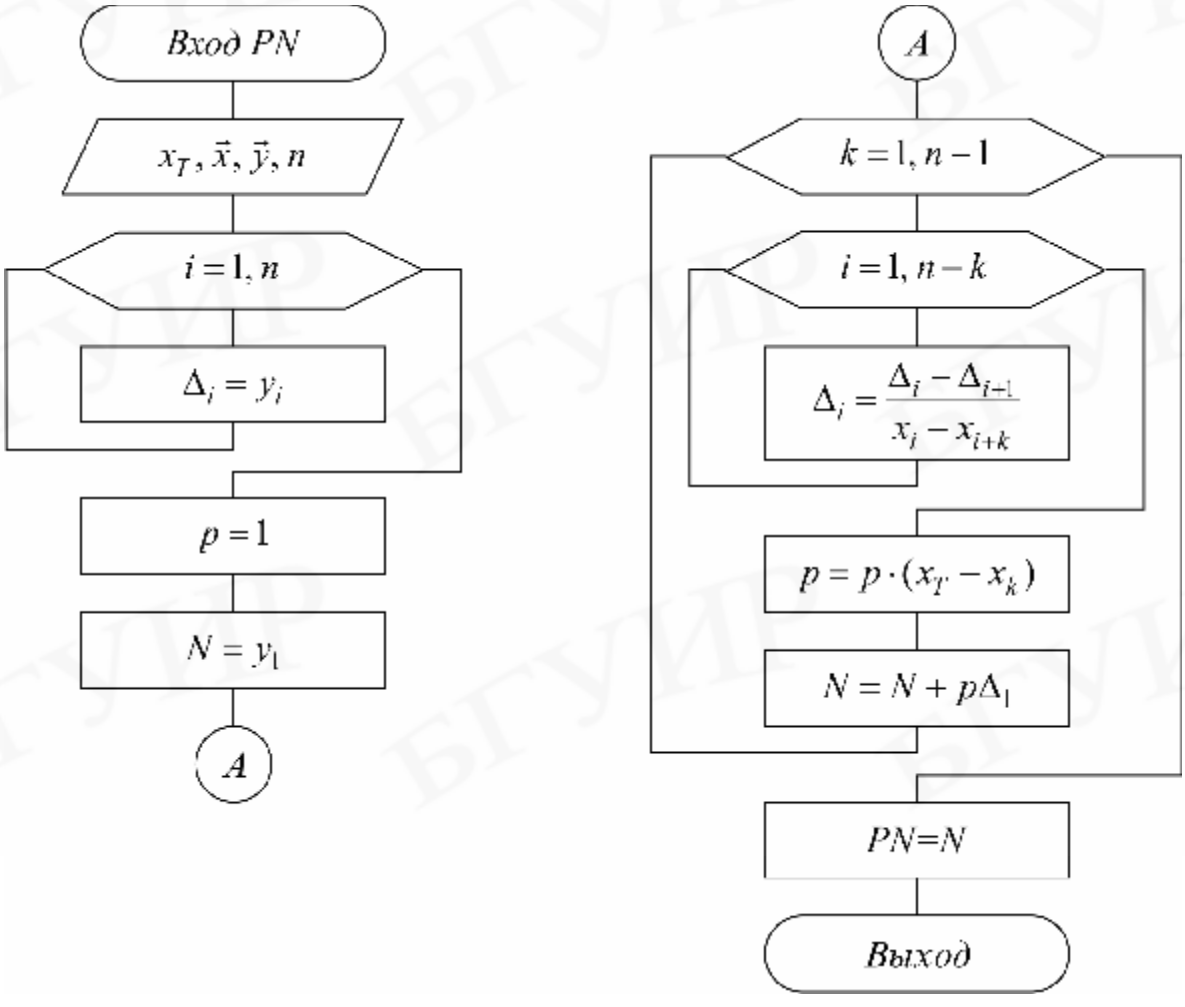
Наряду с общим представлением (3.3) наиболее часто в приложениях используют интерполяционные многочлены в форме Лагранжа и Ньютона. Их особенность в том, что не надо находить параметры $\vec{c}$, так как многочлены в этой форме прямо записаны через значения таблицы $\{(x_i,\ y_i)\ \ i = 1\ \ldots\ n\}$.

Интерполяционный многочлен Ньютона (PN)

$$N_{n-1}(x_T) = y_1 + \sum_{k=1}^{n-1} (x_T - x_1)(x_T - x_2)\ldots(x_T - x_k)\Delta_1^k .$$

(3.6)

Здесь x_T – текущая точка, в которой надо вычислить значение многочлена,



Δ_1^k – разделенные разности порядка *k*, которые вычисляются по следующим рекуррентным формулам:

$$\Delta_i^1 = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}, \quad i = 1...n;$$

$$\Delta_i^2 = \frac{\Delta_i^1 - \Delta_{i+1}^1}{x_i - x_{i+2}}, \quad i = 1...(n-2);$$

$$\Delta_i^k = \frac{\Delta_i^{k-1} - \Delta_{i+1}^{k-1}}{x_i - x_{i+k}}, \; i = 1...(n-k).$$

Схема расчета многочлена Ньютона представлена на рис. 3.2.

Линейная (PNL) и квадратичная(PNS) интерполяция

Вычисления по интерполяционной формуле (3.6) для $n > 3$ используют

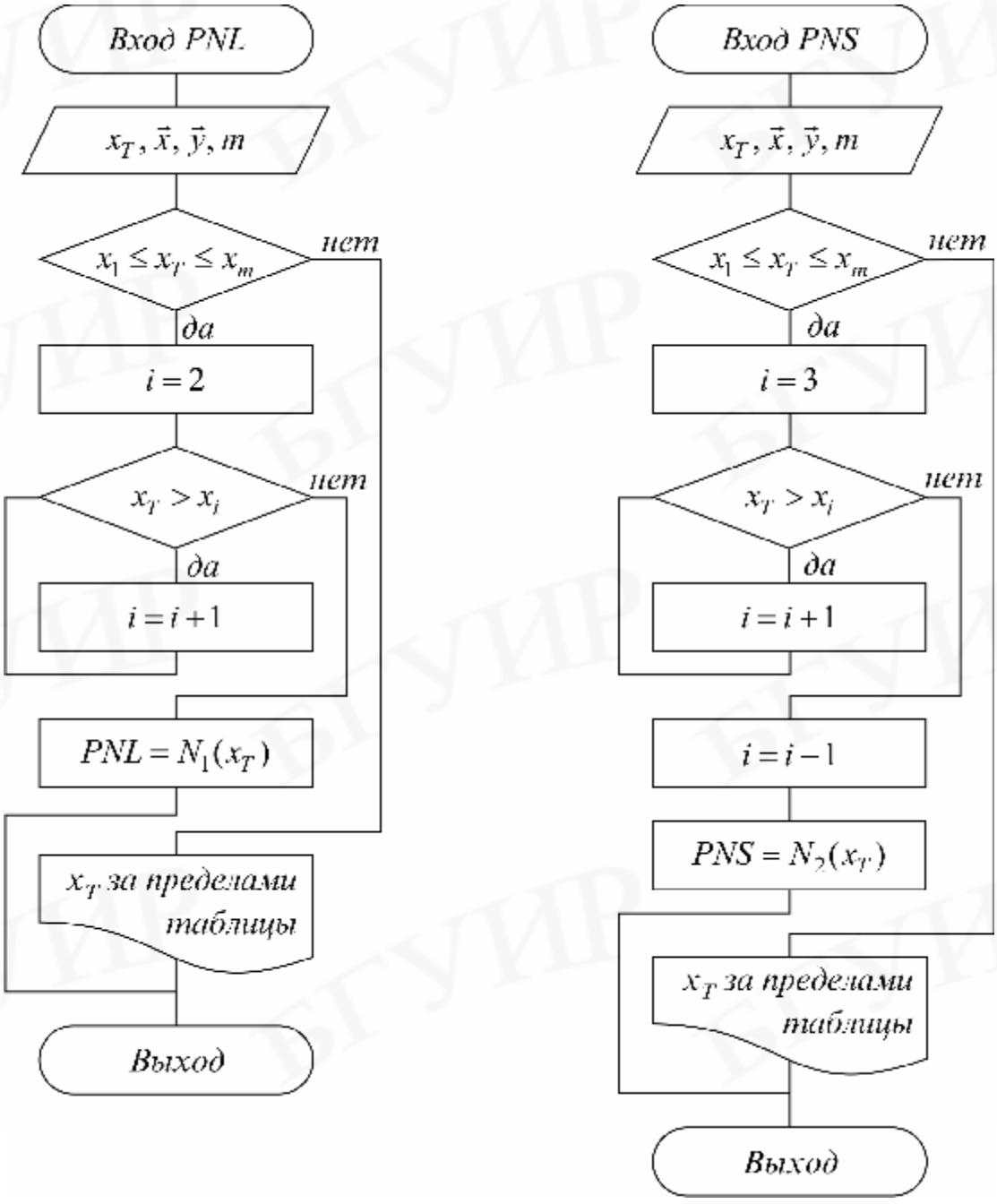


Рис. 3.3

редко. Обычно 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 в точке x находят в таблице ближайший к этой точке i-узел из общей таблицы, строят интерполяционный многочлен Ньютона первой или второй степени по формулам

$$N_1(x_T) = y_{i-1} + (x_T - x_{i-1})\frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \quad x_{i-1} \le x_T \le x_i;$$

$$N_2(x_T) = N_1(x_T) + (x_T - x_{i-1})(x_T - x_i)\frac{\left(\frac{y_{i-1} - y_i}{x_{i-1} - x_i}\right) - \left(\frac{y_i - y_{i+1}}{x_i - x_{i+1}}\right)}{x_{i-1} - x_{i+1}}; \quad x_{i-1} \le x_T \le x_{i+1} \tag{3.7}$$

и за значение $f(x)$ принимают $N_1(x)$ (*линейная интерполяция*) или $N_2(x)$ (*квадратичная интерполяция*). Схема расчета для линейной и квадратичной интерполяции приведена на рис. 3.3.

Интерполяционный многочлен Лагранжа (PL)

$$L_{n-1}(x_T) = \sum_{k=1}^{n} y_k e_k(x_T); \quad e_k(x_T) = \prod_{\substack{i=1 \\ i \ne k}}^{n} \frac{x_T - x_i}{(x_k - x_{i)}} \ . \tag{3.8}$$

Многочлены $e_k^{n-1}(x_j)$ выбраны так, что во всех узлах, кроме k-го, они обращаются в ноль, в k-м узле они равны единице:

$$e_k^{n-1}(x_j) = \begin{cases} 1, & при \quad j = k; \\ 0, & при \quad j \ne k. \end{cases}$$

Поэтому из выражения (3.8) видно, что $L_{n-1}(x_i) = y_i$.

Схема расчета интерполяционного многочлена Лагранжа представлена на рис. 3.4.

Интерполяция общего вида, использующая прямое решение системы (3.2) методом Гаусса (POG)

Следует отметить, что ввиду громоздкости многочлены Ньютона и Лагранжа уступают по эффективности расчета многочлену общего вида (3.3), если предварительно найдены коэффициенты $\vec{c}$.

Поэтому когда требуется производить много вычислений многочлена, построенного по одной таблице, оказывается выгодно вначале один раз найти коэффициенты $\vec{c}$ и затем использовать формулу (3.3). Коэффициенты $\vec{c}$ находят прямым решением системы (3.2) с матрицей (3.4), затем вычисляют его значения по экономно программируемой формуле (алгоритм Горнера)

$$P_{n-1}(x) = c_1 + x_T(c_2 + \ldots \ x_T(c_{n-2} + x_T(c_{n-1} + x_T c_n) \ \ldots).$$

(3.9)

Схема расчета интерполяционного многочлена общего вида по формуле (3.9) с прямым решением системы (3.2) приведена на рис. 3.5.

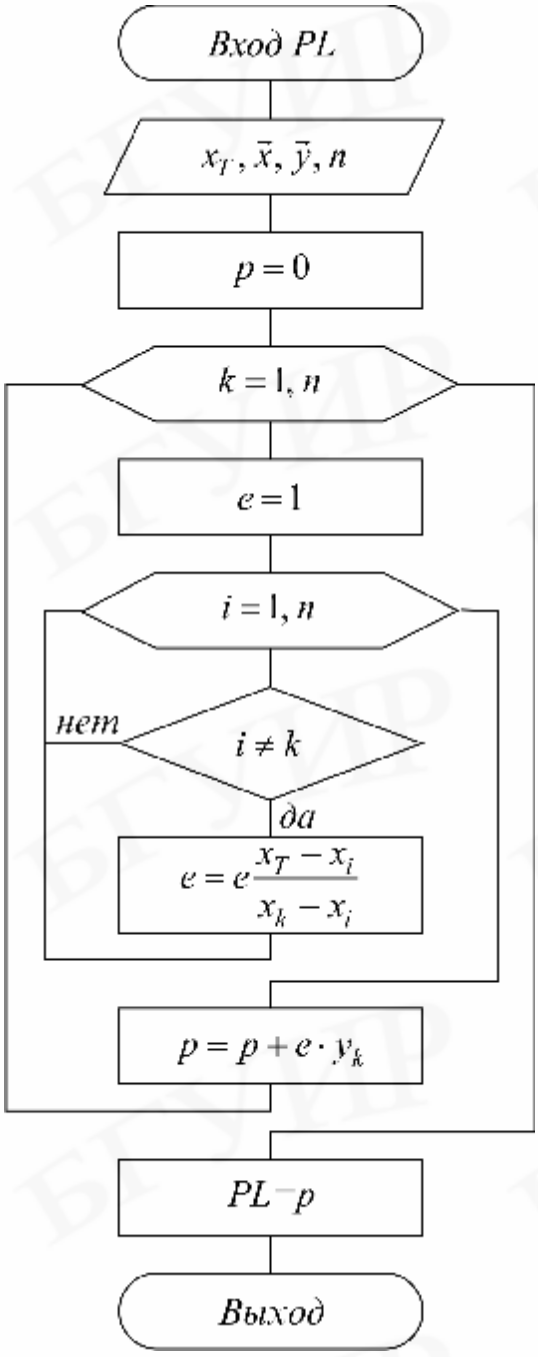


Рис. 3.4

Интерполяция общего вида, использующая расчет коэффициентов многочлена (3.3) через многочлен Лагранжа (POL)

Находить коэффициенты $\vec{c}$ многочлена (3.3) можно, не решая прямо систему (3.2), а используя разложение коэффициентов Лагранжа (3.8):

$$e_k^{n-1}(x) = a_{k,1}^{n-1} + a_{k,2}^{n-1} \cdot x + \ \ldots \ + a_{k,n}^{n-1} \cdot x^{n-1}, \qquad c_i = \sum_{k=1}^{n} y_k \cdot a_{k,i}^{n-1}. \tag{3.10}$$

Рекуррентные формулы для нахождения коэффициентов $a_{k,j}^{n-1}$:

$$a_{k,1}^{m} = a_{k,1}^{m-1} \frac{-x_m}{x_k - x_m}; \quad a_{k,m+1}^{m} = a_{k,m}^{m-1} \frac{1}{x_k - x_m}; \tag{3.11}$$

$$a_{k,j}^{m} = \frac{a_{k,\,j-1}^{m-1} - a_{k,j}^{m-1} \cdot x_m}{x_k - x_m}; \quad 1 < j < m; \qquad m = 2,\ \ldots,\ n-1$$

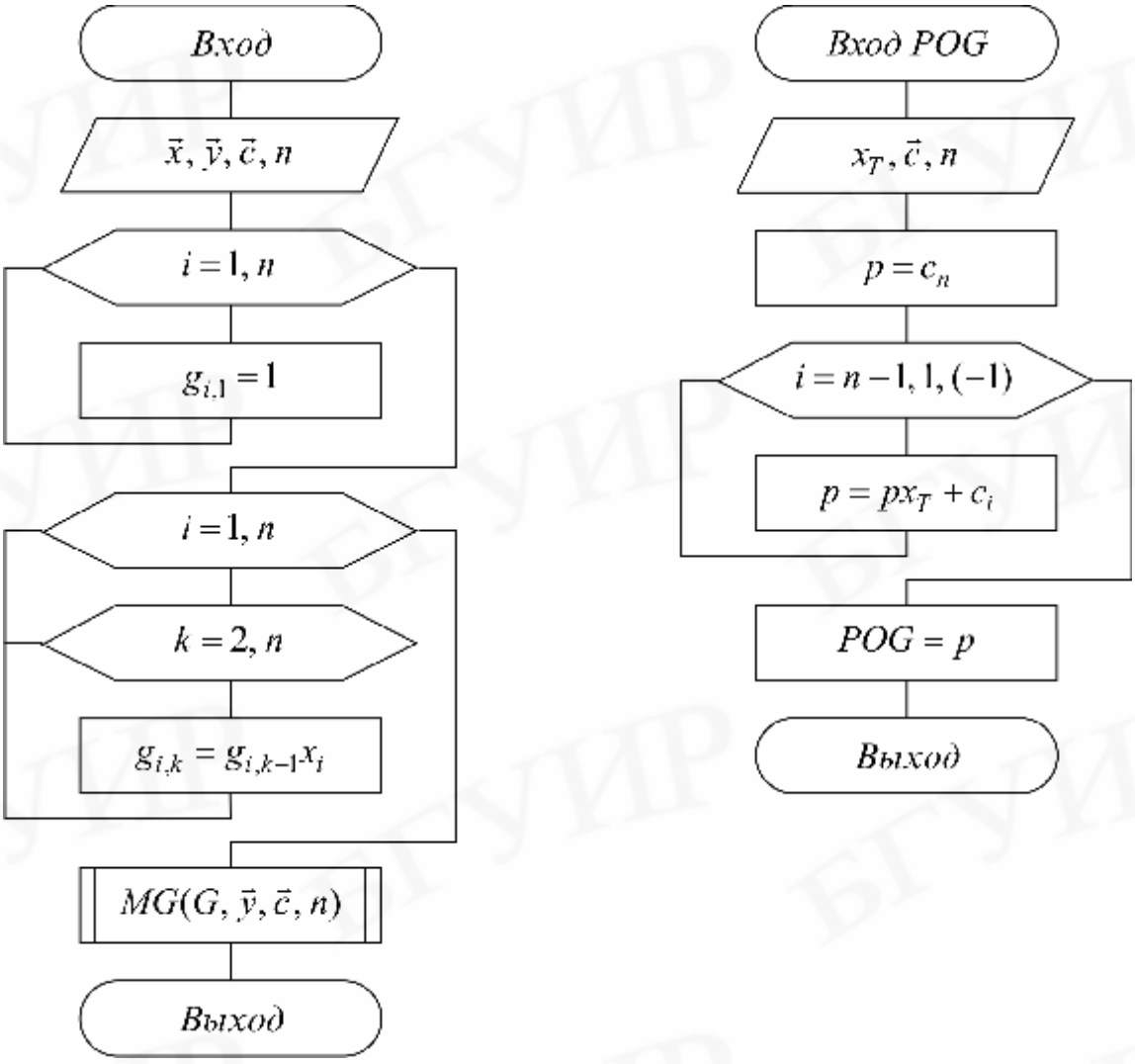
получаются из вида многочленов $e_k^{n-1}(x)$, если использовать очевидное представление

$$e_k^{m}(x) = e_k^{m-1} \cdot \frac{x - x_{n-m}}{x_k - x_{n+m}}; \quad e_k^0 = a_{k,1}^0 = 1; \quad m = 1,\ 2,\ \ldots,\ n; \quad m \neq k.$$

Схема алгоритма вычисления коэффициентов многочлена общего вида по формулам (3.10), (3.11) представлена на рис. 3.6.

3.4. ЧТО ТАКОЕ 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 $\varphi(x,\vec{c})$ в пространстве $L_2[a\,b]$ (см. подразд. 1.3), т.е. из условия



а **б**

Рис. 3.5

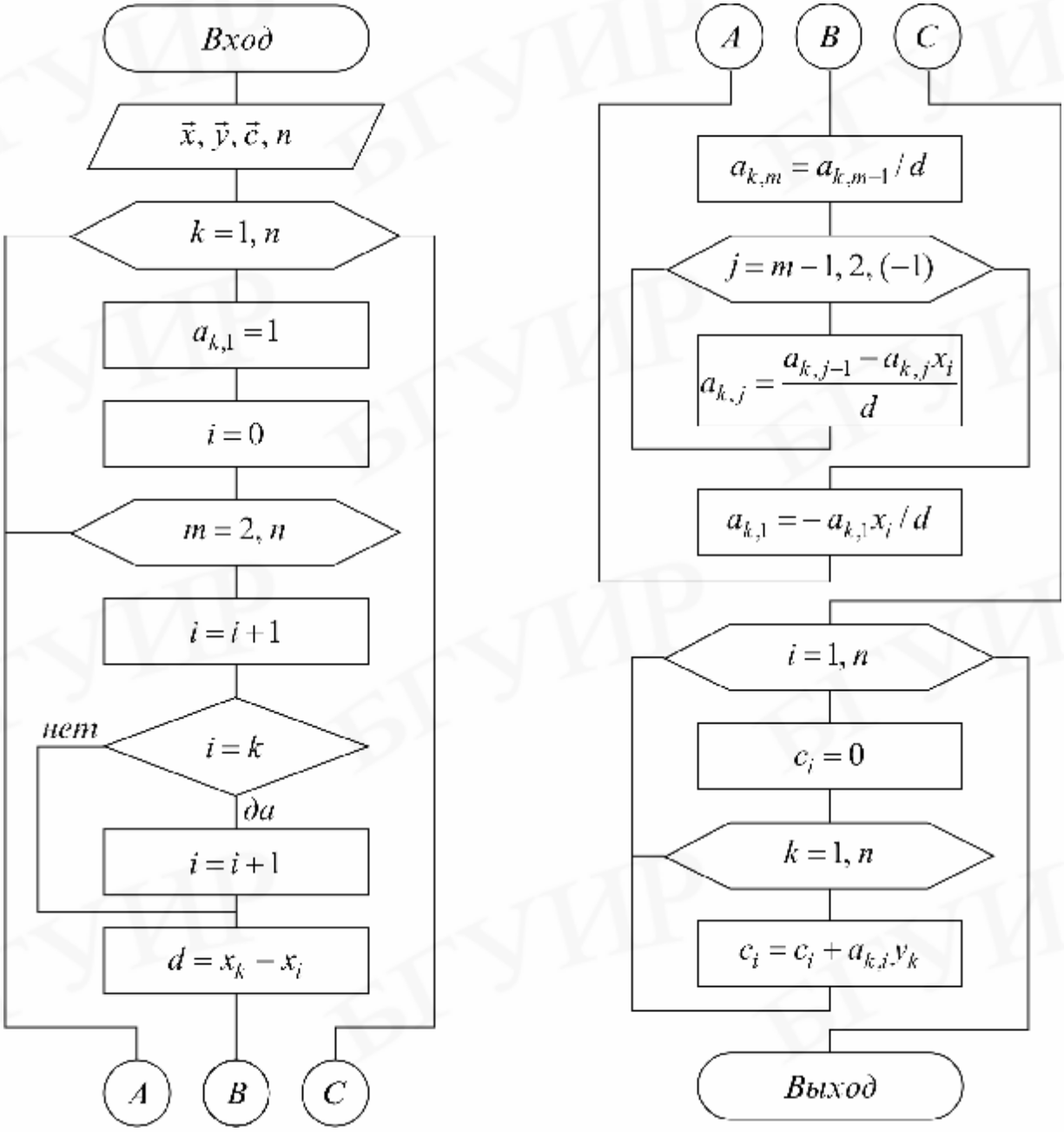


Рис. 3.6

$$\min_{c_1,\ldots,c_n} \left\| f(x) - \varphi\left(x, \vec{c}\right) \right\|_{L_2}. \tag{3.12}$$

В случае линейной аппроксимации (3.1) задача (3.12) сводится к решению СЛАУ для нахождения необходимых коэффициентов $\vec{c}$:

$$\sum_{k=1}^{n} \left(\varphi_i \varphi_k\right)_{L_2} \cdot c_k = \left(f, \varphi_i\right)_{L_2}; \quad i = 1 \ldots n. \tag{3.13}$$

Здесь $\left(\varphi_i \varphi_k\right)_{L_2}$, $\left(f, \varphi_i\right)_{L_2}$ – скалярные произведения в L_2.

Матрица системы (3.13) 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 (3.13) диагональная и параметры $\vec{c}$ находятся по формуле

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

В этом случае представление (3.1) называется *обобщенным рядом Фурье*, а c_k называются *коэффициентами Фурье*.

Метод наименьших квадратов (МНК)

МНК является частным случаем среднеквадратичной аппроксимации. При использовании МНК в области значений x, представляющей некоторый интервал $[a, b]$, где функции f и j должны быть близки, выбирают систему различных точек (узлов) $x_1, ..., x_m$, число которых обычно больше, чем количество искомых параметров $c_1, ..., c_n$, $m \geq n$.

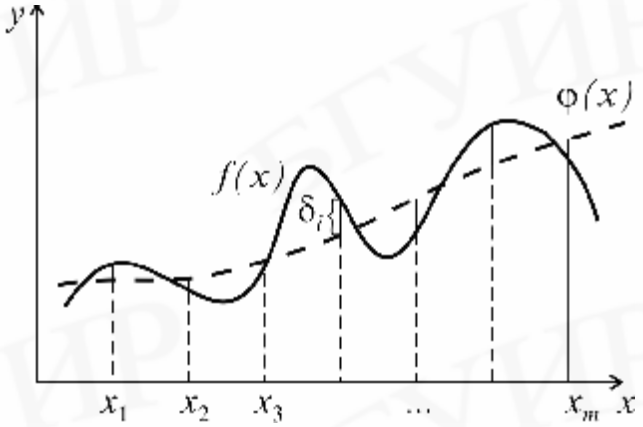


Рис. 3.7

Далее, потребовав, чтобы сумма квадратов невязок во всех узлах была минимальна (рис. 3.7):

$$\min_{\vec{c}} \sum_{i=1}^{m} [y_i - \varphi(x_i, \vec{c})]^2 = \min_{\vec{c}} \sum_{i=1}^{m} \delta_i^2 = \min_{\vec{c}} \delta(\vec{c}),$$

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 (3.1), составляя условия минимума суммы квадратов невязок во всех точках $\delta(\vec{c})$ (в точке минимума все частные производные должны быть равны нулю):

$$\frac{\partial \delta(c_1, c_2, ..., c_n)}{\partial c_i} = 0, \ i = 1, \ n, \tag{3.14}$$

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

$$\sum_{k=1}^{n}(\vec{\varphi}_i\ \vec{\varphi}_k)c_k = (\vec{y},\ \vec{\varphi}_i),\ \ i=1,\ n \quad \text{или}$$

$$G\vec{c} = \vec{b}. \qquad (3.15)$$

Здесь $\vec{\varphi}_i = (\varphi_i(x_1),\ \varphi_i(x_2),\ \ldots,\ \varphi_i(x_m))$, $\vec{y} = (y_1,\ \ldots,\ y_m)$ – векторы-таблицы функций. Элементы матрицы G и вектора $\vec{b}$ в (3.15) определяются выражениями

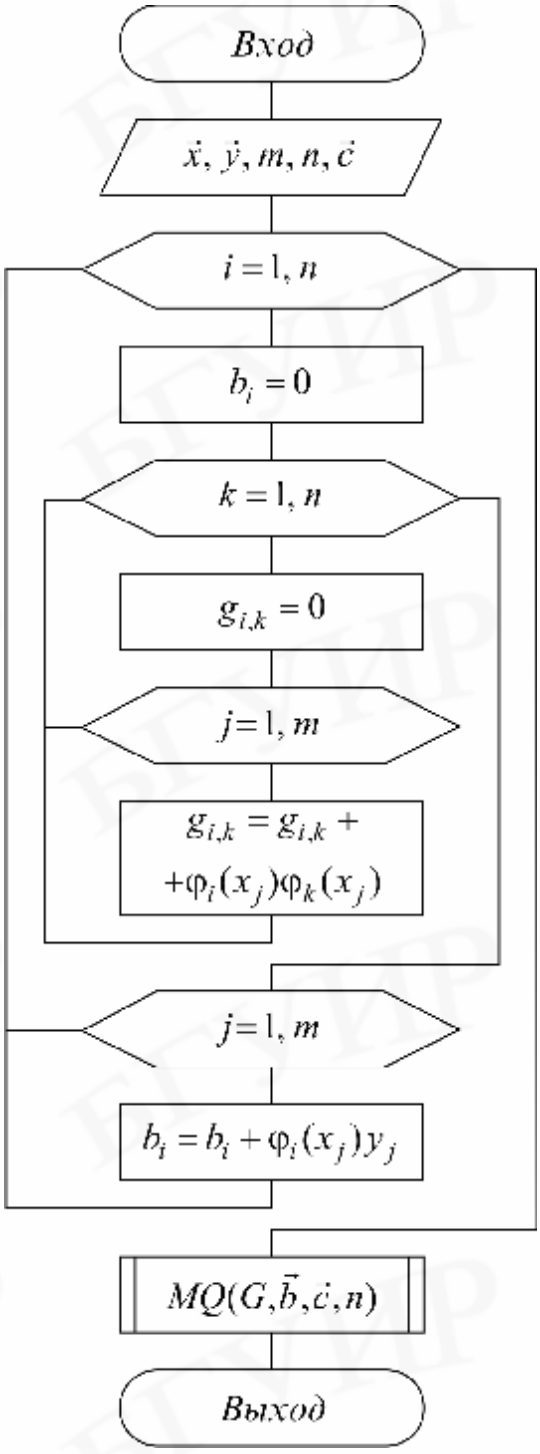


Рис. 3.8

$$\left.\begin{aligned} g_{i,k} &= (\overset{\mathbf{r}}{\varphi}_i \ \overset{\mathbf{r}}{\varphi}_k) = \sum_{j=1}^{m} \varphi_i(x_j)\varphi_k(x_j); \\ b_i &= (\overset{\mathbf{r}}{y}, \ \overset{\mathbf{r}}{\varphi}_i) = \sum_{j=1}^{m} y_j \varphi_i(x_j). \end{aligned}\right\}$$ – скалярные произведения векторов.

Система (3.15) имеет симметричную матрицу G и решается методом квадратного корня.

При аппроксимации по МНК обычно применяют такие функции $\{\varphi_i(x)\}$, которые используют особенности решаемой задачи и удобны для последующей обработки.

Схема расчета коэффициентов многочлена вида (3.3) по методу наименьших квадратов представлена на рис. 3.8.

Приведем пример аппроксимации по МНК. Предположим, что известна таблица значений *f(x)*: {x_1=1, y_1=0.5, x_2=2, y_2=1.2, x_3=3, y_3=0.8}, т.е. m=3. Требуется найти параметры аппроксимирующей функции $\varphi(x, c_1, c_2)$ вида $\varphi = c_1 + c_2 \cdot x$, ($n$=2).

Составляем сумму квадратов невязок

$$\delta(\overset{\mathbf{r}}{c}) = \sum_{i=1}^{3} (y_i - c_1 - c_2 \cdot x_i)^2 = (0{,}5 - c_1 - c_2 \cdot 1)^2 + (1.2 - c_1 - c_2 \cdot 2)^2 + (0.8 - c_1 - c_2 \cdot 3)^2.$$

Условия минимума (3.14):

$$\begin{cases} \dfrac{\partial \delta}{\partial c_1} = -2 \cdot (0.5 - c_1 - 1 \cdot c_2) - 2 \cdot (1.2 - c_1 - 2 \cdot c_2) - 2 \cdot (0.8 - c_1 - 3 \cdot c_2) = 0; \\ \dfrac{\partial \delta}{\partial c_2} = -2 \cdot (0.5 - c_1 - 1 \cdot c_2) - 4 \cdot (1.2 - c_1 - 2 \cdot c_2) - 6 \cdot (0.8 - c_1 - 3 \cdot c_2) = 0. \end{cases}$$

Приводя подобные члены, получим окончательно систему двух уравнений с симметричной матрицей относительно неизвестных c_1 и c_2:

$$\begin{cases} 3c_1 + 6c_2 = 2.5; \\ 6c_1 + 14c_2 = 5.3. \end{cases}$$

Решая ее, находим $c_1 \approx 0.53$, $c_2 = 0.15$.

На рис. 3.9 приведена таблица функции *f(x)* и полученная по МНК функция $j(x)$.

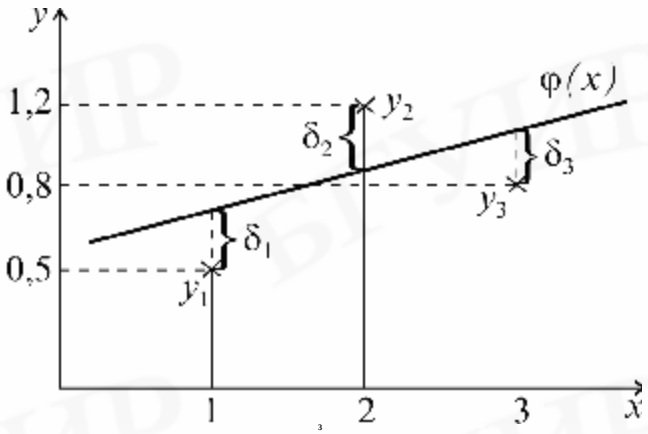


Рис. 3.9

Порядок расчета по МНК следующий. Вначале по исходной таблице формируется матрица G и рассчитываются коэффициенты (см. рис. 3.8) (в качестве $j_k(x)$ здесь берется функция x^{k-1}). Затем, используя полученные

коэффициенты, рассчитывается значение функции в искомой точке (см. рис. 3.5, б).

3.5. ВАРИАНТЫ ЗАДАНИЙ

Во всех вариантах (табл. 3.1.) требуется аппроксимировать заданную исходную функцию *f(x)* многочленом на интервале [*a*, *b*]. Задано количество неизвестных параметров *n*, вид аппроксимации и *m* – количество точек, 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 $x_j = a + (j-1)(b-a)/20;\ \ j=1, 21$, построить графики и проанализировать качество полученной аппроксимации.

Таблица 3.1

N вар.	*Функция f(x)*	*a*	*b*	*m*	*n*	*Вид аппроксимации*
1	$4x - 7\sin(x)$	–2	3	11	3	МНК
2	$x^2 - 10\sin^2(x)$	0	3	4	4	Ньютона PN
3	$\ln(x) - 5\cos(x)$	1	8	4	4	Лагранжа PL
4	$e^x / x^3 - \sin^3(x)$	4	7	4	4	Общего вида POG
5	$\sqrt{x} - \cos^2(x)$	5	8	4	4	Общего вида POL
6	$\ln(x) - 5\sin^2(x)$	3	6	11	2	Линейная PNL
7	$x - 5\sin^2(x)$	1	4	11	4	МНК
8	$\sin^2(x) - x/5$	0	4	11	3	Квадратичная PNS
9	$x^3 + 10x^2$	–8	2	5	5	Ньютона PN
10	$x^3 - 5x^2$	–2	5	5	5	Лагранжа PL
11	$x^3 + 6x^2 - 0.02e^x$	–5	3	5	5	Общего вида POG
12	$x^2 + 5\cos(x)$	–1	4	5	5	Общего вида POL
13	$\sin^2(x) - 3\cos(x)$	1	7	11	5	МНК
14	$x^3 - 50\cos(x)$	–2	5	11	3	Квадратичная PNS
15	$0.1x^3 + x^2 - 10\sin(x)$	–4	2	11	2	Линейная PNL

3.6. 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?

6. Выведите СЛАУ относительно коэффициентов функции $\varphi = c_1 + c_2 x$ по МНК для таблицы $\{(x_1 y_1), (x_2 y_2), (x_3 y_3)\}$.

ТЕМА 4. ВЫЧИСЛЕНИЕ ПРОИЗВОДНЫХ И ИНТЕГРАЛОВ

4.1. КАК АППРОКСИМИРУЮТ ОПЕРАТОРЫ?

Задачи численного дифференцирования и интегрирования являются частным случаем общей *задачи аппроксимации операторов*, состоящей в том, что заданный на некотором множестве функций оператор $U = \Phi[f(x)]$ требуется заменить более простым, удобным для последующего использования, оператором $\tilde{U} = \Lambda[f(x)]$, близким в некотором смысле к исходному.

Аппроксимацию операторов обычно осуществляют следующим образом: функцию $f(x)$ аппроксимируют такой функцией $\varphi(x, \vec{c})$, от которой оператор Ф легко вычисляется, после чего полагают $\tilde{U} = \Lambda[f(x)] = F[\varphi(x, \vec{c})]$.

При аппроксимации операторов численного дифференцирования и интегрирования наибольшее распространение ввиду своей простоты нашли интерполяционные формулы Ньютона. В случае когда функция *f(x)* задана в виде таблицы, полученной с некоторой погрешностью, используют среднеквадратичную аппроксимацию. Ниже рассматривается аппроксимация *f(x)* интерполяционным многочленом.

4.2. 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 $y_i = f(x_i)$ в узлах шаблона, строится интерполяционный многочлен Ньютона (3.6) 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, т.к. в этом случае 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

$$\varepsilon = \max_{x_1 < x < x_n} \left| \frac{d^m f}{dx^m} - \Lambda_m^n[f] \right| \leq \frac{\max_x \left| f^{(m)}(x) \right|}{(n-m)} \max_i \left| x - x_i \right| \leq C h^{n-m}, \qquad (4.1)$$

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

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 $n\text{-}m \geq 1$, и для повышения точности можно либо увеличивать n, либо уменьшать h (последнее более предпочтительно).

Приведем несколько важных формул для равномерного шаблона, получаемых с использованием (3.7):

$$\frac{df}{dx} \approx \frac{dP_1}{dx} = \Lambda_1^2[f(x)] = \frac{y_2 - y_1}{h}; \quad x_1 \leq x \leq x_2. \tag{4.2}$$

Простейшая формула (4.2) имеет второй порядок погрешности в центре интервала и первый по краям:

$$\frac{df}{dx} \approx \frac{dP_2}{dx} = \Lambda_1^3[f(x)] = \frac{y_2 - y_1}{h} + (2x - x_1 - x_2)\frac{y_1 - 2y_2 + y_3}{2h^2}. \tag{4.3}$$

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 (4.3) получается три важные формулы второго порядка точности:

$$\frac{df(x_2)}{dx} = \Lambda_1^3[f(x_2)] = \frac{y_3 - y_1}{2h}; \tag{4.4}$$

$$\frac{df(x_1)}{dx} = \Lambda_1^3[f(x_1)] = -\frac{3y_1 - 4y_2 + y_3}{2h}; \tag{4.5}$$

$$\frac{df(x_3)}{dx} = \Lambda_1^3[f(x_3)] = \frac{y_1 - 4y_2 + 3y_3}{2h}. \tag{4.6}$$

Для вычисления второй производной часто используют следующую простейшую формулу:

$$\frac{d^2f}{dx^2} \approx \frac{d^2P_2}{dx^2} = \Lambda_2[f(x)] = \frac{y_1 - 2y_2 + y_3}{h^2}; \quad x_1 \leq x \leq x_3, \tag{4.7}$$

которая имеет второй порядок погрешности в центральной точке x_2.

4.3. ФОРМУЛЫ ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ

Формулы для вычисления интеграла $U = \int_a^b f(x)dx$ получают следующим образом. Область интегрирования $[a, b]$ разбивают на малые отрезки $\{a = x_1 < x_2 < \ldots < x_{m+1} = b,\ h_i = x_{i+1} - x_i\}$, в общем случае разной длины. Значение интеграла по всей области равно сумме интегралов на отрезках $\int_a^b f(x)dx = \sum_{i=1}^{m} \int_{x_i}^{x_{i+1}} f(x)dx$. Выбирают на каждом отрезке $[x_i, x_{i+1}]$ 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.***

Приведем наиболее часто используемые квадратурные формулы для равных отрезков длиной $h=(b-a)/m,\ x_i=a+(i-1)\cdot h,\ i=1\ \ldots\ m.$

Формула средних

Формула средних получается, если на каждом i-м отрезке взять один центральный узел $x_{i+1/2}=(x_i+x_{i+1})/2$, соответствующий середине отрезка. Функция на каждом отрезке аппроксимируется многочленом нулевой степени (константой) $P_0(x)=y_{i+1/2}=f(x_{i+1/2})$. Заменяя площадь криволинейной фигуры площадью прямоугольника высотой $y_{i+1/2}$ и основанием h, получим (рис. 4.1):

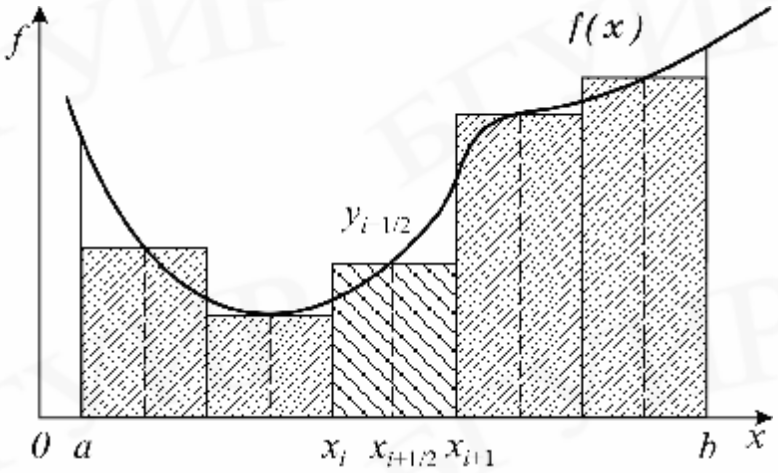


Рис. 4.1

$$\int_a^b f(x)dx \approx \sum_{i=1}^{m}\int_{x_i}^{x_{i+1}} P_0(x)dx = h\sum_{i=1}^{m} y_{i+1/2} = \Sigma_{\text{ср}} f\ . \tag{4.8}$$

Погрешность формулы средних имеет второй порядок по h:

$$\varepsilon_{\text{ср}} = \max\left|\int_a^b f dx - \Sigma_{\text{ср}} f\right| \le \frac{h^2}{24}\int_a^b f''(x)dx. \tag{4.9}$$

Формула трапеций

Формула трапеций получается при аппроксимации функции $f(x)$ на каждом отрезке $[x_i,\ x_{i+1}]$ интерполяционным многочленом первого порядка, т.е. прямой, проходящей через точки (x_i, y_i), (x_{i+1}, y_{i+1}). Площадь криволинейной фигуры заменяется площадью трапеции с основаниями $y_i,\ y_{i+1}$ и высотой h (рис. 4.2):

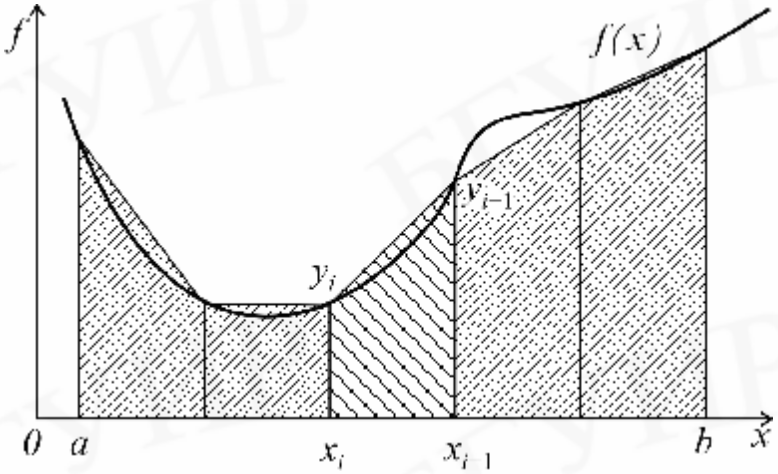


Рис. 4.2

$$\int_a^b f(x)dx \approx \sum_{i=1}^{m} \int_{x_i}^{x_{i+1}} P_1(x)dx = h\sum_{i=1}^{m} \frac{y_i + y_{i+1}}{2} = h\left[\frac{y_1 + y_{m+1}}{2} + \sum_{i=2}^{m} y_i\right] = \Sigma_{\text{тр}} f\,. \qquad (4.10)$$

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

$$\varepsilon_{\text{тр}} = \max\left|\int_a^b f dx - \Sigma_{\text{тр}}\, f\right| \le \frac{h^2}{12}\int_a^b f''(x)dx\,. \qquad (4.11)$$

Формула Симпсона

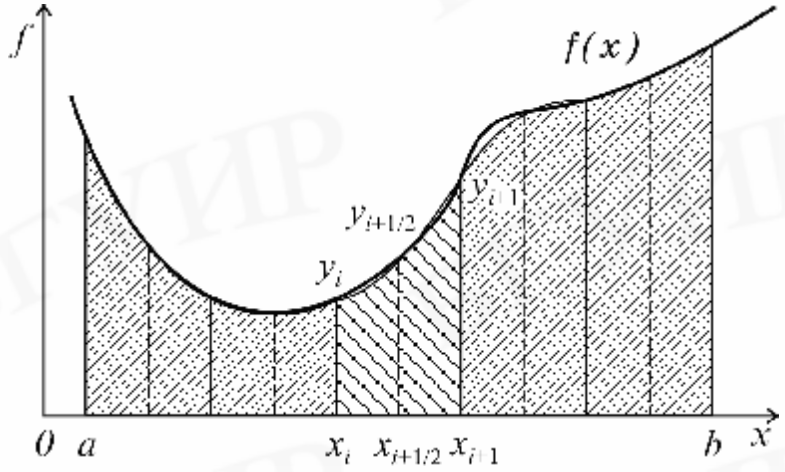


Рис. 4.3

Формула Симпсона получается при аппроксимации функции *f(x)* на каждом отрезке $[x_i, x_{i+1}]$ интерполяционным многочленом второго порядка (параболой) с узлами x_i, $x_{i+1/2}$, x_{i+1}. После интегрирования параболы получаем (рис. 4.3)

$$\int_a^b f(x)dx \approx \sum_{i=1}^{m} \int_{x_i}^{x_{i+1}} P_2(x)dx = \frac{h}{6}\sum_{i=1}^{m} (y_i + 4y_{i+0.5} + y_{i+1}) = \Sigma_{\text{сим}} f\,. \qquad (4.12)$$

После приведения подобных членов формула (4.12) приобретает удобный для программирования вид:

$$\sum\nolimits_{\text{сим}} f = \frac{h}{3}\cdot\left[\frac{y_1 + 4y_{1+0.5} + y_{m+1}}{2} + \sum_{i=2}^{m}\left(2y_{i+0.5} + y_i\right)\right].$$

Погрешность формулы Симпсона имеет четвертый порядок по *h*:

$$\varepsilon = \max\left|\int_a^b f dx - \Sigma_{\text{сим}}\, f\right| \le \frac{h^4}{2880}\int_a^b f^{(4)}(x)dx. \qquad (4.13)$$

Схема с автоматическим выбором шага по заданной точности

Анализ формул (4.8),(4.10),(4.12) показывает, что точное значение интеграла находится между значениями $\Sigma_{\text{ср}}$ и $\Sigma_{\text{тр}}$, при этом имеет место соотношение

$$\Sigma_{\text{сим}} = (\Sigma_{\text{тр}} + 2\Sigma_{\text{ср}})/3. \qquad (4.14)$$

Это соотношение часто используется для контроля погрешности вычислений. Расчет начинается с $m = 2$ и производится по двум методам, в результате

получают $\Sigma_{ср}$, $\Sigma_{тр}$. Если $|\Sigma_{ср} - \Sigma_{тр}| \geq \delta$ (δ – заданная погрешность), то шаг h уменьшают вдвое *($m = m \cdot 2$)* и расчет повторяют. Если точность достигается, то окончательное значение интеграла получается по формуле (4.14). При существенном уменьшении шага h начинают сказываться ошибки округления, поэтому шаг должен быть ограничен снизу некоторой величиной, зависящей от разрядной сетки ЭВМ *($m \leq n$).*

Схема алгоритма с автоматическим выбором шага представлена на рис. 4.4.

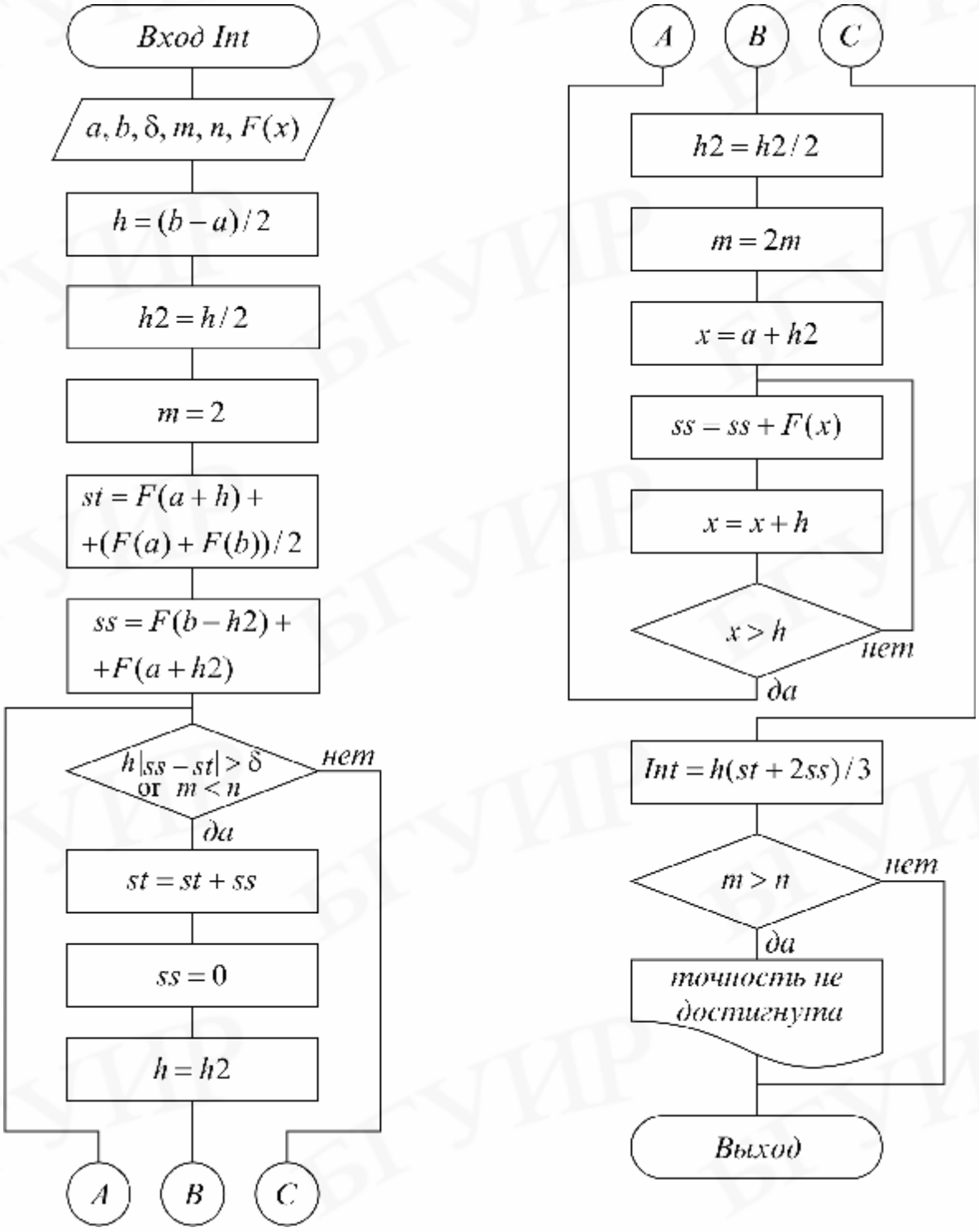


Рис. 4.4

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*** порядка n состоит в таком расположении n узлов интерполяционного многочлена на отрезке $[x_i, x_{i+1}]$, при котором достигается минимум погрешности квадратурной формулы. Детальный анализ показывает, что узлами, удовлетворяющими такому условию, являются нули ортогонального многочлена Лежандра степени n (см. подразд. 3.1). Так, для n = 1 один узел должен быть выбран в центре. Следовательно, метод средних является методом Гаусса порядка 1.

Для n=2 узлы должны быть выбраны следующим образом:

$$x_i^{1,2} = x_{i+1/2} \pm \frac{h}{2} \cdot 0.5773502692,$$

и соответствующая формула Гаусса 2-го порядка имеет вид

$$\int_a^b f(x)dx \approx \frac{h}{2}\sum_{i=1}^{n}[f(x_i^1)+f(x_i^2)]. \qquad (4.15)$$

Порядок погрешности этой формулы при $h \to 0$ равен p = 4, т.е. такой же, как у метода Симпсона, хотя используется только 2 узла!

Для n = 3 узлы выбираются следующим образом:

$$x_i^0 = x_{i+1/2}, \quad x_i^{1,2} = x_i^0 \pm \frac{h}{2} \cdot 0.7745966692,$$

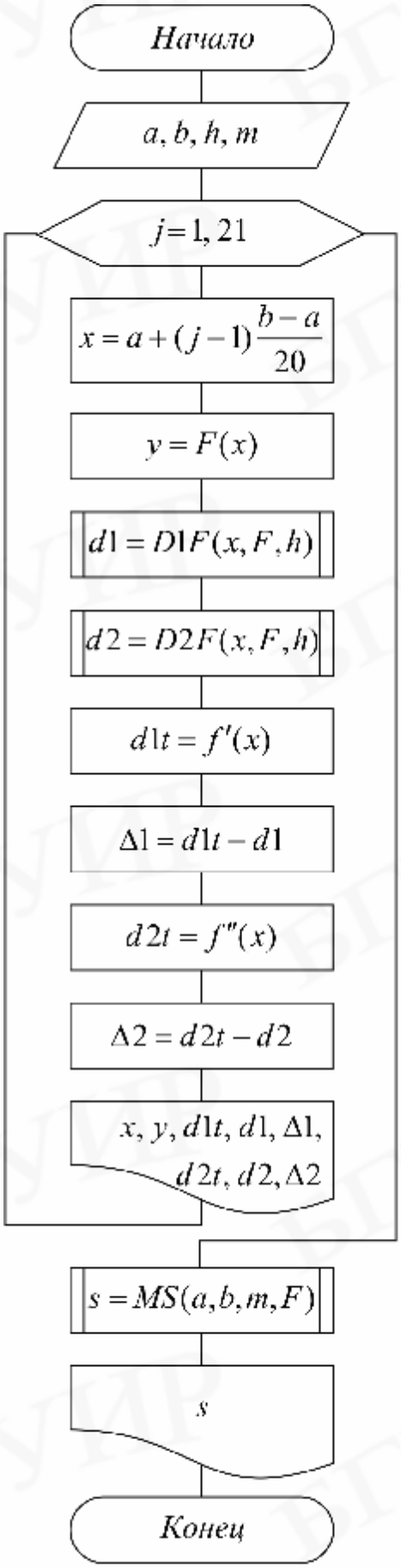
и соответствующая формула Гаусса 3-го порядка имеет вид

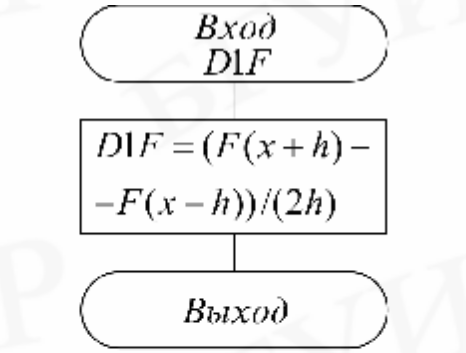
$$\int_a^b f(x)dx \approx \frac{h}{2}\sum_{i=1}^{n}(\frac{5}{9}f(x_i^1)+\frac{8}{9}f(x_i^0)+\frac{5}{9}f(x_i^2)). \qquad (4.16)$$

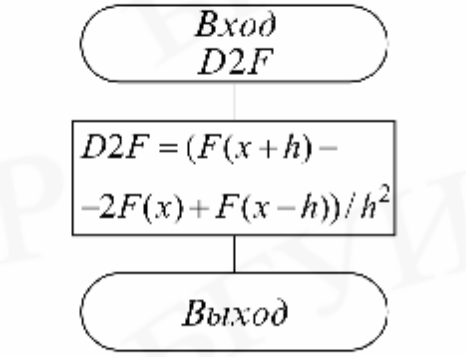
Порядок погрешности этой формулы при $h \to 0$ равен p = 7, т.е. значительно выше, чем у формулы Симпсона, практически при одинаковых затратах на вычисления. Следует отметить, что формулы Гаусса, особенно широко применяются для вычисления несобственных интегралов специального вида, когда подынтегральная функция имеет достаточно высокие производные.

4.4. ВАРИАНТЫ ЗАДАНИЙ

Для каждого варианта задан интервал $[a, b]$, функция $f(x)$ и указан метод вычисления интеграла. Вначале вычислить точные выражения для первой, второй производных $\frac{df(x)}{dx}$, $\frac{d^2f(x)}{dx^2}$ и для интеграла. Затем необходимо составить подпрограмму для вычисления первой и второй производных по формулам (4.4), (4.7) и подпрограмму вычисления интеграла указанным методом. Составить основную программу, которая вычисляет таблицу значений функции, ее


Начало
a, b, h, m
j=1, 21
x = a + (j − 1) (b − a)/20
y = F(x)
d1 = D1F(x, F, h)
d2 = D2F(x, F, h)
d1t = f'(x)
Δ1 = d1t − d1
d2t = f''(x)
Δ2 = d2t − d2
x, y, d1t, d1, Δ1, d2t, d2, Δ2
s = MS(a, b, m, F)
s
Конец


Вход D1F
D1F = (F(x + h) − −F(x − h))/(2h)
Выход


Вход D2F
D2F = (F(x + h) − −2F(x) + F(x − h))/h²
Выход

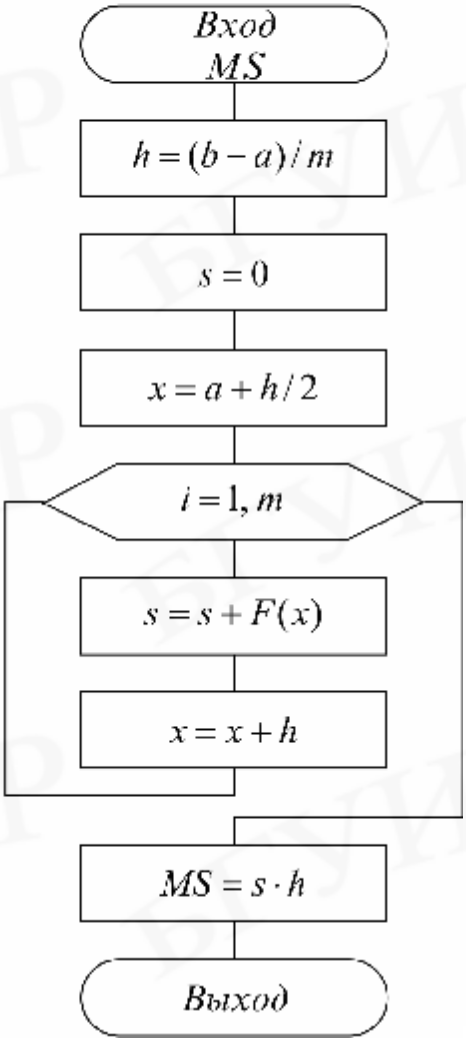

Вход MS
h = (b − a)/m
s = 0
x = a + h/2
i = 1, m
s = s + F(x)
x = x + h
MS = s · h
Выход

Рис. 4.5

точных и приближенных производных f, f', $\Lambda_1^3 f$, f'', $\Lambda_2^3 f$ в точках $x_j = a + (j-1)(b-a)/20;\ \ j = 1..21$, а также точное и приближенное значения интеграла. Расчет первой производной в крайних точках a и b выполнить по формулам (4.5) и (4.6) соответственно, вторую производную вычислять только во внутренних точках. Схема выполнения расчетов приведена на рис. 4.5.

Расчеты производной произвести для h_p =0.2, 0.1 и 0.05. Расчеты интеграла произвести для m=10, 20 и 40.

При использовании алгоритма вычисления интеграла с автоматическим выбором шага по данной точности расчет произвести для δ=0.1, 0.01, 0.001 и получить зависимость $m(\delta)$.

Проанализировать погрешность вычислений, для чего построить графики и вычислить погрешности производных и интеграла.

Таблица 4.1

N вар.	*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	Автомат	6.118
5	$\sqrt{x} - \cos^2(x)$	5	8	Гаусса 2	6.067
6	$\ln(x) - 5\sin^2(x)$	3	6	Гаусса 3	–3.367
7	$x - 5\sin^2(x)$	1	4	Средних	0.100
8	$\sin^2(x) - x/5$	0	4	Трапеций	0.153
9	$x^3 + 10x^2$	–8	2	Симпсона	713.3
10	$x^3 - 5x^2$	–2	5	Автомат	–69.42
11	$x^3 + 6x^2 - 0.02e^x$	–5	3	Гаусса 2	167.6
12	$x^2 + 5\cos(x)$	–1	4	Гаусса 3	22.09
13	$\sin^2(x) - 3\cos(x)$	1	7	Средних	3.533
14	$x^3 - 50\cos(x)$	–2	5	Автомат	154.73
15	$0.1x^3 + x^2 - 10\sin(x)$	–4	2	Симпсона	20.375

4.5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как производится аппроксимация оператора общего вида?
2. Получите формулы (4.2), (4.3) и (4.7).
3. Дайте геометрическую интерпретацию методов: средних, трапеций, Симпсона. Какой порядок погрешности они имеют?
4. В чем суть метода Гаусса?

ТЕМА 5. МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

5.1. 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 ноль, т.е. решение уравнения

$$f(x)=0. \qquad (5.1)$$

Точное решение удается получить в исключительных случаях, и обычно для нахождения корней уравнения применяются численные методы. Решение уравнения (5.1) при этом осуществляется в два этапа:

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

2. Вычисление выбранного корня с заданной точностью ε.

Первая задача решается графическим методом: на заданном отрезке $[a, b]$ вычисляется таблица значений функции с некоторым шагом h, строится ее график и определяются интервалы (α_i, β_i) длиной h, на которых находятся корни.

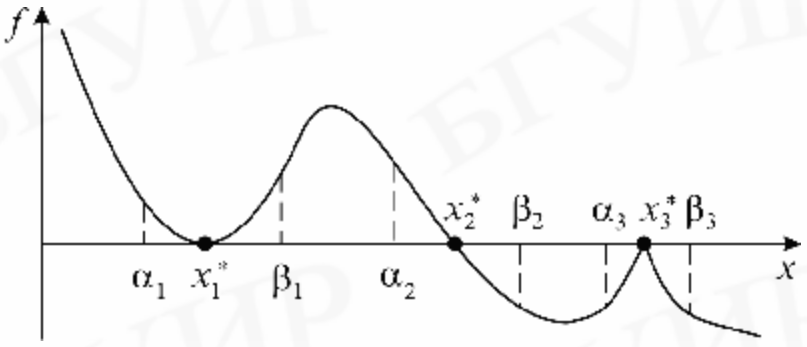


Рис. 5.1

На рис. 5.1 представлены три наиболее часто встречающиеся ситуации:

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. 5.1, в случаях «а» и «в» значение корня совпадает с точкой экстремума функции, и для нахождения таких корней можно использовать методы поиска минимума функции, описанные в теме 6.

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* (см. подразд. 1.5) на интервале (α, β), где находится интересующий нас корень x^*, выбирается m начальных значений $x_0, x_1, \ldots, x_{m-1}$ (обычно $x_0 = \alpha$, $x_1 = \beta$), 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 x_n выбирается в качестве приближенного значения корня ($x^* \approx x_n$).

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

5.2. ИТЕРАЦИОННЫЕ МЕТОДЫ УТОЧНЕНИЯ КОРНЕЙ

Метод простой итерации (MI)

Очень часто в практике вычислений встречается ситуация, когда уравнение (5.1) записано в виде, разрешенном относительно x:

$$x = \varphi(x). \qquad (5.2)$$

Заметим, что переход от записи уравнения (5.1) к эквивалентной записи (5.2) можно сделать многими способами, например, положив

$$\varphi(x) = x + \psi(x) f(x), \qquad (5.3)$$

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

В этом случае корни уравнения (5.2) являются также корнями (5.1) и наоборот.

Исходя из записи (5.2) члены рекуррентной последовательности в методе простой итерации вычисляются по закону

$$x_k = \varphi(x_{k-1}), \quad k = 1, 2, \ldots . \qquad (5.4)$$

Метод является одношаговым, так как последовательность имеет первый порядок (m = 1) и для начала вычислений достаточно знать одно начальное приближение $x_0 = \alpha$ или $x_0 = \beta$, или $x_0 = (\alpha + \beta)/2$.

Геометрическая иллюстрация сходимости и расходимости метода простой итерации представлена на рис. 5.2, а, б, из которого видно, что метод не всегда сходится к точному решению.

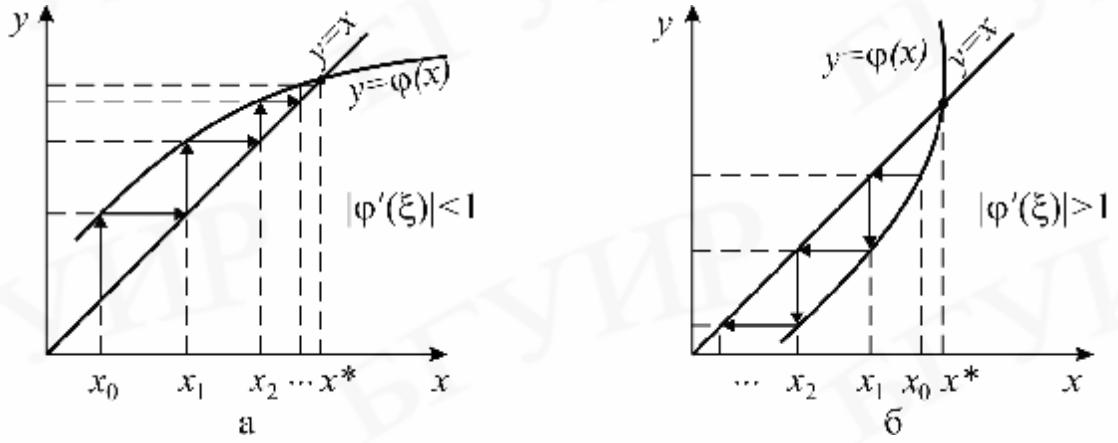


Рис. 5.2

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

$$\xi = (\alpha, \beta), \quad x^* \in (\alpha, \beta). \qquad (5.5)$$

Максимальный интервал **(**α, β**),** для которого выполняется неравенство (5.5), называется ***областью сходимости***. При выполнении условия (5.5) метод сходится, если начальное приближение x_0 выбрано из области сходимости. При этом ***скорость сходимости погрешности*** $\varepsilon_k = \left| x^* - x_k \right|$ к нулю вблизи корня приблизительно такая же, как у геометрической прогрессии $\varepsilon_k \approx \varepsilon_{k-1} q$ со знаменателем $q \cong \left| \varphi'(x^*) \right|$, т.е. чем меньше q, тем быстрее сходимость, и наоборот. Поэтому при переходе от (5.1) к (5.2) функцию $\psi(x)$ в (5.3) выбирают так, чтобы выполнялось условие сходимости (5.5) для как можно большей области (α, β) и с наименьшим q. Удачный выбор этих условий гарантирует эффективность расчетов. Часто положительные результаты дает применение ***релаксации*** (см. подразд. 2.7). Схема алгоритма метода простой итерации представлена на рис.

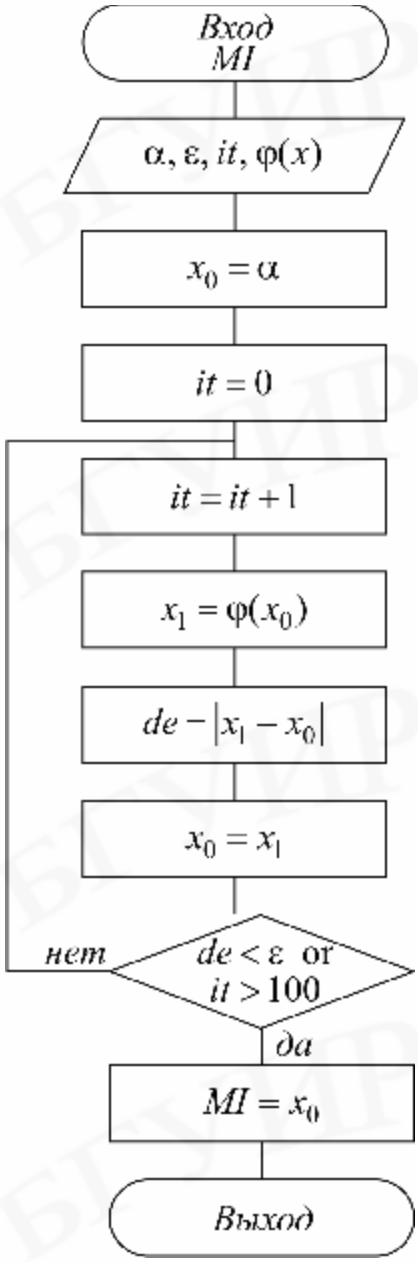


Рис. 5.3

Метод Ньютона (MN)

Этот метод является модификацией метода простой итерации и часто называется методом касательных. Если *f(x)* имеет непрерывную производную, тогда, выбрав в (5.3) $\psi(x) = -1/f'(x)$, получаем эквивалентное уравнение $x = x - f(x)/f'(x) = \varphi(x)$, в котором $q \cong \varphi'(x^*) \equiv 0$. Поэтому скорость сходимости рекуррентной последовательности метода Ньютона

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})} = \varphi(x_{k-1}) \qquad (5.6)$$

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

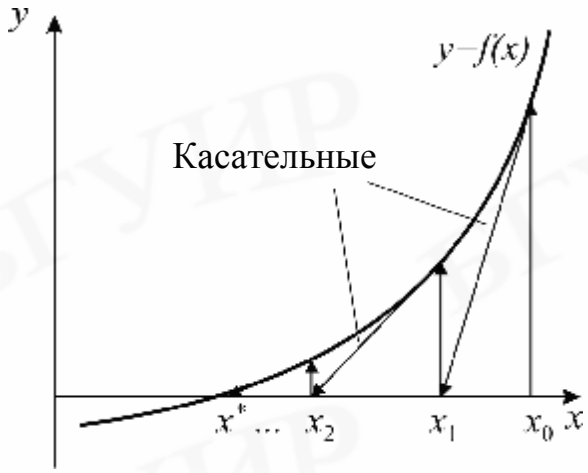


Рис. 5.4

Из (5.6) видно, что этот метод одношаговый (m = 1), и для начала вычислений требуется задать одно начальное приближение x_0 из ***области сходимости***, определяемой неравенством $|f \cdot f''|/(f')^2 < 1$. Метод Ньютона получил также второе название ***метод касательных*** благодаря геометрической иллюстрации его сходимости, представленной на рис. 5.4. Этот метод позволяет находить как простые, так и кратные корни. Основной его недостаток – малая область сходимости и необходимость вычисления производной *f'(x)*.

Метод секущих (MS)

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. 5.4 проводится секущая. Тогда вместо процесса (5.6) получаем

$$x_k = x_{k-1} - \frac{f(x_{k-1})h}{f(x_{k-1}) - f(x_{k-1} - h)} = \varphi(x_{k-1}). \qquad (5.7)$$

Здесь h – некоторый малый параметр метода, который подбирается из условия наиболее точного вычисления производной (см. тему 4).

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

Метод Вегстейна (MV)

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 (5.7) точку x_{k-2}, полученную на предыдущей итерации (рис. 5.5). Расчетная формула метода Вегстейна:

$$x_k = x_{k-1} - \frac{f(x_{k-1})(x_{k-1} - x_{k-2})}{f(x_{k-1}) - f(x_{k-2})} = \varphi(x_{k-1}, x_{k-2}). \quad (5.8)$$

Метод является двухшаговым (m = 2), и для начала вычислений требуется задать два начальных приближения x_0, x_1. Лучше всего $x_0 = \alpha, x_1 = \beta$ (см. рис. 5.1). Метод Вегстейна сходится медленнее метода секущих, однако, требует в 2 раза меньшего числа вычислений *f(x)* и за счет этого оказывается более эффективным.

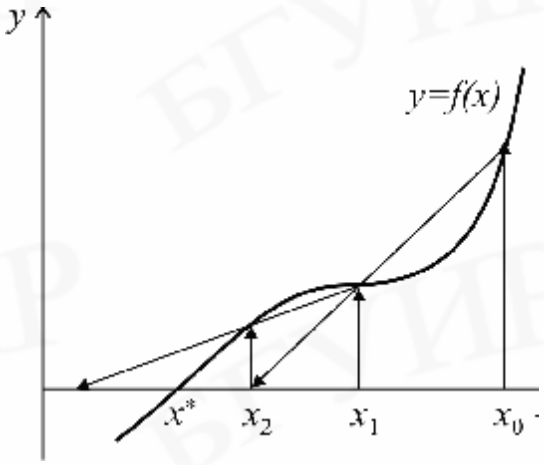


Рис. 5.5

Этот метод иногда называется улучшенным методом простой итерации и в применении к записи уравнения в форме (5.2) имеет вид

$$x_k = x_{k-1} - \frac{x_{k-1} - \varphi(x_{k-1})}{\left(1 - \dfrac{\varphi(x_{k-1}) - \varphi(x_{k-2})}{x_{k-1} - x_{k-2}}\right)}. \quad (5.9)$$

Схема алгоритма метода Вегстейна представлена на рис. 5.6.

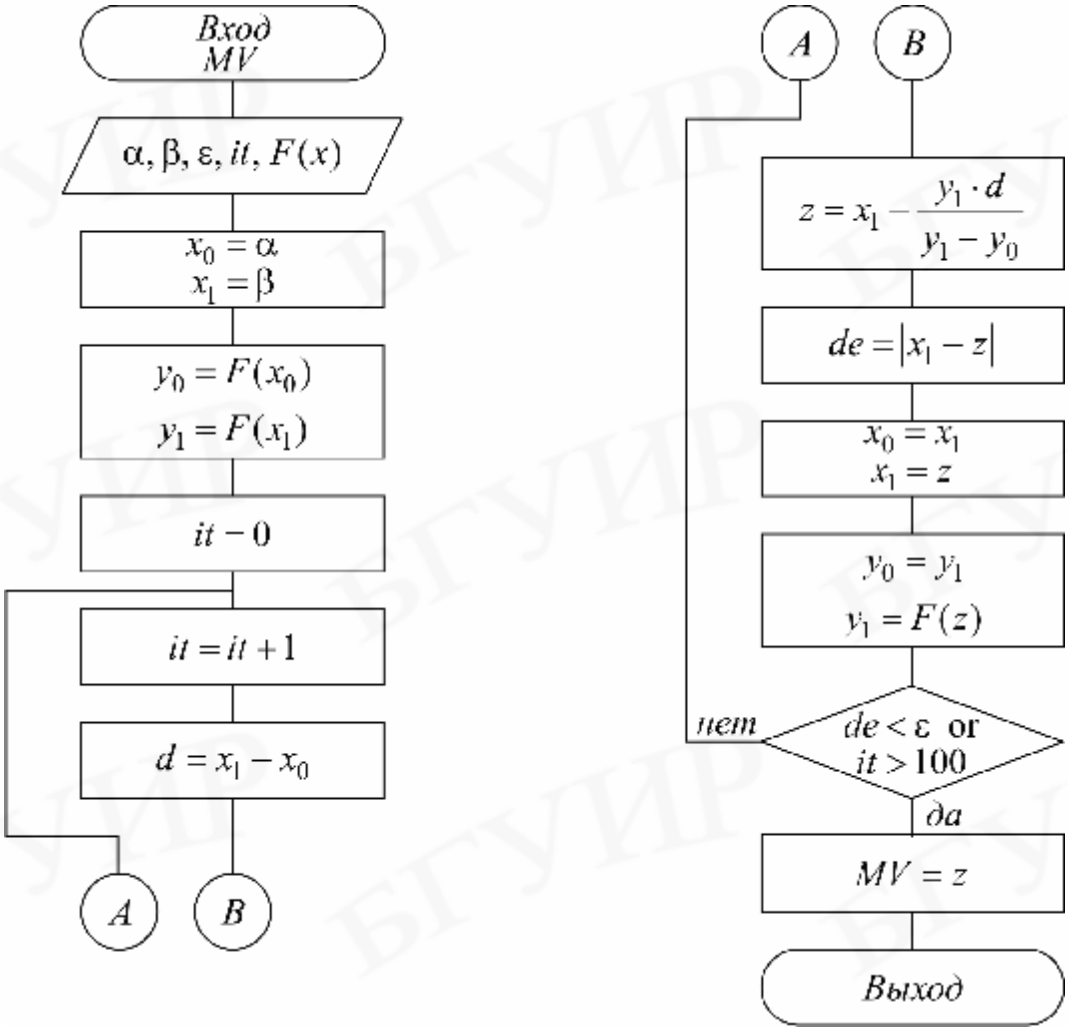


Рис. 5.6

Метод парабол (MP)

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. Его суть: задаются три начальные точки x_1, x_2, x_3 (например $x_1 = x_0 + h$, $x_2 = x_0$, $x_3 = x_0 - h$), в этих точках рассчитываются три значения функции $y = f(x)$, y_1, y_2, y_3 и строится интерполяционный многочлен второго порядка (рис. 5.7), который удобно записать в форме

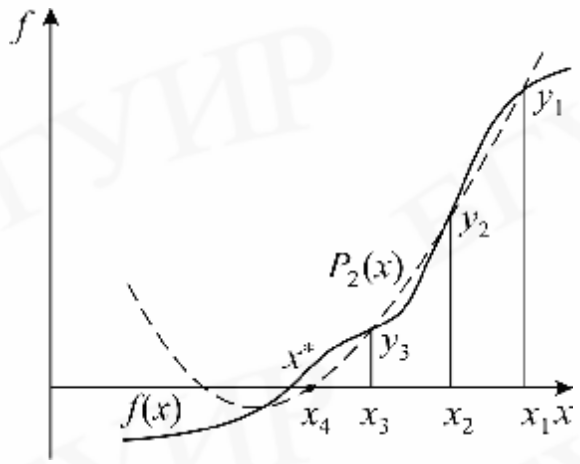


Рис. 5.7

$$P_2 = p(x - x_3)^2 + q(x - x_3) + r = pz^2 + qz + r\,. \qquad (5.10)$$

Коэффициенты этого многочлена вычисляются по формулам

$$z = x - x_3; \quad z_1 = x_1 - x_3; \quad z_2 = x_2 - x_3; \quad r = y_3;$$

$$p = \frac{(y_1 - y_3)z_2 - (y_2 - y_3)z_1}{z_1 z_2 (z_1 - z_2)};\ q = \frac{(y_1 - y_3)z_2^2 - (y_2 - y_3)z_1^2}{z_1 z_2 (z_2 - z_1)}\ . \qquad (5.11)$$

Полином (5.10) имеет два корня:

$$z_m^{1,2} = \frac{-q \pm \sqrt{q^2 - 4pr}}{2p},$$

из которых выбирается наименьший по модулю z_m и рассчитывается следующая точка $x_4 = x_3 + z_m$, в результате получается рекуррентная формула метода парабол:

$$x_k = x_{k-1} + z_m(x_{k-1},\ x_{k-2},\ x_{k-3}) = \varphi(x_{k-1},\ x_{k-2},\ x_{k-3})\,. \qquad (5.12)$$

Метод трехшаговый (m = 3). Скорость сходимости его больше, чем у метода Вегстейна, однако, не лучше, чем у метода Ньютона вблизи корня. Схема алгоритма метода парабол представлена на рис. 5.8.

Метод деления отрезка пополам (MD)

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 (α, β) применяют надежный метод деления отрезка пополам. Его алгоритм основан на построении рекуррентной

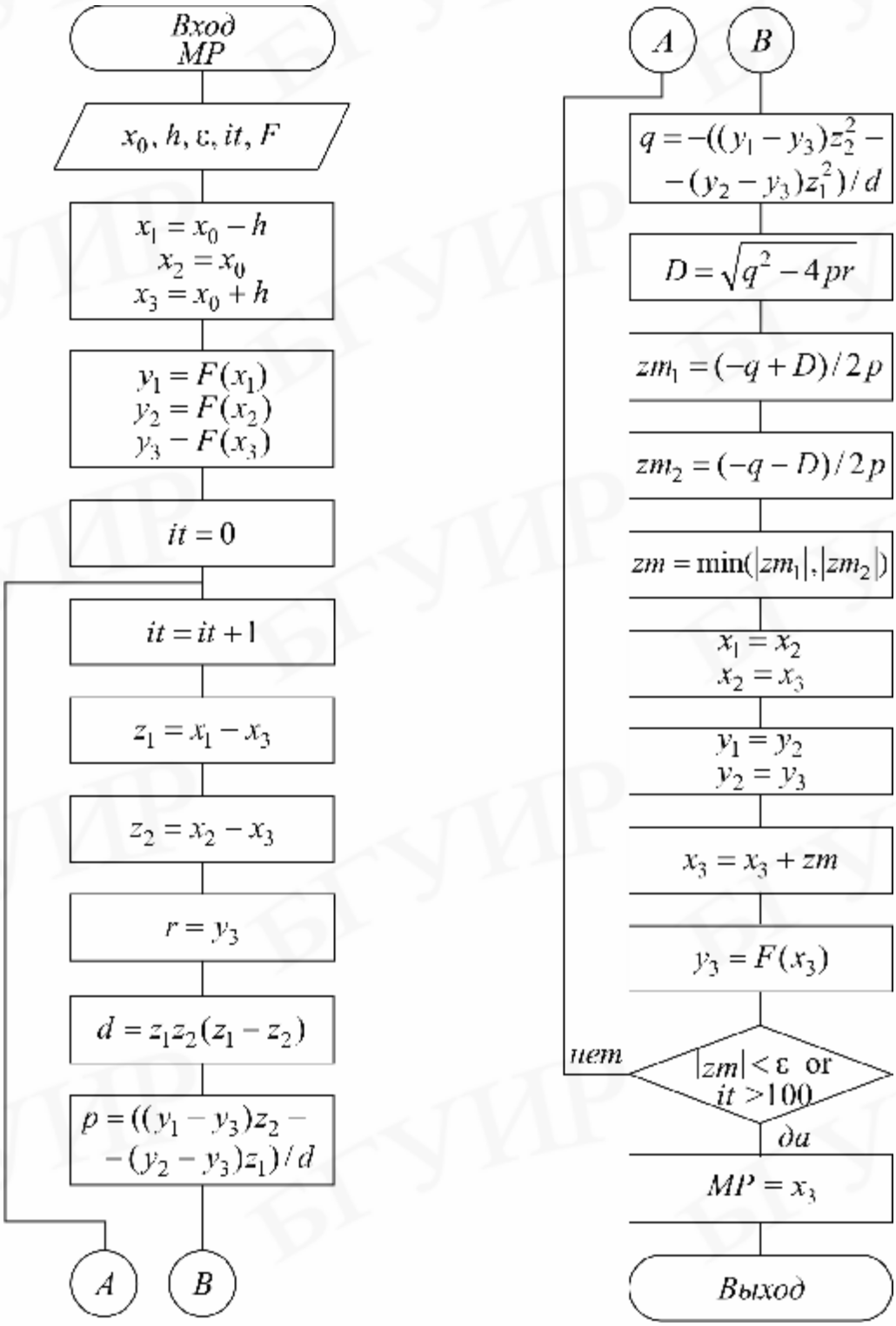


Рис. 5.8

последовательности по следующему закону: в качестве начального приближения выбираются границы интервала, на котором точно имеется один простой корень $x_0 = \alpha$, $x_1 = \beta$, далее находится его середина $x_2 = \frac{x_0 + x_1}{2}$; очередная точка x_3 выбирается как середина того из смежных с x_2 интервалов

$[x_0, x_2]$ или $[x_2, x_1]$, на котором находится корень. В результате получается следующий алгоритм метода деления отрезка пополам:

1. Вычисляем $y_0 = f(x_0), \quad y_1 = f(x_1)$.
2. Вычисляем $x_2 = (x_0 + x_1)/2, \quad y_2 = f(x_2)$.
3. Если $y_0 \cdot y_2 > 0$, тогда $x_0 = x_2, \; y_0 = y_2$, иначе $x_1 = x_2, \; y_1 = y_2$.
4. Если $x_1 - x_0 > \varepsilon$, тогда повторять с п.2.
5. Вычисляем $x^* = (x_0 + x_1)/2$.
6. Конец.

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

5.3. ВАРИАНТЫ ЗАДАНИЙ

По схеме на рис. 5.9 отладить программу определения всех корней функции *f(x)* в указанном интервале [*a, b*], использовать метод в соотвествии с полученным вариантом из табл. 5.1.

Таблица 5.1

N вар.	*f(x)*	*Интервал*		*Метод*
		a	*b*	
1	$4x - 7\sin(x)$	–2	2	MI
2	$x^2 - 10\sin^2(x) + 2$	–1	3	MN
3	$\ln(x) - 5\cos(x)$	1	8	MS
4	$e^x / x^3 - \sin^3(x) - 2$	4	7	MV
5	$\sqrt{x} - \cos^2(x) - 2$	4	8	MP
6	$\ln(x) - 5\sin^2(x)$	2	6	MD
7	$x - 5\sin^2(x) - 5$	3	9	MI
8	$\sin^2(x) - x/5 - 1$	–4	0	MN
9	$x^3 + 10x^2 - 50$	–12	5	MS
10	$x^3 - 5x^2 + 12$	–2	5	MV
11	$x^3 + 6x^2 - 0.02e^x - 14$	–6	2	MP
12	$x^2 + 5\cos(x) - 3$	–4	2	MD
13	$\sin^2(x) - 3\cos(x)$	–7	3	MI
14	$x^3 - 50\cos(x)$	–4	3	MN
15	$0.1x^3 + x^2 - 10\sin(x) - 8$	–4	4	MS

Примечание. В табл. 5.1 все функции на указанном интервале имеют три корня.

Программа работает следующим образом: сначала на экран выдается таблица значений функции и делается запрос на ввод начального приближения (это

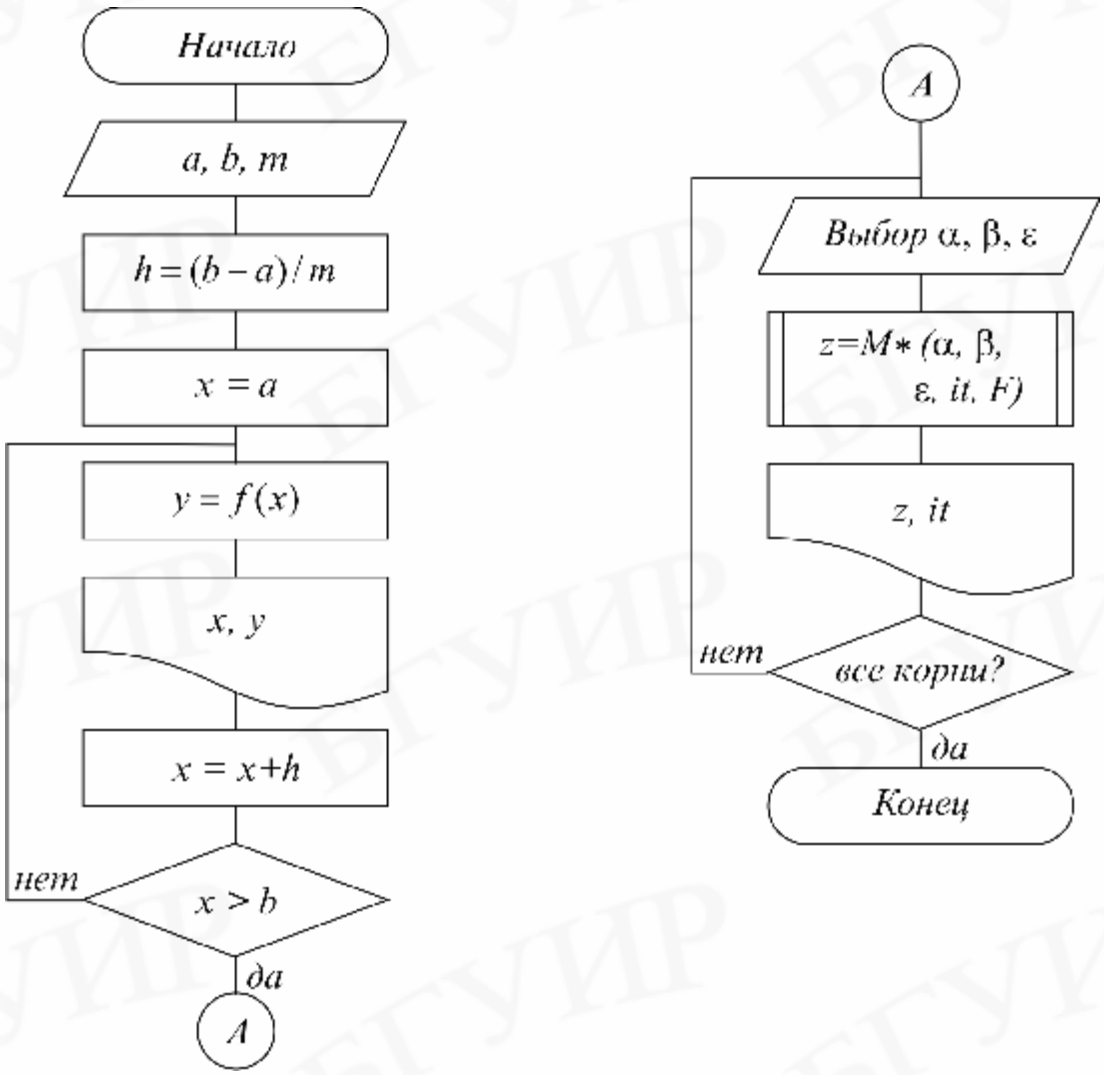


Рис. 5.9

может быть α, β или x_0) к тому корню, который надо получить с заданной точностью. После того как введены требуемые данные, идет обращение к подпрограмме и печать результатов.

Расчет функции, а также метод нахождения корня оформить в виде отдельных подпрограмм. Вариант метода и функции взять из таблицы. Выбрать точность $\varepsilon=10^{-4}$, а значение m по усмотрению.

После выполнения расчетов нарисовать график функции, а также график сходимости x_k к указанному корню x^*, для чего предусмотреть в процедуре нахождения корня возможность вывода значений x_k и $d_k = x_k - x_{k-1}$.

5.4. КОНТРОЛЬНЫЕ ВОПРОСЫ

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

ТЕМА 6. МЕТОДЫ НАХОЖДЕНИЯ МИНИМУМА ФУНКЦИИ ОДНОЙ ПЕРЕМЕННОЙ

6.1. ПОСТАНОВКА ЗАДАЧИ

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 все новых, более эффективных одномерных методов оптимизации продолжается и сейчас, несмотря на кажущуюся простоту задачи.

Наиболее часто используемые методы можно разбить на два класса:

1) *методы уточнения минимума на заданном интервале* $[a,\ b]$ (метод деления пополам, метод золотого сечения);

2) *методы спуска к минимуму* из некоторой начальной точки x_0 (метод последовательного перебора, метод квадратичной параболы, метод кубической параболы).

Методы из класса 1 предназначены для нахождения *условного минимума*. Задача ставится следующим образом: требуется найти такое значение x_m из отрезка $[a,\ b]$, при котором достигается минимум функции $y_m = f(x_m)$, т.е. для любого $x \in [a,\ b]$ выполняется условие $y_m \le f(x)$.

Методы из класса 2 предназначены для поиска и уточнения *безусловного локального минимума*. Задача ставится следующим образом: требуется найти такое значение x_m, $|x_m| < \infty$, при котором достигается локальный минимум $y_m = f(x_m)$, т.е. для любого x из некоторой ε окрестности $E = \{x,\ |x - x_m| < \varepsilon\}$ выполняется $y_m \le f(x)$. В этом случае при нахождении точки x_m обычно нет достаточно точной информации о ее положении, более того, локальных минимумов может быть несколько. Поэтому из соображений физического характера задают некоторое начальное приближение x_0, с которого начинают спуск к точке минимума.

Нахождение требуемого минимума функции осуществляется в два этапа.

1. Приближенное определение местоположения минимума из анализа таблицы значений функции.

2. Вычисление точки минимума x_m с заданной точностью одним из нижеприведенных методов.

6.2. КАКИЕ МЕТОДЫ МИНИМИЗАЦИИ ИСПОЛЬЗУЮТСЯ?

Метод деления отрезка пополам (MDP)

Задается интервал $[\alpha, \beta]$ и погрешность ε. Вычисляются значения функции в двух точках вблизи середины интервала и отбрасывается та часть интервала, которая содержит точку с большим значением функции. Расчет происходит до

тех пор, пока длина интервала не станет меньше заданной погрешности ε. Схема алгоритма представлена на рис. 6.1.

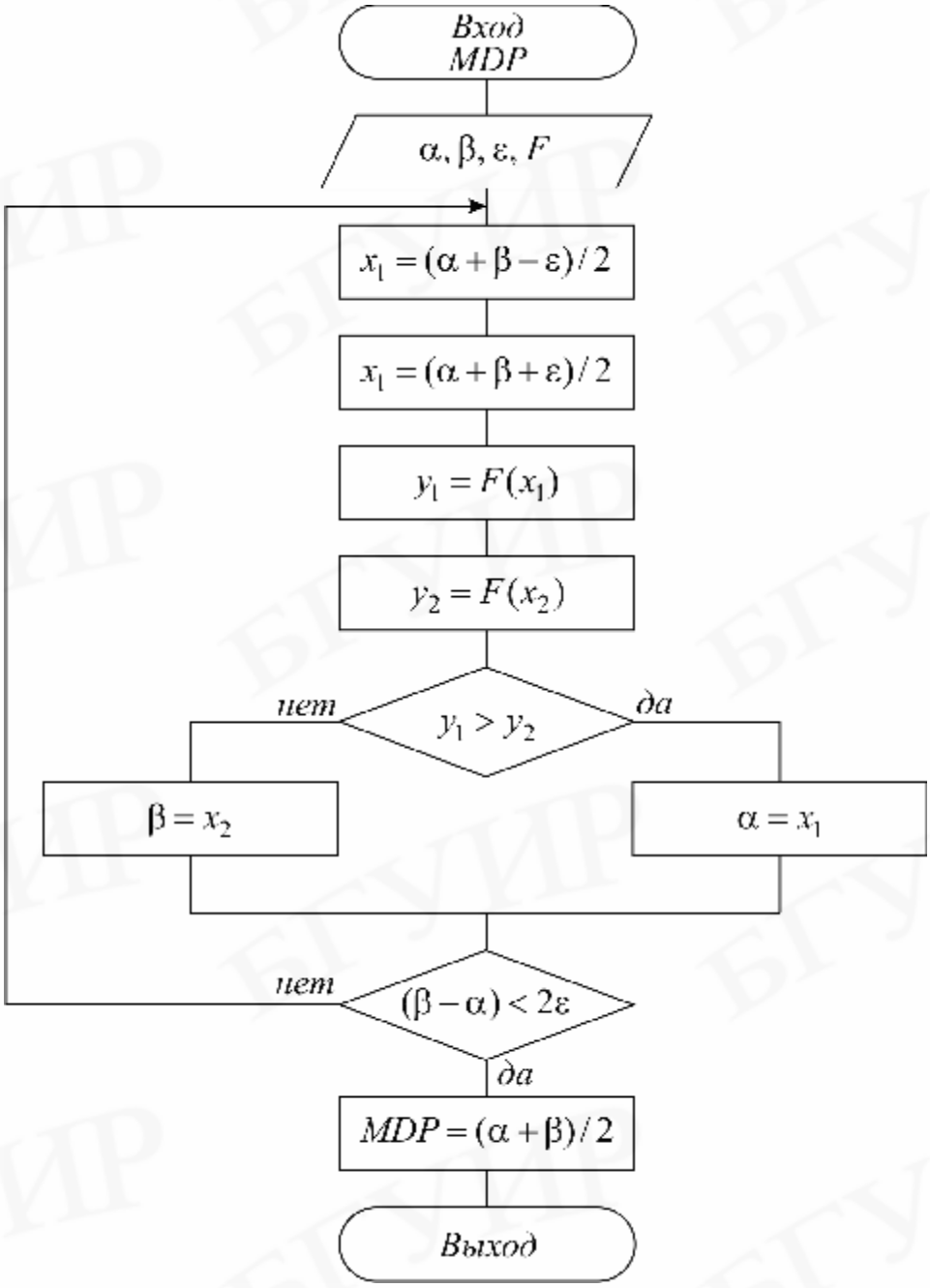


Рис. 6.1

В среднем за одно вычисление функции отрезок, на котором находится *x*, уменьшается примерно в 1.33 раза. Этот метод прост в реализации, позволяет находить минимум разрывной функции, однако требует большого числа вычислений функции для обеспечения заданной точности.

Метод золотого сечения (MZS)

Золотое сечение – это такое деление отрезка [α, β] на две неравные части [α, *x*] и [*x*, β], при котором имеет место следующее соотношение:

$$x\beta / \alpha\beta = \alpha x / x\beta = 1 - \xi, \quad \xi = 2/(3 + \sqrt{5}) \cong 0.381966011.$$

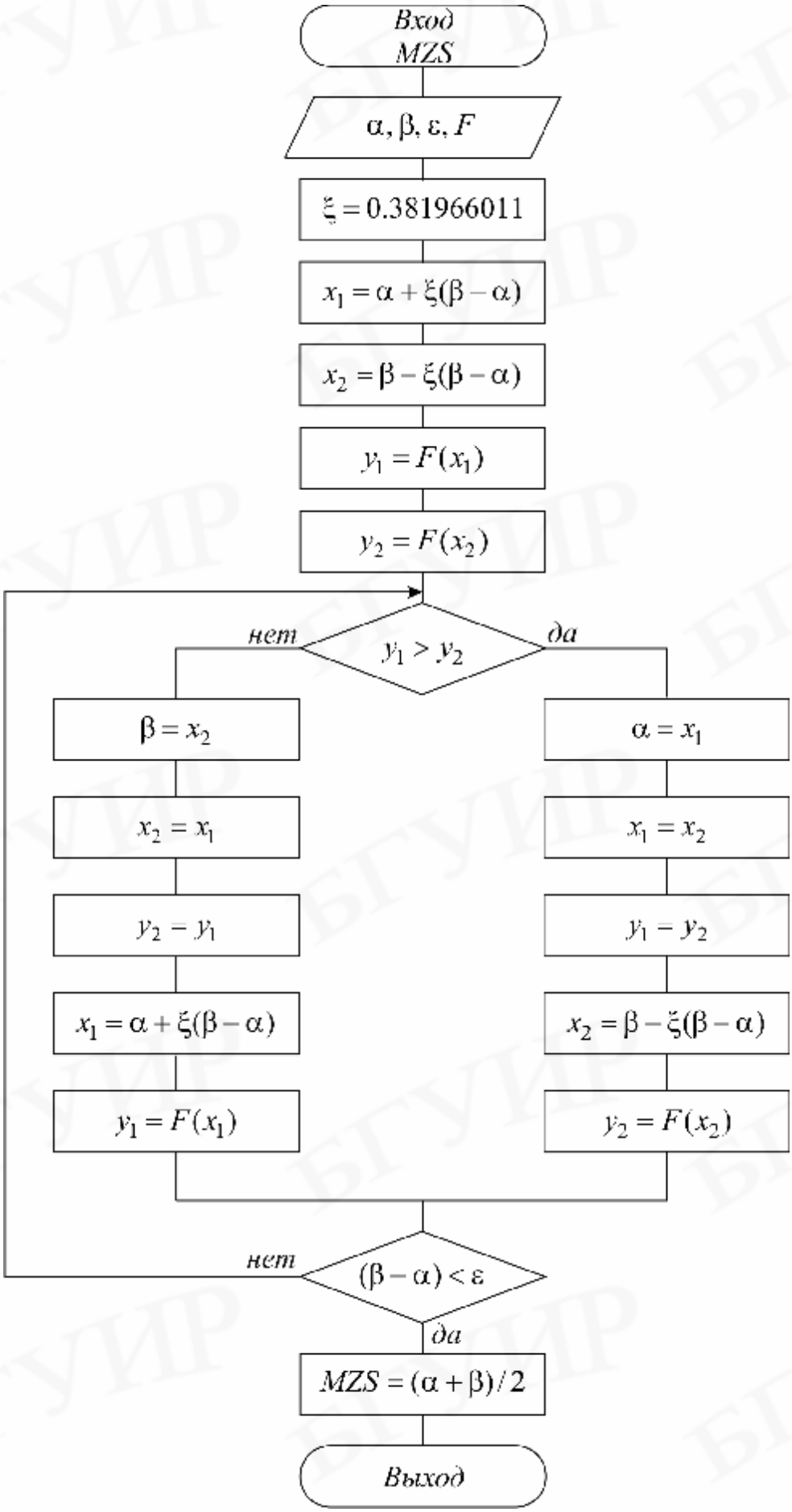

Вход
MZS
α, β, ε, F
ξ = 0.381966011
x1 = α + ξ(β − α)
x2 = β − ξ(β − α)
y1 = F(x1)
y2 = F(x2)
нет
y1 > y2
да
β = x2
x2 = x1
y2 = y1
x1 = α + ξ(β − α)
y1 = F(x1)
α = x1
x1 = x2
y1 = y2
x2 = β − ξ(β − α)
y2 = F(x2)
нет
(β − α) < ε
да
MZS = (α + β) / 2
Выход

Рис. 6.2

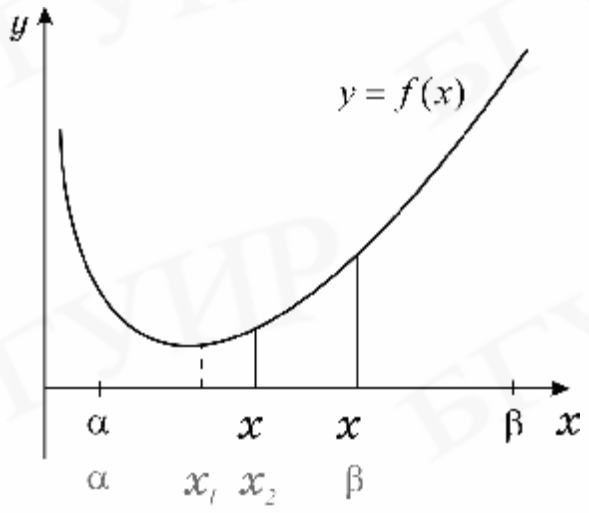


Рис. 6.3

Алгоритм поиска минимума аналогичен вышеописанному MDP и отличается тем, что вначале точки x_1 и x_2 выбираются так, чтобы выполнялось золотое отношение, и вычисляются значения функции в этих точках.

Затем, после очередного сокращения интервала путем отбрасывания неблагоприятной крайней точки на оставшемся *отрезке уже имеется точка, делящая его в золотом отношении* (точка x_1 на рис. 6.3), известно и значение функции в этой точке. Остается лишь выбрать ей симметричную и вычислить значение функции в этой точке для того, чтобы решить, какую из крайних точек отбросить. Схема алгоритма представлена на рис. 6.2.

За одно вычисление функции отрезок, на котором находится x_m, уменьшается в 1-ξ ≅ 1.61 раза, т.е. быстрее чем МДР. Данный метод является наилучшим среди методов класса 1.

Метод последовательного перебора (МРР)

Идея этого метода состоит в том, что, спускаясь из точки x_0 с заданным шагом h в направлении уменьшения функции, устанавливают интервал длиной h, на котором находится минимум, и затем его уточняют. Уточнение можно

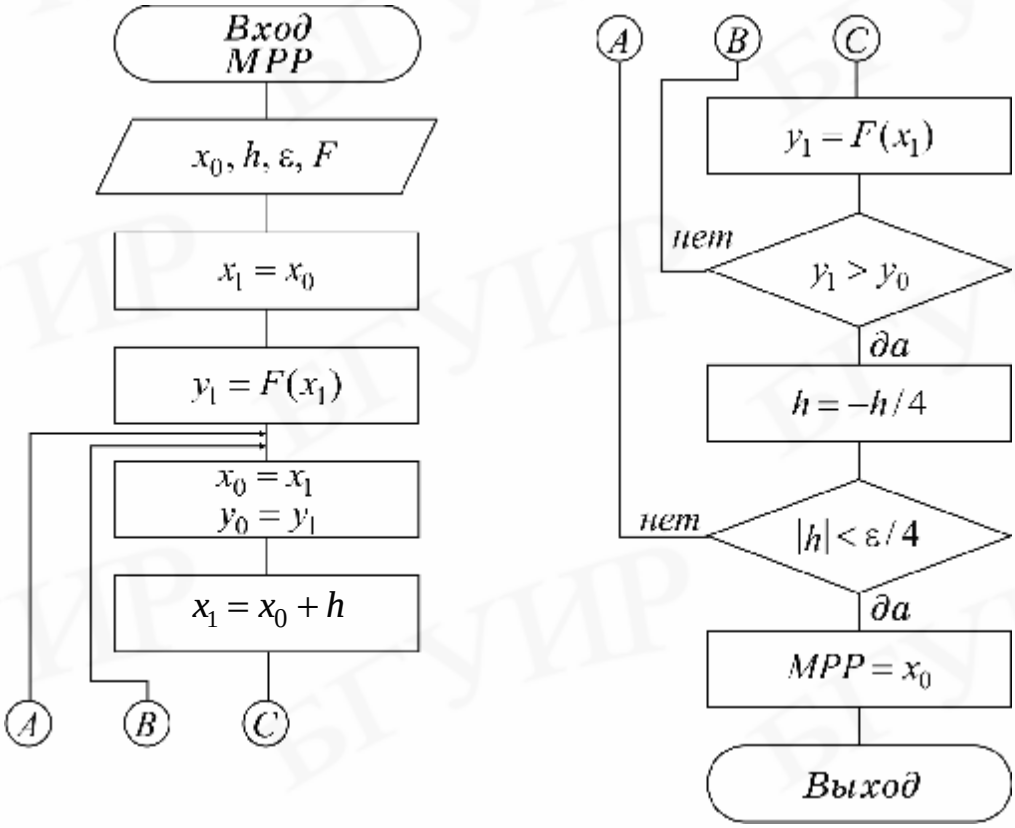


Рис. 6.4

осуществить либо методом золотого сечения, либо повторяя спуск из последней точки, уменьшив шаг и изменив его знак. Алгоритм последнего варианта приведен на рис. 6.4.

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_m.

Метод квадратичной параболы (MP2)

Для ускорения спуска к минимуму из некоторой точки x_0 используют локальные свойства функции вблизи этой точки. Так, скорость и направление убывания можно определить по величине и знаку первой производной. Вторая производная характеризует направление выпуклости: если $f'' > 0$, то функция имеет выпуклость вниз, иначе – вверх. Вблизи локального безусловного

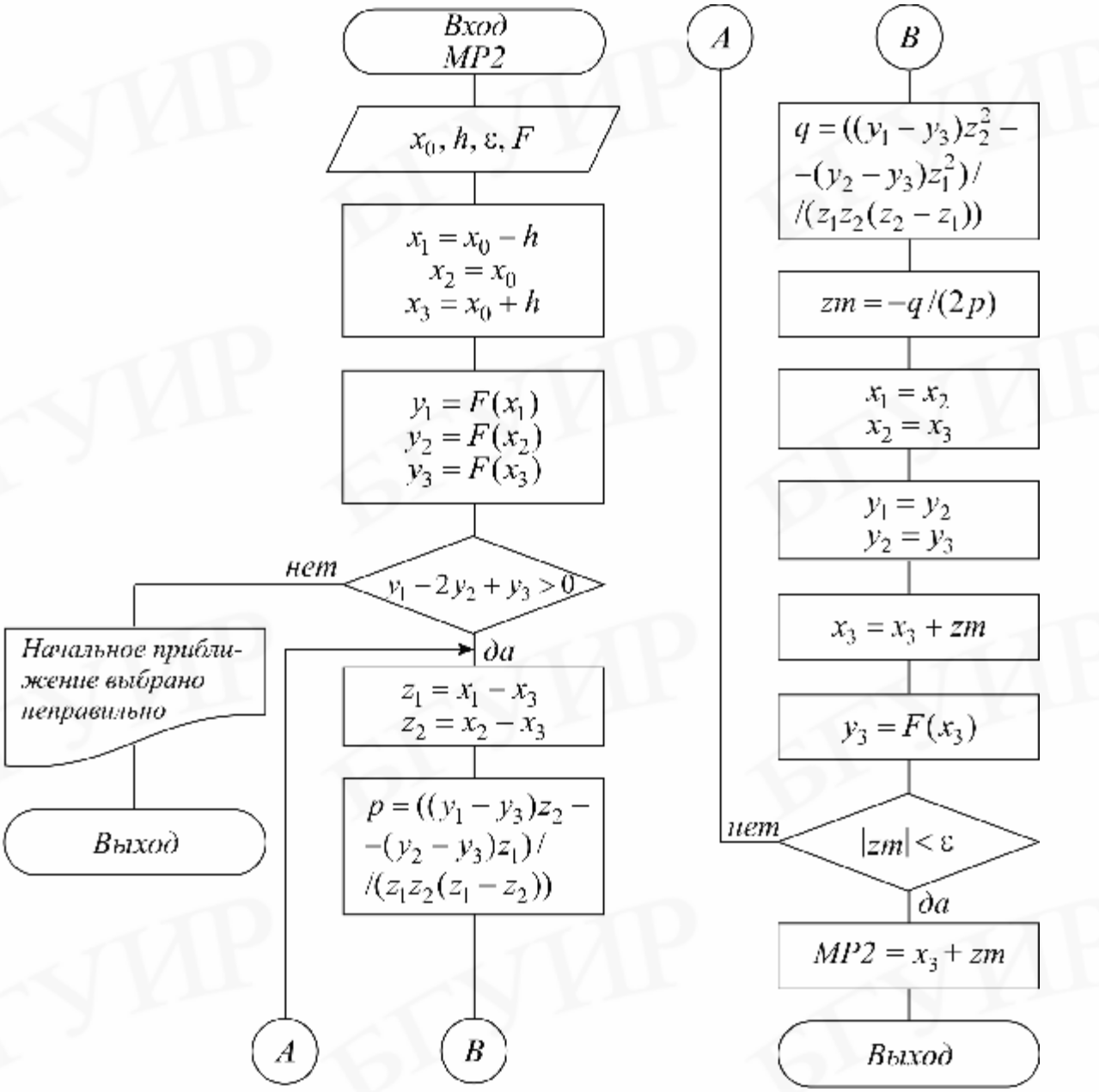


Рис. 6.5

минимума дважды дифференцируемая функция всегда выпукла вниз. Поэтому, если вблизи точки минимума функцию аппроксимировать квадратичной параболой, то она будет иметь минимум. Это свойство и используется в методе квадратичной параболы, суть которого в следующем.

Вблизи точки x_0 выбираются три точки x_1, x_2, x_3. Вычисляются значения y_1, y_2, y_3. Через эти точки проводится квадратичная парабола:

$$p(x-x_3)^2+q(x-x_3)+r=pz^2+qz+r,$$
$$z=x-x_3,\quad z_1=x_1-x_3,\quad z_2=x_2-x_3,\quad r=y_3, \qquad (6.1)$$
$$p=\frac{(y_1-y_3)z_2-(y_2-y_3)z_1}{z_1z_2(z_1-z_2)},\quad q=\frac{(y_1-y_3)z_2^2-(y_2-y_3)z_1^2}{z_1z_2(z_2-z_1)}.$$

Если $p>0$, то парабола имеет минимум в точке $z_m=-q/(2p)$. Следовательно, можно аппроксимировать положение минимума функции значением $x_{m1}=x_3+z_m$ и, если точность не достигнута, следующий спуск производить, используя эту новую точку и две предыдущие. Получается последовательность x_{m1}, x_{m2}, x_{m3}, ... , сходящаяся к точке x_m. Схема алгоритма представлена на рис. 6.5.

Данный метод сходится очень быстро и является одним из наилучших методов спуска. Следует отметить, однако, что вблизи минимума расчет по приведенным здесь формулам для p и q приводит к накоплению погрешности из-за потери значащих цифр при вычитании близких чисел. Поэтому разные авторы предлагают свои эквивалентные формулы, счет по которым более устойчив. Кроме того, в алгоритм вносятся некоторые поправки, позволяющие предусмотреть различные неприятные ситуации – переполнение, деление на 0, уход от корня.

Метод кубической параболы (МРЗ)

Данный метод аналогичен предыдущему, но за счет использования аппроксимации кубической параболой имеет более высокую сходимость, *если функция допускает простое вычисление производной*. При его использовании вблизи точки x_0 выбираются две точки x_1 и x_2 (обычно x_1 = x_2), вычисляются значения функции y_1, y_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 производных параболы и функции:

$$p(x-x_2)^3+q(x-x_2)^2+r(x-x_2)+s=pz^3+qz^2+rz+s=P(z),$$
$$z=x-x_2,\ z_1=x_1-x_2,$$
$$P(x_2)=y_2,\quad P'(x_2)=D_2,\quad P(z_1)=y_1,\quad P'(z_1)=D_1.$$

Как нетрудно убедиться, коэффициенты параболы вычисляются по следующим формулам:

$$s=y_1,\ r=D_1,$$

$$p = (D_1 - D_2 - 2(y_1 - y_2 - D_2 \cdot z_1)/z_1)z_1^2,$$
$$q = (D_2 - D_1 + 3(y_1 - y_2 - D_2 \cdot z_1)/z_1)/z_1.$$

Известно, что кубическая парабола имеет минимум в точке

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

Поэтому приближенное положение минимума можно получить по формуле $x_{m1} = x_2 + z_m$ и, если точность не достигнута, следующий спуск следует производить уже из точек x_2, x_{m1} (точка x_1 отбрасывается). Если подкоренное

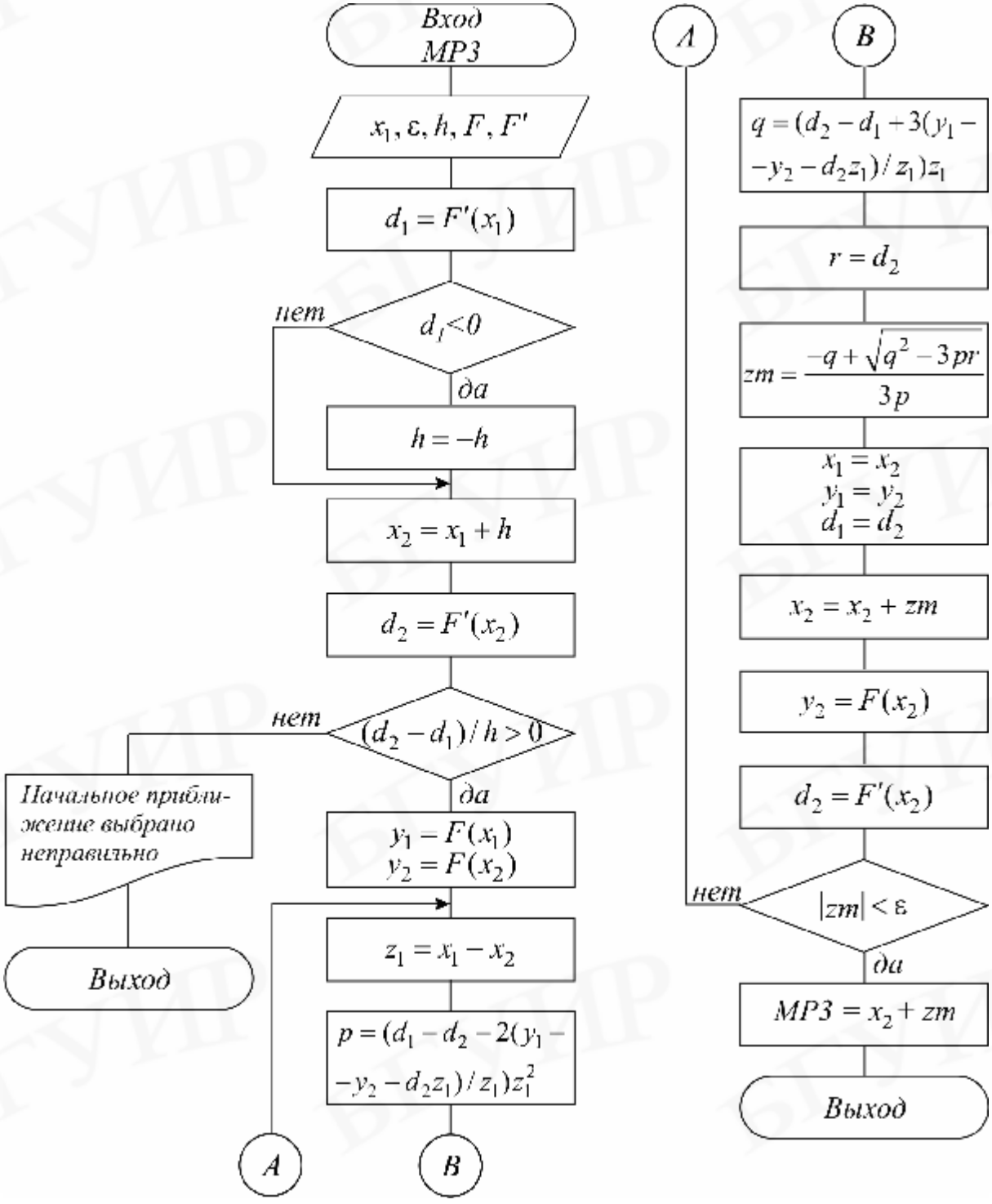


Рис. 6.6

выражение окажется отрицательным, то спуск следует производить до точки перегиба параболы $z_{m1} = -q/3p$. Следует также убедиться, что в начальной точке функция вогнута вниз $\frac{D_2 - D_1}{x_2 - x_1} > 0$. Схема алгоритма представлена на рис. 6.6.

Следует отметить, что вблизи точки минимума расчет по приведенным здесь простейшим формулам для p, q не всегда устойчив из-за ошибок округления, поэтому различные авторы рекомендуют использовать несколько преобразованные формулы.

6.3. ВАРИАНТЫ ЗАДАНИЙ

В соответствии со схемой (рис. 6.7) требуется отладить программу определения минимума указанной в таблице функции заданным методом. Сначала на экране выдается таблица значений функции и делается запрос на ввод начального приближения (α, β или x_0, h) для вычисления требуемого локального минимума. В качестве функции *Fun* использовать метод в соответствии с заданным вариантом. Расчет функции, а также метод нахождения минимума оформить в виде отдельных подпрограмм. Выбрать m и

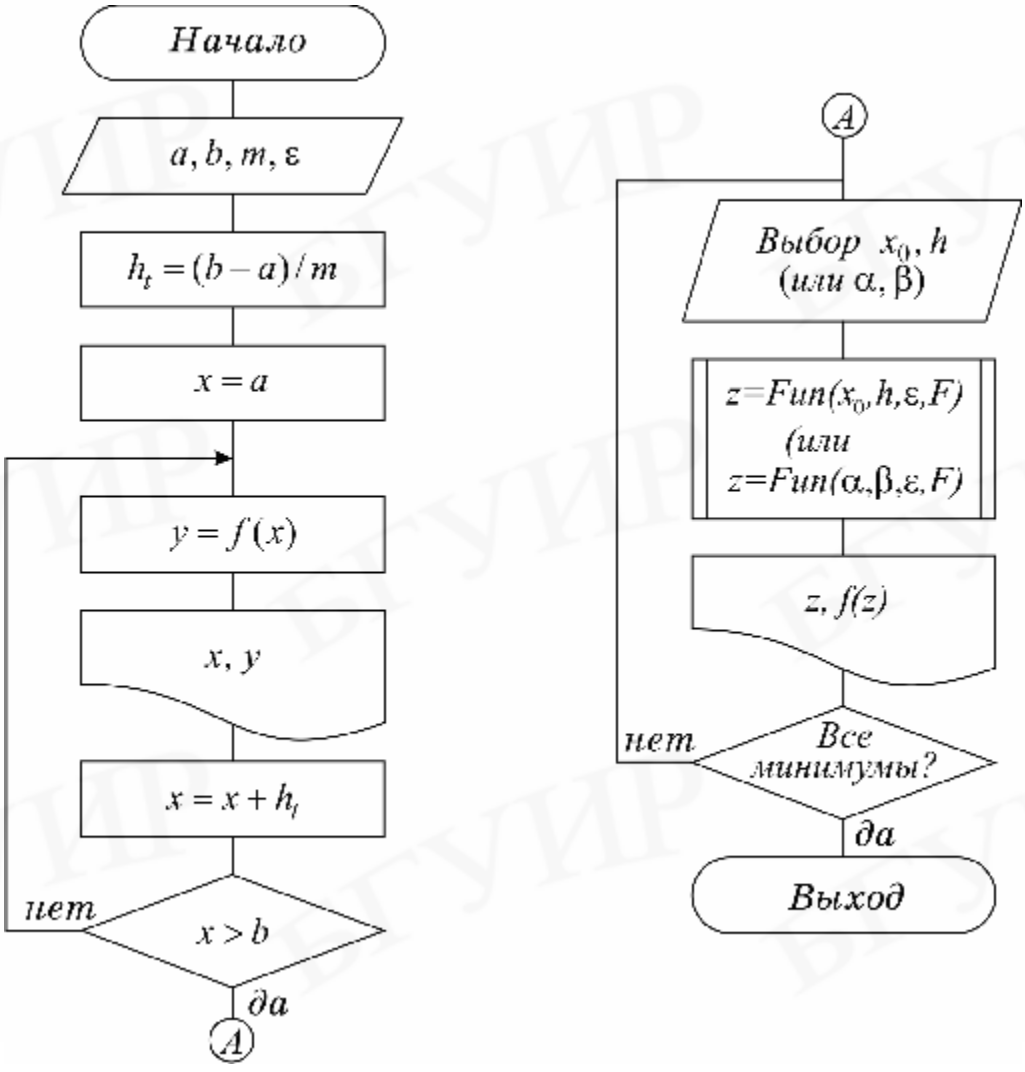


Рис. 6.7

ε по усмотрению. Все функции из табл. 6.1 на указанном интервале имеют три локальных минимума.

После выполнения расчетов построить график исследуемой функции и проанализировать зависимость количества итераций от ε ($\varepsilon=10^{-2}$, $\varepsilon=10^{-3}$, $\varepsilon=10^{-4}$, $\varepsilon=10^{-5}$), для чего встроить в алгоритм счетчик количества вычислений функции.

Таблица 6.1

N вар.	*Минимизируемая функция f(x)*	*Интервал*		*Метод*
		a	*b*	
1	$x-7\sin^2(x)$	–3	6	MDP
2	$x\sin(x)-10\sin^2(x)$	–6	3	MZS
3	$\ln(x)-5\cos^2(x)$	2	11	MPP
4	$e^x/x^3-30\cos(x)$	0.2	12	MP2
5	$\sqrt{x}-\cos(x)$	4	20	MP3
6	$\ln^2(x)-10\cos^2(x)$	2	10	MDP
7	$x-5\sin^2(x)$	1	9	MZS
8	$20x\cdot\sin^2(x)-x^2$	–9	–1	MPP
9	$x^2-100\sin(x)$	–6	10	MP2
10	$20x^3\cdot\sin(x)$	–6	6	MP3
11	$\sin^4(x)-\ln(x)$	2	11	MDP
12	$\sin^2(x)+\cos(x)$	–4	4	MZS
13	$x^2-4x\cdot\sin(x)+\cos(x)$	–4	9	MPP
14	$x^2\cdot\cos(x)+2\sin(x)$	8	24	MP2
15	$x\cdot\cos(x)-x$	2	18	MP3

6.4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое условный и локальный минимумы, в чем их отличие?
2. В чем суть метода последовательного перебора?
3. Объясните графически, почему метод золотого сечения эффективнее метода деления пополам?
4. Дайте геометрическую интерпретацию методов квадратичной и кубической парабол.

ТЕМА 7. РЕШЕНИЕ ЗАДАЧИ КОШИ ДЛЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

7.1. КАКИЕ БЫВАЮТ 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, 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

$$\begin{cases} \dfrac{du_1(x)}{dx} = f_1(x,\ u_1,\ ...,\ u_m), \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \dfrac{du_m(x)}{dx} = f_m(x,\ u_1,\ ...,\ u_m). \end{cases} \qquad (7.1)$$

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 (7.1) 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 и их производных на границах;

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

$$u_1(a) = u_1^0;\ ...;\ u_m(a) = u_m^0. \qquad (7.2)$$

При изложении методов решения задачи Коши воспользуемся компактной записью задачи (7.1), (7.2) в векторной форме.

$$\frac{d\vec{u}}{dx} = f(x, \vec{u});\quad \vec{u}(a) = \vec{u}^0. \qquad (7.3)$$

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

7.2. ОСНОВНЫЕ ПОЛОЖЕНИЯ МЕТОДА СЕТОК ДЛЯ РЕШЕНИЯ ЗАДАЧИ КОШИ

Чаще всего задача (7.3) решается методом сеток.

Суть метода сеток состоит в следующем:

1) в области интегрирования выбирается упорядоченная система точек $a = x_0 < x_1 < x_2 < \ldots < x_n < x_{n+1} = b$, 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 упрощения в дальнейшем будим считать сетку равномерной;

2) 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 – ***интегро-интерполяционный метод***.

Согласно этому способу для получения конечно-разностной схемы проинтегрируем уравнение (7.3) на каждом интервале $[x_k,\ x_{k+1}]$ для k=0, ..., n-1 и разделим на длину этого интервала:

$$\frac{1}{h}\int_{x_k}^{x_{k+1}} \frac{d\vec{u}}{dx}\,dx = \frac{\vec{u}^{k+1} - \vec{u}^k}{h} = \frac{1}{h}\int_{x_k}^{x_{k+1}} \vec{f}(x,\ \vec{u}(x))\,dx. \qquad (7.4)$$

Интеграл в правой части (7.4) аппроксимируем одной из квадратурных формул (см. подразд. 4.3), после чего получаем систему уравнений относительно ***приближенных*** неизвестных значений искомой функции, которые в отличие от точных обозначим $\vec{y}^k \approx \vec{u}^k$.

$$\frac{\vec{y}^{k+1} - \vec{y}^k}{h} = \frac{1}{h}\sum_j \alpha_j \vec{F}^j, \quad \vec{F}^j = \vec{f}(x_j, \vec{y}^j), \quad x_k \le x_j \le x_{k+1}. \qquad (7.5)$$

Здесь x_j – точки внутри интервала, используемые для получения квадратурной формулы (см. подразд. 4.3).

Структура конечно-разностной схемы для задачи Коши (7.5) такова, что она устанавливает закон рекуррентной последовательности $\vec{y}^{k+1} = \varphi\left(\vec{y}^k\right)$ для искомого решения $\vec{y}^0$, $\vec{y}^1$, $\vec{y}^2$, ..., $\vec{y}^n$. Поэтому, используя начальное условие задачи (7.2) и задавая $\vec{y}^0 = \vec{u}^0$, затем по рекуррентным формулам последовательно находят все $\vec{y}^k$, $k = 1, \ldots, n$.

При замене интеграла приближенной квадратурной формулой вносится ***погрешность аппроксимации*** дифференциального уравнения разностным,

которая получается как невязка, если в конечно-разностном уравнении (7.5) подставить вместо $\vec{y}^k$ значение точного решения $\vec{u}^k$:

$$\psi_k = \left| \frac{\vec{u}^{k+1} - \vec{u}^k}{h} - \frac{1}{h} \sum_j \alpha_j f(x_j, \vec{u}^j) \right| .$$

Воспользовавшись соотношением (7.4), получаем простое выражение для вычисления $\psi_k(h)$:

$$\psi_k(h) = \frac{1}{h} \left| \int_{x_k}^{x_{k+1}} \vec{f}(x, \vec{u}(x)) dx - \sum_j \alpha_j \vec{F}^j \right| , \qquad (7.6)$$

которая зависит от шага сетки.

Говорят, что разностная схема (7.5) ***аппроксимирует*** исходную дифференциальную задачу с порядком p, если при $h \to 0$, $\psi_k(h) \le Ch^p$, $C - const$. Из (7.6) следует, что порядок аппроксимации на 1 меньше, чем порядок погрешности используемой квадратурной формулы на интервале $[x_k, x_{k+1}]$.

Чем больший ***порядок аппроксимации p*** , тем выше ***точность решения***:

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

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:

$$\varepsilon(h) \le C_0 \cdot \max_k \left| \psi_k(h) \right| \le C_0 \cdot C \cdot h^p , \qquad (7.7)$$

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.

7.3. КАКИЕ БЫВАЮТ КОНЕЧНО-РАЗНОСТНЫЕ СХЕМЫ?

Большое разнообразие методов обусловлено возможностью по-разному выбирать узлы и квадратурные формулы для аппроксимации интеграла в (7.4) при получении схемы (7.5).

MI. Явная схема 1-го порядка (Эйлера)

Заменим интеграл в (7.4) по формуле левых прямоугольников:

$$\frac{1}{h}\int_{x_k}^{x_{k+1}} \mathbf{f}(x,\mathbf{u}(x))dx \approx \frac{\mathbf{F}^k h}{h} = \mathbf{F}^k .$$

Получим

$$\frac{\mathbf{y}^{k+1} - \mathbf{y}^k}{h} = \mathbf{f}(x_k, \mathbf{y}^k), \quad k = 0,\ 1,\ 2,\ ...,\ n . \tag{7.8}$$

Задавая $\mathbf{y}^0 = \mathbf{u}^0$, с помощью (7.8) легко получить все последующие значения $\mathbf{y}^k$, $k = 1, 2, ..., n1$, так как формула явно разрешается относительно $\mathbf{y}^{k+1}$. Погрешность аппроксимации $\psi(h)$ и соответственно точность $\varepsilon(h)$ имеют первый порядок в силу того, что формула левых прямоугольников на интервале $[x_x,\ x_{k+1}]$ имеет погрешность второго порядка, а схема устойчива.

Алгоритм представлен на рис.7.1.

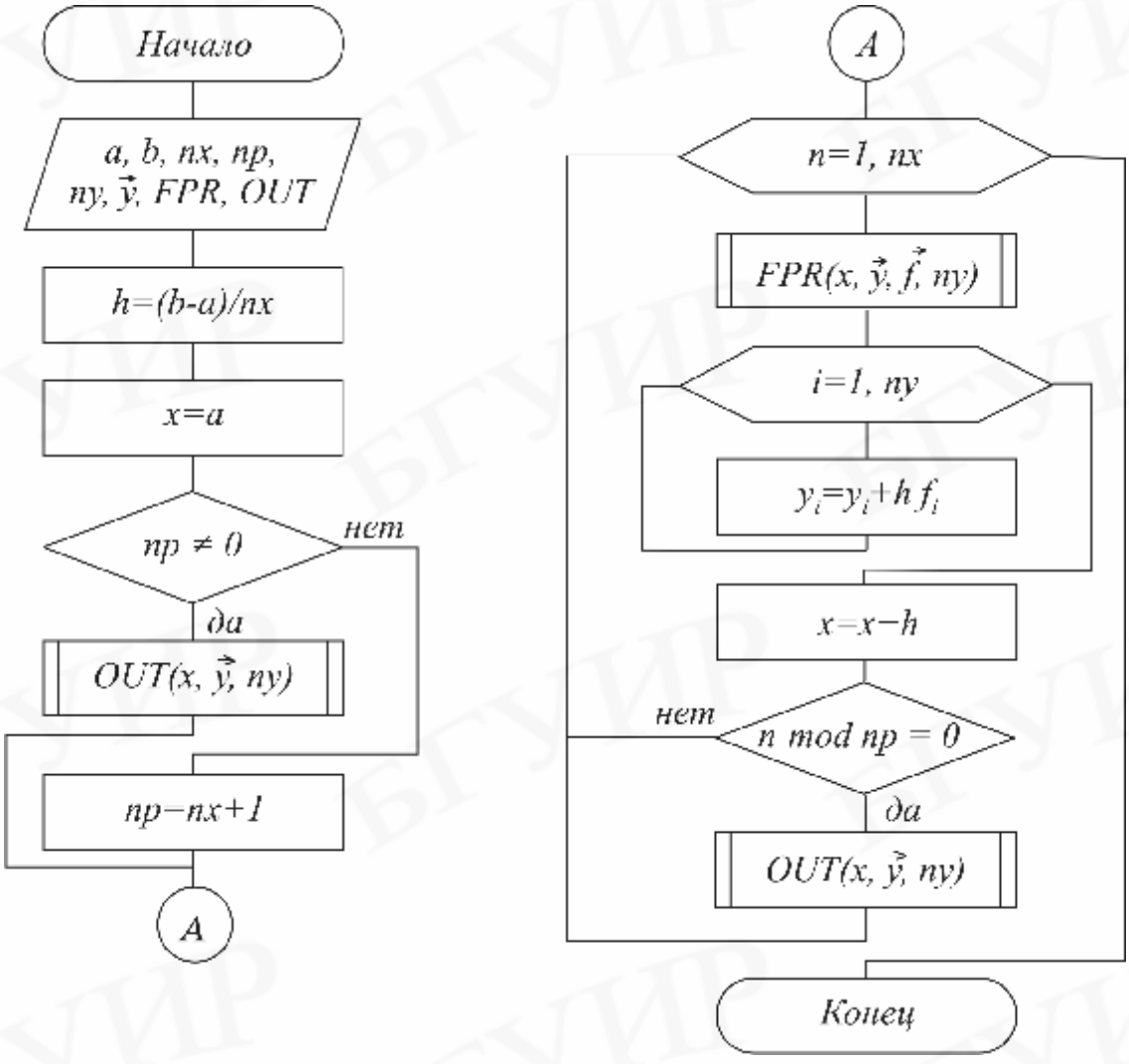


Рис. 7.1

М2. Неявная схема 1-го порядка

Используя в (7.5) формулу правых прямоугольников, получим

$$\frac{\mathbf{y}^{k+1} - \mathbf{y}^{k}}{h} = \mathbf{f}(x_{k+1}, \mathbf{y}^{k+1}). \qquad (7.9)$$

Эта схема неразрешима явно относительно $\mathbf{y}^{k+1}$, поэтому для получения $\mathbf{y}^{k+1}$ требуется использовать итерационную процедуру решения уравнения (7.9) (см. метод простой итерации в подразд. 5.2):

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

За начальное приближение берется значение $\mathbf{y}^{k+1,0} = \mathbf{y}^{k}$ с предыдущего шага. Обычно, если h выбрано удачно, достаточно сделать 2–3 итерации для достижения заданной погрешности $\left\| y^{k+1,s} - y^{k+1,s-1} \right\| < \varepsilon$. Эффективность неявной схемы заключается в том, что у нее константа устойчивости C_0 значительно меньше, чем у явной схемы.

М3. Неявная схема 2-го порядка

Используя в (7.5) формулу трапеций, получим

$$\frac{\mathbf{y}^{k+1} - \mathbf{y}^{k}}{h} = \frac{\mathbf{f}(x,\ \mathbf{y}^{k}) + \mathbf{f}(x_{k+1},\ \mathbf{y}^{k+1})}{2}. \qquad (7.10)$$

Так как формула трапеций имеет третий порядок точности на интервале $[x_x,\ x_{k+1}]$, то погрешность аппроксимации $\psi(h)$ – второй.

Схема (7.10) не разрешена относительно $\mathbf{y}^{k+1}$, поэтому требуется итерационная процедура (см. М2):

$$\mathbf{y}^{k+1,s} = \mathbf{y}^{k} + \frac{h}{2}(\mathbf{f}(x_k,\ \mathbf{y}^{k}) + \mathbf{f}(x_{k+1}, \mathbf{y}^{k+1,s-1}),\quad s = 1,\ 2,\ \ldots,\ \mathbf{y}^{k+1,0} = \mathbf{y}^{k}.$$

Алгоритм представлен на рис. 7.2.

М4. Схема предиктор–корректор (Рунге–Кутта) 2-го порядка

Используя в (7.5) формулу средних, получим

$$\frac{\mathbf{y}^{k+1} - \mathbf{y}^{k}}{h} = \mathbf{f}(x_{k+1/2},\ \mathbf{y}^{k+1/2}). \qquad (7.11)$$

Уравнение разрешено явно относительно $\mathbf{y}^{k+1}$, однако в правой части присутствует неизвестное значение $\mathbf{y}^{k+1/2}$ в середине отрезка $[x_k,\ x_{k+1}]$. Для решения этого уравнения существует следующий способ. Вначале по явной схеме (7.8) рассчитывают $\mathbf{y}^{k+1/2}$ (предиктор):

$$\mathbf{y}^{k+1/2} = \mathbf{y}^{k} + \frac{h}{2}\mathbf{f}(x_k,\ \mathbf{y}^{k}).$$

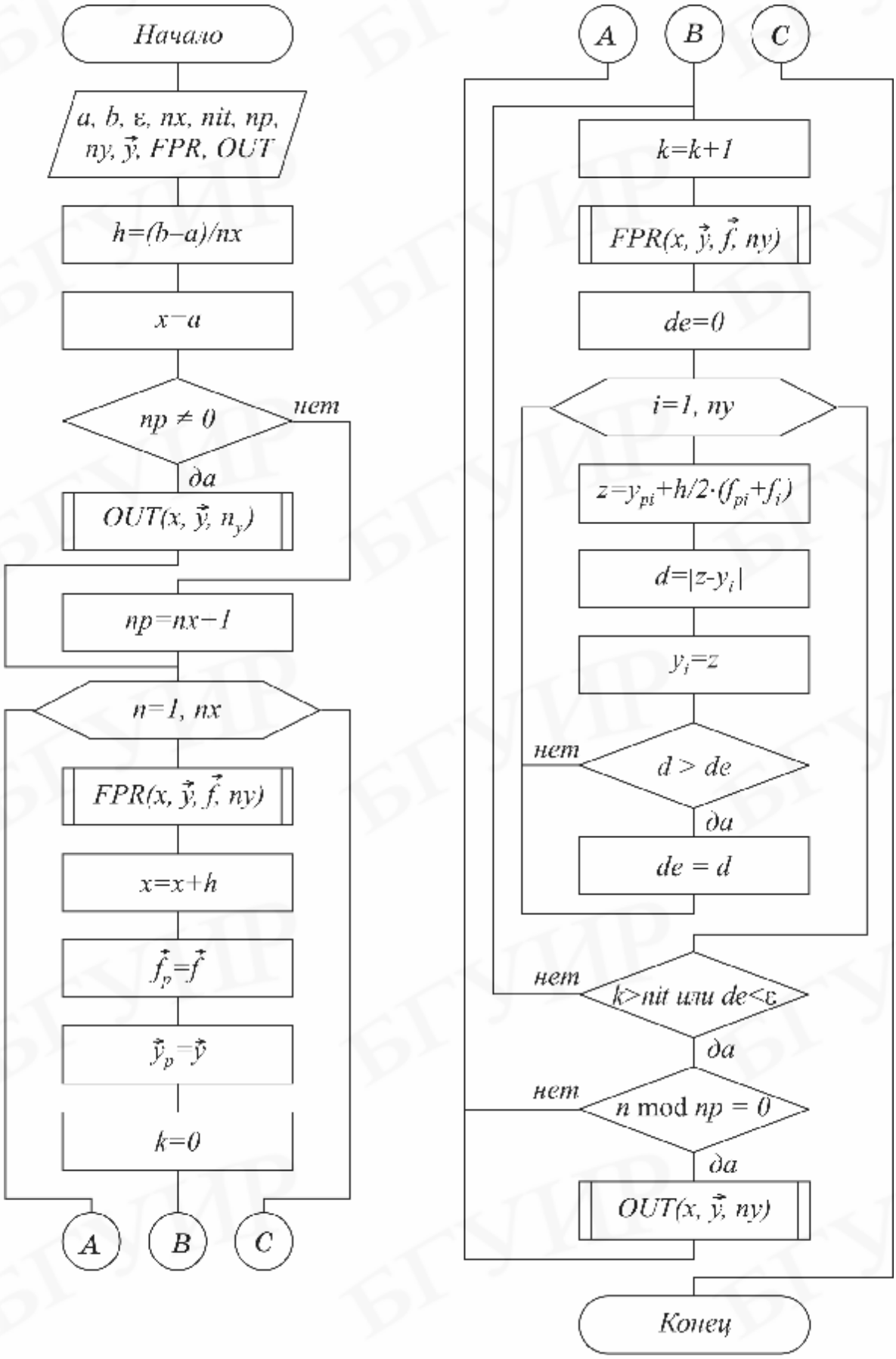

Начало
a, b, ε, nx, nit, np, ny, ŷ, FPR, OUT
h=(b–a)/nx
x=a
np ≠ 0
нет
да
OUT(x, ŷ, n_y)
np=nx–1
n=1, nx
FPR(x, ŷ, f̂, ny)
x=x+h
f̂_p=f̂
ŷ_p=ŷ
k=0
A
B
C
A
B
C
k=k+1
FPR(x, ŷ, f̂, ny)
de=0
i=1, ny
z=y_pi+h/2·(f_pi+f_i)
d=|z-y_i|
y_i=z
нет
d > de
да
de = d
нет
k>nit или de<ε
да
нет
n mod np = 0
да
OUT(x, ŷ, ny)
Конец

Рис. 7.2

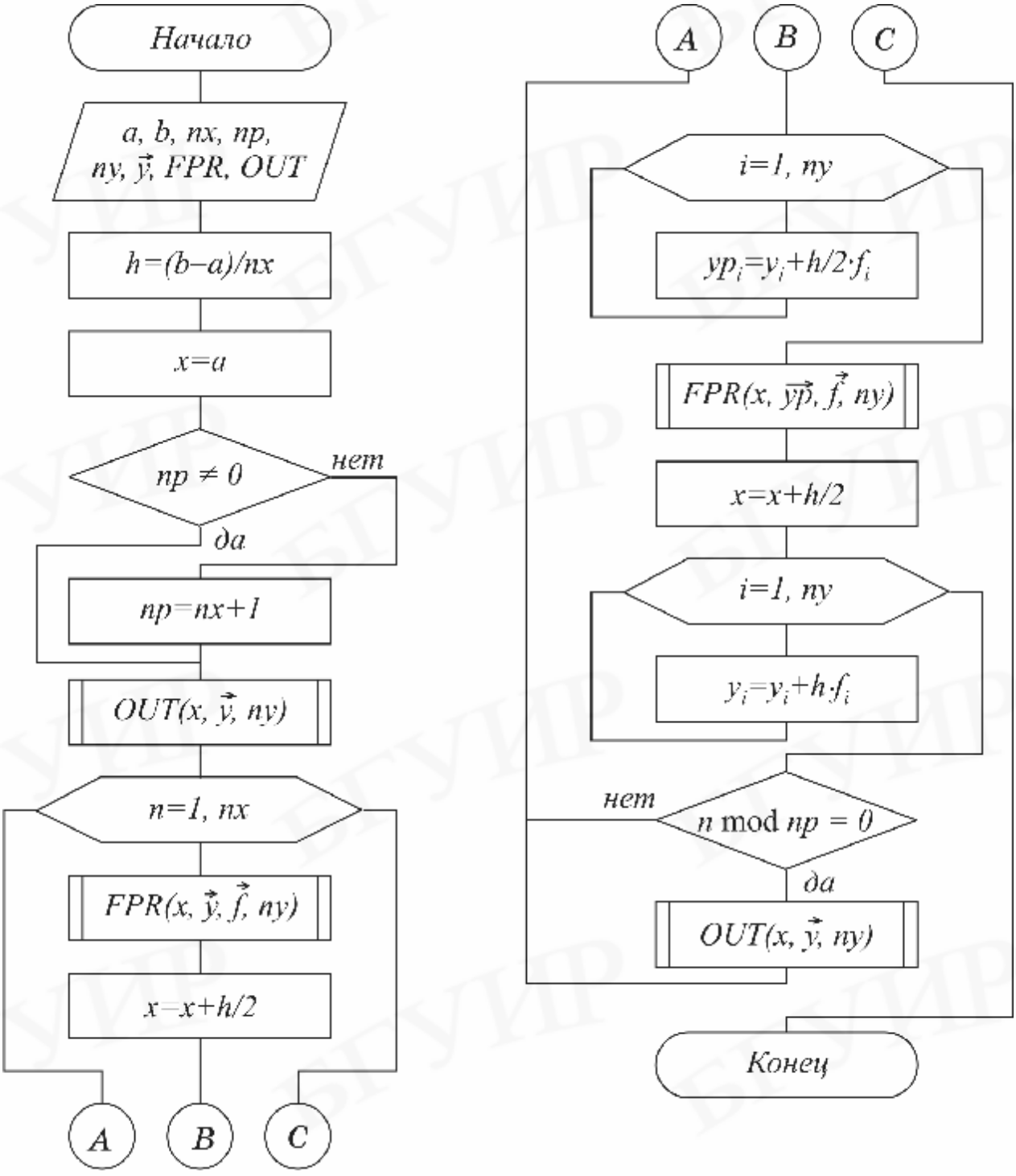


Рис. 7.3

После этого рассчитывают $\mathbf{r}_{y^{k+1}}$ по (7.11) (корректор). В результате схема оказывается явной и имеет второй порядок. Заметим, что схема получается из схемы М3, если в ней выполнять только две итерации (s = 1, 2).

Алгоритм представлен на рис.7.3.

М5. Схема Рунге–Кутта 4-го порядка

Используя в (7.5) формулу Симпсона, получим

$$\frac{\mathbf{r}_{k+1} - \mathbf{r}_k}{h} = \frac{1}{6}\,[\,\mathbf{r}f(x_k,\ \mathbf{r}_k) + 4\,\mathbf{r}f(x_{k+1/2},\ \mathbf{r}_{k+1/2}) + \mathbf{r}f(x_{k+1},\ \mathbf{r}_{k+1})]. \qquad (7.12)$$

Так как формула Симпсона имеет пятый порядок точности, то погрешность аппроксимации данной схемы имеет четвертый порядок.

Можно по-разному реализовать расчет неявного по $\mathbf{r}_{k+1}$ уравнения (7.12), однако наибольшее распространение получил следующий способ. Делают предиктор вида

$$\mathbf{r}_{k+1/2,1} = \mathbf{r}_k + \frac{h}{2}\,\mathbf{r}f(x_k,\ \mathbf{r}_k),$$

$$\mathbf{r}_{k+1/2,2} = \mathbf{r}_k + \frac{h}{2}\,\mathbf{r}f(x_{k+1/2},\ \mathbf{r}_{k+1/2,1}),$$

$$\mathbf{r}_{k+1,1} = \mathbf{r}_k + h\,\mathbf{r}f(x_{k+1/2},\ \mathbf{r}_{k+1/2,2}),$$

затем корректор по формуле

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \frac{h}{6}[\,f(x_k,\ \mathbf{r}_k) + 2\,\mathbf{r}f(x_{k+1/2,}\ \mathbf{r}_{k+1/2,1}) +$$

$$+2\,\mathbf{r}f(x_{k+1/2,}\ \mathbf{r}_{k+1/2,2}) + \mathbf{r}f(x_{k+1},\mathbf{r}_{k+1,1})].$$

7.4. МНОГОШАГОВЫЕ СХЕМЫ АДАМСА

При построении всех предыдущих схем для вычисления интеграла в правой части (7.4) использовались лишь точки в диапазоне одного шага $[x_k,\ x_{k+1}]$. Поэтому при реализации таких схем для вычисления следующего значения $\mathbf{r}_{k+1}$ требуется знать только одно предыдущее значение $\mathbf{r}_k$, т.е. рекуррентная последовательность получается первого порядка. Такие схемы называют *одношаговыми*. Мы, однако, видели, что для повышения точности при переходе от x_k к x_{k+1} приходилось использовать и значения функции F внутри интервала $[x_{k+1/2},\ \mathbf{r}_{k+1/2}]$. Схемы, в которых это используется (М4,М5,...), называют *схемами с дробными шагами*. В этих схемах повышение точности достигается за счет дополнительных затрат на вычисление функции $F(x)$ в промежуточных точках интервала $[x_k,\ x_{k+1}]$.

Идея методов Адамса заключается в том, чтобы для повышения точности использовать уже вычисленные на предыдущих шагах значения $\mathbf{r}_k,\ \mathbf{r}_{k-1},\ \mathbf{r}_{k-2}, \ldots$.

Заменим в (7.4) $F(x)$ интерполяционным многочленом Ньютона вида

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

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

После интегрирования на интервале $[x_k,\ x_{k+1}]$ получим ***явную экстраполяционную схему*** Адамса. (***Экстраполяцией*** называется получение значений интерполяционного многочлена в точках x, выходящих за крайние узлы сетки). В нашем случае интегрирование производится на интервале $[x_k,\ x_{k+1}]$, а полином строится по узлам $x_k,\ x_{k-1},\ x_{k-2}$.

Порядок аппроксимации схемы в этом случае определяется количеством использованных при построении полинома узлов (например если используются $x_k,\ x_{k-1},$ то схема второго порядка).

Если в (7.4) $F(x)$ заменить многочленом Ньютона вида

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

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

то после интегрирования получим ***неявную интерполяционную схему Адамса***. Заметим, что неявная интерполяционная схема второго порядка совпадает со схемой М3.

М6. Явная экстраполяционная схема Адамса 2-го порядка

$$\frac{\mathbf{r}_{y^{k+1}} - \mathbf{r}_{y^k}}{h} = 1.5 \cdot \mathbf{r}_f(x_k,\ \mathbf{r}_{y^k}) - 0.5 \cdot \mathbf{r}_f(x_{k-1},\ \mathbf{r}_{y^{k-1}}). \qquad (7.13)$$

Схема двухшаговая, поэтому для начала расчетов необходимо найти $\mathbf{r}_{y^1}$ по методу М4, после чего $\mathbf{r}_{y^2},\ \mathbf{r}_{y^3}$, ... вычислять по (7.13).

М7. Явная экстраполяционная схема Адамса 3-го порядка

$$\frac{\mathbf{r}_{y^{k+1}} - \mathbf{r}_{y^k}}{h} = \frac{23}{12}\mathbf{r}_f(x_k,\ \mathbf{r}_{y^k}) - \frac{16}{12}\mathbf{r}_f(x_{k-1},\ \mathbf{r}_{y^{k-1}}) + \frac{5}{12}\mathbf{r}_f(x_{k-2}, \mathbf{r}_{y^{k-2}}). \qquad (7.14)$$

Схема трехшаговая, поэтому для начала расчетов необходимо найти $\mathbf{r}_{y^1},\ \mathbf{r}_{y^2}$ по методу М5, после чего $\mathbf{r}_{y^3},\ \mathbf{r}_{y^4}$, ... вычислить по (7.14).

М8. Неявная схема Адамса 3-го порядка

$$\frac{\mathbf{r}_{y^{k+1}} - \mathbf{r}_{y^k}}{h} = \frac{5}{12}\mathbf{r}_f(x_{k+1},\ \mathbf{r}_{y^{k+1}}) + \frac{8}{12}\mathbf{r}_f(x_k,\ \mathbf{r}_{y^k}) - \frac{1}{12}\mathbf{r}_f(x_{k-1},\ \mathbf{r}_{y^{k-1}}). \qquad (7.15)$$

Так как схема двухшаговая, то для начала расчетов необходимо найти $\mathbf{r}_{y^1}$ по методу М5, после чего $\mathbf{r}_{y^2},\ \mathbf{r}_{y^3}$, ... вычислить по (7.15).

Для нахождения $\mathbf{r}_{y^{k+1}}$ требуется использовать метод простой итерации:

$$\mathbf{r}_{y^{k+1,s}} = \mathbf{r}_{y^k} + h\left[\frac{5}{12}\mathbf{r}_f(x_{k+1},\ \mathbf{r}_{y^{k+1,s-1}}) + \frac{8}{12}\mathbf{r}_f(x_k,\ \mathbf{r}_{y^k}) - \frac{1}{12}\mathbf{r}_f(x_{k-1},\ \mathbf{r}_{y^{k-1}})\right].$$

Значение $\mathbf{r}_{y^{k+1,0}}$ следует рассчитать по формуле (7.13):

$$\vec{y}^{k+1,0} = \vec{y}^{k} + h \cdot \left[1.5 \cdot \vec{f}(x_k,\ \vec{y}^{k}) - 0.5 \cdot \vec{f}(x_{k-1},\ \vec{y}^{k-1}) \right].$$

Чаще всего бывает достаточно одной итерации. Если при этом разность $\left| \vec{y}^{k+1,0} - \vec{y}^{k+1,1} \right|$ оказывается большой, то следует уменьшить h. Схема алгоритма представлена на рис. 7.4.

7.5. ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ АЛГОРИТМОВ

Для решения задачи Коши составляется стандартная подпрограмма, которая помещается в библиотеку стандартных программ. Передача всех необходимых данных из основной программы пользователя в подпрограмму организуется через список формальных параметров, который включает:

a, b – начало и конец области интегрирования;

nx – количество шагов сетки;

np – параметр, указывающий, через какое количество шагов организовывать вывод данных;

ny – количество уравнений;

$\vec{y}$ – массив, в который сначала при обращении к подпрограмме помещается вектор начальных значений $\vec{y}(0) = \vec{u}^0$, а по окончании работы в нем помещается решение на конце интервала $\vec{y}(b)$.

Для итерационных методов дополнительно вводятся:

nit – максимально допустимое количество итераций;

ε – точность (погрешность, до которой выполняются итерации).

В подпрограмме $FPR(x,\ \vec{y},\ \vec{F},\ ny)$ для заданных x и $\vec{y}$ вычисляется вектор $\vec{f} = \vec{f}(x,\ \vec{y})$, в подпрограмме $OUT(x,\ \vec{y},\ ny)$ осуществляется вывод данных.

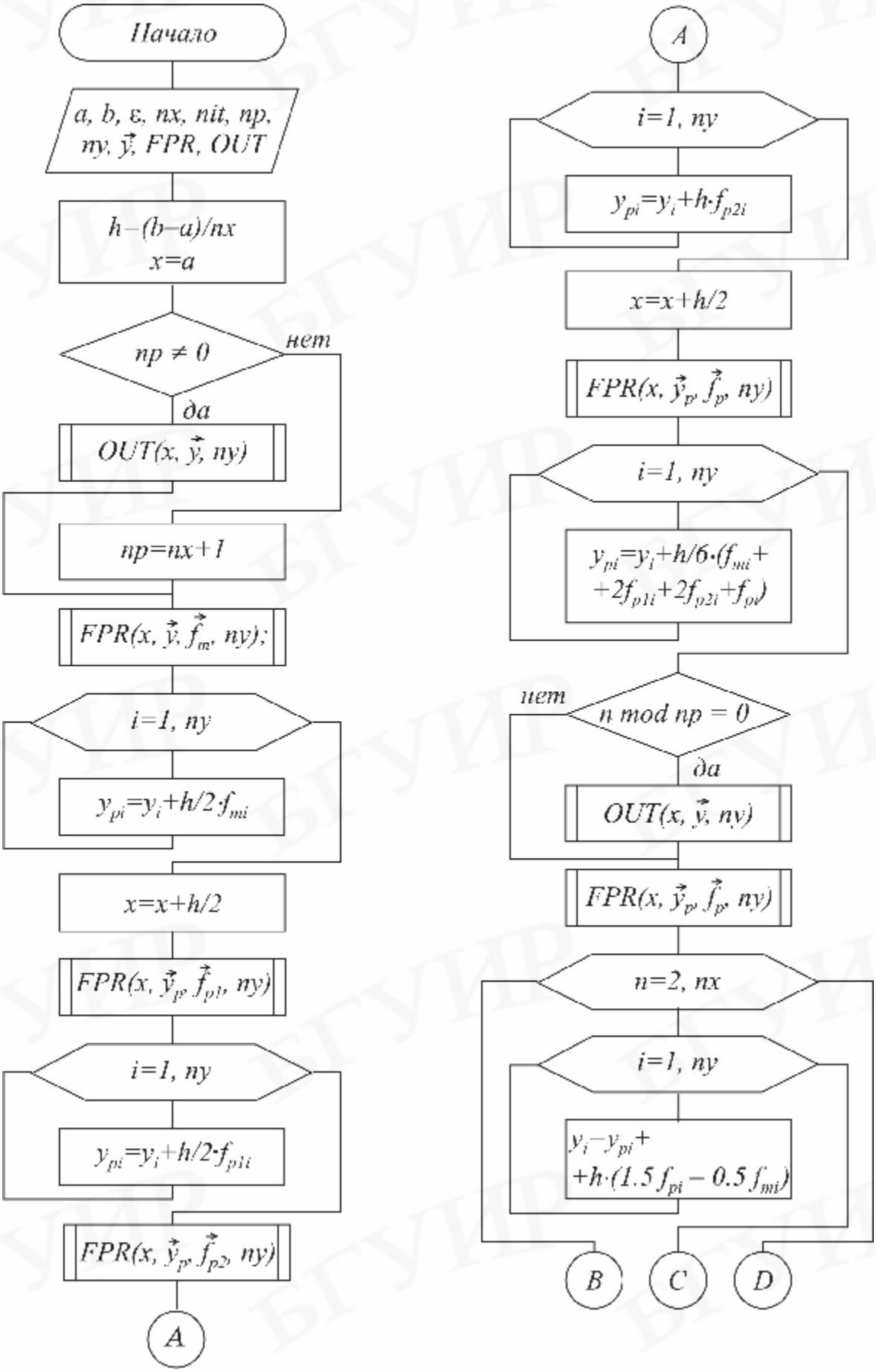


Рис. 7.4. (окончание см. на с . 76)

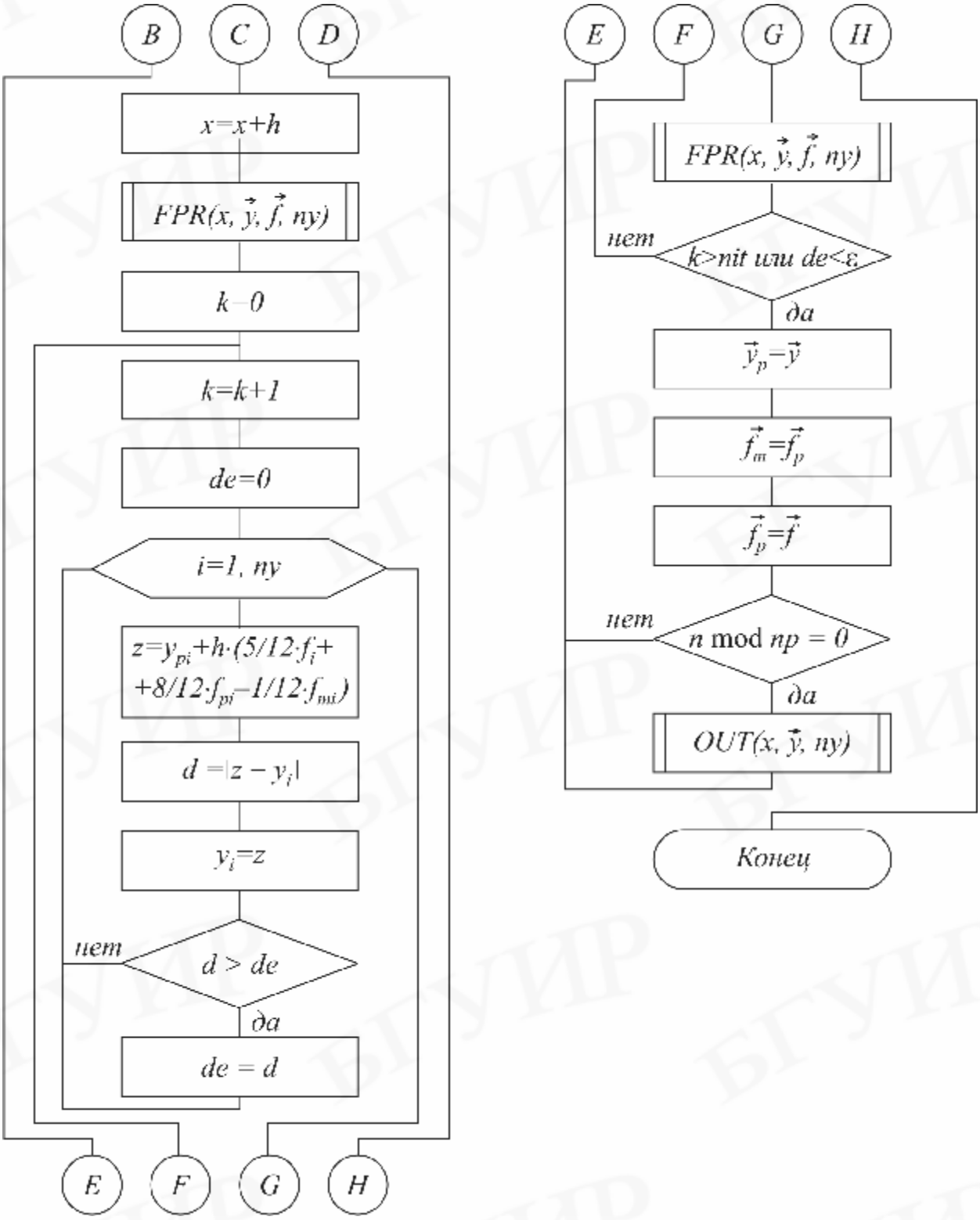


Рис. 7.4. Окончание
(начало см. на с . 75)

7.6. ВАРИАНТЫ ЗАДАНИЙ

Составить отдельную подпрограмму, оформленную в виде модуля, для решения задачи Коши в соответствии со схемой для своего варианта (табл. 7.1).

С помощью этой подпрограммы решить задачу для системы двух уравнений в соответствии с вариантом (см. табл. 7.1):

$$\frac{du_1}{dx} = f_1(x, u_1, u_2),$$

$$\frac{du_2}{dx} = f_2(x, u_1, u_2),$$

$$a \le x \le b,$$

$$u_1(a) = u_1^0,$$

$$u_2(a) = u_2^0$$

Точное решение этой задачи при $u_1^0 = 2a$, $u_2^0 = e^a$ одинаково для всех вариантов и имеет вид $u_1 = 2x$, $u_2 = e^x$.

Процедура FPR для варианта 15 имеет вид

```
Procedure FPR(x : real; var y, F : mas; ny : integer);
    begin
    F[1] := (y[1]*exp(x))/(y[2]*x)
    F[2] := 2*x/y[1] + y[2]-1;
    end;
```

Вариант процедуры OUT, обеспечивающей вывод таблицы значений приближенного и точного решения и погрешностей (d_1, d_2), имеет вид

```
Procedure OUT(x:real; var y:mas; ny:integer);
    begin
    Writeln('x= ',x:6:4,'  y1=',y[1]:8:4,'  u1=',2*x:8:4,' d1=', u1-y1:8:4,
                '  y2=',y[2]:8:4,'  u2=',exp(x):8:4,'  d2=',u2-y2:8:4);
    end;
```

Начальные условия следует задать y[1]:=2*a; y[2]:=exp(a);.

Расчеты произвести для последовательности сгущающихся сеток, $h = h_1 = (b-a)/10$, $h = h_1/2$, $h = h_1/4$, ..., и сравнивая полученное решение с точным решением, добиваться того, чтобы погрешность на втором конце *(x=b)* была не больше 0.0001.

Построить графики полученных решений для $h = h_1$, сравнить их с точным решением.

Таблица 7.1

N вар.	$f_1(x, u_1, u_2)$	$f_2(x, u_1, u_2)$	*[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, 3]	2	e^1	M1
2	$u_1 + u_2 - 2x - e^x + 2$	$u_1 + u_2 - 2x$	[1, 2]	2	e^1	M2
3	$u_1 + 2u_2 / e^x - 2x$	$u_1 /(2x) - e^x / u_2 + u_2$	[2, 3]	4	e^2	M3
4	$(u_1 \cdot e^x)/(x \cdot u_2)$	$2u_1 + u_2 - 4x$	[1, 4]	2	e^1	M4
5	$2u_1 + (u_2 + e^x)/e^x - 4x$	$2x \cdot u_2 / u_1$	[2, 4]	4	e^2	M5
6	$u_1 \cdot u_2 /(e^x \cdot x)$	$2x / u_1 + 2u_2 - e^x - 1$	[1, 3]	2	e^1	M6
7	$u_1 / 2x + u_2 / e^x + 1$	$u_1 \cdot u_2 / 2x$	[2, 3]	4	e^2	M7
8	$u_1 / x + u_2 - e^x$	$2x / u_1 + u_2^2 / e^x - 1$	[1, 4]	2	e^1	M8
9	$u_1 + 2e^x / u_2 - 2x$	$u_1^2 / x^2 + u_2 - 4$	[1, 2]	2	e^1	M7
10	$4x / u_1 - u_2 + e^x$	$u_1 / 2x - u_2 / e^x + e^x$	[2, 4]	4	e^2	M6
11	$2x / u_1 + u_2 / e^x$	$u_1 \cdot e^{2x} /(u_2 \cdot 2x)$	[3, 4]	6	e^3	M5
12	$u_1 \cdot u_2 /(2e^x) - x + 2$	$u_1 + 2u_2 - 2x - e^x$	[1, 3]	2	e^1	M4
13	$u_1^2 + u_2 - 4x^2 - e^x + 2$	$u_1 \cdot e^x / u_2 + u_2 - 2x$	[1, 2]	2	e^1	M3
14	$u_1^2 / 2x^2 - u_2 + e^x$	$u_1 \cdot e^x / 2x + u_2 / e^x - 1$	[2, 4]	4	e^2	M2
15	$u_1 \cdot e^x /(x \cdot u_2)$	$2x / u_1 + u_2 - 1$	[3, 4]	6	e^3	M1

7.7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как формулируется задача Коши для системы из *n* уравнений?
2. В чем суть метода сеток?
3. Что такое конечно-разностная схема, погрешность аппроксимации, устойчивость?
4. Сформулируйте содержание основной теоремы метода сеток.
5. Назовите известные вам схемы решения дифференциального уравнения.
6. В чем отличие методов Адамса от методов Рунге–Кутта?

ЛИТЕРАТУРА

1. Калиткин, Н. Н. Численные методы / Н. Н. Калиткин. – М.: Наука, 1978. – 512 с.

2. Бахвалов, Н. С. Численные методы / Н. С. Бахвалов. – М.: Наука, 1975. – 632 с.

3. Егоров, А. А. Вычислительные алгоритмы линейной алгебры : учеб. пособие / А. А. Егоров. – Минск : БГУ, 2005. – 190 с.

4. Волков, Е. А. Численные методы / Е. А. Волков. – М. : Наука, 1982. – 255 с.

5. Васильков, Ю. В. Компьютерные технологии вычислений в математическом моделировании / Ю. В. Васильков, Н. Н. Василькова. – М. : Финансы и статистика, 2001. – 250 с.

6. Крылов, В. И. Вычислительные методы высшей математики. Т.1 / В. И. Крылов и др. – Минск : Выш. шк., 1972. – 584 с.

7. Крылов, В. И. Вычислительные методы высшей математики. Т.2. / В. И. Крылов и др. – Минск : Выш. шк., 1975. – 672 с.

8. Форсайт, Дж. Машинные методы математических вычислений / Дж. Форсайт и др. – М. : Мир, 1980. – 280 с.

9. Шуп, Т. Решение инженерных задач на ЭВМ / Т. Шуп. – М.: Мир, 1982. – 238 с.

10. Копченова, Н. В., Марон, И. А. Вычислительная математика в примерах и задачах Н. В. Копченова, И. А. Марон. – М.: Наука, 1972. – 368 с.

11. Самарский, А. А. Введение в численные методы / А. А Самарский. – М.: Наука, 1982.– 272 с.

12. Березин, И. С. Методы вычислений. Т.1 / И. С. Березин, Н. П. Жидков. – М. : Физматгиз, 1962. – 464 с.

13. Березин, И. С. Методы вычислений. Т.2 / И. С. Березин, Н. П. Жидков. – М. : Физматгиз, 1970. – 620 с.

14. Банди, Б. Методы оптимизации. Вводный курс / Б. Банди – М. : Мир, 1989. – 277 с.

15. Мак–Кракен, Д. Численные методы и программирование на Фортране / Д. Мак–Кракен, У. Дорн. – М. : Мир, 1972. – 586 с.

16. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. – Томск : МП «РАСКО», 1991. – 272 с.

17. Методические указания к индивидуальным заданиям по курсу «Вычислительная математика». Раздел «Основы численных методов» для студ. всех спец. / Сост. А. К. Синицын, В. А. Новиков, В. В. Соловьев. – Минск : МРТИ, 1994.

18. Синицын, А. К. Практикум по курсу «Алгоритмы вычислительной математики»: учеб. пособие для студ. 1–2 го курсов всех спец. / А. К Синицын. – Минск : БГУИР, 1996.

19. Синицын, А. К. Алгоритмы вычислительной математики : лаб. практикум по курсу «Программирование» для студ. 1–2 го курсов всех спец / А. К Синицын, А. А. Навроцкий. – Минск : БГУИР, 2002.

Св.план 2007, поз. 52

Учебное издание

Синицын Анатолий Константинович
Навроцкий Анатолий Александрович

АЛГОРИТМЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

Учебно-методическое пособие по курсу
«Основы алгоритмизации и программирования»

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

Подписано в печать 15.03.2007. Формат 60x84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 4,77.
Уч.-изд. л. 4,2. Тираж 1000 экз. Заказ 658.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6