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

В. С. Муха

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ И КОМПЬЮТЕРНАЯ АЛГЕБРА

2-е издание
исправленное и дополненное

Рекомендовано УМО вузов Республики Беларусь
по образованию в области информатики и радиоэлектроники
в качестве учебно-методического пособия для студентов учреждений,
обеспечивающих получение высшего образования по специальности
«Автоматизированные системы обработки информации»

Минск БГУИР 2010

УДК 519.6 (075.8)
ББК 22.19я73
М92

Р е ц е н з е н т ы:

кафедра «Высшая математика №1» Белорусского национального технического университета (заведующий кафедрой, доктор технических наук, профессор Н. А. Микулик);

профессор кафедры математического моделирования и анализа данных Белорусского государственного университета, доктор физико-математических наук, профессор Е. Е. Жук;

профессор кафедры высшей математики Белорусского государственного университета информатики и радиоэлектроники, кандидат физико-математических наук А. А. Карпук

Муха, В. С.

М92 Вычислительные методы и компьютерная алгебра : учеб.-метод. пособие / В. С. Муха. – 2-е изд., испр. и доп. – Минск : БГУИР, 2010. – 148 с. : ил.
ISBN 978-985-488-522-3

Излагаются основы теории погрешностей вычислений и вычислительных методов: решение систем линейных алгебраических уравнений, интерполирование, интегрирование, решение нелинейных уравнений, решение обыкновенных дифференциальных уравнений и систем обыкновенных дифференциальных уравнений. Рассматриваются такие вопросы, как вычисление полиномов, решение систем нелинейных уравнений, трехдиагональных систем линейных алгебраических уравнений и интерполирование сплайнами. Описываются основные функции символьных вычислений в системе Matlab.

Пособие предназначено для студентов специальности «Автоматизированные системы обработки информации», других специальностей технических вузов, а также преподавателей.

УДК 519.6 (075.8)
ББК 22.19я73

ISBN 978-985-488-522-3

ПРЕДИСЛОВИЕ

Вычислительные (или численные) методы – это методы решения математических задач в численном виде. Отличительной чертой численных методов является то, что исходные данные в задаче задаются в виде числа или набора чисел, и решение получается также в виде числа или набора чисел. В отличие от численных методов компьютерная алгебра занимается разработкой и реализацией *аналитических* методов решения математических задач на ЭВМ. В компьютерной алгебре предполагается, что исходные данные сформулированы в аналитическом (символьном) виде, и результаты решения также получаются в символьном виде.

Как вычислительные методы, так и компьютерная алгебра являются важными составляющими в системе подготовки инженеров технических специальностей высших учебных заведений. Особенно они важны для инженеров специальности «Автоматизированные системы обработки информации», так как рассматривают методы обработки математической информации.

В пособии излагаются основы вычислительных методов: решение систем линейных алгебраических уравнений, интерполирование, численное интегрирование, численное решение нелинейных уравнений, численное решение обыкновенных дифференциальных уравнений. Излагаются также основы теории погрешностей вычислений и основы символьных вычислений в системе Matlab. Кроме того, в разделе «Дополнение» рассматриваются некоторые специфические вопросы, такие, как вычисление корней полиномов, решение систем нелинейных уравнений методом Ньютона, решение трехдиагональных систем линейных алгебраических уравнений методом прогонки, интерполирование функций сплайнами.

Пособие рассчитано на студентов младших курсов высших технических учебных заведений, что определило выбор простых математических средств для изложения материала.

ПРЕДИСЛОВИЕ КО ВТОРОМУ ИЗДАНИЮ

Второе издание отличается от первого включением ряда новых вопросов и более детальным изложением некоторых существующих. В частности, дополнительно рассматриваются методы LU-разложения и квадратного корня решения систем линейных алгебраических уравнений (СЛАУ), метод хорд решения скалярных нелинейных уравнений, вопросы анализа вычислительной сложности метода Гаусса решения СЛАУ и задачи интерполирования функций. Более детально рассматриваются полиномы Чебышева 1-го рода, ортогональные полиномы скалярной переменной, задача выбора наилучших узлов интерполирования. Добавлены схема алгоритма Гаусса решения СЛАУ и схема алгоритма расчета значения интерполяционного полинома Лагранжа в одной точке, полезные как для программной реализации этих методов, так и для анализа их вычислительной сложности. Исправлены замеченные опечатки.

1. МАТЕМАТИЧЕСКИЕ МОДЕЛИ. ЧИСЛЕННЫЕ МЕТОДЫ. ПОГРЕШНОСТИ ВЫЧИСЛЕНИЙ

1.1. Математические модели и моделирование

Построение любой технической системы или запуск любого производственного процесса начинается с их проектирования и исследования. На первой стадии проектирования обычно строится модель системы. Использование при проектировании технической системы ее модели называется моделированием. Можно выделить *физические* и *математические* модели реальных технических систем. *Физическая модель* системы воспроизводит реальную техническую систему, но в уменьшенных размерах. Физическое моделирование позволяет получить ответы на поставленные вопросы, но слишком медленным и дорогим способом. Подчас построить физическую модель не проще, чем реальную систему. Другой способ – построение математической модели технической системы. *Математическая модель* представляет собой математические соотношения, описывающие техническую систему. Это может быть функция $y = f(x)$, выражающая зависимость выходной переменной системы y от входной переменной x, или дифференциальное уравнение относительно y и x. Математическая модель должна правильно отражать важнейшие связи между переменными системы, т.е. должна быть адекватной реальной системе. В противном случае она будет бесполезной, а то и вредной, поскольку выводы на ее основе не будут соответствовать тому, что происходит в действительности. В то же время математическая модель должна быть достаточно простой, с тем чтобы ее исследование было не слишком трудоемким. Следующая стадия проектирования – это анализ или исследование полученной модели. Анализ физической модели состоит в загрузке построенной установки сырьем, запуске ее в работу и исследовании полученной продукции. Анализ математической модели заключается в получении

общих и частных аналитических решений сформулированной математической задачи и их интерпретации. Аналитические решения можно получить для наиболее простых (наиболее грубых) моделей. Для более точных и более сложных моделей аналитические решения удается получить сравнительно редко. В этих случаях на помощь приходят численные методы, позволяющие получить частные численные решения практически любых задач. Получение частных численных решений сформулированной задачи на основе аналитических решений или с помощью численных методов иногда называют *имитационным моделированием* реального процесса.

В данном пособии мы будем иметь дело с *математическими моделями* технических систем и процессов, а также *численными методами* решения задач по их исследованию на ЭВМ.

1.2. Этапы численного решения задач на ЭВМ

Процесс численного решения задачи на ЭВМ состоит из следующих этапов:

1) постановка задачи;
2) разработка метода и алгоритма решения задачи;
3) написание компьютерной программы;
4) отладка программы;
5) проведение расчетов и анализ результатов.

Рассмотрим кратко содержание каждого этапа.

Постановка задачи заключается в построении математических зависимостей, адекватно описывающих техническую систему, т.е. в построении математической модели системы. Например, техническая система может быть описана системой линейных или нелинейных уравнений, дифференциальным уравнением и т.д.

Следующим этапом является *разработка метода и алгоритма решения задачи*. Дело в том, что многие задачи не могут быть решены в аналитическом

виде. Поэтому существует необходимость разработки методов решения задачи в арифметической форме, требующей выполнения лишь арифметических операций сложения, вычитания, умножения, деления. Например, вычисление интеграла сводится к вычислению суммы, вычисление производной – к вычислению приращений функции и аргумента. Такие методы называются численными. Иногда разработанный численный метод представляют в виде алгоритма, т.е. в виде упорядоченной последовательности операций, позволяющих из исходных данных получить конечный результат.

Полученный алгоритм записывается на одном из алгоритмических языков *в виде программы* для решения на ЭВМ. Далее осуществляется *отладка программы*, во время которой выявляются допущенные в ней синтаксические ошибки, и проверяется правильность решения задачи на примерах с известным решением. После того как программа отлажена и есть уверенность, что она дает правильные результаты, *выполняется большой объем расчетов по исследованию составленной модели.*

1.3. Виды погрешностей решения задач

Любая задача не может быть решена абсолютно точно. Всегда будет существовать погрешность ее решения. Существуют 4 источника погрешности результата при решении любой задачи:

1) погрешность математической модели;
2) погрешность исходных данных;
3) погрешность численного метода;
4) погрешность округления.

Погрешность математической модели зависит от допущений при получении математической модели физического процесса. Например, чаще всего для простоты пытаются построить линейную модель. Для этого реальную зависимость заменяют линейной частью ряда Тейлора. При этом допускается опреде-

ленная погрешность. Анализ погрешности математической модели относится к проблеме адекватности математической модели. В рамках данного пособия этот вопрос не рассматривается, т.е. всегда предполагается, что модель адекватна, и погрешность математической модели отсутствует.

Погрешность исходных данных появляется из-за неточности измерений, грубых недосмотров или невозможности представить исходные данные конечной десятичной дробью. Например, при использовании числа $1/3$ в качестве исходного данного погрешность появится, если мы представим это число конечной десятичной дробью $0{,}3$ или $0{,}33$. Часто случается, что дроби, которые являются конечными в одной системе счисления, становятся бесконечными в другой системе счисления. Например, конечная десятичная дробь $0{,}1$, будучи переведенной в двоичную систему счисления, становится бесконечной дробью $0{,}00011(0011)$. Из-за конечности разрядной сетки ЭВМ мы не сможем удержать все двоичные цифры данного числа. Ошибки в исходных данных вызывают погрешность результата вычислений независимо от того, каким методом эти вычисления производятся. Поэтому погрешность результата, вызванная погрешностью исходных данных, называется неустранимой. В ряде случаев неустранимая погрешность может становиться недопустимо большой. Это происходит при решении так называемых некорректно поставленных задач. Задача называется некорректно поставленной (по А. Н. Тихонову), если небольшие погрешности в исходных данных приводят к большим погрешностям решения. В рамках данного пособия предполагается, что исходные данные могут содержать погрешности, однако все решаемые задачи являются корректно поставленными.

Погрешность численного метода связана с тем, что точные операторы заменяются приближенными, например, интеграл заменяется суммой, функция – многочленом или строится бесконечный итерационный процесс, который обрывается после конечного числа итераций. Погрешность метода будем рассматривать для каждого численного метода. Численный метод необходимо подбирать таким образом, чтобы погрешность метода была в 2 – 5 раз меньше

погрешности исходных данных. Очень большая погрешность метода снижает точность ответа, а очень малая невыгодна, так как обычно требует значительного увеличения объема вычислений.

Погрешность округления связана с округлением чисел, участвующих в операциях, поскольку числа в памяти ЭВМ хранятся с удержанием лишь конечного числа их значащих цифр.

1.4. Погрешности арифметических операций

Погрешность, возникшая в определенном месте вычислений, распространяется дальше в процессе вычислений. При этом может произойти накопление погрешности до больших значений, что может поставить под сомнение конечный результат. В связи с этим весьма важным является вопрос анализа погрешностей вычислений и их распространения в вычислениях, который рассматривается ниже.

Пусть $\tilde{x}$ – приближенное значение числа x. *Абсолютной погрешностью* числа x называется разность его точного и приближенного значений:

$$\Delta_x = x - \tilde{x}.$$

Это значит, что точное значение числа равно сумме приближенного значения и абсолютной погрешности:

$$x = \tilde{x} + \Delta_x.$$

Относительная погрешность числа x определяется как отношение абсолютной погрешности к приближенному значению:

$$\delta_x = \frac{\Delta_x}{\tilde{x}}.$$

Поскольку в численных методах используются арифметические операции, то определим абсолютные и относительные погрешности операций сложения, вычитания, умножения и деления. Пусть $\tilde{x}$ и $\tilde{y}$ – приближенные значения чисел

x и y соответственно, а также Δ_x и Δ_y – их абсолютные погрешности, так что $x = \tilde{x} + \Delta_x$, $y = \tilde{y} + \Delta_y$. Найдем абсолютные погрешности арифметических операций.

Для сложения получим

$$z = x + y = \tilde{x} + \Delta_x + \tilde{y} + \Delta_y = (\tilde{x} + \tilde{y}) + (\Delta_x + \Delta_y) = \tilde{z} + \Delta_z .$$

Мы видим, что абсолютная погрешность суммы равна сумме абсолютных погрешностей слагаемых:

$$\Delta_{x+y} = \Delta_x + \Delta_y . \qquad (1.1)$$

Аналогично для вычитания получаем

$$\Delta_{x-y} = \Delta_x - \Delta_y . \qquad (1.2)$$

Для операции умножения будем иметь

$$z = xy = (\tilde{x} + \Delta_x)(\tilde{y} + \Delta_y) = \tilde{x}\tilde{y} + \tilde{x}\Delta_y + \tilde{y}\Delta_x + \Delta_x\Delta_y .$$

В этом выражении произведением погрешностей $\Delta_x\Delta_y$ по сравнению с другими слагаемыми можно пренебречь:

$$z = xy \approx \tilde{x}\tilde{y} + (\tilde{x}\Delta_y + \tilde{y}\Delta_x) .$$

Отсюда погрешность умножения

$$\Delta_{xy} \approx \tilde{x}\Delta_y + \tilde{y}\Delta_x . \qquad (1.3)$$

Абсолютную погрешность деления определим как линейную часть приращения функции двух переменных:

$$z = \frac{x}{y} .$$

Для этого представим функцию линейной частью ряда Тейлора в окрестности приближенных значений $\tilde{x}$, $\tilde{y}$. Получим

$$z = \frac{x}{y} \approx \frac{\tilde{x}}{\tilde{y}} + \frac{\partial z(\tilde{x},\tilde{y})}{\partial \tilde{x}}\Delta_x + \frac{\partial z(\tilde{x},\tilde{y})}{\partial \tilde{y}}\Delta_y = \frac{\tilde{x}}{\tilde{y}} + \frac{\Delta_x}{\tilde{y}} - \frac{\tilde{x}\Delta_y}{\tilde{y}^2} = \tilde{z} + \Delta_z .$$

Следовательно, абсолютная погрешность деления

$$\Delta_{x/y} \approx \frac{\tilde{y}\Delta_x - \tilde{x}\Delta_y}{\tilde{y}^2}. \qquad (1.4)$$

Получим также выражения для относительных погрешностей арифметических операций.

Для сложения имеем

$$\delta_{x+y} = \frac{\Delta_{x+y}}{\tilde{x}+\tilde{y}} = \frac{\Delta_x + \Delta_y}{\tilde{x}+\tilde{y}} = \frac{\Delta_x}{\tilde{x}+\tilde{y}} + \frac{\Delta_y}{\tilde{x}+\tilde{y}} = \frac{\tilde{x}}{\tilde{x}+\tilde{y}}\frac{\Delta_x}{\tilde{x}} + \frac{\tilde{y}}{\tilde{x}+\tilde{y}}\frac{\Delta_y}{\tilde{y}},$$

следовательно,

$$\delta_{x+y} = \frac{\tilde{x}}{\tilde{x}+\tilde{y}}\delta_x + \frac{\tilde{y}}{\tilde{x}+\tilde{y}}\delta_y. \qquad (1.5)$$

Аналогично получаем для операции вычитания:

$$\delta_{x-y} = \frac{\tilde{x}}{\tilde{x}-\tilde{y}}\delta_x - \frac{\tilde{y}}{\tilde{x}-\tilde{y}}\delta_y. \qquad (1.6)$$

Для операции умножения имеем

$$\delta_{xy} \approx \frac{\Delta_{xy}}{\tilde{x}\tilde{y}} = \frac{\tilde{x}\Delta_y}{\tilde{x}\tilde{y}} + \frac{\tilde{y}\Delta_x}{\tilde{x}\tilde{y}} = \delta_x + \delta_y \qquad (1.7)$$

и для операции деления

$$\delta_{x/y} \approx \frac{\Delta_{x/y}}{\tilde{x}/\tilde{y}} = \frac{\frac{\Delta_x}{\tilde{y}} - \frac{\tilde{x}}{\tilde{y}^2}\Delta_y}{\frac{\tilde{x}}{\tilde{y}}} = \frac{\tilde{y}\Delta_x - \tilde{x}\Delta_y}{\tilde{x}\tilde{y}} = \delta_x - \delta_y. \qquad (1.8)$$

Таким образом, мы получили формулы (1.1) – (1.4) для абсолютных погрешностей и формулы (1.5) – (1.8) для относительных погрешностей арифметических операций. Знак минус в этих формулах не означает уменьшения погрешности, поскольку знак абсолютной или относительной погрешности может быть любым.

Часто нас интересует абсолютное значение погрешности. Для оценки абсолютных значений погрешности обычно используются неравенства. Приведем некоторые неравенства и равенства относительно абсолютных значений, известные из курса школьной математики:

$$|x+y| \le |x|+|y|,$$

$$|x-y| \le |x|+|y|,$$

$$|xy| = |x||y|,$$

$$\left|\frac{x}{y}\right| = \frac{|x|}{|y|}.$$

Если $|a| \le A$ и $|b| \le B$, то $|a+b| \le A+B$, $|ab| \le AB$.

С учетом этих выражений, в частности, получаем, что

$$|\Delta_{x\pm y}| \le |\Delta_x| + |\Delta_y|,$$

$$|\delta_{xy,x/y}| \le |\delta_x| + |\delta_y|.$$

Отметим, что мы нашли погрешности арифметических операций, считая, что эти операции выполняются абсолютно точно. Так как в каждой из них существует относительная погрешность округления δ_t, то ее нужно добавить отдельно к полученным относительным погрешностям арифметических операций.

1.5. Графы арифметических операций

Как уже указывалось, погрешность распространяется в процессе вычислений, переходя в конечном итоге в погрешность результата счета. Распространение погрешности в процессе вычислений можно проследить с помощью специального графа вычислительного процесса. Вообще *графом называется некоторое множество точек, соединенных каким-либо образом линиями друг с другом.* Точки называются вершинами, а линии – ребрами графа. Вершины, принадлежащие ребру, называются вершинами, инцидентными данному ребру. Ребро называется неориентированным, если порядок инцидентных ему вершин не учитывается, и ориентированным, если такой порядок учитывается. Ориентированные ребра называются дугами и обозначаются стрелками. Граф называется ориентированным, если все его ребра ориентированы, и неориентированным, если все его ребра не ориентированы. Граф вычислительного процесса –

это ориентированный граф. Вершины такого графа обозначаются кружками. Каждая вершина соответствует выполняемой операции или числу, участвующему в операции (операнду). Внутри кружка записывается выполняемая операция или операнд. Дуги графа указывают, какие операнды участвуют в операции и какой результат получается. Возле каждой дуги пишется коэффициент, на который умножается относительная погрешность операнда. Графы четырех арифметических операций приведены на рис. 1.1 – 1.4.

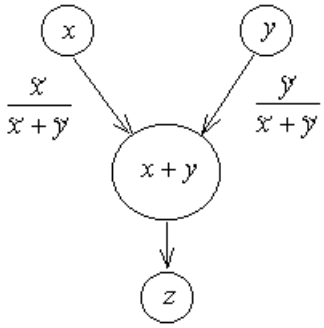

Рис. 1.1. Граф операции сложения

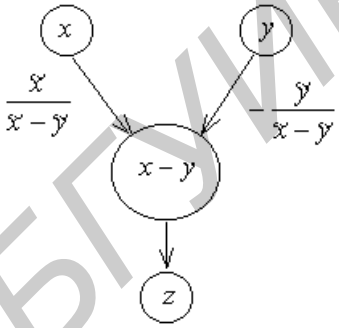

Рис. 1.2. Граф операции вычитания

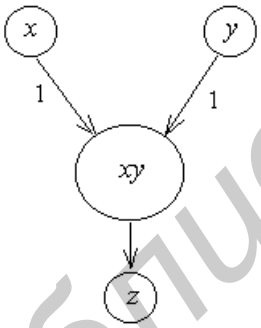

Рис. 1.3. Граф операции умножения

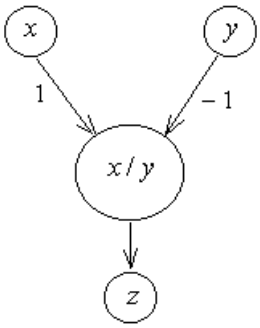

Рис. 1.4. Граф операции деления

Эти графы наглядно представляют формулы для относительных погрешностей арифметических операций и позволяют их воспроизвести. Предположим, что δ_x, δ_y – относительные погрешности чисел x, y, а r – относительная погрешность округления при выполнении операции. Тогда по графам, приведенным на рис. 1.1 – 1.4, можно получить следующие формулы для относительных погрешностей результата z:

$$\delta_{x+y} = \frac{\widetilde{x}}{\widetilde{x}+\widetilde{y}}\delta_x + \frac{\widetilde{y}}{\widetilde{x}+\widetilde{y}}\delta_y + r,$$

$$\delta_{x-y} = \frac{\widetilde{x}}{\widetilde{x}-\widetilde{y}}\delta_x - \frac{\widetilde{y}}{\widetilde{x}-\widetilde{y}}\delta_y + r,$$

$$\delta_{xy} \approx \delta_x + \delta_y + r,$$

$$\delta_{x/y} \approx \delta_x - \delta_y + r.$$

1.6. Распространение погрешностей в вычислениях

Построив на основе графов арифметических операций граф всего вычислительного процесса, можно подсчитать погрешность результата. Проиллюстрируем это на примерах, которые позволят сделать некоторые полезные выводы относительно погрешностей вычислений.

Пример 1.1. Сложение положительных чисел

Пусть требуется найти погрешность суммы четырех положительных чисел, т.е. погрешность величины

$$y = x_1 + x_2 + x_3 + x_4,$$

если исходные данные x_1, x_2, x_3, x_4 имеют относительные погрешности δ_{x_1}, δ_{x_2}, δ_{x_3}, δ_{x_4} и r_1, r_2, r_3 – относительные погрешности округления после каждой операции сложения. Предположим, что сложение выполняется последовательно в порядке написания чисел. Граф такого вычислительного процесса приведен на рис. 1.5. Применяя в соответствии с рис. 1.5 правила подсчета относительных погрешностей, получим для относительной погрешности результата следующую формулу:

$$\frac{\Delta_y}{\widetilde{y}} = \left(\left(\frac{\widetilde{x}_1}{\widetilde{x}_1+\widetilde{x}_2}\delta_{x_1} + \frac{\widetilde{x}_2}{\widetilde{x}_1+\widetilde{x}_2}\delta_{x_2} + r_1\right)\frac{\widetilde{x}_1+\widetilde{x}_2}{\widetilde{x}_1+\widetilde{x}_2+\widetilde{x}_3} + \frac{\widetilde{x}_3}{\widetilde{x}_1+\widetilde{x}_2+\widetilde{x}_3}\delta_{x_3} + r_2\right)\times$$

$$\times\frac{\widetilde{x}_1+\widetilde{x}_2+\widetilde{x}_3}{\widetilde{x}_1+\widetilde{x}_2+\widetilde{x}_3+\widetilde{x}_4} + \frac{\widetilde{x}_4}{\widetilde{x}_1+\widetilde{x}_2+\widetilde{x}_3+\widetilde{x}_4}\delta_{x_4} + r_3.$$

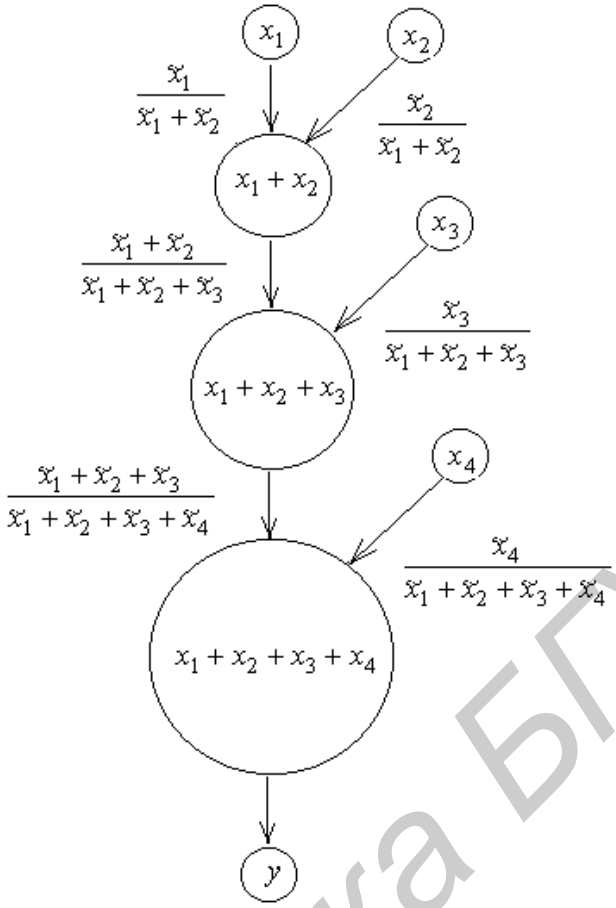

Рис. 1.5. Граф последовательного сложения положительных чисел

Предположим теперь, что исходные данные не содержат погрешностей, т.е.

$$\delta_{x_1} = \delta_{x_2} = \delta_{x_3} = \delta_{x_4} = 0 .$$

Тогда предыдущая формула для относительной погрешности результата упростится к виду

$$\frac{\Delta_y}{\tilde{y}} = \left(r_1 \frac{\tilde{x}_1 + \tilde{x}_2}{\tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3} + r_2 \right) \frac{\tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3}{\tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3 + \tilde{x}_4} + r_3 .$$

Умножая обе части этого равенства на $\tilde{y} = \tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3 + \tilde{x}_4$, получим выражение для абсолютной погрешности результата:

$$\Delta_y = r_1(\tilde{x}_1 + \tilde{x}_2) + r_2(\tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3) + r_3(\tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3 + \tilde{x}_4) .$$

Если погрешности округления одни и те же для каждой операции сложения, т.е. $r_1 = r_2 = r_3 = r$, то получим, что

$$\Delta_y = (3\tilde{x}_1 + 3\tilde{x}_2 + 2\tilde{x}_3 + x_4)r. \qquad (1.9)$$

Из последнего выражения видно, что числа, складываемые сначала, входят в выражение с большими коэффициентами, т.е. вносят больший вклад в погрешность результата. Погрешность становится меньше, если сначала складывать меньшие числа.

Из данного примера вытекает следующее правило. *Если необходимо произвести сложение (вычитание) длинной последовательности чисел, то для уменьшения погрешности результата необходимо работать сначала с наименьшими числами.*

Пример 1.2. Сложение почти равных положительных чисел

Пусть мы снова складываем четыре положительных числа x_1, x_2, x_3, x_4, но на этот раз они почти равны друг другу. Утверждение «числа почти равны друг другу» можно записать в виде

$$\tilde{x}_i = x_0 + e_i,\ i = \overline{1,4},$$

где $|e_i| << x_0$ (e_i много меньше x_0).

Применяя полученную в предыдущем примере формулу (1.9) к таким числам, для абсолютной погрешности суммы получим формулу

$$\Delta_y = (9x_0 + 3e_1 + 3e_2 + 2e_3 + e_4)r.$$

Так как $|e_i|$ малы по сравнению с x_0, то приближенно можно записать

$$\Delta_y = 9x_0 r.$$

Рассмотрим теперь другой способ вычисления этой суммы, а именно:

$$y = (x_1 + x_2) + (x_3 + x_4).$$

Граф такого вычислительного процесса представлен на рис. 1.6. Обозначая (как и в примере 1.1) δ_{x_1}, δ_{x_2}, δ_{x_3}, δ_{x_4} относительные погрешности чисел x_1, x_2, x_3, x_4 и r_1, r_2, r_3 – относительные погрешности округления после каждой

операции сложения, получим следующую формулу относительной погрешности суммы:

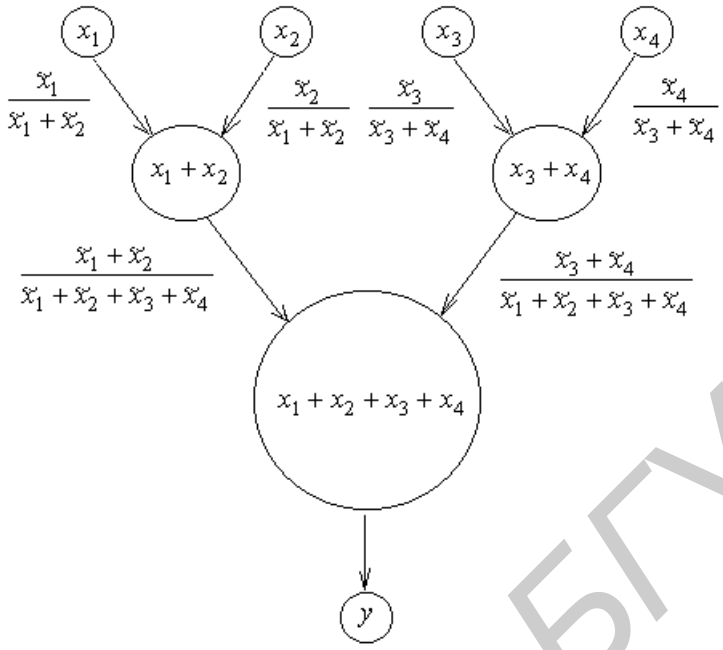

Рис. 1.6. Граф параллельного сложения положительных чисел

$$\delta_y = \frac{\Delta_y}{\tilde{y}} = \left(\frac{\tilde{x}_1}{\tilde{x}_1+\tilde{x}_2}\delta_{x_1} + \frac{\tilde{x}_2}{\tilde{x}_1+\tilde{x}_2}\delta_{x_2} + r_1 \right) \frac{\tilde{x}_1+\tilde{x}_2}{\tilde{x}_1+\tilde{x}_2+\tilde{x}_3+\tilde{x}_4} +$$

$$+ \left(\frac{\tilde{x}_3}{\tilde{x}_3+\tilde{x}_4}\delta_{x_3} + \frac{\tilde{x}_4}{\tilde{x}_3+\tilde{x}_4}\delta_{x_4} + r_2 \right) \frac{\tilde{x}_3+\tilde{x}_4}{\tilde{x}_1+\tilde{x}_2+\tilde{x}_3+\tilde{x}_4} + r_3 .$$

Считая $\delta_{x_1} = \delta_{x_2} = \delta_{x_3} = \delta_{x_4} = 0$, будем иметь

$$\delta_y = \frac{\Delta_y}{\tilde{y}} = r_1 \frac{\tilde{x}_1+\tilde{x}_2}{\tilde{x}_1+\tilde{x}_2+\tilde{x}_3+\tilde{x}_4} + r_2 \frac{\tilde{x}_3+\tilde{x}_4}{\tilde{x}_1+\tilde{x}_2+\tilde{x}_3+\tilde{x}_4} + r_3,$$

$$\Delta_y = r_1(\tilde{x}_1+\tilde{x}_2) + r_2(\tilde{x}_3+\tilde{x}_4) + r_3(\tilde{x}_1+\tilde{x}_2+\tilde{x}_3+\tilde{x}_4).$$

При одинаковых погрешностях округления $r_1 = r_2 = r_3 = r$ получим следующую абсолютную погрешность результата:

$$\Delta_y = (2\tilde{x}_1 + 2\tilde{x}_2 + 2x_3 + 2\tilde{x}_4)r .$$

Подставляя сюда $\tilde{x}_i = x_0 + e_i$ и пренебрегая e_i по сравнению с x_0, окончательно получим

$$\Delta_y = 8x_0 r .$$

Сравнивая ошибку округления для двух способов вычисления суммы, замечаем, что второй способ имеет несколько меньшую погрешность.

Проведенный анализ позволяет сформулировать следующее правило: при сложении n^2 положительных чисел приблизительно одинаковой величины общая ошибка округления уменьшится, если числа сложить сначала по n чисел, а потом сложить n частичных сумм.

Пример 1.3. Вычитание почти равных чисел

Предположим, что необходимо найти $z = x - y$ и $|\delta_x| < r$, $|\delta_y| < r$. Относительная погрешность разности определяется формулой (1.6). Из этой формулы видно, что если разность $\tilde{x} - \tilde{y}$ мала, то относительная погрешность разности может быть большой, даже если абсолютные погрешности операндов малы. Так как в дальнейших вычислениях эта большая относительная погрешность будет распространяться, то она может поставить под сомнение точность окончательного результата. Отсюда вывод: если возможно, необходимо избегать вычитания почти равных чисел. Часто формулы, содержащие такое вычитание, можно преобразовать так, чтобы избежать подобного вычитания. Например, вычисления по формуле

$$u = \sqrt{2{,}01} - \sqrt{2}$$

можно заменить вычислениями по преобразованной формуле:

$$u = \frac{(\sqrt{2{,}01} - \sqrt{2})(\sqrt{2{,}01} + \sqrt{2})}{\sqrt{2{,}01} + \sqrt{2}} = \frac{2{,}01 - 2}{\sqrt{2{,}01} + \sqrt{2}} = \frac{0{,}01}{\sqrt{2{,}01} + \sqrt{2}} .$$

2. РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Решение систем линейных алгебраических уравнений (СЛАУ) является достаточно важной вычислительной задачей. По сведениям из некоторых литературных источников 75 % всех расчетных математических задач приходится на решение СЛАУ. Это понятно, поскольку чаще всего сразу строят наиболее простые математические модели реальных процессов, т.е. линейные модели. С решением СЛАУ связаны такие задачи, как вычисление определителей, обращение матриц, вычисление собственных значений и собственных векторов матриц, интерполирование, аппроксимация по методу наименьших квадратов и многие другие. Современная вычислительная математика располагает большим арсеналом методов решения СЛАУ, а математическое обеспечение ЭВМ – многими пакетами программ и программными системами, позволяющими решать СЛАУ. Поэтому очень важно уметь ориентироваться в этом море методов и программ с тем, чтобы выбрать для себя наиболее подходящие или даже предложить собственные их модификации.

2.1. Постановка задачи. Методы решения

Определение. Системой линейных алгебраических уравнений (СЛАУ) называется совокупность равенств вида

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \ldots + a_{1,n}x_n = b_1, \\ a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 + \ldots + a_{2,n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m,1}x_1 + a_{m,2}x_2 + a_{m,3}x_3 + \ldots + a_{m,n}x_n = b_m, \end{cases} \qquad (2.1)$$

которые при некоторых значениях переменных $x_1, x_2, \ldots, x_n$ могут обращаться в тождества.

Переменные $x_1, x_2, ..., x_n$ этой совокупности равенств называются неизвестными, переменные $a_{i,j}$, $i, j = \overline{1,n}$, – коэффициентами, а переменные b_i, $i = \overline{1,n}$, – свободными членами СЛАУ. Значения $x_1^*, x_2^*, ..., x_n^*$ переменных $x_1, x_2, ..., x_n$, которые обращают равенства (2.1) в тождества, называются решением СЛАУ. Таким образом, задача решения СЛАУ состоит в том, чтобы по известным коэффициентам системы $a_{i,j}$, $i, j = \overline{1,n}$, и свободным членам b_i, $i = \overline{1,n}$, найти значения переменных $x_1, x_2, ..., x_n$, обращающие равенства (2.1) в тождества.

СЛАУ вида (2.1) – это система m уравнений с n неизвестными. Не останавливаясь на общей теории таких систем уравнений, в дальнейшем ограничимся рассмотрением системы n уравнений с n неизвестными:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \ldots + a_{1,n}x_n = b_1, \\ a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 + \ldots + a_{2,n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n,1}x_1 + a_{n,2}x_2 + a_{n,3}x_3 + \ldots + a_{n,n}x_n = b_n. \end{cases} \tag{2.2}$$

Эту систему уравнений можно записать в краткой форме:

$$\sum_{j=1}^{n} a_{i,j}x_j = b_i, \; i = \overline{1,n}, \; j = \overline{1,n}. \tag{2.3}$$

Часто систему (2.2) записывают в векторно-матричной форме. Для этого вводят в рассмотрение $(n \times n)$-матрицу коэффициентов $A = (a_{i,j})$ и векторы-столбцы неизвестных $X = (x_j)$ и свободных членов $B = (b_i)$, $i = \overline{1,n}$, $j = \overline{1,n}$:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}, \quad X = (x_j) = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad B = (b_i) = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Тогда СЛАУ (2.2) записывается в виде

$$AX = B, \tag{2.4}$$

поскольку сумма в левой части выражения (2.3) есть формула для расчета элементов матрицы AX.

Если определитель матрицы A не равен нулю ($\det(A) = |A| \neq 0$), то система (2.2) (или (2.4)) имеет единственное решение, которое определяется формулой

$$X = A^{-1}B, \tag{2.5}$$

где A^{-1} – матрица, обратная матрице A. Известно также правило Крамера для решения СЛАУ (2.4), в соответствии с которым неизвестные определяются по формуле

$$x_i = \frac{\Delta_i}{\Delta}, \quad i = \overline{1,n}, \tag{2.6}$$

где $\Delta = \det(A)$ – определитель матрицы A, Δ_i – определитель матрицы A, в которой столбец коэффициентов при x_i заменен столбцом свободных членов.

Однако методы решения СЛАУ с помощью обратной матрицы (2.5) или с помощью правила Крамера (2.6) встречают возражения с различных сторон. Прежде всего отмечается их большая трудоемкость (вычислительная сложность).

Если при реализации формул Крамера определители вычислять путем понижения порядка, т.е. путем разложения по элементам какой-нибудь строки или какого-нибудь столбца, то на вычисление определителя потребуется $n!$ операций умножения, а на решение СЛАУ – $n!n$ таких операций. Факториальный рост количества арифметических операций с увеличением размерности задачи (и вообще очень быстрый рост) называется «проклятьем размерности». По оценке авторов работы [3], число операций $30! \approx 10^{32}$ недоступно для современных ЭВМ. Если оценить величину $100! \approx 10^{158}$ и прикинуть потенциальные возможности развития вычислительной техники, то можно сделать вывод о том, что в обозримом будущем системы сотого порядка в принципе не могут быть решены по формулам Крамера [4]. Вместе с тем размерность $n = 30$ или $n = 100$

для современных задач считается небольшой. Довольно часто приходится решать системы с тысячами и более переменных.

Если находить решение с помощью обратной матрицы по формуле (2.5) и обратную матрицу находить через алгебраические дополнения, то такое решение СЛАУ будет равнозначным применению формул Крамера с теми же недостатками.

Другой причиной, по которой классические методы считаются неприемлемыми, является сильное влияние на окончательный результат округлений: погрешности за счет округлений катастрофически нарастают с увеличением числа переменных n.

В связи с указанными недостатками классических методов разработаны и применяются другие методы решения СЛАУ. Вообще все методы решения СЛАУ можно разделить на конечные и итерационные. Конечные методы также называются точными. Точные методы – это методы, которые в условиях отсутствия округлений приводят к точному решению за конечное число арифметических и логических операций. Итерационные методы – это методы, которые в условиях отсутствия округлений могут привести к точному решению путем бесконечного повторения единообразных действий (итераций). Понятно, что при наличии округлений и точные, и итерационные методы приведут к приближенному решению.

Точными следует считать метод решения путем обращения матрицы СЛАУ, правило Крамера, а также рассматриваемый в разд. 2.2 метод исключения Гаусса. Рассматриваемый в разд. 2.5 метод Гаусса – Зейделя является итерационным.

2.2. Метод Гаусса

2.2.1. Описание метода Гаусса

Дадим сначала общее описание метод Гаусса для решении СЛАУ (2.2). Этот метод состоит из двух этапов, которые называются прямым и обратным ходом. В процессе прямого хода система уравнений путем исключения переменных приводится к так называемому верхнему треугольному виду. В процессе обратного хода находится решение системы.

Прямой ход состоит из $n-1$ шагов $k=1,2,...,n-1$. На шаге $k=1$ исключается неизвестная x_1 из всех уравнений, начиная со второго. На шаге $k=2$ исключается x_2 из всех уравнений, начиная с третьего. На любом k-м шаге исключается x_k из всех уравнений, начиная с $k+1$ уравнения. На последнем шаге $k=n-1$ исключается x_{n-1} из последнего уравнения. В результате выполнения прямого хода мы получаем систему уравнений с так называемой верхней треугольной матрицей коэффициентов.

Обратный ход позволяет последовательно получить неизвестные системы уравнений. Сначала определяют x_n из последнего n-го уравнения. Затем это значение подставляют в $(n-1)$-е уравнение и определяют x_{n-1}, и т. д. до определения x_1 из первого уравнения.

2.2.2. Расчетные формулы метода Гаусса

Получим расчетные формулы метода Гаусса. Начнем с прямого хода. Прямой ход базируется на том, что решение системы уравнений не изменится, если из некоторого уравнения вычесть любое другое уравнение, умноженное на некоторый коэффициент. Коэффициенты подбираются таким образом, чтобы при вычитании исключались определенные переменные.

На первом шаге для i-го уравнения начиная с $i = 2$ вводится коэффициент

$$m_i^{(1)} = \frac{a_{i1}}{a_{11}}\ , \quad i = \overline{2,n}$$

и из i-го уравнения вычитается 1-е уравнение, умноженное на этот коэффициент. Результирующее уравнение записывается на место i-го. Это приводит к исключению переменной x_1 из i-го уравнения. После этого шага система уравнений примет следующий вид:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \ldots + a_{1,n}x_n &= b_1, \\ a_{2,2}^{(1)}x_2 + a_{2,3}^{(1)}x_3 + \ldots + a_{2,n}^{(1)}x_n &= b_2^{(1)}, \\ a_{3,2}^{(1)}x_2 + a_{3,3}^{(1)}x_3 + \ldots + a_{3,n}^{(1)}x_n &= b_3^{(1)}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n,2}^{(1)}x_2 + a_{n,3}^{(1)}x_3 + \ldots + a_{n,n}^{(1)}x_n &= b_n^{(1)}, \end{aligned}$$

где $a_{i,j}^{(1)}, b_i^{(1)}$ для всех $i, j = \overline{2,n}$ – коэффициенты, полученные на первом шаге прямого хода. Они определяются следующими выражениями:

$$a_{i,j}^{(1)} = a_{i,j} - m_i^{(1)} a_{1,j}\,,$$

$$b_i^{(1)} = b_i - m_i^{(1)} \; b_1\,,$$

$$i = \overline{2,n}, \quad j = \overline{1,n}\,.$$

На втором шаге для i-го уравнения начиная с $i = 3$ вводится коэффициент

$$m_i^{(2)} = \frac{a_{i,2}^{(1)}}{a_{2,2}^{(1)}}\ , \quad i = \overline{3,n}$$

и из i-го уравнения вычитается 2-е уравнение, умноженное на этот коэффициент. Это приводит к исключению из i-го уравнения переменной x_2. После второго шага система уравнений примет вид

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \ldots + a_{1,n}x_n = b_1,$$
$$a^{(1)}_{2,2}x_2 + a^{(1)}_{2,3}x_3 + \ldots + a^{(1)}_{2,n}x_n = b^{(1)}_2,$$
$$a^{(2)}_{3,3}x_3 + \ldots + a^{(2)}_{3,n}x_n = b^{(2)}_3,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a^{(2)}_{n,3}x_3 + \ldots + a^{(2)}_{2,n}x_n = b^{(2)}_n,$$

где $a^{(2)}_{i,j}, b^{(2)}_i$ – коэффициенты, полученные на втором шаге прямого хода. Они определяются выражениями

$$a^{(2)}_{i,j} = a^{(1)}_{i,j} - m^{(2)}_i a^{(1)}_{2,j},$$
$$b^{(2)}_i = b^{(1)}_i - m^{(2)}_i \quad b^{(1)}_2,$$
$$i = \overline{3,n}, \quad j = \overline{2,n}.$$

Вообще на k-м шаге для i-го уравнения начиная с $i = k+1$ вводится коэффициент

$$m^{(k)}_i = \frac{a^{(k-1)}_{i,k}}{a^{(k-1)}_{k,k}}, \quad i = \overline{k+1,n}, \tag{2.7}$$

и из i-го уравнения вычитается k-е уравнение, умноженное на этот коэффициент. Результирующее уравнение записывается на место i-го. При этом из i-го уравнения исключается переменная x_k. Коэффициенты системы уравнений на k-м шаге пересчитываются по формулам

$$a^{(k)}_{i,j} = a^{(k-1)}_{i,j} - m^{(k)}_i a^{(k-1)}_{k,j}, \tag{2.8}$$
$$b^{(k)}_i = b^{(k-1)}_i - m^{(k)}_i \quad b^{(k-1)}_k, \tag{2.9}$$
$$k = \overline{1,n-1}, \quad i = \overline{k+1,n}, \quad j = \overline{k,n}.$$

При $k = n-1$ происходит исключение x_{n-1} из последнего уравнения, и окончательная верхняя треугольная система записывается следующим образом:

$$
\begin{aligned}
a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \ldots + a_{1,n}x_n &= b_1 \ , \\
a^{(1)}_{2,2}x_2 + a^{(1)}_{2,3}x_3 + \ldots + a^{(1)}_{2,n}x_n &= b^{(1)}_2 , \\
a^{(2)}_{3,3}x_3 + \ldots + a^{(2)}_{3,n}x_n &= b^{(2)}_3 , \\
\ldots\ldots\ldots\ldots\ldots & \\
a^{(n-1)}_{n,n}x_n &= b^{(n-1)}_n .
\end{aligned}
\qquad (2.10)
$$

Теперь выполняется *обратный ход*. Видно, что из последнего уравнения можно сразу определить x_n:

$$x_n = \frac{b^{(n-1)}_n}{a^{(n-1)}_{n,n}} .$$

Подставляя это значение в предпоследнее уравнение, находим x_{n-1}:

$$x_{n-1} = \frac{b^{(n-2)}_{n-1} - a^{(n-2)}_{n-1,n}x_n}{a^{(n-2)}_{n-1,n-1}} .$$

Для нахождения любой переменной x_j применяется формула

$$x_j = \frac{b^{(j-1)}_j - a^{(j-1)}_{j,j+1}x_{j+1} - \ldots - a^{(j-1)}_{j,n}x_n}{a^{(j-1)}_{j,j}}, \quad j = n-1, n-2, \ldots, 1 .$$

Замечание. В процессе решения СЛАУ легко может быть получен определитель системы $\det(A)$. Он равен произведению диагональных элементов матрицы верхней треугольной системы:

$$\det(A) = a_{1,1}a^{(1)}_{2,2}a^{(2)}_{3,3}\ldots a^{(n-1)}_{n,n} .$$

Алгоритм Гаусса реализуется по схеме, приведенной на рис. 2.1, в том случае, когда блок №5 «Выбор главного элемента» пропускается. Назначение этого блока описывается в п. 2.2.3.

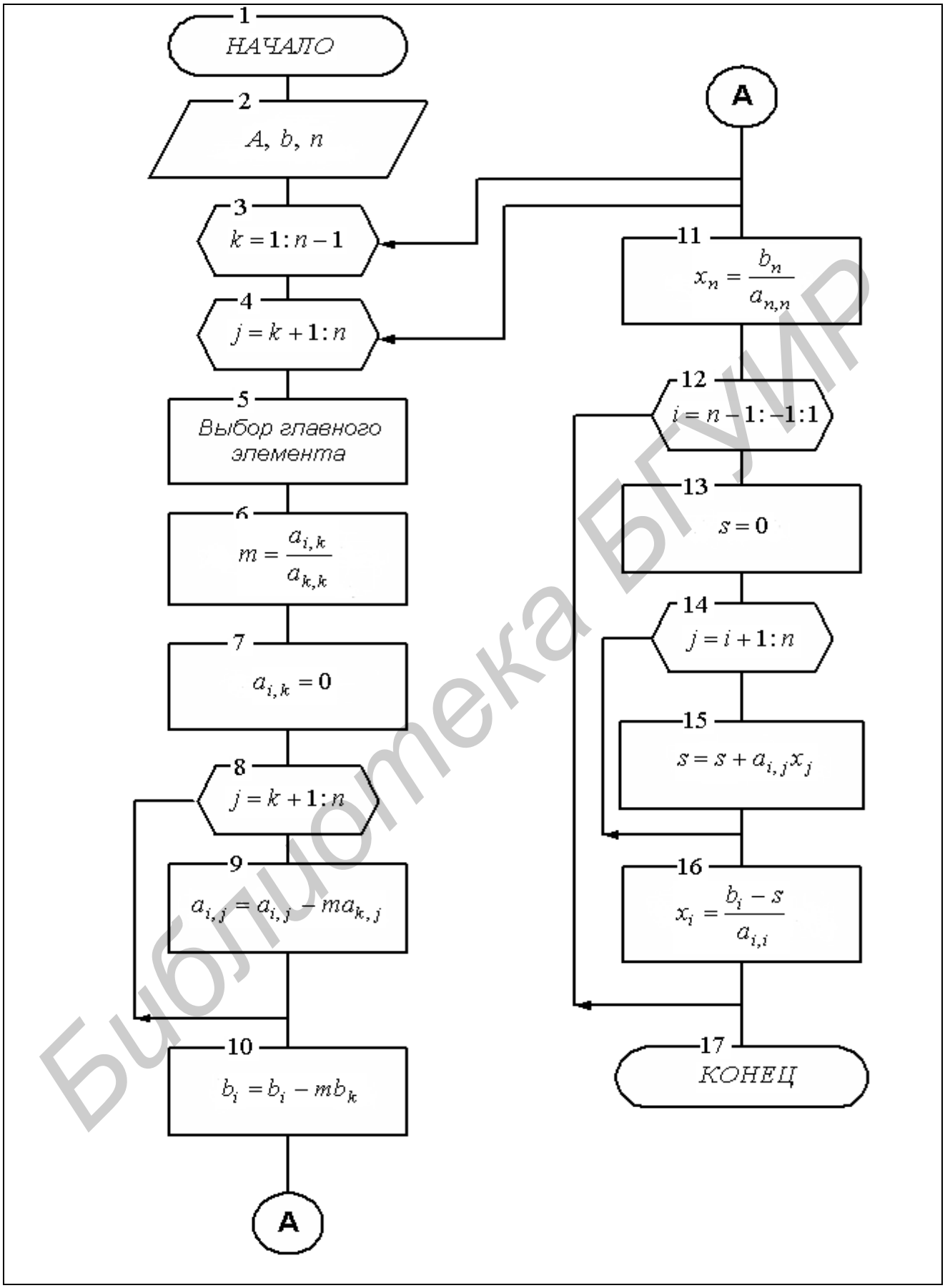

Рис. 2.1. Схема алгоритма Гаусса

2.2.3. Погрешность метода Гаусса. Метод Гаусса с выбором главного элемента

Основные вычисления методом исключений Гаусса выполняются с помощью формул (2.7) – (2.9). Граф процесса вычисления коэффициентов $a_{i,j}^{(k)}$ по формуле (2.8) представлен на рис. 2.2. Этот граф позволяет проследить накопление погрешностей в процессе вычислений по методу Гаусса.

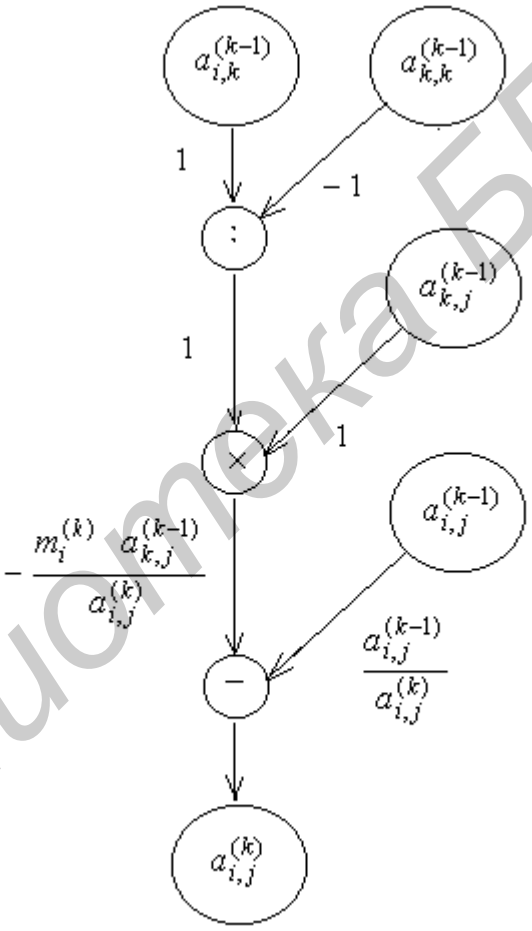

Рис. 2.2. Граф процесса вычисления коэффициента $a_{i,j}^{(k)}$

Обозначим $\delta_{i,j}$ относительную погрешность, содержащуюся в коэффициенте $a_{i,j}^{(k-1)}$, α,β,γ – относительные погрешности округления соответственно

при делении, умножении и вычитании. Тогда для относительной погрешности $\delta_{i,j}^{(k)}$ коэффициента $a_{i,j}^{(k)}$ можно получить выражение

$$\delta_{i,j}^{(k)} = \frac{\Delta_{i,j}^{(k)}}{a_{i,j}^{(k)}} = -(\delta_{i,k} - \delta_{k,k} + \alpha + \delta_{k,j} + \beta)\frac{m_i^{(k)} a_{k,j}^{(k-1)}}{a_{i,j}^{(k)}} + \delta_{i,j}\frac{a_{i,j}^{(k-1)}}{a_{i,j}^{(k)}} + \gamma .$$

Отсюда получаем формулу для оценки абсолютной погрешности $\Delta_{i,j}^{(k)}$:

$$|\Delta_{i,j}^{(k)}| \le (|\delta_{i,k}| + |\delta_{k,k}| + |\alpha| + |\delta_{k,j}| + |\beta|)\cdot|m_i^{(k)}|\cdot|a_{k,j}^{(k-1)}| + \\ + |\delta_{i,j}|\cdot|a_{i,j}^{(k-1)}| + |\gamma|\cdot|a_{i,j}^{(k)}| .$$

Если предположить, что погрешности $\delta_{i,j}$, α, β, γ не превышают некоторой величины C, то получим следующую формулу для оценки погрешности:

$$|\Delta_{i,j}^{(k)}| \le (5\cdot|m_i^{(k)}|\cdot|a_{k,j}^{(k-1)}| + |a_{i,j}^{(k-1)}| + |a_{i,j}^{(k)}|)C .$$

Из последнего выражения видно, что погрешность вычисления коэффициента $a_{i,j}^{(k)}$ в основном определяется первым слагаемым правой части и уменьшается с уменьшением $|m_i^{(k)}|$. Чтобы $|m_i^{(k)}|$ было по возможности меньшим, необходимо, чтобы $|a_{k,k}^{(k-1)}|$ было по возможности большим (см. формулу (2.7)). Поэтому перед выполнением шага исключения каждой переменной желательно переставить уравнения системы так, чтобы

$$|a_{k,k}^{(k-1)}| \ge |a_{i,k}^{(k-1)}|, \; i = \overline{k+1, n},$$

потому что тогда

$$|m_i^{(k)}| \le 1.$$

Если в методе Гаусса выполняется такая перестановка, то метод называется методом Гаусса с выбором главного элемента. Этот метод имеет меньшую погрешность при определении решения СЛАУ.

Алгоритм Гаусса с выбором главного элемента реализуется с помощью схемы рис. 2.1 при использовании блока №5 «Выбор главного элемента». Схема блока №5 приведена на рис. 2.3.

2.3. Вычислительная сложность метода Гаусса

Под вычислительной сложностью любого численного метода или алгоритма будем понимать число операций, необходимых для его выполнения. Число операций алгоритма легко посчитать по схеме алгоритма. Выполним такой расчет для метода Гаусса. Число операций будем обозначать числом с индексом, указывающим на вид операции (сложение, вычитание, умножение, деление и т.д.). По схеме рис. 2.1 для прямого хода получим

$$n_{\text{пр.х}} = \sum_{k=1}^{n-1}\sum_{i=k+1}^{n}[1_{\text{д}} + \sum_{j=k+1}^{n}(1_{\text{ум}} + 1_{\text{выч}}) + 1_{\text{ум}} + 1_{\text{выч}}] =$$

$$= \sum_{k=1}^{n-1}\sum_{i=k+1}^{n}[1_{\text{д}} + (n-k)_{\text{ум}} + (n-k)_{\text{выч}} + 1_{\text{ум}} + 1_{\text{выч}}].$$

Приравнивая умножения к делениям и вычитания к сложениям, будем иметь

$$n_{\text{пр.х}} = \sum_{k=1}^{n-1}\sum_{i=k+1}^{n}[(n-k+2)_{\text{ум}} + (n-k+1)_{\text{сл}}] =$$

$$= \sum_{k=1}^{n-1}\sum_{i=k+1}^{n}(n-k+2)_{\text{ум}} + (n-k+1)_{\text{сл}} =$$

$$= \sum_{k=1}^{n-1}(n-k)(n-k+2)_{\text{ум}} + (n-k)(n-k+1)_{\text{сл}} =$$

$$= \left(\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}\right)_{\text{ум}} + \left(\frac{n^3}{3} - \frac{n}{3}\right)_{\text{сл}}.$$

Хотя различные операции отличаются по сложности выполнения, но в последних моделях персональных компьютеров путем усовершенствования блока вычислений с плавающей точкой добиваются того, чтобы любая команда с плавающей точкой выполнялась за один такт микропроцессора. Поэтому будем учитывать все выполняемые операции. В итоге получим, что для выполнения прямого хода по методу Гаусса потребуется выполнить

$$n_{\text{пр.х}} = \frac{2n^3}{3} + \frac{n^2}{2} - \frac{7n}{6}$$

операций. Руководствуясь схемой рис. 2.1, сделаем аналогичные расчеты для обратного хода:

$$n_{\text{обр.х}} = \sum_{i=1}^{n-1}\left(\sum_{j=i+1}^{n} (1_{\text{ум}} + 1_{\text{сл}}) + 1_{\text{выч}} + 1_{\text{д}} \right) = \sum_{i=1}^{n-1}\left((n-i)_{\text{ум}} + (n-i)_{\text{сл}} + 1_{\text{выч}} + 1_{\text{д}} \right) =$$

$$= \sum_{i=1}^{n-1}\left((n-i+1)_{\text{ум}} + (n-i+1)_{\text{сл}} \right) = \left(\frac{n^2}{2} + \frac{n}{2} - 1 \right)_{\text{ум}} + \left(\frac{n^2}{2} + \frac{n}{2} - 1 \right)_{\text{сл}} = n^2 + n - 2 .$$

Общее число операций для выполнения алгоритма Гаусса *без выбора главного элемента* $n_{\text{б.выб}}$ будет равно

$$n_{\text{б.выб}} = n_{\text{пр.х}} + n_{\text{обр.х}} = \frac{2n^3}{3} + \frac{3n^2}{2} - \frac{n}{6} . \qquad (2.11)$$

В этом выражении опущено слагаемое, не зависящее от n, ввиду его малости.

Выполним также расчет числа операций, необходимых *для выбора главного элемента* $n_{\text{выб}}$. Из схемы рис. 2.3 видно, что здесь выполняются только операции *сравнения* (ср) и *присваивания* (пр). Число таких операций будет равно:

$$n_{\text{выб}} = \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} \left(2_{\text{пр}} + \sum_{\nu=л+1}^{m} (1_{\text{ср}} + 2_{\text{пр}}) + \sum_{j=k}^{n} (3_{\text{пр}}) + 3_{\text{пр}} \right) =$$

$$= \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} \left(2_{\text{пр}} + (n-k)_{\text{ср}} + (n-k)_{\text{пр}} + 3(n-k)_{\text{пр}} + 3_{\text{пр}} \right) =$$

$$= \sum_{k=1}^{n-1} (n-k)_{\text{ср}} + \left(4(n-k) + 5 \right)_{\text{пр}} = \left(\frac{n^2}{2} - \frac{n}{2} \right)_{\text{ср}} + \left(2n^2 + 3n - 5 \right)_{\text{пр}} = \frac{5n^2}{2} - \frac{5n}{2} .$$

Общее число операций для выполнения алгоритма Гаусса *с выбором главного элемента* $n_{\text{с выб}}$ будет равно

$$n_{\text{с выб}} = n_{\text{б.выб}} + n_{\text{выб}} = \frac{2n^3}{3} + 4n^2 + \frac{7n}{3} . \qquad (2.12)$$

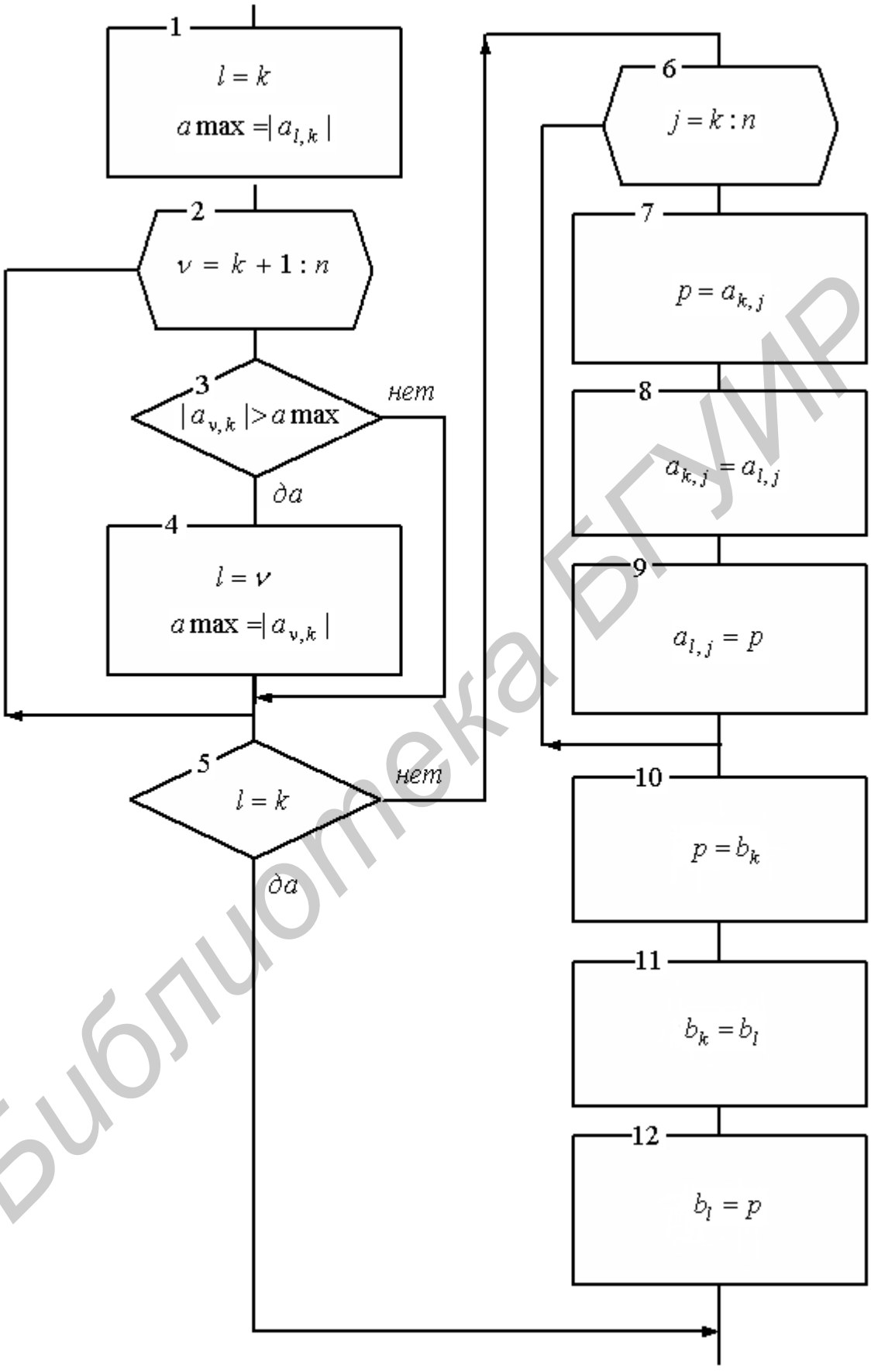

Рис. 2.3. Схема алгоритма выбора главного элемента

2.4. Обращение матрицы

Как известно, матрица A^{-1} называется обратной матрице $A=(a_{i,j}),\ i,j=\overline{1,n}$, если выполняется соотношение

$$AA^{-1}=E, \tag{2.13}$$

где E – единичная матрица.

Обозначим $A^{-1}=(a^{i,j})$, $E=(\delta_{i,j})$, $i,j=\overline{1,n}$, где $\delta_{i,j}$ – символ Кронекера,

$$\delta_{i,j}=\begin{cases}1, & \text{если} \quad i=j,\\ 0, & \text{если} \quad i\neq j.\end{cases}$$

Тогда соотношение (2.13) можно записать так:

$$\sum_{k=1}^{n} a_{i,k}a^{k,j}=\delta_{i,j},\ \ i,j=\overline{1,n}. \tag{2.14}$$

Из этого выражения видно, что если рассматривать j-й столбец $a^{k,j}$ обратной матрицы как вектор, то он является решением системы линейных алгебраических уравнений (2.14) с матрицей A и вектором правой части, все элементы которого равны нулю, кроме j-го, который равен 1. Таким образом, элементы обратной матрицы могут быть получены решением системы линейных алгебраических уравнений. Для обращения матрицы необходимо решить n систем линейных уравнений с одинаковой матрицей A и разными правыми частями. Приведение матрицы A к треугольному виду по формулам (2.8), (2.9) делается при этом только один раз. В дальнейшем необходимо формировать новую правую часть, преобразовывать ее по формуле (2.9) при помощи чисел $m_i^{(k)}$ (2.7) и для каждой правой части выполнять обратный ход. Понятно, что преобразующие числа $m_i^{(k)}$, $k=\overline{1,n-1}$, $i=\overline{k+1,n}$, необходимо сохранять по мере их получения на этапе прямого хода. Видно, что эти числа образуют нижнюю треугольную матрицу, совпадающую по размерам с матрицей исключаемых коэффициентов системы. Поэтому число $m_i^{(k)}$ можно поместить на место элемента $a_{i,k}$.

Для обращения матрицы этим методом требуется примерно $2n^2$ ячеек оперативной памяти и примерно $2n^3$ арифметических операций.

Описанный прием вычислений можно применить при решении систем уравнений с одной и той же матрицей A и различными правыми частями. В этом случае выгодно привести матрицу к треугольному виду только однажды, используя сохраненные числа $m_i^{(k)}$ во всех последующих вычислениях.

2.5. Метод LU-разложения

Матрица $L=(l_{i,j})$, $i,j=\overline{1,n}$, называется нижней треугольной, если все ее элементы, расположенные выше главной диагонали, равны нулю. Матрица $U=(u_{i,j})$, $i,j=\overline{1,n}$, называется верхней треугольной, если все ее элементы, расположенные ниже главной диагонали, равны нулю. Общепринятые обозначения L и U связаны с английскими словами lower (нижний) и upper (верхний). Пусть $A=(a_{i,j})$, $i,j=\overline{1,n}$, $n\times n$-матрица, например матрица СЛАУ (2.4). Справедлива следующая теорема.

Теорема. Если все главные миноры квадратной матрицы A отличны от нуля, то существуют такие нижняя L и верхняя U треугольные матрицы, что $A=LU$. Если элементы диагонали одной из матриц L или U фиксированы (ненулевые), то такое разложение единственно.

Получим формулы для разложения матрицы A при фиксированной диагонали матрицы L. В развернутой форме разложение $A=LU$ имеет вид

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{2,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n,1} & l_{n,2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ 0 & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{n,n} \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}.$$

Выполняя умножение матриц в левой части и приравнивая соответствующие элементы обеих частей равенства, получим $n \times n$-матрицу уравнений

$$\begin{array}{llll} u_{1,1} = a_{1,1}, & u_{1,2} = a_{1,2}, & \cdots & u_{1,n} = a_{1,n}, \\ l_{2,1}u_{1,1} = a_{2,1}, & l_{2,1}u_{1,2} + u_{2,2} = a_{2,2}, & \cdots & l_{2,1}u_{1,n} + u_{2,n} = a_{2,n}, \\ \vdots & \vdots & \ddots & \vdots \\ l_{n,1}u_{1,1} = a_{n,1}, & l_{n,1}u_{1,2} + l_{n,2}u_{2,2} = a_{n,2}, & \cdots & l_{n,1}u_{1,n} + \ldots + l_{n,n-1}u_{n-1,n} + u_{n,n} = a_{n,n} \end{array}$$

относительно $n \times n$-матрицы неизвестных

$$\begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ l_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n,1} & l_{n,2} & \cdots & u_{n,n} \end{pmatrix}. \qquad (2.15)$$

Специфика этой системы такова, что позволяет определить неизвестные одно за другим. Сначала из первой строки системы находим

$$u_{1,j} = a_{1,j},\ j = \overline{1,n}.$$

Затем из оставшейся части первого столбца системы находим

$$l_{i,1} = \frac{a_{i,1}}{u_{1,1}},\ i = \overline{2,n}.$$

Далее, из оставшейся части второй строки

$$u_{2,j} = a_{2,j} - l_{2,1}u_{1,j},\ j = \overline{2,n},$$

из оставшейся части второго столбца

$$l_{i,2} = \frac{a_{i,2} - l_{i,1}u_{1,2}}{u_{2,2}},\ i = \overline{3,n},$$

и т.д. Последним находим элемент

$$u_{n,n} = a_{n,n} - \sum_{k=1}^{n-1} l_{n,k}u_{k,n}.$$

Легко заметить, что все отличные от нуля и единицы элементы матриц L, U могут быть однозначно вычислены с помощью всего двух формул:

$$u_{i,j} = a_{i,j} - \sum_{k=1}^{n-1} l_{i,k} u_{k,j}\ ,\ i \le j\ , \tag{2.16}$$

$$l_{i,j} = \frac{1}{u_{j,j}} \left(a_{i,j} - \sum_{k=1}^{n-1} l_{i,k} u_{k,j} \right),\ i > j\ . \tag{2.17}$$

При организации вычислений по формулам (2.16), (2.17) необходимо переключать вычисления с одной формулы на другую, как указано выше. Это удобно делать, ориентируясь на матрицу неизвестных (2.15), а именно, первая строка матрицы (2.15) вычисляется по формуле (2.16) при $i = 1,\ j = \overline{1,n}$; первый столбец (2.15) (без первого элемента) – по формуле (2.17) при $j = 1,\ i = \overline{2,n}$, и т.д.

Если матрица A исходной СЛАУ (2.4) разложена в произведение треугольных матриц L, U, то вместо (2.4) мы можем записать эквивалентное уравнение

$$LUX = B\ .$$

Введя вектор вспомогательных переменных $Y = (y_1, y_2, ..., y_n)^T$, перепишем его в виде системы

$$LY = B,$$
$$UX = Y\ .$$

Таким образом, решение СЛАУ с квадратной матрицей коэффициентов свелось к решению двух СЛАУ с треугольными матрицами коэффициентов, которые могут быть легко решены. Уравнение $LY = B$ в развернутом виде имеет вид

$$\begin{cases} y_1 = b_1, \\ l_{2,1} y_1 + y_2 = b_2\ , \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ l_{n,1} y_1 + l_{n,2} y_2 + \ldots + l_{n,n-1} y_{n-1} + y_n = b_n\ . \end{cases}$$

Все y_i могут быть последовательно найдены по формуле

$$y_i = b_i - \sum_{k=1}^{i-1} l_{i,k} y_k\ . \tag{2.18}$$

Уравнение $UX = Y$ в развернутом виде имеет вид

$$\begin{cases} u_{1,1}x_1 + u_{1,2}x_2 + a_{1,3}x_3 + \ldots + u_{1,n}x_n = y_1, \\ u_{2,2}x_2 + u_{2,3}x_3 + \ldots + u_{2,n}x_n = y_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ u_{n,n}x_n = y_n. \end{cases} \quad (2.19)$$

Отсюда получаем

$$x_i = \frac{1}{u_{i,i}}\left(y_i - \sum_{k=i+1}^{n} u_{i,k}x_k \right). \quad (2.20)$$

Итак, решение СЛАУ методом LU-разложения сводится к расчетам по четырем формулам: совокупности формул (2.16), (2.17) для получения элементов матриц L и U, формуле (2.18) для получения вспомогательного вектора Y и формуле (2.20) для получения решения СЛАУ.

Отметим, что система уравнений (2.19) полностью совпадает с системой уравнений (2.10), получающейся после завершения прямого хода метода Гаусса. Это означает, что решения СЛАУ с помощью LU-разложения – это просто другая реализация метода Гаусса. Эта реализация называется схемой Холецкого.

Следует отметить, что в отличие от метода Гаусса схема Холецкого менее удобна для усовершенствования с целью уменьшения влияния вычислительных погрешностей путем выбора подходящих ведущих элементов.

2.6. Метод квадратного корня решения симметричных СЛАУ

СЛАУ (2.4) называется симметричной, если матрица ее коэффициентов $A = (a_{i,j})$, $i, j = \overline{1,n}$, симметричная, т.е. если $A^T = A$. Симметричная СЛАУ вполне может быть решена методом Гаусса. Однако если для решения симметричной СЛАУ воспользоваться LU-разложением (схемой Холецкого), то объем вычислений можно сократить почти вдвое. Ввиду симметричности матрицы схема Холецкого упрощается по сравнению со случаем матрицы общего вида.

Будем строить представление симметричной матрицы в виде $A = U^T U$. Запишем это разложение в развернутом виде:

$$\begin{pmatrix} u_{1,1} & 0 & \cdots & 0 \\ u_{1,2} & u_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ u_{1,n} & u_{2,n} & \cdots & u_{n,n} \end{pmatrix} \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ 0 & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{n,n} \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}.$$

Умножая матрицы в левой части и приравнивая соответствующие элементы обеих частей, получим систему $n(n+1)/2$ уравнений относительно такого же количества неизвестных (элементов матрицы U):

$$\begin{aligned} u_{1,1}^2 = a_{1,1}, \quad u_{1,1}u_{1,2} = a_{1,2}, \quad \cdots \quad & u_{1,1}u_{1,n} = a_{1,n}, \\ u_{1,2}^2 + u_{2,2}^2 = a_{2,2}, \quad \cdots \quad & u_{1,2}u_{1,n} + u_{2,2}u_{2,n} = a_{2,n}, \\ & \vdots \\ & u_{1,n}^2 + u_{2,n}^2 + \ldots + u_{n,n}^2 = a_{n,n}. \end{aligned}$$

Из первой строки уравнений находим сначала $u_{1,1} = \sqrt{a_{1,1}}$, затем $u_{1,j} = \dfrac{a_{1,j}}{u_{1,1}}$ при $j = \overline{2,n}$. Из второй – $u_{2,2} = \sqrt{a_{2,2} - u_{1,2}^2}$, затем $u_{2,j} = \dfrac{a_{2,j} - u_{1,2}u_{1,j}}{u_{2,2}}$ при $j = \overline{3,n}$ и т.д. Завершается процесс вычислением

$$u_{n,n} = \sqrt{a_{n,n} - \sum_{k=1}^{n-1} u_{k,n}^2}.$$

Таким образом, матрица U может быть определена совокупностью формул

$$u_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{i-1} u_{k,i}^2}, \; i = \overline{1,n}, \tag{2.21}$$

$$u_{i,j} = \frac{a_{i,j} - \sum_{k=1}^{i-1} u_{k,i}u_{k,j}}{u_{i,i}}, \; j = \overline{2,n}, \; j > i, \tag{2.22}$$

$$u_{i,j} = 0, \; j < i. \tag{2.23}$$

Известно, что для важного в приложениях класса симметричных положительно определенных матриц разложение по формулам (2.21) – (2.23) выполнимо.

При наличии разложения $A = U^T U$ мы можем вместо уравнения (2.4) записать эквивалентное уравнение

$$U^T U X = B.$$

Введя вектор вспомогательных переменных $Y = (y_1, y_2, ..., y_n)^T$, перепишем его в виде системы

$$U^T Y = B,$$
$$UX = Y.$$

Первое уравнение этой системы имеет вид

$$\begin{cases} u_{1,1} y_1 = b_1, \\ u_{1,2} y_1 + u_{2,2} y_2 = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ u_{1,n} y_1 + u_{2,n} y_2 + \ldots + u_{n,n} y_n = b_n, \end{cases}$$

откуда получаем

$$y_i = \frac{1}{u_{i,i}} \left(b_i - \sum_{k=1}^{i-1} u_{k,i} y_k \right), \; i = \overline{1,n}. \qquad (2.24)$$

Из второго уравнения

$$\begin{cases} u_{1,1} x_1 + u_{1,2} x_2 + u_{1,3} x_3 + \ldots + u_{1,n} x_n = y_1, \\ u_{2,2} x_2 + u_{2,3} x_3 + \ldots + u_{2,n} x_n = y_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ u_{n,n} x_n = y_n \end{cases}$$

находим искомые значения x_i:

$$x_i = \frac{1}{u_{i,i}} \left(y_i - \sum_{k=i+1}^{n} u_{i,k} x_k \right), \; i = n, n-1, \ldots, 1. \qquad (2.25)$$

Решение симметричных СЛАУ по формулам (2.21) – (2.25) называют методом квадратного корня.

2.7. Метод Гаусса – Зейделя

Метод Гаусса – Зейделя – это итерационный метод решения задачи, или метод последовательных приближений.

2.7.1. Расчетные формулы метода Гаусса – Зейделя

Получим расчетные формулы метода Гаусса – Зейделя. Пусть решается система уравнений (2.1). Выразим из 1-го уравнения x_1, из 2-го уравнения x_2 и т.д. В результате получим

$$x_1 = \frac{1}{a_{1,1}}(b_1 - a_{1,2}x_2 - a_{1,3}x_3 - \ldots - a_{1,n}x_n),$$

$$x_2 = \frac{1}{a_{2,2}}(b_2 - a_{2,1}x_1 - a_{2,3}x_3 - \ldots - a_{2,n}x_n),$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$x_n = \frac{1}{a_{n,n}}(b_n - a_{n,1}x_1 - a_{n,2}x_2 - \ldots - a_{n,n-1}x_{n-1})$$

или вообще для любого i

$$x_i = \frac{1}{a_{i,i}}(b_i - a_{i,1}x_1 - a_{i,2}x_2 - \ldots - a_{i,i-1}x_{i-1} - a_{i,i+1}x_{i+1} - \ldots - a_{i,n}x_n), \quad i = \overline{1,n}. \quad (2.26)$$

Предположим, что на некоторой k-й итерации мы получили решение $x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}$. Используем известные к моменту расчета x_i значения этих переменных в правой части уравнения (2.26) для расчета значения x_i в левой части. В результате мы получим следующую рекуррентную формулу, которая и составляет метод Гаусса – Зейделя:

$$x_i^{(k+1)} = \frac{1}{a_{i,i}}(b_i - a_{i,1}x_1^{(k+1)} - \ldots - a_{i,i-1}x_{i-1}^{(k+1)} - a_{i,i+1}x_{i+1}^{(k)} - \ldots - a_{i,n}x_n^{(k)}), \quad i = \overline{1,n}.$$

Расчеты по последней формуле продолжаются при $k = 1,2,3,...$ до тех пор, пока не будет выполняться условие

$$\max_i |x_i^{(k+1)} - x_i^{(k)}| < \varepsilon,$$

или условие

$$\max_i \left| \frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k+1)}} \right| < \delta,$$

где $\varepsilon > 0$, $\delta > 0$, причем ε – допустимая абсолютная погрешность нахождения решения СЛАУ, а δ – допустимая относительная погрешность.

2.7.2. Сходимость метода Гаусса – Зейделя

Итерационные методы могут сходиться к решению или не сходиться. То же самое относится и к методу Гаусса – Зейделя. Исследуем сходимость метода на примере системы из двух уравнений

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 = b_1, \\ a_{2,1}x_1 + a_{2,2}x_2 = b_2. \end{cases}$$

Запишем данную систему в виде

$$\begin{cases} x_1 = \dfrac{1}{a_{1,1}}(b_1 - a_{1,2}x_2), \\ x_2 = \dfrac{1}{a_{2,2}}(b_2 - a_{2,1}x_1). \end{cases} \tag{2.27}$$

Итерации метода Гаусса – Зейделя здесь осуществляются по формулам

$$\begin{cases} x_1^{(k)} = \dfrac{1}{a_{1,1}}(b_1 - a_{1,2}x_2^{(k-1)}), \\ x_2^{(k)} = \dfrac{1}{a_{2,2}}(b_2 - a_{2,1}x_1^{(k)}). \end{cases} \tag{2.28}$$

Обозначим

$$\begin{cases} \Delta x_1^{(k)} = x_1 - x_1^{(k)}, \\ \Delta x_2^{(k)} = x_2 - x_2^{(k)}. \end{cases}$$

Вычитая из выражений (2.27) выражения (2.28), получим

$$\begin{cases} \Delta x_1^{(k)} = -\dfrac{a_{1,2}}{a_{1,1}} \Delta x_2^{(k-1)}, \\ \Delta x_2^{(k)} = -\dfrac{a_{2,1}}{a_{2,2}} \Delta x_1^{(k)}. \end{cases}$$

Подставляя первое из этих выражений во второе, получим

$$\Delta x_2^{(k)} = \frac{a_{1,2}a_{2,1}}{a_{1,1}a_{2,2}} \Delta x_2^{(k-1)}.$$

Аналогично можно найти

$$\Delta x_2^{(k-1)} = \frac{a_{1,2}a_{2,1}}{a_{1,1}a_{2,2}} \Delta x_2^{(k-2)},$$

так что

$$\Delta x_2^{(k)} = \left(\frac{a_{1,2}a_{2,1}}{a_{1,1}a_{2,2}} \right)^2 \Delta x_2^{(k-2)}.$$

Продолжая этот процесс дальше, будем иметь

$$\Delta x_2^{(k)} = \left(\frac{a_{1,2}a_{2,1}}{a_{1,1}a_{2,2}} \right)^k \Delta x_2^{(0)}.$$

Точно так же можно получить

$$\Delta x_1^{(k)} = \left(\frac{a_{1,2}a_{2,1}}{a_{1,1}a_{2,2}} \right)^k \Delta x_1^{(0)}.$$

Поэтому если

$$\left| \frac{a_{1,2}a_{2,1}}{a_{1,1}a_{2,2}} \right| < 1, \qquad (2.29)$$

то $\Delta x_1^{(k)} \underset{k\to\infty}{\to} 0$, $\Delta x_2^{(k)} \underset{k\to\infty}{\to} 0$, и метод Гаусса-Зейделя сходится к решению x_1, x_2.

Соотношение (2.29) – это достаточное условие сходимости метода Гаусса – Зейделя для системы из двух уравнений.

Содержательный смысл приведенного условия сходимости можно выяснить, проанализировав его. Неравенство (2.29) выполняется, если

$$\begin{cases} |a_{1,1}| > |a_{1,2}|, \\ |a_{2,2}| \geq |a_{2,1}|, \end{cases}$$

или если

$$\begin{cases} |a_{1,1}| \geq |a_{1,2}|, \\ |a_{2,2}| > |a_{2,1}|. \end{cases}$$

Последние два условия можно назвать условиями преобладания диагональных элементов системы уравнений: для сходимости метода Гаусса – Зейделя диагональные элементы системы уравнений по абсолютной величине должны превышать недиагональные.

Полученные достаточные условия сходимости для системы двух уравнений можно распространить на систему n уравнений. Без доказательства скажем, что метод Гаусса – Зейделя сходится для системы n уравнений, если выполняются условия

$$|a_{i,i}| \geq |a_{i,1}| + |a_{i,2}| + \ldots + |a_{i,i-1}| + |a_{i,i+1}| + \ldots + |a_{i,n}|$$

для всех $i = \overline{1,n}$ кроме одного, для которого выполняется более жесткое условие

$$|a_{i,i}| > |a_{i,1}| + |a_{i,2}| + \ldots + |a_{i,i-1}| + |a_{i,i+1}| + \ldots + |a_{i,n}|.$$

Сформулированные условия также являются достаточными, но не необходимыми. Возможны случаи невыполнения данных условий при сходимости метода. Эти условия также можно назвать условиями преобладания диагональных элементов. В связи с этим часто для обеспечения сходимости метода Гаусса – Зейделя бывает достаточно поменять местами уравнения системы с тем, чтобы на главную диагональ матрицы системы попали наибольшие по абсолютному значению коэффициенты.

2.7.3. Графическая иллюстрация метода Гаусса – Зейделя

Проиллюстрируем сходимость или расходимость метода Гаусса – Зейделя на примере системы уравнений

$$\begin{cases} x_1 - 2x_2 = -2, \\ 2x_1 + x_2 = 2. \end{cases}$$

Сразу отметим, что достаточное условие сходимости метода Гаусса – Зейделя (2.29) для данной системы не выполняется, поскольку

$$\left|\frac{a_{1,2}a_{2,1}}{a_{1,1}a_{2,2}}\right| = \left|\frac{-2\cdot 2}{1\cdot 1}\right| = 4 > 1.$$

В соответствии с рекомендациями метода Гаусса–Зейделя перепишем эту систему в виде

$$\begin{cases} x_1 = -2 + 2x_2, \\ x_2 = 2 - 2x_1 \end{cases}$$

и получим следующие формулы итераций:

$$\begin{cases} x_1^{(k)} = -2 + 2x_2^{(k-1)}, \\ x_2^{(k)} = 2 - 2x_1^{(k)}. \end{cases} \qquad (2.30)$$

Решение x_1^*, x_2^* данной системы определяется как точка пересечения двух прямых $x_1 - 2x_2 = -2$ и $2x_1 + x_2 = 2$. Эти прямые изображены на рисунке 2.4. В соответствии с формулой итераций (2.30) мы выбираем начальное приближение $x_2^{(0)}$, подставляем это значение в первое уравнение системы $x_1 - 2x_2 = -2$ и из него определяем новое приближение $x_1^{(1)}$. Затем это значение $x_1^{(1)}$ подставляем во второе уравнение системы $2x_1 + x_2 = 2$ и из него определяем новое приближение $x_2^{(1)}$. На этом первая итерация заканчивается. Далее этот процесс повторяется. Описанные итерации изображены на рис. 2.4 прямыми линиями со

стрелками. Мы видим, что итерационный процесс не сходится к решению системы.

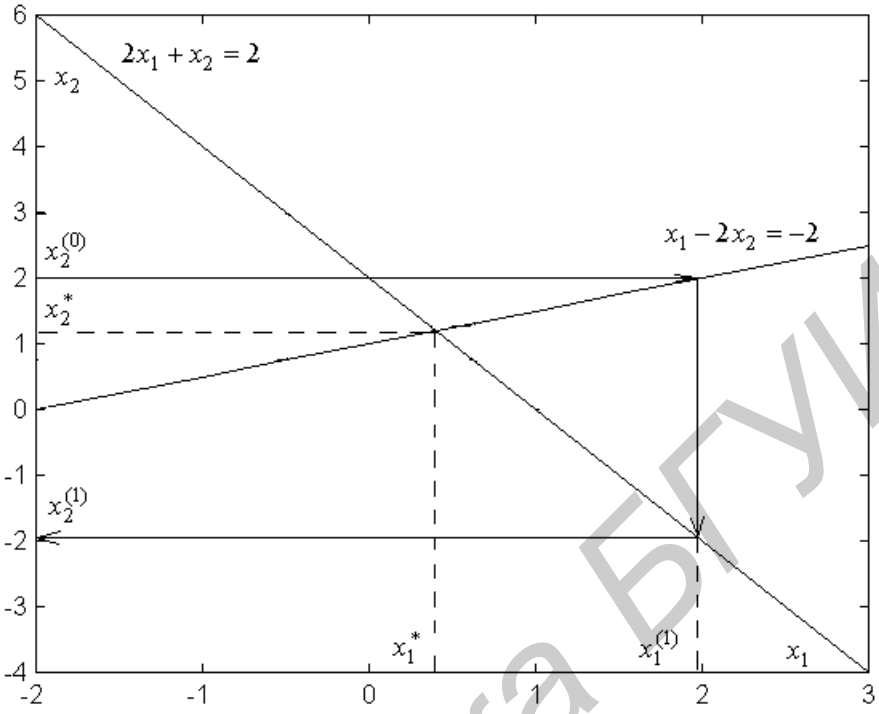

Рис. 2.4. Иллюстрация расходимости метода Гаусса – Зейделя

Переставим теперь уравнения системы, т.е. будем решать систему

$$\begin{cases} 2x_1 + x_2 = 2, \\ x_1 - 2x_2 = -2. \end{cases}$$

Достаточное условие сходимости метода Гаусса – Зейделя (2.29) теперь выполняется:

$$\left|\frac{a_{1,2}a_{2,1}}{a_{1,1}a_{2,2}}\right| = \left|\frac{1\cdot 1}{2\cdot 2}\right| = \frac{1}{4} < 1.$$

В соответствии с рекомендациями метода Гаусса – Зейделя перепишем эту систему в виде

$$\begin{cases} x_1 = 1 - \dfrac{x_2}{2}, \\ x_2 = 1 + \dfrac{x_1}{2} \end{cases}$$

и организуем итерационный процесс по формулам

$$\begin{cases} x_1^{(k)} = 1 - \dfrac{x_2^{(k-1)}}{2}, \\ x_2^{(k)} = 1 + \dfrac{x_1^{(k)}}{2}. \end{cases} \tag{2.31}$$

Этот процесс изображен прямыми линиями со стрелками на рис. 2.5.

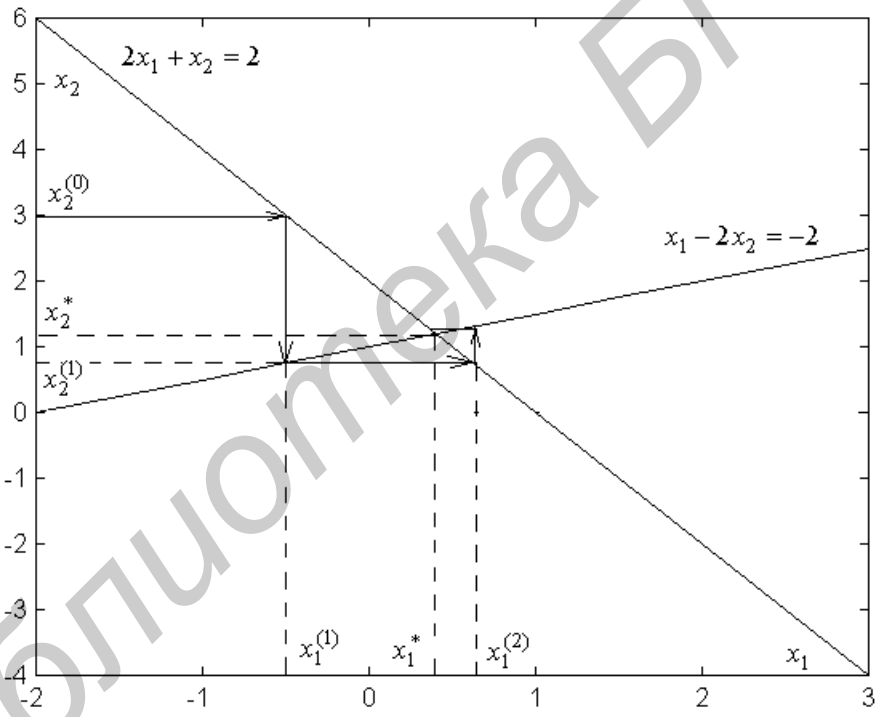

Рис. 2.5. Иллюстрация сходимости метода Гаусса – Зейделя

В соответствии с формулой итераций (2.31) выбираем начальное приближение $x_2^{(0)}$, подставляем это значение в первое уравнение системы $2x_1 + x_2 = 2$ и из него определяем новое приближение $x_1^{(1)}$. Затем это значение $x_1^{(1)}$ подставляем

во второе уравнение системы $x_1 - 2x_2 = -2$ и из него определяем новое приближение $x_2^{(1)}$. Далее этот процесс повторяется. Мы видим, что процесс сходится. Таким образом, простой перестановкой уравнений системы мы добились сходимости метода Гаусса – Зейделя.

3. АППРОКСИМАЦИЯ ФУНКЦИЙ

3.1. Понятие аппроксимации функций

Аппроксимация функции $y = f(x)$ – это замена этой функции другой более простой функцией $y = \varphi(x)$, близкой к $f(x)$ в некотором смысле. В зависимости от критерия близости функций $f(x)$ и $\varphi(x)$ существуют различные методы аппроксимации.

Если расстояние с между функциями $f(x)$ и $\varphi(x)$ на некотором отрезке $[a,b]$ действительной прямой определить выражением

$$\rho(f(x),\varphi(x)) = \int_a^b (f(x) - \varphi(x))^2 dx,$$

то аппроксимация функции $f(x)$ по критерию минимума такого расстояния ρ будет называться аппроксимацией с минимальной интегральной квадратичной погрешностью.

Если критерий близости функций $f(x)$ и $\varphi(x)$ состоит в том, чтобы $f(x)$ и $\varphi(x)$ совпадали в дискретном ряде точек $x_0, x_1, \ldots, x_n$ отрезка $[a,b]$, то такой способ аппроксимации функции $f(x)$ называется *интерполированием* функции $f(x)$.

Если расстояние ρ между функциями $f(x)$ и $\varphi(x)$ на некотором отрезке $[a,b]$ действительной прямой определить выражением

$$\rho(f(x),\varphi(x)) = \sum_{i=0}^{n} (f(x_i) - \varphi(x_i))^2,$$

то аппроксимация функции $f(x)$ по критерию минимума такого расстояния ρ будет называться аппроксимацией по методу наименьших квадратов.

3.2. Постановка задачи интерполирования функций

Задача интерполирования функции $f(x)$ на некотором отрезке $[a,b]$ формулируется следующим образом. На отрезке $[a,b]$ задано $n+1$ точек $x_0, x_1, \ldots, x_n \in [a,b]$, которые называют узлами. Обычно считают, что первая и последняя точки совпадают с концами отрезка $[a,b]$: $x_0 = a$, $x_n = b$. Известны значения $y_i = f(x_i)$ функции $f(x)$ в этих точках, $i = \overline{0,n}$. Требуется заменить эту функцию некоторой другой функцией $\varphi(x)$ таким образом, чтобы значения обеих функций совпадали в узлах, т.е. чтобы выполнялись равенства

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

Искомой неизвестной в данной задаче является функция $\varphi(x)$.

Сформулированную задачу иногда интерпретируют следующим образом. Некоторая функция $f(x)$ задана на отрезке $[a,b]$ таблицей своих значений

x_i	x_0	x_1	x_2	…	x_n
$y_i = f(x_i)$	y_0	y_1	y_2	…	y_n

и требуется найти способ определения значений этой функции в любых других точках отрезка $[a,b]$.

При формулировке задачи интерполирования обычно предполагают, что аппроксимирующая функция $\varphi(x)$ задана с точностью до $(n+1)$ параметров $a_0, a_1, \ldots, a_n$, т.е. в виде $\varphi(x, a_0, a_1, \ldots, a_n)$. Тогда нам необходимо отыскать неизвестные параметры $a_0, a_1, \ldots, a_n$ исходя из заданных равенств

$$\varphi(x_i, a_0, a_1, \ldots, a_n) = y_i, \quad i = \overline{0,n}. \tag{3.1}$$

Эти равенства можно рассматривать как систему $n+1$ уравнений относительно неизвестных параметров $a_0, a_1, \ldots, a_n$.

Чаще всего функцию $\varphi(x, a_0, a_1, \ldots, a_n)$ представляют в виде полинома n-й степени

$$\varphi(x, a_0, a_1, \ldots, a_n) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n . \qquad (3.2)$$

Тогда система уравнений (3.1) принимает следующий вид:

$$\begin{aligned} &a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n = y_0, \\ &a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n = y_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n = y_n . \end{aligned} \qquad (3.3)$$

Определитель этой системы имеет вид

$$\begin{vmatrix} 1 & x_0 & x_0^2 & \ldots x_0^n \\ 1 & x_1 & x_1^2 & \ldots x_1^n \\ \ldots & \ldots & \ldots & \ldots\ldots \\ 1 & x_n & x_n^2 & \ldots x_n^n \end{vmatrix}$$

и называется определителем Вандермонда на системе точек $x_0, x_1, \ldots, x_n$. Доказано, что если точки $x_0, x_1, \ldots, x_n$ попарно различны, что предполагается при постановке задачи, то определитель Вандермонда не равен нулю. В таком случае система уравнений (3.3) имеет единственное решение, т.е. существует единственный полином (3.2) степени n, коэффициенты которого удовлетворяют системе уравнений (3.3). Этот полином называется интерполяционным полиномом для функции $f(x)$.

3.3. Интерполяционный полином Лагранжа

Интерполяционный полином может быть представлен в различных формах. Одной из них является форма Лагранжа. Полином Лагранжа имеет следующий вид:

$$P_n(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)},$$

или в компактной форме

$$P_n(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} = \sum_{i=0}^{n} y_i L_i(x), \tag{3.4}$$

где

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}, \quad i = \overline{0, n},$$

так называемый полином влияния i-го узла.

Формула (3.4) легко может быть получена путем следующих рассуждений. Понятно, что полином влияния i-го узла $L_i(x)$ есть полином n-й степени, который равен нулю во всех узлах, кроме x_i, а в узле x_i равен 1. Вид этого полинома приведен на рис. 3.1 для случая $i = 3$. Произведение $y_i L_i(x)$ есть полином n-й степени, который во всех узлах, кроме x_i, равен нулю, а в узле x_i равен y_i. На рис. 3.1 изображен полином $y_3 L_3(x)$. Просуммировав произведения $y_i L_i(x)$ по всем i (по всем узлам), мы получим полином (3.4), удовлетворяющий постановке задачи интерполирования. Следовательно, построенный полином будет интерполяционным.

Полином Лагранжа можно записать в другом виде. Для этого рассмотрим полином

$$A(x) = (x - x_0)(x - x_1) \cdots (x - x_n) = \prod_{j=0}^{n} (x - x_j). \tag{3.5}$$

Это полином $(n+1)$-й степени со старшим коэффициентом, равным 1, и обращающийся в нуль во всех узлах. Выделим в нем множитель $(x - x_i)$ и найдем производную $A'(x_i)$. Получим

$$A'(x) = \{[(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)](x - x_i)\}' =$$
$$= [(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)]'(x - x_i) +$$
$$+ (x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n),$$

$$A'(x_i) = (x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n) = \prod_{\substack{j=0 \\ j\neq i}}^{n}(x_i - x_j).$$

Тогда полином влияния i-го узла $L_i(x)$ можно записать в виде

$$L_i(x) = \frac{A(x)}{A'(x_i)(x - x_i)}, \tag{3.6}$$

а сам полином Лагранжа (3.4) – в виде

$$P_n(x) = \sum_{i=0}^{n} y_i \frac{A(x)}{A'(x_i)(x - x_i)}. \tag{3.7}$$

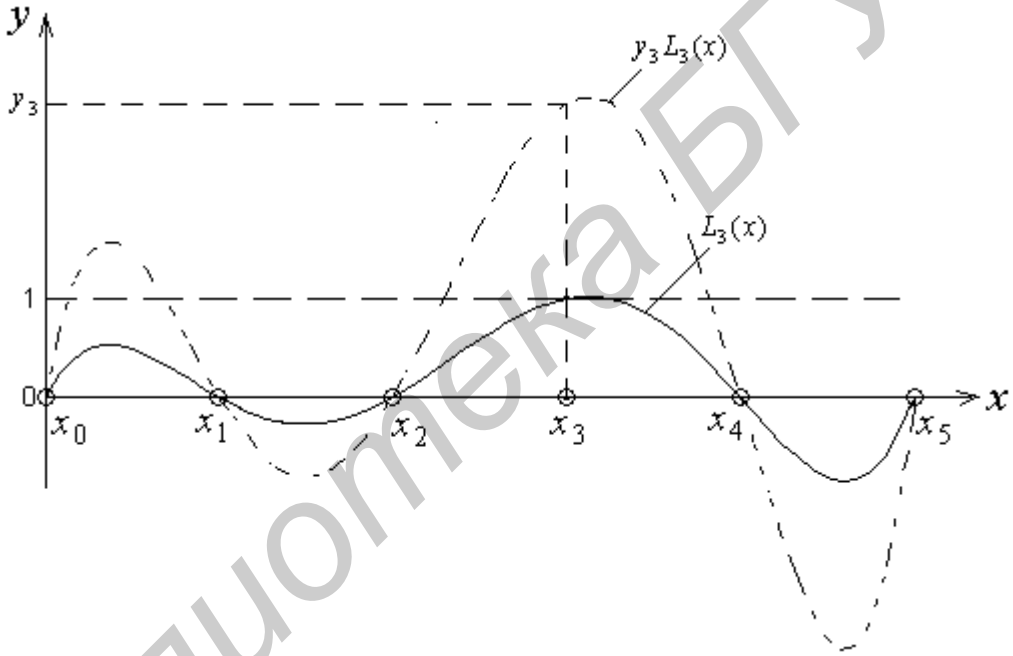

Рис. 3.1. К построению интерполяционного полинома Лагранжа

На рис. 3.2 представлена схема алгоритма расчета значения интерполяционного полинома Лагранжа в одной точке x по формуле (3.4). По этой схеме можно найти число операций, затрачиваемых на ее выполнение:

$$n_{\text{Лаг}} = \sum_{i=0}^{n}\sum_{j=0}^{n}(2_{\text{выч}} + 1_{\text{ум}} + 1_{\text{дел}}) + 1_{\text{ум}} + 1_{\text{сл}} =$$

$$= \sum_{i=0}^{n} 4(n+1) + 2 = (n+1)\big(4(n+1) + 2\big) = 4n^2 + 10n.$$

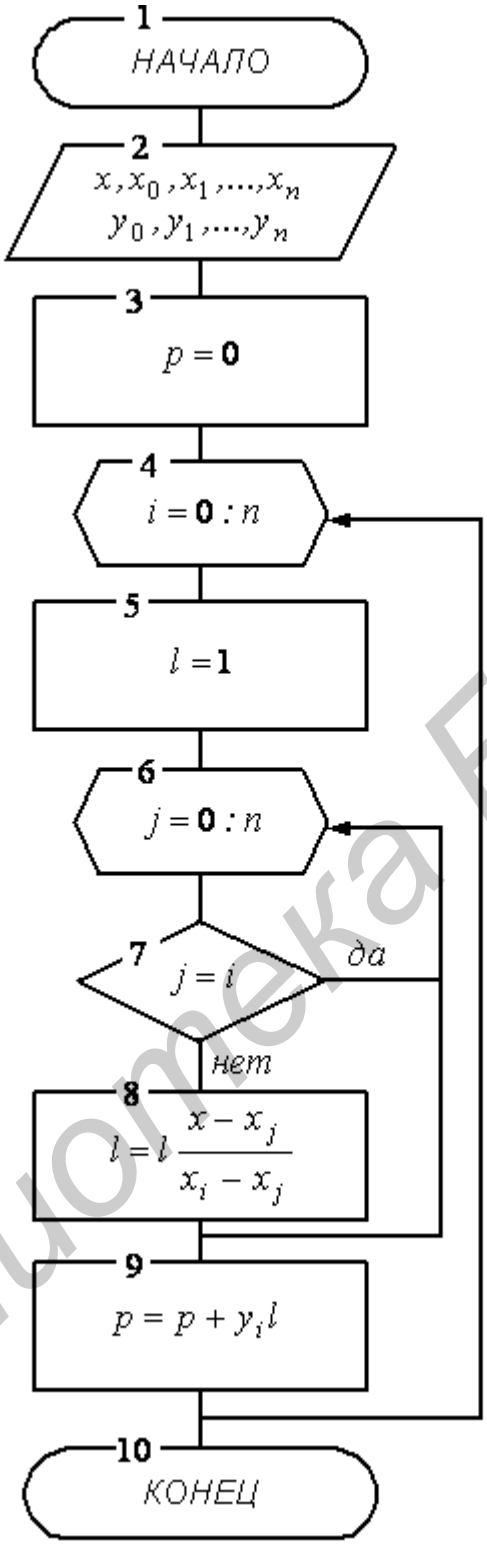

Рис. 3.2. Схема алгоритма расчета значения интерполяционного полинома Лагранжа в одной точке

Для расчета значений полинома n-й степени в k точках нужно выполнить

$$N_{\text{Лаг}} = 4kn^2 + 10kn \tag{3.8}$$

операций.

Пример 3.1. Построим полином Лагранжа для функции, заданной табл. 3.1.

Таблица 3.1

x_i	$x_0 = -2$	$x_1 = -1$	$x_2 = 3$
y_i	$y_0 = 12$	$y_1 = 6$	$y_2 = 2$

Поскольку мы имеем $n = 2$, то ожидаем интерполяционный полином второй степени. Общая формула этого полинома следующая:

$$P_2(x) = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}.$$

Подставив сюда данные из таблицы 3.1, получим

$$P_2(x) = 12 \frac{(x+1)(x-3)}{(-2+1)(-2-3)} + 6 \frac{(x+2)(x-3)}{(-1+2)(-1-3)} + 2 \frac{(x+2)(x+1)}{(3+2)(3+1)} =$$

$$= 12 \frac{x^2 - 2x - 3}{5} + 6 \frac{x^2 - x - 6}{-4} + 2 \frac{x^2 + 3x + 2}{20} =$$

$$= \frac{48x^2 - 96x - 144 - 30x^2 + 30x + 180 + 2x^2 + 6x + 4}{20} = x^2 - 3x + 2.$$

Легко обнаружить, что этот полином в узлах имеет значения, совпадающие с табличными.

3.4. Вычисление значений полиномов

Во многих случаях приходится вычислять значения полинома m-й степени

$$P_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{m-1} x^{m-1} + a_m x^m, \tag{3.9}$$

как это имеет место, например, в задаче интерполирования. Простейший способ сделать это – возвести x в соответствующую степень, умножить на коэффици-

ент и полученные произведения сложить. Расчет выражения $a_k x^k$ требует выполнения k умножений, так что расчет значения полинома этим способом можно выполнить за $1+2+...+m=\dfrac{m(m+1)}{2}$ умножений и m сложений. Однако существует более экономичный способ расчета значений полиномов, который называется правилом или схемой Горнера. Изложим его.

Разделим полином (3.9) на $(x-x_0)$. В результате деления получим полином степени $m-1$ и постоянный остаток b_0, так что полином $P_m(x)$ можно представить в виде

$$P_m(x)=(x-x_0)(b_1+b_2x+b_3x^2+...+b_{m-1}x^{m-2}+b_mx^{m-1})+b_0, \qquad (3.10)$$

где $b_0, b_1, ..., b_m$ – некоторые новые коэффициенты.

Из этой формулы видно, что

$$P_m(x_0)=b_0. \qquad (3.11)$$

Раскрыв скобки в правой части выражения (3.10), получим

$$P_m(x)=b_1x+b_2x^2+b_3x^3+...+b_{m-1}x^{m-1}+b_mx^m+$$
$$+b_0-x_0b_1-x_0b_2x-x_0b_3x^2-...-x_0b_{m-1}x^{m-2}-x_0b_mx^{m-1}. \qquad (3.12)$$

Приравнивая коэффициенты при одинаковых степенях x в выражениях (3.9) и (3.12), получим

$$a_m=b_m,$$
$$a_{m-1}=b_{m-1}-x_0b_m,$$
$$\cdots\cdots\cdots\cdots$$
$$a_j=b_j-x_0b_{j+1},$$
$$\cdots\cdots\cdots\cdots$$
$$a_0=b_0-x_0b_1.$$

Нами получены рекуррентные формулы следующего вида:

$$b_m=a_m,$$

$$b_j = a_j - x_0 b_{j+1},\ j = m-1,\ m-2,\ \ldots,\ 0. \qquad (3.13)$$

В конце расчетов по этим формулам мы получаем b_0, которое согласно (3.11) является значением полинома $P_m(x)$ в точке x_0. Формулы (3.13) и есть правило Горнера для расчета значения полинома. Для вычисления значения полинома степени m по правилу Горнера требуется m умножений и m сложений, т.е.

$$n_{\text{Гор}} = 2m \qquad (3.14)$$

операций, что значительно меньше изложенного выше способа. Кроме того, расчеты по правилу Горнера в ряде случаев имеют меньшую погрешность от округлений.

Для выяснения структуры расчетов по правилу Горнера рассмотрим полином третьей степени ($m = 3$) и выпишем формулы (3.13):

$$b_3 = a_3,$$

$$b_2 = a_2 - x_0 b_3 = a_2 + x_0 a_3,$$

$$b_1 = a_1 - x_0 b_2 = a_1 + x_0 (a_2 + x_0 a_3),$$

$$b_0 = a_0 - x_0 b_1 = a_0 + x_0 (a_1 + x_0 (a_2 + x_0 a_3)) = P_3(x_0).$$

Поскольку x_0 взято произвольно, то индекс 0 при x можно опустить. Из последних выражений видно, что правило Горнера для полинома произвольной степени $P_m(x)$ можно представить в виде

$$P_m(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{m-1} + x a_m))\ldots).$$

3.5. Вычислительная сложность задачи интерполирования

Как следует из изложенного ранее, возможны две схемы решения задачи интерполирования функции. Первая схема состоит в решении системы линейных алгебраических уравнений (3.3) относительно коэффициентов полинома и последующем вычислении значения полинома. Вторая схема состоит в расчете значения полинома по формуле Лагранжа (3.4). Будем называть ее схемой Ла-

гранжа. Какая из этих схем более экономична по затратам машинного времени? Для ответа на этот вопрос найдем число операций расчета для каждой схемы и сравним эти числа между собой.

Для первой схемы будем считать, что при решении СЛАУ используется метод Гаусса без выбора главного элемента, а при расчете значения полинома используется схема Горнера. Назовем эту схему схемой Гаусса – Горнера. Учитывая, что для интерполяционного полинома n-й степени решается СЛАУ из $n+1$ уравнений, получим, что для определения коэффициентов полинома затрачивается

$$n_{\text{б.в}} = \frac{2(n+1)^3}{3} + \frac{3(n+1)^2}{2} - \frac{(n+1)}{6} \approx \frac{2}{3}n^3 + \frac{7}{2}n^2 + \frac{31}{6}n \qquad (3.15)$$

операций (см. формулу (2.11)). Для расчета значения полинома степени n в одной точке по схеме Горнера затрачивается $n_{\text{Гор}} = 2n$ операций (см. формулу (3.14)), а значений в k точках – $N_{\text{Гор}} = 2kn$ операций. В итоге по схеме Гаусса – Горнера затрачивается

$$n_1 \approx \frac{2}{3}n^3 + \frac{7}{2}n^2 + \frac{31n}{6} + 2nk \qquad (3.16)$$

операций.

При расчетах по схеме Лагранжа на расчет интерполяционного полинома степени n в k точках затрачивается

$$n_2 = \left(4n^2 + 10n\right)k$$

операций (см. формулу (3.8)). Так как в формуле для n_2 при k стоит коэффициент $4n^2 + 10n$, а в формуле для n_1 при k стоит коэффициент $2n$, то ясно, что при большом k получим $n_2 > n_1$. Это значит, что при необходимости рассчитать интерполяционный полином в достаточно большом числе точек более выгодной является схема Гаусса – Горнера.

На рис. 3.3 представлены графики функций (3.15) и (3.16) в зависимости от k при $n = 10$. Видно, что схема Лагранжа более выгодна при расчете значения полинома 10-й степени не более чем в двух точках.

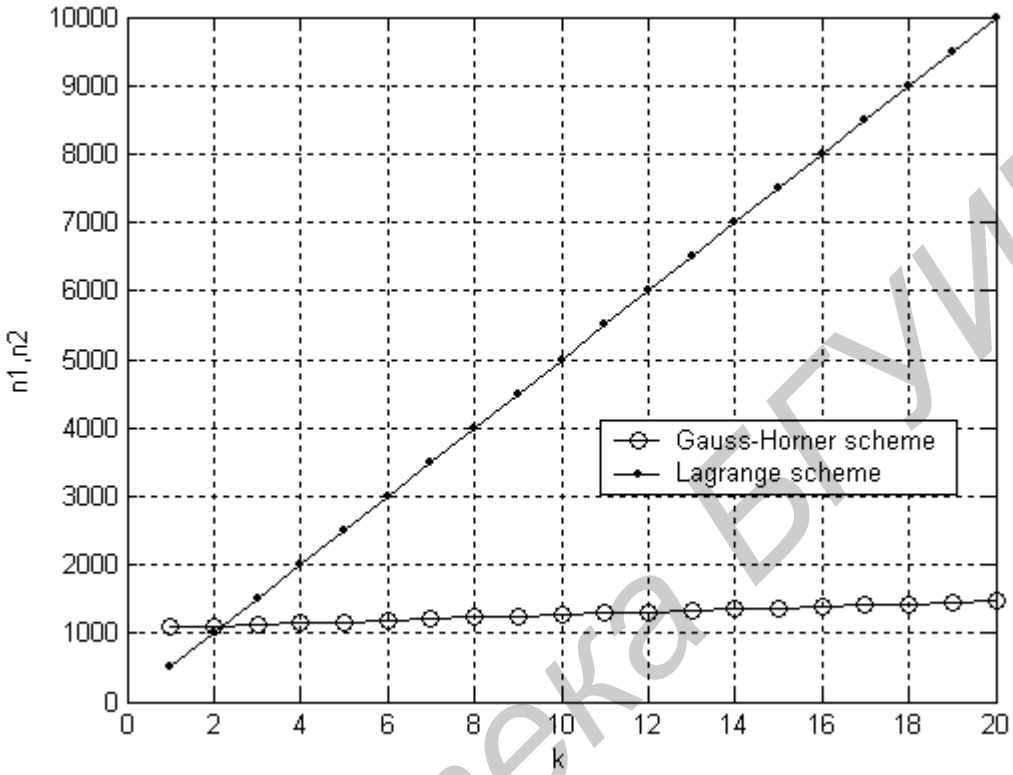

Рис. 3.3. Зависимости числа операций от числа точек расчета интерполяционного полинома по схемам Гаусса – Горнера и Лагранжа

3.6. Конечные и разделенные разности функции

Пусть $x_0, x_1, x_2, \ldots$ – точки действительной прямой и $y_i = f(x_i)$, $i = 0, 1, \ldots$, – значения функции $f(x)$ в этих точках. Назовем значения $f(x_i)$ конечными разностями нулевого порядка функции $f(x)$. Конечными разностями первого порядка функции $f(x)$ называются приращения

$$\Delta f(x_i) = f(x_{i+1}) - f(x_i),\ i = 0,1,\ldots\ .$$

Конечные разности второго порядка определяются как конечные разности от конечных разностей первого порядка:

$$\Delta^{(2)} f(x_i) = \Delta(\Delta f(x_i)) = \Delta f(x_{i+1}) - \Delta f(x_i) =$$

$$= f(x_{i+2}) - f(x_{i+1}) - f(x_{i+1}) + f(x_i) = f(x_{i+2}) - 2f(x_{i+1}) + f(x_i),\ i = 0,1,\ldots .$$

Конечные разности третьего порядка определяются как конечные разности от конечных разностей второго порядка:

$$\Delta^{(3)} f(x_i) = \Delta(\Delta^{(2)} f(x_i)),\ i = 0,1,\ldots .$$

Вообще конечные разности n-го порядка определяются с помощью следующего рекуррентного соотношения:

$$\Delta^{(n)} f(x_i) = \Delta(\Delta^{(n-1)} f(x_i)),\ i = 0,1,\ldots .$$

Если функция есть полином n-й степени

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n,$$

то можно показать, что его конечная разность n-го порядка есть величина постоянная ($\Delta^{(n)} P_n(x) = \text{const}$), а конечная разность $(n+1)$-го порядка равна нулю ($\Delta^{(n+1)} P_n(x) = 0$).

Введем теперь понятие разделенных разностей функции $f(x)$. Назовем значения $f(x_i)$ в точках x_i, $i = 0,1,\ldots$, разделенными разностями нулевого порядка функции $f(x)$. Разделенными разностями первого порядка функции $f(x)$ называются отношения

$$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i},\ i = 0,1,2,\ldots .$$

Разделенными разностями второго порядка функции $f(x)$ называются отношения

$$f(x_i, x_{i+1}, x_{i+2}) = \frac{f(x_{i+1}, x_{i+2}) - f(x_i, x_{i+1})}{x_{i+2} - x_i}\ i = 0,1,2,\ldots .$$

Вообще разделенные разности n-го порядка определяются через разделенные разности $(n-1)$-го порядка с помощью рекуррентного соотношения

$$f(x_i, x_{i+1}, \ldots, x_{i+n}) = \frac{f(x_{i+1}, \ldots, x_{i+n}) - f(x_i, \ldots, x_{i+n-1})}{x_{i+n} - x_i},\ n = 1,2,\ldots,\ i = 0,1,2,\ldots .$$

Разделенные разности $f(x_i, x_{i+1},..., x_{i+n})$ являются симметричными функциями своих аргументов, т.е. не меняются при их перестановке. Например,

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

Если $P_n(x)$ – полином степени n, то можно показать, что его разделенная разность $(n+1)$-го порядка тождественно равна нулю

$$P_n(x, x_0, x_1,..., x_n) = 0$$

для любой системы попарно различных точек $x, x_0, x_1,..., x_n$.

В случае равноотстоящих на величину h точек $x_0, x_1, x_2,...$ разделенные разности можно выразить через конечные разности. Действительно, легко заметить, что

$$f(x_i, x_{i+1}) = \frac{\Delta f(x_i)}{h},\ i = 0,1,2,...,$$

$$f(x_i, x_{i+1}, x_{i+2}) = \frac{\Delta^{(2)} f(x_i)}{2! h^2},\ i = 0,1,2,... .$$

В общем случае справедлива формула

$$f(x_i, x_{i+1},..., x_n) = \frac{\Delta^{(n-i)} f(x_i)}{(n-i)! h^{n-i}},\ i = 0,1,2,...,$$

которая при $i = 0$ приобретает вид

$$f(x_0, x_1,..., x_n) = \frac{\Delta^{(n)} f(x_0)}{n! h^n}. \qquad (3.17)$$

Из приведенного изложения очевидно, что конечные и разделенные разности являются прообразами производных. Действительно, производная n-го порядка получается при равномерном шаге h из конечной разности n-го порядка путем предельного перехода

$$f^{(n)}(x) = \lim_{h \to 0} \frac{\Delta^{(n)} f(x)}{h^n}.$$

3.7. Интерполяционный полином Ньютона

Пусть в точках $x_0, x_1, ..., x_n$ отрезка $[a,b]$ известны значения $f(x_0), f(x_1), ..., f(x_n)$ функции $f(x)$. Построим по этим данным интерполяционный полином $P_n(x)$ в форме полинома Ньютона. Построение основывается на том, что разделенные разности интерполяционного полинома и функции совпадают, т.е.

$$\begin{aligned} P_n(x_i) &= f(x_i), \\ P_n(x_0, x_1) &= f(x_0, x_1), \\ P_n(x_0, x_1, x_2) &= f(x_0, x_1, x_2), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ P_n(x_0, x_1, ..., x_n) &= f(x_0, x_1, ..., x_n). \end{aligned} \tag{3.18}$$

Кроме того, мы знаем, что

$$P_n(x, x_0, x_1, ..., x_n) = 0. \tag{3.19}$$

Запишем разделенную разность первого порядка для полинома $P_n(x)$:

$$P_n(x, x_0) = \frac{P_n(x) - P_n(x_0)}{x - x_0}.$$

Отсюда

$$P_n(x) = P_n(x_0) + P_n(x, x_0)(x - x_0). \tag{3.20}$$

Разделенная разность произвольного порядка имеет вид

$$P_n(x, x_0, ..., x_m) = \frac{P_n(x, x_0, ..., x_{m-1}) - P_n(x_0, x_1, ..., x_m)}{x - x_m},$$

откуда

$$P_n(x, x_0, ..., x_{m-1}) = P_n(x_0, x_1, ..., x_m) + P_n(x, x_0, ..., x_m)(x - x_m).$$

Последняя формула позволяет перейти от разделенной разности m-го порядка к разделенной разности более высокого $(m+1)$-го порядка. В частности,

$$P_n(x, x_0) = P_n(x_0, x_1) + P_n(x, x_0, x_1)(x - x_1),$$

$$P_n(x, x_0, x_1) = P_n(x_0, x_1, x_2) + P_n(x, x_0, x_1, x_2)(x - x_2).$$

Используя эти формулы, из формулы (3.20) последовательно получаем

$$P_n(x) = P_n(x_0) + P_n(x_0, x_1)(x - x_0) + P_n(x, x_0, x_1)(x - x_0)(x - x_1),$$

$$P_n(x) = P_n(x_0) + P_n(x_0, x_1)(x - x_0) + P_n(x_0, x_1, x_2)(x - x_0)(x - x_1) +$$

$$+ P_n(x, x_0, x_1, x_2)(x - x_0)(x - x_1)(x - x_2).$$

Продолжая этот процесс, в итоге получим формулу

$$P_n(x) = P_n(x_0) + P_n(x_0, x_1)(x - x_0) + P_n(x_0, x_1, x_2)(x - x_0)(x - x_1) +$$

$$+ P_n(x_0, x_1, x_2, x_3)(x - x_0)(x - x_1)(x - x_2) + \ldots +$$

$$+ P_n(x_0, x_1, \ldots, x_n)(x - x_0)(x - x_1)\cdots(x - x_{n-1}) +$$

$$+ P_n(x, x_0, \ldots, x_n)(x - x_0)(x - x_1)\cdots(x - x_n).$$

Учитывая равенства (3.18), (3.19), получаем полином Ньютона для неравноотстоящих узлов, выраженный через разделенные разности в начальной точке x_0:

$$P_n(x) = f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + \ldots +$$

$$+ f(x_0, x_1, \ldots, x_n)(x - x_0)(x - x_1)\cdots(x - x_{n-1}). \qquad (3.21)$$

Отметим, что при выводе формулы полинома Ньютона мы не предполагали, что узлы располагаются в порядке возрастания. Поэтому порядок узлов в выражении полинома Ньютона можно изменять. Изменив порядок узлов на обратный, мы получим следующее выражение полинома Ньютона, записанное через разделенные разности в конечной точке x_n:

$$P_n(x) = f(x_n) + f(x_n, x_{n-1})(x - x_n) + f(x_n, x_{n-1}, x_{n-2})(x - x_n)(x - x_{n-1}) + \ldots +$$

$$+ f(x_n, x_{n-1}, \ldots, x_0)(x - x_n)(x - x_{n-1})\cdots(x - x_0). \qquad (3.22)$$

Если точки $x_0, x_1, \ldots, x_n$ отстоят одна от другой на равном расстоянии h, то с учетом равенства (3.17) получаем полином Ньютона, выраженный через конечные разности в начальной точке x_0:

$$P_n(x) = f(x_0) + \frac{\Delta f(x_0)}{1!h}(x - x_0) + \frac{\Delta^{(2)} f(x_0)}{2!h^2}(x - x_0)(x - x_1) + \ldots +$$

$$+\frac{\Delta^{(n)} f(x_0)}{n! h^n}(x-x_0)(x-x_1)\cdots(x-x_{n-1}). \qquad (3.23)$$

Аналогичным образом получаем полином Ньютона, выраженный через конечные разности в конечной точке x_n:

$$P_n(x) = f(x_n) + \frac{\Delta f(x_n)}{1! h}(x-x_n) + \frac{\Delta^{(2)} f(x_n)}{2! h^2}(x-x_n)(x-x_{n-1}) + \ldots +$$

$$+\frac{\Delta^{(n)} f(x_n)}{n! h^n}(x-x_n)(x-x_{n-1})\cdots(x-x_0). \qquad (3.24)$$

Полиномы Ньютона удобны тогда, когда необходимо изменять количество узлов интерполирования. При увеличении количества узлов нет необходимости пересчитывать весь полином, достаточно прибавить к нему новые слагаемые, соответствующие новым узлам. В случае полинома Лагранжа при добавлении новых узлов приходится пересчитывать весь полином.

Полиномы (3.21), (3.23), полученные по разностям в начальной точке x_0, используются для интерполирования функции вблизи начальной точки x_0, а полиномы (3.22), (3.24), полученные по разностям в конечной точке x_n, – для интерполирования вблизи конечной точки x_n.

Пример 3.2. Построим интерполяционный полином Ньютона по данным примера 3.1 подразд. 3.3. Для этого по табл. 3.1 составим таблицу разделенных разностей (табл. 3.2).

Таблица 3.2

x_i	$f(x_i)$	$f(x_i, x_{i+1})$	$f(x_i, x_{i+1}, x_{i+2})$
-2	12		
		-6	
-1	6		1
		-1	
3	2		

Разделенные разности образуют верхнюю убывающую диагональ, содержащую разделенные разности в начальной точке x_0 (числа 12, –6, 1), и нижнюю возрастающую диагональ, содержащую разделенные разности в конечной точке x_n (числа 2, –1, 1). Используя разделенные разности в начальной точке x_0, получаем полином Ньютона

$$P_2(x) = f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) =$$

$$= 12 - 6(x + 2) + (x + 2)(x + 1) = 12 - 6x - 12 + x^2 + 3x + 2 = x^2 - 3x + 2.$$

Используя разделенные разности в конечной точке x_n, получаем полином

$$P_2(x) = f(x_2) + f(x_2, x_1)(x - x_2) + f(x_2, x_1, x_0)(x - x_2)(x - x_1) =$$

$$= 2 - (x - 3) + (x - 3)(x + 1) = 2 - x + 3 + x^2 - 2x - 3 = x^2 - 3x + 2.$$

В обоих случаях мы получили один и тот же полином, совпадающий с полиномом Лагранжа в примере 3.1.

3.8. Погрешность интерполирования

Абсолютная погрешность интерполяционного полинома Лагранжа – это разность $R_n(x) = f(x) - P_n(x)$, где $f(x)$ – интерполируемая функция, $P_n(x)$ – интерполяционный полином Лагранжа. Поскольку абсолютная погрешность интерполирования равна нулю в узлах интерполирования, то ее можно представить в виде

$$R_n(x) = f(x) - P_n(x) = kA(x), \qquad (3.25)$$

где $A(x)$ – полином (3.5). Будем считать, что на отрезке интерполирования $[a, b]$ функция $f(x)$ имеет производные до $(n+1)$-го порядка включительно. Введем вспомогательную функцию

$$u(x) = f(x) - P_n(x) - kA(x). \qquad (3.26)$$

Функция $u(x)$, очевидно, имеет $(n+1)$ корень в точках $x_0, x_1, ..., x_n$. Подберем коэффициент k в выражении (3.26) так, чтобы $u(x)$ имела $(n+2)$-й корень в некоторой точке x отрезка $[a,b]$, не совпадающей с узлами интерполирования. Для этого достаточно положить

$$u(x) = f(x) - P_n(x) - kA(x) = 0.$$

Так как в предполагаемой точке x $A(x) \neq 0$, то этот коэффициент вполне определяется последним уравнением. При таком значении коэффициента k функция $u(x)$ имеет $(n+2)$ корня на отрезке $[a,b]$ и обращается в нуль на концах каждого из следующих $(n+1)$ отрезков: $[x_0, x_1]$, $[x_1, x_2]$,…, $[x_i, x]$, $[x, x_{i+1}]$,…, $[x_{n-1}, x_n]$. Применяя к каждому из этих отрезков теорему Ролля[1], убеждаемся, что производная $u'(x)$ не менее $(n+1)$ раз обращается в нуль на $[a,b]$. Применяя теорему Ролля к производной $u'(x)$, убеждаемся, что вторая производная $u''(x)$ не менее n раз обращается в нуль на $[a,b]$. Продолжая эти рассуждения, приходим к заключению, что производная $u^{(n+1)}(x)$ хотя бы 1 раз обращается в нуль на $[a,b]$. Это значит, что существует точка ξ, для которой

$$u^{(n+1)}(\xi) = 0.$$

Из формулы (3.26) получаем

$$u^{(n+1)}(x) = f^{(n+1)}(x) - P_n^{(n+1)}(x) - kA^{(n+1)}(x).$$

Так как производные

$$P_n^{(n+1)}(x) = 0,\ A^{(n+1)}(x) = (n+1)!,$$

то

$$u^{(n+1)}(x) = f^{(n+1)}(x) - k(n+1)!,$$

а в точке $x = \xi$

[1] *Теорема Ролля.* Пусть функция $f(x)$, дифференцируемая в замкнутом промежутке $[a,b]$, обращается в нуль на концах промежутка. Тогда производная $f'(x)$ по меньшей мере один раз обращается в нуль внутри промежутка.

$$0 = f^{(n+1)}(\xi) - k(n+1)!$$

Отсюда

$$k = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

и из выражения (3.25) получим следующую формулу для абсолютной погрешности интерполяционного полинома Лагранжа:

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} A(x),\ \xi \in [a,b]. \qquad (3.27)$$

Из последнего выражения получаем оценку абсолютной погрешности интерполирования в виде

$$|f(x) - P_n(x)| \le \frac{M_{(n+1)}}{(n+1)!} |A(x)|, \qquad (3.28)$$

где $M_{(n+1)} = \max\limits_{x\in[a,b]} |f^{(n+1)}(x)|$ – максимальное по модулю значение $(n+1)$-й производной функции $f(x)$ на отрезке интерполирования $[a,b]$.

3.9. Полиномы Чебышева 1-го рода

Полиномы Чебышева 1-го рода встречаются в задаче наилучшего выбора узлов интерполирования, рассмотренной далее в подразд. 3.10.

Рассмотрим следующую задачу: среди всех полиномов $T_n(z)$ степени n со старшим коэффициентом, равным единице, найти такой, для которого величина

$$\max_{z\in[-1,1]} |T_n(z)|$$

является минимальной. Такой полином называется полиномом, наименее уклоняющимся от нуля на отрезке $[-1,1]$. Поставленная задача была решена русским математиком П. Л. Чебышевым. Найденный Чебышевым полином, наименее

уклоняющийся от нуля на отрезке $[-1,1]$, получил название полинома Чебышева 1-го рода.

Полиномом Чебышева 1-го рода степени n называется функция вида

$$T_n(z) = \cos(n \arccos z),\ n = 0,1,2,\ldots,\ z \in [-1,1].$$

Обозначив $\alpha = \arccos z$, получим

$$T_n(z) = \cos(n\alpha),$$

$$T_0(z) = \cos(0),$$

$$T_1(z) = \cos(\alpha),$$

$$T_{n-1}(z) = \cos((n-1)\alpha),$$

$$T_{n+1}(z) = \cos((n+1)\alpha).$$

Так как по формуле суммы косинусов

$$\cos((n-1)\alpha) + \cos((n+1)\alpha) = 2\cos(\alpha)\cos(n\alpha),$$

то справедливо соотношение $T_{n-1}(z) + T_{n+1}(z) = 2T_1(z)T_n(z)$. Отсюда получаем рекуррентную формулу для полиномов Чебышева:

$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z). \qquad (3.29)$$

По этой формуле, в частности, имеем:

$$T_0(z) = 1,$$

$$T_0(z) = 1,$$

$$T_2(z) = 2z^2 - 1,$$

$$T_3(z) = 4z^3 - 3z,$$

$$T_4(z) = 8z^4 - 8z^2 + 1,$$

$$T_5(z) = 16z^5 - 20z^3 + 5z.$$

Из рекуррентной формулы (3.29) следует, что коэффициент при старшей степени полинома $T_n(z)$ равен 2^{n-1}. Нормированным полиномом Чебышева 1-го рода степени n называется полином

$$\widehat{T}_n(z) = \frac{1}{2^{n-1}} T_n(z) = 2^{1-n} T_n(z). \qquad (3.30)$$

Коэффициент при старшей степени нормированного полинома Чебышева $\widehat{T}_n(z)$ равен единице. Если z_i, $i = \overline{0, n-1}$, – корни полинома $\widehat{T}_n(z)$, то

$$\widehat{T}_n(z) = \prod_{i=0}^{n-1}(z - z_i),\ T_n(z) = 2^{n-1}\prod_{i=0}^{n-1}(z - z_i). \qquad (3.31)$$

Из определения полинома $T_n(z)$ ясно, что $|T_n(z)| \le 1$. Тогда

$$|\widehat{T}_n(z)| \le 2^{1-n},\ |\widehat{T}_{n+1}(z)| \le 2^{-n}.$$

Можно доказать, что это максимальное значение на отрезке $[-1,1]$ достигается, т.е.

$$\max_{z\in[-1,1]} |\widehat{T}_{n+1}(z)| = 2^{-n}. \qquad (3.32)$$

Полиномы Чебышева 1-го рода ортогональны на отрезке $[-1,1]$ по весу

$$\rho(z) = \frac{1}{\sqrt{1-z^2}},$$

т.е.

$$\int_{-1}^{1} \frac{T_k(z)T_m(z)}{\sqrt{1-z^2}}\,dz = \begin{cases} 0, & k \ne m, \\ \pi/2, & k = m \ne 0, \\ \pi, & k = m = 0. \end{cases}$$

Они также являются ортогональными на системе точек $z_0, \ldots, z_{n-1}$, т.е.

$$\sum_{i=0}^{n-1} T_k(z_i)T_m(z_i) = 0,\ k \ne m.$$

3.10. Наилучший выбор узлов интерполирования

Из формулы (3.27) видно, что погрешность интерполирования зависит от двух множителей, один из которых $f^{(n+1)}(x)$ зависит от свойств интерполируемой функции и не поддается регулированию, а величина другого $A(x)$ оп-

ределяется исключительно выбором узлов интерполирования. Чаще всего узлы интерполирования располагают на отрезке интерполирования с равномерным шагом. Вместе с тем, выбирая неравномерную сетку узлов, можно увеличить точность интерполирования. Задача о наилучшем выборе узлов интерполирования была сформулирована и решена русским математиком П. Л. Чебышевым. Наилучшие узлы интерполирования выбираются равными корням полинома, наименее уклоняющегося от нуля на отрезке интерполирования. Действительно, полином n-й степени, наименее уклоняющийся от нуля на отрезке $[-1,1]$, – это нормированный полином Чебышева $\widehat{T}_n(z)$ (3.30), допускающий представление (3.31). Выполним линейное преобразование независимой переменной:

$$x = \frac{b-a}{2} z + \frac{a+b}{2}.$$

Это преобразование осуществляет взаимно-однозначное соответствие между $x \in [a,b]$ и $z \in [-1,1]$. Благодаря этому преобразованию каждому узлу $x_i \in [a,b]$ можно поставить в соответствие точку $z_i \in [-1,1]$ по формуле

$$x_i = \frac{b-a}{2} z_i + \frac{a+b}{2},\ i = \overline{0,n}, \tag{3.33}$$

в результате чего образующие полином $A(x)$ сомножители $(x - x_i)$ в формуле для погрешности интерполирования (3.28) преобразуются по формуле

$$(x - x_i) = \frac{b-a}{2}(z - z_i).$$

Следовательно,

$$A(x) = \prod_{i=0}^{n} (x - x_i) = \left(\frac{b-a}{2}\right)^{n+1} \prod_{i=0}^{n} (z - z_i). \tag{3.34}$$

Из последней формулы видно, что значение $\max\limits_{z \in [a,b]} |A(x)|$ будет минимальным, когда будет минимальным $\max\limits_{z \in [-1,1]} |\prod\limits_{i=0}^{n} (z - z_i)|$. Так как коэффициент при стар-

шей степени полинома $\prod_{i=0}^{n}(z-z_i)$ равен 1, то на основании представления (3.31) это есть нормированный полином Чебышева 1-го рода, а z_i – его корни.

Итак, наилучшие узлы интерполирования $z_0, z_1, ..., z_n$ на отрезке $[-1,1]$ выбираются из условия

$$\widehat{T}_{n+1}(z) = 0,$$

т.е. как решение уравнения

$$\cos((n+1)\arccos z) = 0.$$

Из этого уравнения получаем

$$(n+1)\arccos z = \frac{\pi}{2} + i\pi = \frac{(2i+1)\pi}{2},\ i = \overline{0,n},$$

$$\arccos z = \frac{(2i+1)}{2(n+1)}\pi,$$

так что наилучшие узлы интерполирования определяются формулой

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

Наилучшие узлы интерполирования на произвольном отрезке $[a,b]$ находятся по формуле (3.33).

Подчеркнем, что при таком выборе узлов мы сводим к минимуму максимальное значение погрешности интерполирования. Найдем это максимальное значение погрешности интерполирования в случае наилучшего выбора узлов интерполирования. Учитывая выражение (3.32) и формулу (3.34), получим

$$\max_{z\in[a,b]} |A(x)| = \left(\frac{b-a}{2}\right)^{n+1} \max_{z\in[-1,1]} |\widehat{T}_{n+1}(z)| = \frac{(b-a)^{n+1}}{2^{2n+1}}.$$

В итоге из формулы (3.28) для погрешности интерполирования при наилучшем выборе узлов будем иметь соотношение

$$\max_{z\in[a,b]} |f(x) - P_n(x)| \le \frac{M_{(n+1)}(b-a)^{n+1}}{(n+1)!2^{2n+1}}.$$

4. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

4.1. Постановка задачи численного интегрирования

Задача численного интегрирования состоит в том, чтобы найти численное значение определенного интеграла

$$I = \int_a^b f(x)dx, \qquad (4.1)$$

где $f(x)$ – функция, непрерывная на отрезке интегрирования $[a,b]$.

Формулы для решения этой задачи называются квадратурными. Квадратурная формула позволяет вместо точного значения интеграла (4.1) найти некоторое его приближенное значение $\widetilde{I}$. Разность точного и приближенного значений интеграла называется абсолютной погрешностью квадратурной формулы (или численного метода),

$$R = I - \widetilde{I}.$$

Квадратурные формулы используют для вычисления интеграла (4.1) значения $y_0 = f(x_0),\ y_1 = f(x_1),\ldots,\ y_n = f(x_n)$ функции $f(x)$ в точках $x_0, x_1, ..., x_n$ отрезка $[a,b]$. Квадратурная формула имеет вид

$$I \approx \sum_{i=0}^{n} c_i y_i,$$

где c_i – некоторые коэффициенты, которые называются весовыми.

Рассмотрим различные квадратурные формулы и их погрешности.

4.2. Метод прямоугольников

Как известно из курса высшей математики, часть плоскости, ограниченную сверху кривой $y = f(x),\ a \le x \le b,\ f(x) \ge 0$, снизу – осью Ox, с боков – прямыми $x = a$ и $x = b$ (рис. 4.1), называют криволинейной трапецией. При $f(x) \ge 0$

площадь криволинейной трапеции считается положительной, а при $f(x) \le 0$ – отрицательной. Если $a < b$, то определенный интеграл (4.1) представляет собой сумму площадей, заключенных между кривой $y = f(x)$, осью Ox и крайними прямыми $x = a$, $x = b$, взятых со знаком плюс там, где $f(x) \ge 0$, и со знаком минус там, где $f(x) \le 0$.

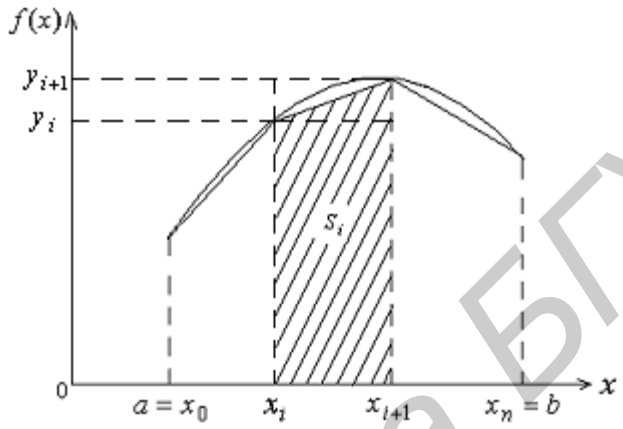

Рис. 4.1. К методам прямоугольников и трапеций

Разобьем отрезок интегрирования $[a,b]$ на n частей точками $x_0, x_1, \ldots, x_n$, как это показано на рис. 4.1, и заменим площадь криволинейной трапеции суммой площадей прямоугольников, построенных на частичных отрезках $[x_i, x_{i+1}]$, $i = \overline{0, n-1}$, как на основаниях. Если высоту i-го прямоугольника взять равной значению функции $f(x)$ в левой точке основания прямоугольника, т.е. принять

$$s_i = f(x_i)(x_{i+1} - x_i) = y_i h_i,$$

то получим квадратурную формулу левых прямоугольников:

$$\widetilde{I} = \sum_{i=0}^{n-1} s_i = \sum_{i=0}^{n-1} y_i h_i .$$

Для равноотстоящих на величину h узлов

$$h = \frac{b-a}{n},$$

формула левых прямоугольников имеет вид

$$\widetilde{I} = h\sum_{i=0}^{n-1} y_i = h(y_0 + y_1 + \ldots + y_{n-1}). \qquad (4.2)$$

Мы видим, что все весовые коэффициенты формулы левых прямоугольников в случае равноотстоящих узлов равны h, кроме коэффициента при y_n, который равен нулю.

Если интеграл на i-м отрезке $[x_i, x_{i+1}]$ заменить площадью прямоугольника с высотой, равной значению функции $f(x)$ в правой точке основания прямоугольника, т.е. принять

$$s_i = f(x_{i+1})(x_{i+1} - x_i) = y_{i+1}h_i,$$

то получим квадратурную формулу правых прямоугольников

$$\widetilde{I} = \sum_{i=0}^{n-1} s_i = \sum_{i=0}^{n-1} y_{i+1}h_i .$$

Для равноотстоящих на величину h узлов формула правых прямоугольников имеет вид

$$\widetilde{I} = h\sum_{i=0}^{n-1} y_{i+1} = h(y_1 + y_2 + \ldots + y_n). \qquad (4.3)$$

Мы видим, что все весовые коэффициенты формулы правых прямоугольников в случае равноотстоящих узлов равны h, кроме коэффициента при y_0, который равен нулю.

4.3. Погрешность метода прямоугольников

Абсолютная погрешность метода левых прямоугольников есть разность

$$R = \int_a^b f(x)dx - \sum_{i=0}^{n-1} h_i y_i .$$

Она складывается из погрешностей r_i, получаемых на частичных отрезках интегрирования,

$$R = \sum_{i=0}^{n-1} r_i ,$$

где

$$r_i = \int_{x_i}^{x_{i+1}} f(x)dx - h_i y_i ,\ i = \overline{0, n-1}.$$

Оценка погрешности определяется выражением

$$|R| \le \sum_{i=0}^{n-1} |r_i|.$$

Чтобы оценить r_i, рассмотрим функцию

$$r = \int_{\alpha}^{\beta} f(x)dx - f(\alpha)(\beta - \alpha) = \int_{\alpha}^{\beta} (f(x) - f(\alpha))dx ,\ \beta \ge \alpha .$$

Понятно, что если $\alpha = x_i$, $\beta = x_{i+1}$, то $r_i = r$. По теореме Лагранжа

$$f(x) - f(\alpha) = f'(\xi)(x - \alpha) ,\ \xi \in [\alpha, x],$$

и

$$r = \int_{\alpha}^{\beta} f'(\xi)(x - \alpha)dx = f'(\xi)\frac{(\beta - \alpha)^2}{2}.$$

Если обозначить

$$M_1 = \max_{x \in [a,b]} |f'(x)|,$$

то

$$|r| \le M_1 \frac{(\beta - \alpha)^2}{2}, \quad |r_i| \le \frac{M_1}{2} h_i^2 .$$

На этом основании получаем оценку для частичной погрешности при равномерном шаге интегрирования $h = \frac{b-a}{n}$:

$$|r_i| \le \frac{M_1}{2}h^2 = \frac{M_1}{2}\left(\frac{b-a}{n}\right)^2 = \frac{M_1(b-a)^2}{2n^2}.$$

Оценка абсолютной погрешности на всем отрезке интегрирования определяется выражением

$$|R| \le \sum_{i=0}^{n-1}|r_i| \le \frac{M_1(b-a)^2}{2n} = \frac{M_1(b-a)}{2}h.$$

Мы видим, что погрешность метода прямоугольников имеет тот же порядок, что и шаг интегрирования h. Поскольку для функций вида $f(x) = \text{const}$ $M_1 = 0$, то для таких функций формула прямоугольников является точной.

4.4. Метод трапеций

Заменим площадь криволинейной трапеции суммой площадей трапеций, построенных на частичных отрезках $[x_i, x_{i+1}]$, $i = \overline{0, n-1}$, (см. рис. 4.1),

$$\tilde{I} = \sum_{i=0}^{n-1} s_i,$$

где

$$s_i = \frac{(f(x_i) + f(x_{i+1}))(x_{i+1} - x_i)}{2} = \frac{y_i + y_{i+1}}{2}h_i.$$

В результате получим квадратурную формулу трапеций:

$$\tilde{I} = \sum_{i=0}^{n-1}\frac{y_i + y_{i+1}}{2}h_i.$$

Для равноотстоящих на величину h узлов формула трапеций имеет вид

$$\tilde{I} = \frac{h}{2}\sum_{i=0}^{n-1}(y_i + y_{i+1}) = \frac{h}{2}(y_0 + 2y_1 + 2y_2 + \ldots + 2y_{n-1} + y_n). \qquad (4.4)$$

4.5. Погрешность метода трапеций

Абсолютная погрешность метода трапеций при равномерном шаге интегрирования есть величина

$$R = \int_a^b f(x)dx - \frac{h}{2}\sum_{i=0}^{n-1}(y_i + y_{i+1}).$$

Она складывается из погрешностей r_i, получаемых на частичных отрезках интегрирования,

$$R = \sum_{i=0}^{n-1} r_i ,$$

где

$$r_i = \int_{x_i}^{x_{i+1}} f(x)dx - \frac{h}{2}(y_i + y_{i+1}),\ i = \overline{0, n-1}.$$

Оценка погрешности определяется выражением

$$|R| \le \sum_{i=0}^{n-1} |r_i|.$$

Чтобы оценить r_i, рассмотрим функцию

$$r(t) = \int_{x_i}^{t} f(x)dx - \frac{f(x_i) + f(t)}{2}(t - x_i),\ t \in [x_i, x_{i+1}].$$

Очевидно, что $r(x_i) = 0$, $r(x_{i+1}) = r_i$. Кроме того,

$$r'(t) = f(t) - \frac{1}{2}(f'(t)(t - x_i) + f(t) + f(x_i)) = \frac{f(t) - f(x_i)}{2} - \frac{f'(t)}{2}(t - x_i),$$

$$r'(x_i) = 0,$$

$$r''(t) = \frac{f'(t)}{2} - \frac{f''(t)}{2}(t - x_i) - \frac{f'(t)}{2} = -\frac{f''(t)}{2}(t - x_i),$$

$$r''(x_i) = 0.$$

Обозначим

$$M_2 = \max_{t \in [a,b]} |f''(t)|,\ t \in [a, b].$$

Тогда

$$|r''(t)| \le \frac{M_2}{2}(t - x_i).$$

По этой причине

$$|r'(t)| = |r'(t) - r'(x_i)| = \left|\int_{x_i}^{t} r''(t)dt\right| \le \int_{x_i}^{t} |r''(t)|\,dt \le \int_{x_i}^{t} \frac{M_2}{2}(t - x_i)dt = \frac{M_2}{4}(t - x_i)^2,$$

$$|r(t)| = |r(t) - r(x_i)| = \left|\int_{x_i}^{t} r'(t)dt\right| \le \int_{x_i}^{t} |r'(t)|\,dt \le \int_{x_i}^{t} \frac{M_2}{4}(t - x_i)^2\,dt = \frac{M_2}{12}(t - x_i)^3.$$

Следовательно,

$$|r_i| \le \frac{M_2}{12}(x_{i+1} - x_i)^3 = \frac{M_2}{12}h^3 = \frac{M_2}{12}\left(\frac{b-a}{n}\right)^3 = \frac{M_2(b-a)^3}{12n^3}.$$

Оценка абсолютной погрешности на всем отрезке интегрирования определяется выражением

$$|R| \le \sum_{i=0}^{n-1} |r_i| \le \frac{M_2(b-a)^3}{12n^2} = \frac{M_2(b-a)}{12}h^2,$$

где $M_2 = \max\limits_{x\in[a,b]} |f''(x)|$ – максимальное по модулю значение второй производной подынтегральной функции $f(x)$ на отрезке интегрирования $[a,b]$.

Мы видим, что погрешность метода прямоугольников имеет тот же порядок, что и h^2. Поскольку для функций вида $f(x) = c_0 + c_1 x$ имеем $f''(x) = 0$, $M_2 = 0$, то для таких функций (для полиномов первой степени) формула трапеций является точной.

4.6. Метод Симпсона

В основе метода Симпсона лежит следующая лемма.

Лемма. Если $F(x) = Bx^2 + Cx + D$ или $F(x) = Ax^3 + Bx^2 + Cx + D$, то

$$\int_a^{a+2h} F(x)dx = \frac{h}{3}(F(a) + 4F(a+h) + F(a+2h)). \qquad (4.5)$$

Выполним доказательство лишь для квадратичной параболы. Подставим функцию $F(x) = Bx^2 + Cx + D$ под интеграл в (4.5) и вычислим его. Получим

$$\int_a^{a+2h} F(x)dx = \int_a^{a+2h} (Bx^2 + Cx + D)dx = \frac{Bx^3}{3}\Big|_a^{a+2h} + \frac{Cx^2}{2}\Big|_a^{a+2h} + Dx\Big|_a^{a+2h} =$$

$$= \frac{B}{3}((a+2h)^3 - a^3) + \frac{C}{2}((a+2h)^2 - a^2) + D(a+2h-a) =$$

$$= \frac{B}{3}(a^3 + 6a^2h + 12ah^2 + 8h^3 - a^3) + \frac{C}{2}(a^2 + 4ah + 4h^2 - a^2) + 2Dh =$$

$$= \frac{h}{3}(6Ba^2 + 12Bah + 8Bh^2 + 6Ca + 6Ch + 6D). \qquad (4.6)$$

С другой стороны,

$$\frac{h}{3}F(a) = \frac{h}{3}(Ba^2 + Ca + D),$$

$$\frac{h}{3}4F(a+h) = \frac{h}{3}(4B(a+h)^2 + 4C(a+h) + 4D),$$

$$\frac{h}{3}F(a+2h) = \frac{h}{3}(B(a+2h)^2 + C(a+2h) + D).$$

Сложив три последних выражения, мы получим выражение (4.6), что и доказывает лемму.

Перейдем к изложению метода Симпсона. Разделим точки $x_0, x_1, ..., x_n$, разбивающие отрезок интегрирования $[a,b]$ на частичные отрезки с равномерным шагом h, на тройки точек x_0, x_1, x_2, $x_2, x_3, x_4, \ldots$, x_{n-2}, x_{n-1}, x_n. Для такого разбиения число отрезков n необходимо выбрать четным. На отрезке, определяемом i-й тройкой точек $x_{2i}, x_{2i+1}, x_{2i+2}$, $i = 0,1,2,...,(n-2)/2$, заменим подынтегральную функцию параболой второго порядка $Bx^2 + Cx + D$, проходящей через точки (x_{2i}, y_{2i}), (x_{2i+1}, y_{2i+1}), (x_{2i+2}, y_{2i+2}), и заменим точное значение ин-

теграла на этом отрезке интегралом s_i от полученной параболы. На основании леммы можно записать, что

$$s_i = \frac{h}{3}(y_{2i} + 4y_{2i+1} + y_{2i+2}).$$

Приближенное значение интеграла на всем отрезке интегрирования $[a,b]$ получим как сумму этих частичных интегралов:

$$\widetilde{I} = \sum_{i=0}^{(n-2)/2} s_i = \frac{h}{3} \sum_{i=0}^{(n-2)/2} (y_{2i} + 4y_{2i+1} + y_{2i+2}) =$$

$$= \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + ... + 2y_{n-2} + 4y_{n-1} + y_n). \qquad (4.7)$$

Мы видим, что в методе Симпсона крайние значения функции y_0, y_n суммируются с весом 1, значения y_i с нечетным номером i – с весом 4 и значения y_i с четным номером i – с весом 2. Суммы значений функции с различными весами удобно вычислить отдельно, а затем их сложить, умножить на шаг h и разделить на 3.

4.7. Погрешность метода Симпсона

Абсолютная погрешность формулы Симпсона (4.7) определяется выражением

$$R = \int_a^b f(x)dx - \frac{h}{3} \sum_{i=0}^{(n-2)/2} (y_{2i} + 4y_{2i+1} + y_{2i+2}).$$

Она складывается из частичных погрешностей r_i, полученных на каждой тройке точек, используемых для аппроксимации:

$$R = \sum_{i=0}^{(n-2)/2} r_i.$$

Частичная погрешность здесь определяется выражением

$$r_i = \int_{x_{2i}}^{x_{2i+2}} f(x)dx - \frac{h}{3}(y_{2i} + 4y_{2i+1} + y_{2i+2}).$$

Оценка общей погрешности имеет вид

$$|R| \le \sum_{i=0}^{(n-2)/2} |r_i|. \qquad (4.8)$$

Получим оценки сначала для частичной, а затем и для полной погрешностей. Для этого рассмотрим вспомогательную функцию

$$r(t) = \int_{\gamma-t}^{\gamma+t} f(x)dx - \frac{t}{3}\left[f(\gamma-t) + 4f(\gamma) + f(\gamma+t)\right],\ t \in [0,h].$$

При $\gamma = x_{2i} + h$ и $t = h$ эта функция совпадает с r_i. Кроме того, $r(0) = 0$. Найдем производные функции $r(t)$ до 3-го порядка включительно. Поскольку

$$\psi(t) = \int_{\gamma-t}^{\gamma+t} f(x)dx = \int_{\gamma-t}^{0} f(x)dx + \int_{0}^{\gamma+t} f(x)dx,$$

то, используя правила дифференцирования интеграла по нижнему и верхнему пределам, получим

$$\psi'(t) = f(\gamma-t) + f(\gamma+t),$$

а также

$$r'(t) = f(\gamma-t) + f(\gamma+t) -$$

$$-\left[\frac{1}{3}(f(\gamma-t) + 4f(\gamma) + f(\gamma+t)) + \frac{t}{3}(f'(\gamma+t) - f'(\gamma-t))\right] =$$

$$= \frac{2}{3}\left[f(\gamma-t) + f(\gamma+t)\right] - \frac{4}{3}f(\gamma) - \frac{t}{3}\left[f'(\gamma+t) - f'(\gamma-t)\right],$$

$$r'(0) = 0.$$

Выполняя последовательное дифференцирование, будем иметь

$$r''(t) = \frac{2}{3}\left[f'(\gamma+t) - f'(\gamma-t)\right] -$$

$$-\frac{1}{3}\left[f'(\gamma+t) - f'(\gamma-t)\right] - \frac{t}{3}\left[f''(\gamma+t) + f''(\gamma-t)\right] =$$

$$= \frac{1}{3}\left[f'(\gamma+t) - f'(\gamma-t)\right] - \frac{t}{3}\left[f''(\gamma+t) + f''(\gamma-t)\right],$$

$$r''(0) = 0,$$

$$r'''(t) = \frac{1}{3}\left[f''(\gamma + t) + f''(\gamma - t)\right] -$$

$$-\frac{1}{3}\left[f''(\gamma + t) + f''(\gamma - t)\right] - \frac{t}{3}\left[f'''(\gamma + t) - f'''(\gamma - t)\right] =$$

$$= -\frac{t}{3}\left[f'''(\gamma + t) - f'''(\gamma - t)\right],$$

$$r'''(0) = 0\,.$$

Применяя к $f'''(t)$ теорему Лагранжа, т.е. используя равенство

$$f'''(\gamma + t) - f'''(\gamma - t) = f^{(4)}(\xi)2t\,,\ \xi \in [\gamma - t, \gamma + t],$$

получим

$$r'''(t) = -\frac{2t^2}{3} f^{(4)}(\xi)\,.$$

Обозначая M_4 максимальное по модулю значение четвертой производной подынтегральной функции $f(x)$ на отрезке интегрирования $[a,b]$,

$$M_4 = \max_{x \in [a,b]} f^{(4)}(x),$$

получим оценку для 3-й производной функции $r(t)$:

$$|r'''(t)| \le \frac{2}{3} M_4 t^2\,.$$

Поскольку

$$r''(t) = r''(t) - r''(0) = \int_0^t r'''(x)dx\,,$$

то

$$|r''(t)| \le \int_0^t |r'''(x)|\,dx \le \int_0^t \frac{2}{3} M_4 x^2 dx = \frac{2}{9} M_4 t^3\,.$$

Аналогично получаем

$$r'(t) = r'(t) - r'(0) = \int_0^t r''(x)dx\,,$$

$$|r'(t)| \le \int_0^t |r''(x)|\,dx \le \int_0^t \frac{2}{9} M_4 x^3 dx = \frac{2}{36} M_4 t^4,$$

$$r(t) = r(t) - r(0) = \int_0^t r'(x)dx,$$

$$|r(t)| \le \int_0^t |r'(x)|\,dx \le \int_0^t \frac{2}{36} M_4 x^4 dx = \frac{2}{180} M_4 t^5.$$

Поскольку $r_i = r(h)$, то для частичной погрешности r_i получаем оценку

$$|r_i| \le \frac{2}{180} M_4 h^5 = \frac{2}{180} M_4 \frac{(b-a)^5}{n^5}.$$

Выражение (4.8) имеет $n/2$ слагаемых, следовательно, абсолютная погрешность формулы Симпсона будет оцениваться выражением

$$|R| \le \sum_{i=0}^{(n-2)/2} |r_i| \le \frac{n}{2}\frac{2}{180} M_4 \frac{(b-a)^5}{n^5} = \frac{M_4}{180}\frac{(b-a)^5}{n^4} = \frac{M_4(b-a)}{180} h^4.$$

Из последней формулы видно, что абсолютная погрешность метода Симпсона имеет тот же порядок, что и h^4. Формула Симпсона точна для полиномов третьей степени, поскольку для полинома 3-й степени $M_4 = 0$. Это является следствием того, что лемма (4.5) справедлива также для полинома 3-й степени.

4.8. Интерполяционные квадратурные формулы

Рассмотрим вычисление следующего интеграла:

$$I = \int_a^b f(x)\rho(x)dx, \qquad (4.9)$$

где $f(x)$ – некоторая достаточно гладкая функция, которую назовем подынтегральной; $\rho(x)$ – некоторая неотрицательная интегрируемая функция, которая называется весовой.

Этот интеграл является более общим по сравнению с рассматриваемым ранее интегралом (4.1). Интеграл вида (4.1) получим из (4.9) при весовой функции $\rho(x)=1$.

Для вычисления интеграла (4.9) применим следующий подход: выберем на отрезке $[a,b]$ $n+1$ точек $x_0,x_1,...,x_n$. В отличие от предыдущих методов не будем вычислять интегралы на частичных отрезках, а заменим подынтегральную функцию на всем отрезке $[a,b]$ интерполяционным полиномом (3.7), построенным по узлам $x_0,x_1,...,x_n$. В результате получим следующую квадратурную формулу:

$$\int_a^b f(x)\rho(x)dx \approx \int_a^b \sum_{i=0}^{n} \frac{A(x)\rho(x)dx}{(x-x_i)A'(x_i)} y_i = \sum_{i=0}^{n} y_i \int_a^b \frac{A(x)\rho(x)dx}{(x-x_i)A'(x_i)} = \sum_{i=0}^{n} c_i y_i , \quad (4.10)$$

где

$$c_i = \int_a^b \frac{A(x)\rho(x)}{(x-x_i)A'(x_i)} dx = \int_a^b L_i(x)\rho(x)dx , \quad (4.11)$$

$$y_i = f(x_i),$$

и $L_i(x)$ – полином влияния i-го узла (3.6).

Формула (4.10), в которой коэффициенты определяются по выражению (4.11), называется *интерполяционной квадратурной формулой*.

Рассмотрим вопрос погрешности интерполяционной квадратурной формулы. Заменяя подынтегральную функцию $f(x)$ полиномом Лагранжа $P_n(x)$, получаем абсолютную грешность $R(x)$ (3.27). Представим функцию $f(x)$ виде

$$f(x) = P_n(x) + R(x)$$

и найдем интеграл (4.9):

$$I = \int_a^b f(x)\rho(x)dx = \int_a^b (P_n(x)+R(x))\rho(x)dx = \int_a^b P_n(x)\rho(x)dx + \int_a^b R(x)\rho(x)dx =$$

$$= \sum_{i=0}^{n} c_i f(x_i) + \int_a^b R(x)\rho(x)dx .$$

Ясно, что второе слагаемое правой части этого выражения есть абсолютная погрешность интерполяционной квадратурной формулы:

$$R_I = \int_a^b R(x)\rho(x)dx.$$

Подставляя сюда выражение (3.27) для погрешности $R(x)$, получим следующую формулу абсолютной погрешности интерполяционной квадратурной формулы (4.10):

$$R_I = \frac{f^{(n+1)}(\xi)}{(n+1)!}\int_a^b A(x)\rho(x)dx \quad \xi \in [a,b].$$

Если обозначить

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

то для оценки абсолютной погрешности интерполяционной квадратурной формулы (4.10) получим выражение

$$|R_I| \le \frac{M_{(n+1)}}{(n+1)!}\int_a^b |A(x)|\rho(x)dx.$$

Из полученных выражений для погрешности видно, что интерполяционная квадратурная формула (4.10) точна для полиномов n-й степени, поскольку в этом случае $M_{(n+1)} = 0$. О такой квадратурной формуле говорят, что ее степень точности равна n.

Таким образом, квадратурная формула интерполяционного типа (4.10), построенная по $n+1$ узлам $x_0, x_1, ..., x_n$, является точной для полиномов n-й степени. Справедливо также и обратное утверждение, которое сформулируем в виде теоремы.

Теорема. Если квадратурная формула

$$\int_a^b f(x)\rho(x)dx \approx \sum_{i=0}^{n} d_i f(x_i) \qquad (4.12)$$

точна для полинома степени n, то она является интерполяционной.

Для доказательства достаточно показать, что если $f(x)$ – полином степени n, то коэффициенты d_i определяются формулой (4.11), т.е. $d_i = c_i$. Выберем в качестве такого полинома полином влияния k-го узла $L_k(x)$ (3.6). Тогда по условию теоремы квадратурная формула (4.12) для него будет точной, т.е.

$$\int_a^b L_k(x)\rho(x)dx = \sum_{i=0}^{n} d_i L_k(x_i),\ k = 0,1,...,n.$$

Полином влияния обладает свойством

$$L_k(x_i) = \begin{cases} 1, & если \quad i = k, \\ 0, & если \quad i \neq k, \end{cases}$$

в связи с чем предыдущий интеграл будет равен

$$\int_a^b L_k(x)\rho(x)dx = d_k.$$

Левая часть последнего равенства есть c_k, т.е. $d_k = c_k$, $k = 0,1,...,n$, и теорема доказана.

4.9. Интерполяционные квадратурные формулы наивысшей алгебраической степени точности (квадратурные формулы Гаусса)

Точность интерполяционной квадратурной формулы (4.10) можно существенно увеличить путем рационального выбора узлов $x_0, x_1, ..., x_n$. Задача получения более точной квадратурной формулы формулируется следующим образом: построить квадратурную формулу

$$\int_a^b f(x)\rho(x)dx \approx \sum_{i=0}^{n} c_i f(x_i), \qquad (4.13)$$

которая при заданном n была бы точной для полиномов возможно большей степени. Построение такой формулы заключается в надлежащем выборе коэффициентов $c_0, c_1, ..., c_n$ и узлов $x_0, x_1, ..., x_n$.

Для решения задачи потребуем, чтобы формула (4.13) была точна для любого полинома степени m. Это эквивалентно требованию, чтобы формула была точна для однородных полиномов $f(x) = x^k$, $k = 0,1,...,m$. Отсюда получаем условия

$$\sum_{i=0}^{n} c_i x_i^k = \int_a^b x^k \rho(x) dx, \; k = 0,1,...,m, \tag{4.14}$$

которые представляют собой систему $m+1$ нелинейных уравнений относительно $2(n+1)$ неизвестных $c_0, c_1, ..., c_n$, $x_0, x_1, ..., x_n$. Для того чтобы число уравнений равнялось числу неизвестных, необходимо выполнение равенства $m+1 = 2(n+1)$, откуда получаем $m = 2n+1$. Оказывается, что при $m = 2n+1$ эта система уравнений имеет единственное решение. Таким образом, существует квадратурная формула (4.13), точная для полиномов степени $2n+1$. Степень точности этой формулы более чем в 2 раза выше степени точности интерполяционной квадратурной формулы. Для получения этой формулы необходимо определить ее параметры $c_0, c_1, ..., c_n$, $x_0, x_1, ..., x_n$ путем решения системы нелинейных уравнений (4.14) при $m = 2n+1$. Однако существует другой путь определения этих параметров, который содержится в следующей теореме.

Теорема. Квадратурная формула (4.13) точна для любого полинома степени $2n+1$ тогда и только тогда, когда выполняются следующие условия:

1) формула является формулой интерполяционного типа, т.е. ее коэффициенты определяются выражением (4.11);

2) полином $A(x)$ в (4.11) ортогонален с весом $\rho(x)$ любому многочлену $q_k(x)$ степени k меньше $n+1$:

$$\int_a^b A(x) q_k(x) \rho(x) dx = 0, \; k = 0,1,...,n,$$

что равносильно требованию

$$\int_a^b A(x) x^k \rho(x) dx = 0, \; k = 0,1,...,n. \tag{4.15}$$

Теорему принимаем без доказательства.

Замечание 1. Полином $A(x)$, удовлетворяющий условиям (4.15), называется ортогональным полиномом на $[a,b]$ с весом $\rho(x)$.

Замечание 2. Поскольку полином $A(x)$ имеет вид $A(x)=(x-x_0)(x-x_1)\cdots$ $\ldots(x-x_n)$, то ясно, что точки $x_0,x_1,...,x_n$ являются корнями этого ортогонального полинома.

Таким образом, для любых n существует, притом единственная, квадратурная формула наивысшей алгебраической степени точности $2n+1$ вида (4.13). Узлы этой формулы совпадают с корнями ортогонального на $[a,b]$ с весом $\rho(x)$ полинома степени $n+1$, а коэффициенты определяются формулой (4.11). Интерполяционная квадратурная формула наивысшей алгебраической степени точности называется также квадратурной формулой Гаусса – Кристоффеля или квадратурной формулой Гаусса. Узлы $x_0,x_1,...,x_n$ и соответствующие им веса $c_0,c_1,...,c_n$ квадратурной формулы Гаусса рассчитываются заранее для различных весовых функций $\rho(x)$ и сводятся в таблицы [8, 9]. В следующих разделах приведены примеры квадратурных формул Гаусса.

4.9.1. Квадратурная формула Гаусса – Лежандра

Квадратурная формула Гаусса – Лежандра используется для вычисления интеграла с единичной весовой функцией $\rho(x)=1$ на конечном отрезке $[a,b]$, т.е. интеграла вида

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

Этот интеграл линейной заменой переменных

$$x=\frac{b-a}{2}z+\frac{b+a}{2}$$

приводится к виду

$$I = \int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f(\frac{b-a}{2}z + \frac{b+a}{z})dz .$$

На отрезке $[-1,1]$ ортогональны с весом $\rho(z) = 1$ полиномы Лежандра:

$$P_n(z) = \frac{1}{2^n n!}\frac{d^n}{dz^n}(z^2 - 1)^n,\ n = 0,1,2,... .$$

Первые четыре полинома Лежандра имеют вид

$$P_0(z) = 1,\ P_1(z) = z,$$

$$P_2(z) = \frac{3}{2}z^2 - \frac{1}{2},\ P_3(z) = \frac{5}{2}z^3 - \frac{3}{2}z .$$

Узлы $x_0, x_1, ..., x_n$ квадратурной формулы в этом случае выбираются равными корням полинома Лежандра $P_{n+1}(z)$. Квадратурная формула имеет вид

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=0}^{n} c_i f(\frac{b-a}{2}z_i + \frac{b+a}{2}).$$

В табл. 4.1 в качестве примера приведены узлы и коэффициенты для этой формулы при использовании от одного до четырех узлов.

Таблица 4.1

Узлы и коэффициенты квадратурной формулы Гаусса – Лежандра

Число узлов $n+1$	Значения узлов z_i	Значения весовых коэффициентов c_i
1	0,000000	2,000000
2	±0,577350	1,000000
3	0,000000 ±0,774597	8/9 5/9
4	±0,339981 ±0,861136	0,652145 0,347855

4.9.2. Квадратурная формула Гаусса – Лагерра

Квадратурная формула Гаусса – Лагерра используется для вычисления интеграла на положительной полуоси вида

$$I = \int_0^{\infty} f(x)e^{-x}dx,$$

т.е. интеграла с весовой функцией $\rho(x) = e^{-x}$. На полуоси $(0,\infty)$ ортогональны с весом $\rho(x) = e^{-x}$ полиномы Лагерра

$$L_n(x) = e^x \frac{d^n}{dx^n}(x^n e^{-x}),\ n = 0,1,2,\ldots\ .$$

Первые четыре полинома Лагерра имеют вид

$$L_0(x) = 1,\ L_1(x) = -x + 1,$$

$$L_2(x) = x^2 - 4x + 2,\ L_3(x) = -x^3 + 9x^2 - 18x + 6.$$

Узлы $x_0, x_1, \ldots, x_n$ квадратурной формулы в этом случае выбираются равными корням полинома Лагерра $L_{n+1}(x)$. Квадратурная формула имеет вид

$$\int_0^{\infty} f(x)e^{-x}dx \approx \sum_{i=0}^{n} c_i f(x_i).$$

В табл. 4.2 приведены узлы и коэффициенты для этой формулы при использовании от одного до четырех узлов.

Таблица 4.2

Узлы и коэффициенты квадратурной формулы Гаусса – Лагерра

Число узлов $n+1$	Значения узлов x_i	Значения весовых коэффициентов c_i
1	1,000000	1,000000
2	0,585786 3,414214	0,853553 0,146447
3	0,415775 2,294280 6,289945	0,711093 0,278518 0,0103893
4	0,322548 1,745761 4,536620 9,395071	0,603154 0,357419 0,0388879 0,000539295

4.9.3. Квадратурная формула Гаусса – Эрмита

Квадратурная формула Гаусса – Эрмита предназначена для вычисления интеграла на всей действительной прямой вида

$$I = \int_{-\infty}^{\infty} f(x)e^{-x^2}dx, \tag{4.16}$$

т.е. интеграла с весовой функцией $\rho(x) = e^{-x^2}$. На действительной прямой ортогональны с весом $\rho(x) = e^{-x^2}$ полиномы Эрмита:

$$H_n(x) = (-1)^n e^{x^2}\frac{d^n}{dx^n}(e^{-x^2}),\ n = 0,1,2,\ldots .$$

Первые четыре полинома Эрмита имеют вид

$$H_0(x) = 1,\ H_1(x) = 2x,$$

$$H_2(x) = 4x^2 - 2,\ H_3(x) = 8x^3 - 12x.$$

Узлы $x_0, x_1, \ldots, x_n$ квадратурной формулы в этом случае выбираются равными корням полинома Эрмита $H_{n+1}(x)$. Квадратурная формула имеет вид

$$\int_{-\infty}^{\infty} f(x)e^{-x^2}dx \approx \sum_{i=0}^{n} c_i f(x_i).$$

В табл. 4.3 приведены узлы и коэффициенты для этой формулы при использовании от одного до четырех узлов.

Таблица 4.3

Узлы и коэффициенты квадратурной формулы Гаусса – Эрмита

Число узлов $n+1$	Значения узлов x_i	Значения весовых коэффициентов c_i
1	0,000000	1,772454
2	±0,707107	0,886227
3	0,000000 ±1,224745	1,181636 0,295409
4	±0,524648 ±1,650680	0,804914 0,0813128

В вероятностных задачах часто приходится вычислять интегралы на всей действительной прямой вида

$$I_1 = \int_{-\infty}^{\infty} f(x) \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-a)^2}{2\sigma^2}} dx.$$

Последний интеграл заменой переменной $\frac{x-a}{\sigma\sqrt{2}} = z$, $x = a + \sigma\sqrt{2}\, z$, $dx = \sigma\sqrt{2}\, dz$ приводится к виду (4.16)

$$I_1 = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} f(a + \sigma\sqrt{2}\, z) e^{-z^2} dz,$$

и квадратурная формула для его вычисления имеет вид

$$I_1 = \int_{-\infty}^{\infty} f(x) \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-a)^2}{2\sigma^2}} dx \approx \frac{1}{\sqrt{\pi}} \sum_{i=0}^{n} c_i f(a + \sigma\sqrt{2}\, x_i).$$

Узлы x_i и коэффициенты c_i этой формулы берутся из табл. 4.3.

5. РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

5.1. Постановка задачи численного решения нелинейных уравнений

Требуется решить уравнение

$$f(x) = 0, \tag{5.1}$$

где $f(x)$ – нелинейная непрерывная функция.

Часто говорят, что требуется найти корни уравнения (5.1), корни или нули функции $f(x)$. Аналитическое решение данного уравнения можно получить лишь в простейших случаях (например в случае квадратного уравнения). В большинстве же случаев такие уравнения решаются численными методами. Полное решение поставленной задачи связано с решением следующих задач.

1. Установить количество, характер и расположение корней уравнения.

2. Найти приближенные значения корней, т.е. указать промежутки, в которых находятся корни (отделить корни).

3. Найти значения корней с требуемой точностью (уточнить корни).

Не будем подробно останавливаться на решении первых двух задач. Отметим лишь, что первая задача решается в основном аналитическими методами. Например, установлено, что полином n-й степени имеет ровно n корней, эти корни могут быть комплексными, а также кратными. Вторая задача решается в основном численными и графическими методами. Например, приближенные значения корней можно установить, построив график функции $f(x)$. Можно также составить таблицу значений функции. Если в двух соседних точках таблицы функция имеет разные знаки, то между этими точками расположено нечетное число корней (хотя бы один корень).

Предположим, что первые две задачи каким-то образом решены, и рассмотрим численные методы решения третьей задачи, т.е. численные методы уточнения корней.

5.2. Метод деления отрезка пополам

Метод деления отрезка пополам или метод половинного деления относится к семейству методов дихотомии. Предполагается известным отрезок $[a,b]$, на котором расположен искомый корень. В этом случае $f(a)f(b)<0$. Методы дихотомии состоят в последовательном делении отрезка $[a,b]$ на две части и отбрасывании той части, для которой установлено, что на ней корня нет. Метод половинного деления состоит в последовательном делении отрезка $[a,b]$ пополам и отбрасывании той половины, на которой корня нет, до тех пор, пока длина отрезка не станет малой (рис. 5.1). Этот алгоритм можно описать следующим образом:

1) если $b-a>\varepsilon$, то перейти к п. 2, иначе – перейти к п. 5;
2) найти $x=\dfrac{a+b}{2}$;
3) если $f(a)f(x)<0$, то положить $b=x$, иначе – положить $a=x$;
4) перейти к п. 1;
5) найти значение корня $x_0=\dfrac{a+b}{2}$ и прекратить вычисления.

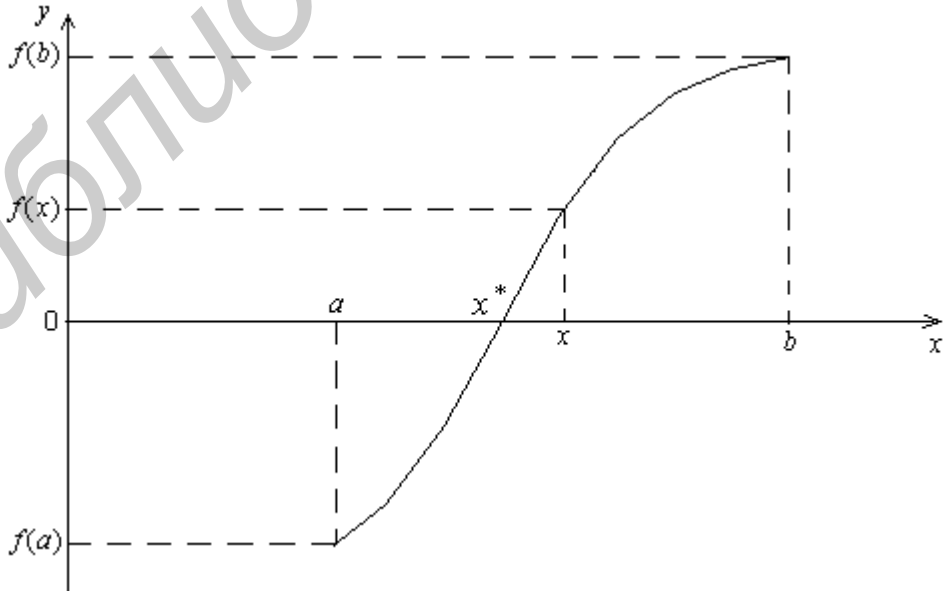

Рис. 5.1. Иллюстрация метода деления отрезка пополам

В описанном алгоритме число $\varepsilon > 0$ определяет допустимую абсолютную погрешность нахождения корня.

Выполнение пп. 1 – 4 называется одной итерацией. За одну итерацию метода половинного деления длина отрезка, на котором находится корень, уменьшается ровно вдвое, $l_1 = (b-a)/2$, так что отклонение полученного значения корня x_0 от истинного значения x^* будет удовлетворять неравенству

$$\left|x_0 - x^*\right| < l_1 = \frac{b-a}{2}.$$

После выполнения k итераций это отклонение будет подчиняться неравенству

$$\left|x_0 - x^*\right| < l_k = \frac{b-a}{2^k}.$$

Данное неравенство позволяет утверждать, что для непрерывной на отрезке $[a,b]$ функции при увеличении числа итераций k имеет место сходимость x_0 к x^*. Кроме того, оно позволяет подсчитать число итераций, достаточное для достижения заданной точности ε. Для этого необходимо разрешить относительно натурального k неравенство

$$\frac{b-a}{2^k} < \varepsilon.$$

5.3. Метод хорд

В рассмотренном выше методе половинного деления процесс деления отрезка на две части (дихотомии) был жестко фиксирован: эти части были равными. Естественно предположить, что в семействе методов дихотомии можно достичь несколько лучших результатов, если отрезок $[a,b]$, на котором расположен искомый корень, делить не пополам, а пропорционально значениям $f(a)$, $f(b)$ функции на концах отрезка. Это означает, что точку x на рис. 5.2 имеет смысл находить как абсциссу точки пересечения оси Ox с прямой, проходящей

через точки $A\big(a, f(a)\big)$ и $B\big(b, f(b)\big)$, иначе – с хордой AB. Уравнение прямой, проходящей через две данные точки A и B, имеет вид

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a}.$$

Отсюда, полагая $y = 0$, находим

$$x = a - \frac{f(a)(b - a)}{f(b) - f(a)}. \qquad (5.2)$$

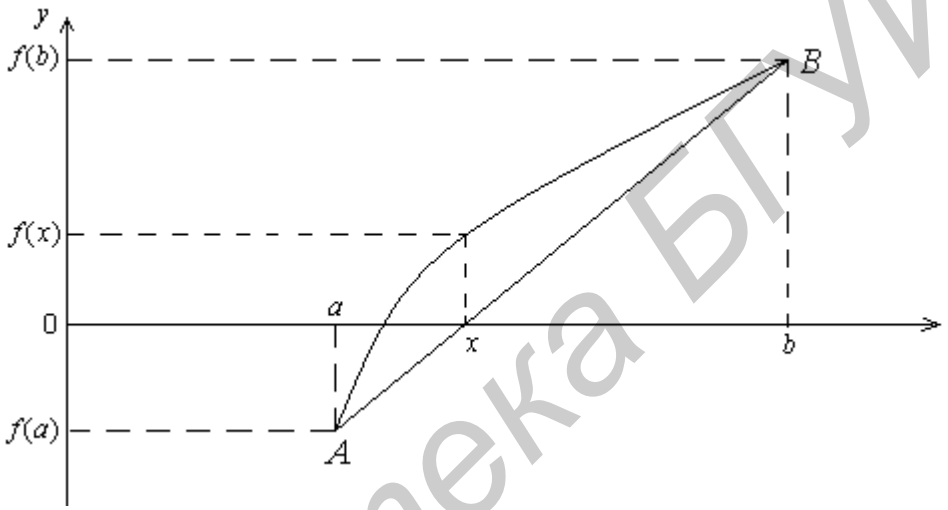

Рис. 5.2. Иллюстрация метода хорд

Данный подход может быть реализован в рамках алгоритма половинного деления, рассмотренного в разд. 5.2. Отличия нового алгоритма будут состоять в том, что в пунктах 2 и 5 алгоритма половинного деления x и x_0 следует рассчитывать по формуле (5.2). Метод, использующий формулу (5.2), получил название метода хорд.

Легко понять, что для линейной функции $f(x)$ метод хорд дает значение корня $x_0 = x^*$ всего за одну итерацию при любом отрезке $[a, b]$. Поэтому можно рассчитывать на его быструю сходимость для функций, близких к линейным. Однако если на функцию $f(x)$ не накладывать дополнительных ограни-

чений, может оказаться, что метод хорд будет проигрывать в скорости методу половинного деления. Например, скорость сходимости метода хорд будет низкой для функции, изображенной на рис. 5.3.

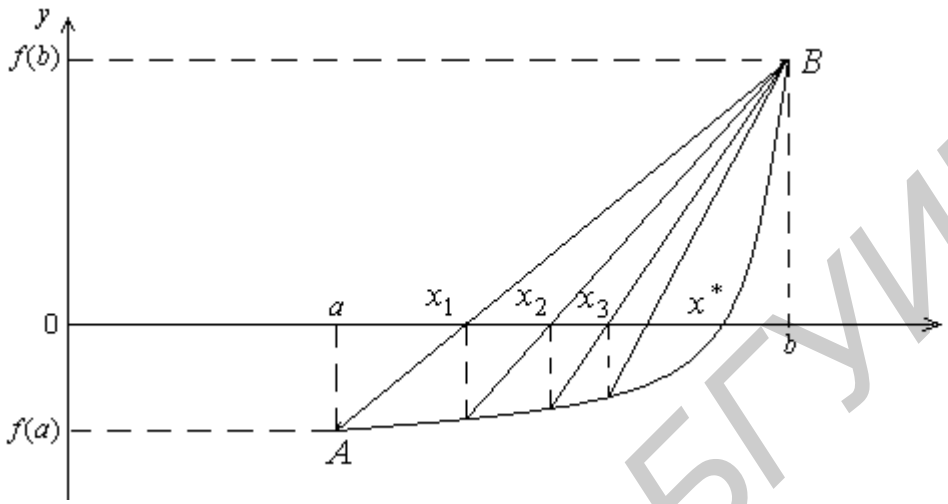

Рис. 5.3. Иллюстрация метода хорд с медленной сходимостью

5.4 Метод простой итерации

Преобразуем уравнение (5.1) следующим образом. Умножим обе части уравнения на некоторую функцию $\psi(x) \neq 0$. Получим эквивалентное уравнение

$$-\psi(x) f(x) = 0 .$$

Прибавив к обеим частям последнего уравнения x, получим уравнение

$$x = \varphi(x) , \tag{5.3}$$

где

$$\varphi(x) = x - \psi(x) f(x) . \tag{5.4}$$

Организуем итерационный вычислительный процесс по формуле

$$x_{n+1} = \varphi(x_n), \quad n = 1,2,\ldots, \tag{5.5}$$

где x_n – начальное значение корня уравнения; x_{n+1} – последующее уточненное значение корня.

Формула (5.5) и есть алгоритм метода простой итерации.

В обозначениях исходного уравнения (5.1) алгоритм простой итерации записывается в виде

$$x_{n+1} = x_n - \psi(x_n) f(x_n). \tag{5.6}$$

Выполнение расчетов по формуле (5.5) или (5.6) называется одной итерацией. Итерации прекращают, когда

$$\frac{|x_{n+1} - x_n|}{|x_{n+1}|} < \varepsilon, \tag{5.7}$$

где число $\varepsilon > 0$ определяет допустимую относительную погрешность нахождения корня.

Исследуем сходимость метода простой итерации. Пусть x^* – корень уравнения (5.3). В этом случае имеем равенство

$$x^* = \varphi(x^*). \tag{5.8}$$

Вычитая из (5.5) выражение (5.8), получим

$$x_{n+1} - x^* = \varphi(x_n) - \varphi(x^*).$$

Применим к правой части последнего выражения теорему Лагранжа, т.е. воспользуемся равенством

$$\varphi(x_n) - \varphi(x^*) = \varphi'(\xi)(x_n - x^*),$$

где ξ лежит между x_n и x^*. Получим

$$x_{n+1} - x^* = \varphi'(\xi)(x_n - x^*).$$

Обозначим

$$M_1 = \max_x |\varphi'(x)|,$$

где максимум берется по всем x, которые могут встретиться в процессе итераций. Тогда можно записать, что

$$|x_{n+1} - x^*| \le M_1 |x_n - x^*|.$$

Заменяя здесь n на $n-1$, получим

$$|x_n - x^*| \le M_1 |x_{n-1} - x^*|.$$

Два последних неравенства позволяют записать, что

$$|x_{n+1} - x^*| \le M_1^2 |x_{n-1} - x^*|.$$

Продолжая процесс уменьшения n и подстановок, в конечном итоге получим

$$|x_{n+1} - x^*| \le M_1^{n+1} |x_0 - x^*|,$$

где x_0 – первоначальное приближение к корню.

Очевидно, что если $M_1 < 1$, то $M_1^{n+1} \underset{n\to\infty}{\to} 0$ и $|x_{n+1} - x^*| \underset{n\to\infty}{\to} 0$, т.е. итерации сходятся к корню уравнения (5.3). Итак, мы показали, что метод простой итерации сходится, если

$$|\varphi'(x)| < 1 \qquad (5.9)$$

для значений x на всех итерациях.

Сходимость метода простой итерации иллюстрируется рис. 5.4.

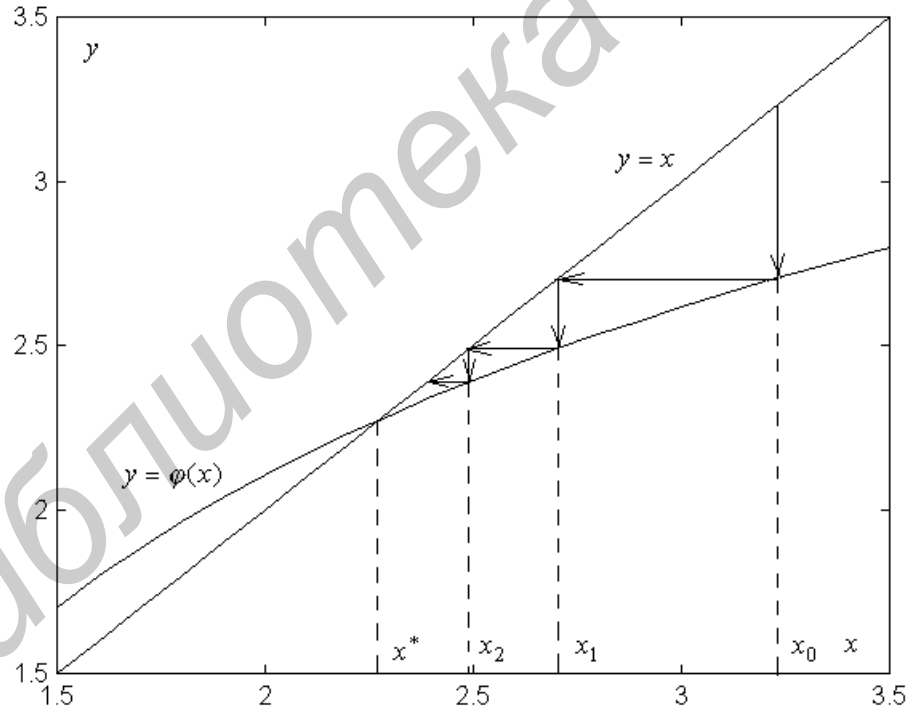

Рис. 5.4. Иллюстрация сходимости метода простой итерации

Корень x^* уравнения (5.3) определяется как точка пересечения кривой $y = \varphi(x)$ и прямой $y = x$. Выбирая первоначальное приближение x_0 и подставляя его в функцию $\varphi(x)$, получаем новое приближение x_1. Итерационный процесс изображен на рисунке 5.4 в виде прямых со стрелками. Горизонтальные прямые со стрелками соответствуют приравниванию значений функций $y = \varphi(x)$ и $y = x$, вертикальные – подстановкам значений x в функцию $y = \varphi(x)$. Функция $y = \varphi(x)$ в данном случае удовлетворяет условию сходимости (5.9), и от итерации к итерации получаем значения x_0, x_1, x_2, …, все более близкие к корню x^*.

Рис. 5.5 иллюстрирует расходимость метода простой итерации.

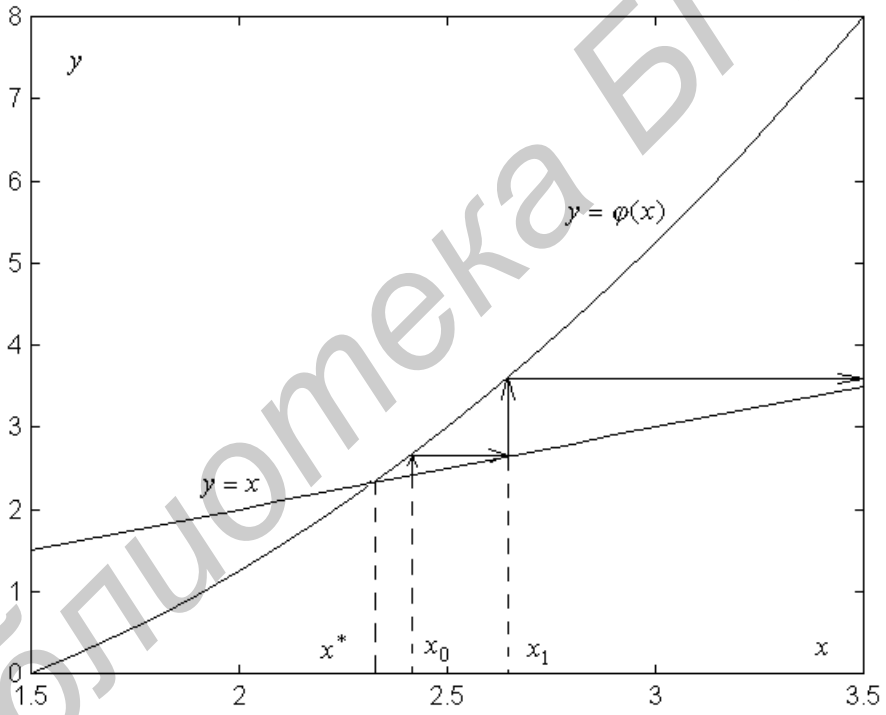

Рис. 5.5. Иллюстрация расходимости метода простой итерации

Функция $y = \varphi(x)$ в данном случае не удовлетворяет условию сходимости (5.9), ее производная по модулю больше единицы (она возрастает быстрее, чем функция $y = x$). В итоге итерации расходятся.

Если решается уравнение (5.1) и оно сводится к уравнению (5.3) с помощью подстановки (5.4), то условие сходимости (5.9) принимает вид

$$|(x-\psi(x)\cdot f(x))'|<1.$$

Выбором функции $\psi(x)$ в формуле для итераций (5.6) можно влиять на сходимость метода. В простейшем случае $\psi(x)=\text{const}$ со знаком плюс или минус.

Пример 5.1. Пусть требуется найти квадратный корень некоторого положительного числа a, т.е. найти $x=\sqrt{a}$. Решение этой задачи эквивалентно решению уравнения

$$x^2-a=0.$$

Воспользуемся для решения этого уравнения методом простой итерации. В данном случае $f(x)=x^2-a$, $\varphi(x)=x-kf(x)=x-k(x^2-a)$. Достаточное условие сходимости метода простой итерации $|\varphi'(x)|<1$ приобретает вид

$$|1-2kx|<1.$$

Отсюда мы можем найти значение коэффициента k, обеспечивающего сходимость метода. Получаем

$$-1<1-2kx<1,\ 0<kx<1.$$

Для положительных значений корня x имеем $0<k<1/x$. Например, для $0<x<100$ имеем $0<k<0{,}01$. Для нахождения корня используется следующая формула итераций:

$$x_{n+1}=x_n-k(x_n^2-a).$$

5.5. Метод Ньютона

Итерации по методу Ньютона осуществляются по формуле

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)},\ \ n=1,2,\ldots\ . \tag{5.10}$$

Итерации прекращаются, если выполняется условие (5.7).

Сравнивая формулу (5.10) с формулой итераций (5.6) для метода простой итерации, приходим к выводу, что метод Ньютона – это метод простой итерации с функцией

$$\psi(x) = \frac{1}{f'(x)}.$$

Получаем то же условие сходимости (5.9), что и для метода простой итерации, но с функцией $\varphi(x)$ вида

$$\varphi(x) = x - \frac{f(x)}{f'(x)}. \tag{5.11}$$

Условие (5.9) с учетом (5.11) можно записать в виде

$$|\varphi'(x)| = \left|\frac{f''(x)\cdot f(x)}{[f'(x)]^2}\right| < 1. \tag{5.12}$$

Последнее неравенство должно выполняться для значений x на всех возможных итерациях. Детальный анализ условия сходимости (5.12) позволяет сделать вывод, что метод Ньютона сходится, если первоначальное приближение выбрано достаточно близко к корню.

На рис. 5.6 приведена графическая иллюстрация метода Ньютона. В точке x_n к кривой $f(x)$ проводится касательная, точка пересечения которой с осью абсцисс дает нам новое приближение x_{n+1}. Затем в точке x_{n+1} опять проводится касательная и т.д.

Пример 5.2. Найдем, как и в примере п. 5.1, квадратный корень положительного числа a, т.е. найдем $x = \sqrt{a}$. Для этого решим уравнение $x^2 - a = 0$ методом Ньютона. В данном случае $f(x) = x^2 - a$, $f'(x) = 2x$, и формула итераций имеет вид

$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n}.$$

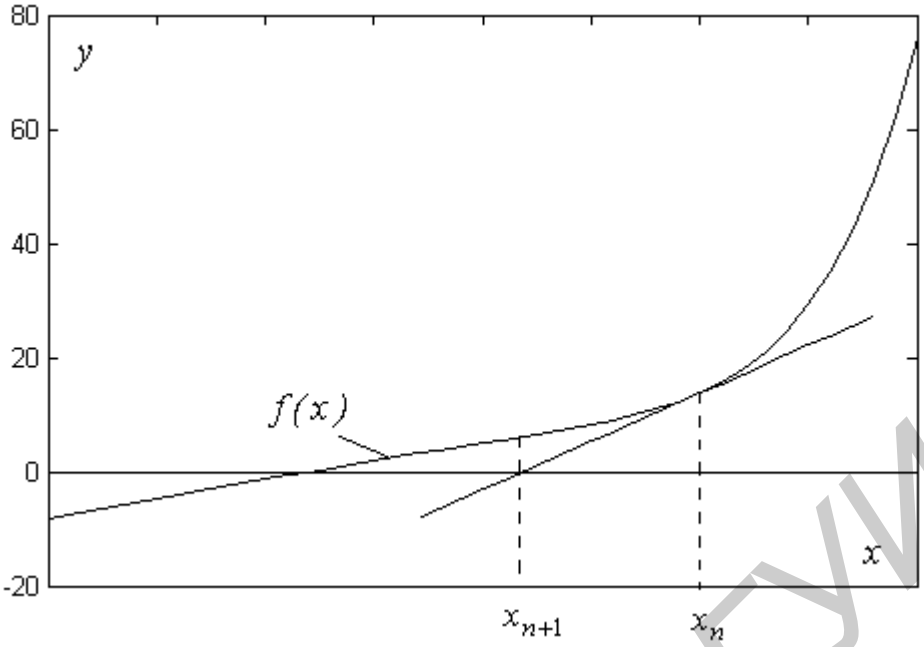

Рис. 5.6. Иллюстрация метода Ньютона

5.6. Метод секущих

В методе Ньютона, рассмотренном в подразд. 5.5, необходимо знать не только функцию $f(x)$, нуль которой мы ищем, но и производную этой функции $f'(x)$, что является недостатком данного метода. Если в методе Ньютона заменить производную на отношение приращений функции и аргумента,

$$f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}},$$

то получим следующую формулу итераций:

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})},\ n = 1,2,\ldots . \qquad (5.13)$$

Итерации прекращаются, если выполняется условие (5.7). Формула (5.13) называется методом секущих для нахождения корня уравнения (5.1). Графическая иллюстрация метода секущих приведена на рис. 5.7. Через две точки, определяемые приближениями x_{n-1} и x_n, проводится секущая и по ней определяется

новое приближение x_{n+1}. Затем проводится следующая секущая через точки, определяемые приближениями x_n и x_{n+1}, и т.д. В отличие от предыдущих методов этот метод является двухшаговым. Это значит, что для определения нового приближения необходимо иметь два предыдущих приближения.

Сравнивая формулы (5.2) и (5.13), приходим к выводу, что метод хорд и метод секущих определяются однотипными формулами. Однако это лишь внешнее сходство. Идеологии применения этих формул различны, что приводит к различиям в свойствах и скорости сходимости порождаемых этими методами последовательностей приближений к корню уравнения.

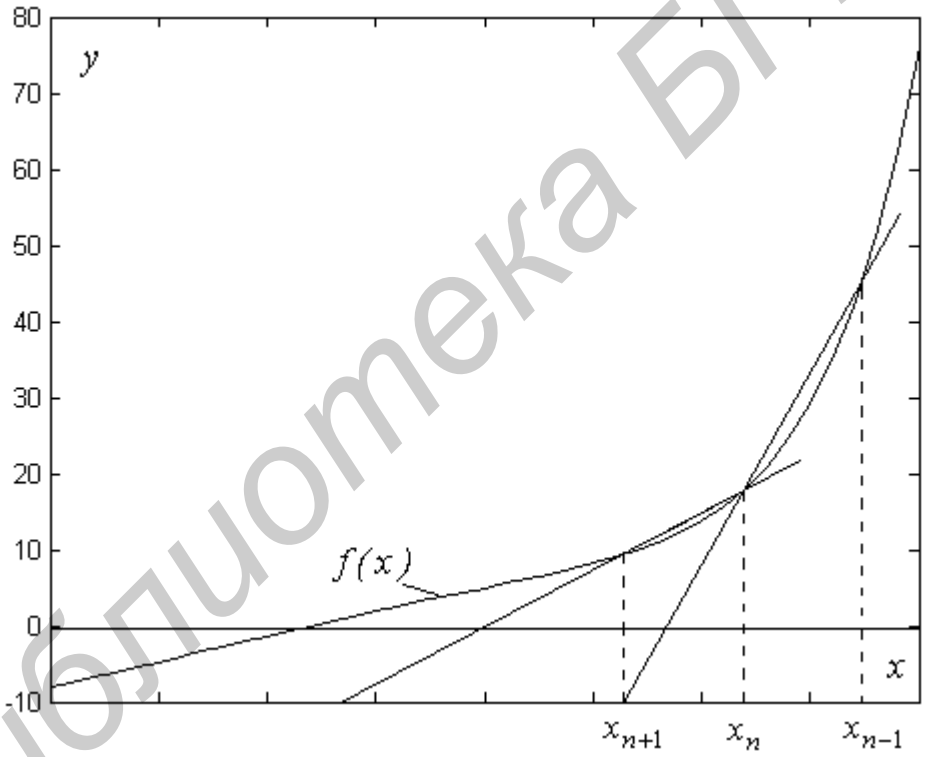

Рис. 5.7. Иллюстрация метода секущих

6. РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ (ОДУ)

6.1. Постановка задачи

Соотношение вида

$$F(x, y, y', y'', \ldots, y^{(n)}) = 0, \tag{6.1}$$

где F – некоторая функция независимой переменной x, функции $y = y(x)$ и ее производных $y' = y'(x) = \frac{dy(x)}{dx}$, $y'' = y''(x) = \frac{dy^2(x)}{dx^2}$, …, $y^{(n)} = \frac{d^n y(x)}{dx^n}$, называется обыкновенным дифференциальным уравнением n-го порядка. Решить уравнение – значит найти функцию $y(x)$, превращающую равенство (6.1) в тождество. Существует понятие *общего* и *частного* решения этого дифференциального уравнения. *Общее* решение (общий интеграл) – это формула, дающая все решения данного уравнения. Обычно общее решение обыкновенного дифференциального уравнения n-го порядка (6.1) зависит от n постоянных $C_1, C_2, \ldots, C_n$, которые могут выбираться произвольно. Решение, которое не зависит от произвольных постоянных, называется *частным* решением дифференциального уравнения (частным интегралом). График каждого частного решения называется *интегральной кривой*. Чаще всего для обыкновенного дифференциального уравнения (6.1) формулируется так называемая задача Коши, когда дополнительно к уравнению (6.1) задают значения функции и ее производных до $(n-1)$-го порядка в некоторой точке x_0. Эти дополнительные данные называются начальными условиями. Наличие начальных условий позволяет получить частное решение дифференциального уравнения.

Процесс решения дифференциального уравнения называется его *интегрированием*. Интегрирование дифференциального уравнения вовсе не означает, что этот процесс сводится к вычислению интеграла. Если же решение диффе-

ренциального уравнения действительно свелось к вычислению интеграла, то говорят, что уравнение решено *в квадратурах*.

Методы решения дифференциальных уравнений бывают точные, приближенные и численные. Точные методы дают решение, которое можно выразить в элементарных функциях. Получить точное решение дифференциального уравнения можно не всегда. Например, решение уравнения $y' = x^2 + y^2$ не выражается в элементарных функциях. Приближенные методы дают решение в виде некоторой последовательности функций $y_m(x)$, сходящейся к решению $y(x)$ при $m \to \infty$. Численные методы дают решение в виде таблицы значений функции $y(x)$. В рамках данной работы мы будем заниматься численными методами решения дифференциальных уравнений и рассмотрим методы решения дифференциального уравнения 1-го порядка

$$y' = f(x, y) \tag{6.2}$$

с начальным условием $y(x_0) = y_0$.

6.2. Метод рядов Тейлора

Пусть требуется решить дифференциальное уравнение 1-го порядка (6.2). Запишем разложение решения в ряд Тейлора в окрестности некоторой точки x_m:

$$y(x) = y_m + y'_m(x - x_m) + \frac{y''_m}{2!}(x - x_m)^2 + \frac{y'''_m}{3!}(x - x_m)^3 + \ldots, \tag{6.3}$$

где $y_m = y(x_m)$ – значение функции $y(x)$ в точке x_m; $y'_m = y'(x_m)$, $y''_m = y''(x_m)$, $y'''_m = y'''(x_m)$ – значения производных функции $y(x)$ в точке x_m.

Предположим, что мы нашли приближенное решение уравнения (6.2) в точках x_0, $x_1, \ldots, x_m$, расположенных на расстоянии h друг от друга, так что $x_i = x_0 + ih$, $i = 0,1,2,\ldots$. Тогда мы можем найти приближенное решение в точке x_{m+1}, подставив в выражение (6.3) $x = x_{m+1} = x_m + h$:

$$y_{m+1} = y_m + hy'_m + \frac{h^2}{2!} y''_m + \frac{h^3}{3!} y'''_m + \dots .$$

Чтобы воспользоваться этим выражением, нам необходимо ограничиться в нем конечным числом слагаемых и получить необходимые производные искомой функции, поскольку из постановки задачи они не известны. Ограничившись тремя слагаемыми, получим

$$y_{m+1} = y_m + hy'_m + \frac{h^2}{2!} y''_m + O(h^3), \qquad (6.4)$$

где $O(h^3)$ означает, что в отброшенные члены ряда h входит в степени, не ниже третьей. Покажем, как найти производные y'_m, y''_m. Непосредственно из дифференциального уравнения (6.2) получаем, что

$$y'_m = f(x_m, y_m). \qquad (6.5)$$

Дифференцируя $f(x, y)$ в правой части выражения (6.2) как сложную функцию, явно зависящую от x, получим

$$y''(x) = \frac{\partial}{\partial x} f(x,y) + \frac{\partial}{\partial y} f(x,y) \frac{dy}{dx} = \frac{\partial}{\partial x} f(x,y) + f(x,y) \frac{\partial}{\partial y} f(x,y).$$

Обозначив

$$\frac{\partial}{\partial x} f(x,y) = f'_x(x,y), \quad \frac{\partial}{\partial y} f(x,y) = f'_y(x,y)$$

и предположив, что $x = x_m$, $y = y_m$, получим

$$y''_m = y''(x_m) = f'_x(x_m, y_m) + f(x_m, y_m) f'_y(x_m, y_m). \qquad (6.6)$$

Подставив производные (6.5), (6.6) в выражение (6.4), получим следующую формулу решения уравнения (6.2) с помощью метода рядов Тейлора:

$$y_{m+1} = y_m + hf(x_m, y_m) + \frac{h^2}{2!}\left(f'_x(x_m, y_m) + f(x_m, y_m) f'_y(x_m, y_m) \right). \qquad (6.7)$$

Формула метода рядов Тейлора (6.7) характеризуется следующим. Она является одношаговой, так как для вычисления нового значения функции y_{m+1} нам необходимо знать лишь предыдущее значение функции y_m. Абсолютная по-

грешность формулы имеет порядок h^3, поскольку при ее получении мы отбросили в разложении Тейлора члены с h в кубе и выше. К ее недостаткам следует отнести необходимость получения производных $f'_x(x,y)$, $f'_y(x,y)$ правой части решаемого уравнения (6.2) и необходимость расчета значений трех функций $y'_m = f(x_m, y_m)$, $f'_x(x_m, y_m)$, $f'_y(x_m, y_m)$ для получения нового значения функции y_m. Ниже будут рассмотрены другие методы, которые позволяют решить уравнение (6.2) с той же точностью, что и формула (6.7), но с меньшими вычислительными затратами.

6.3. Метод Эйлера

Этот метод решения уравнения (6.2) состоит в последовательных расчетах по формуле

$$y_{m+1} = y_m + h \cdot f(x_m, y_m), \tag{6.8}$$

начиная с точки (x_0, y_0), заданной начальными условиями x_0, $y(x_0) = y_0$. Здесь h – шаг интегрирования по независимой переменной x. Формула Эйлера (6.8) получается из формулы (6.7) метода рядов Тейлора удержанием в правой части лишь двух слагаемых.

Геометрический смысл формулы Эйлера (6.3) иллюстрируется на рис. 6.1. В точке $A = (x_m, y_m)$ проводится касательная к интегральной кривой $y(x)$, новое значение функции y_{m+1} определяется как точка B пересечения касательной с вертикальной прямой $x = x_{m+1} = x_m + h$. Следующая касательная проводится в точке $B = (x_{m+1}, y_{m+1})$. Из рисунка видно, что при возрастании переменной x погрешность в определении функции может накапливаться. Абсолютная погрешность формулы Эйлера имеет порядок h^2, что видно из формулы (6.7).

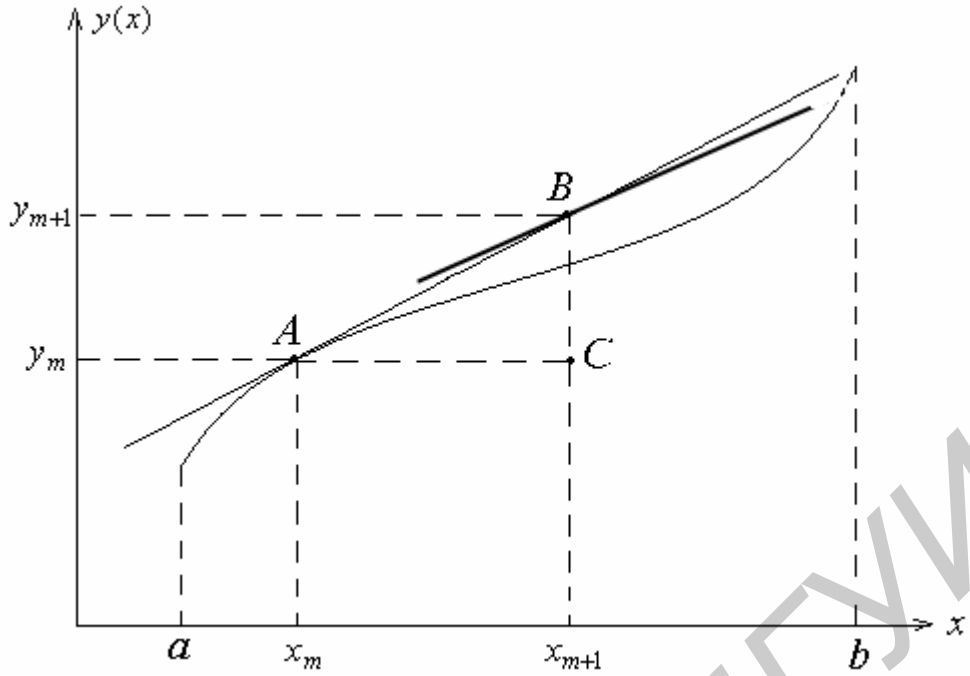

Рис. 6.1. Иллюстрация метода Эйлера

6.4. Метод Рунге – Кутта 2-го порядка

Этот метод состоит в том, чтобы организовать последовательные расчеты по формулам

$$k_1 = f(x_m, y_m),$$
$$k_2 = f(x_m + b_1 h, y_m + b_2 h k_1), \qquad (6.9)$$
$$y_{m+1} = y_m + h(a_1 k_1 + a_2 k_2),$$

начиная с начальных условий (x_0, y_0). Здесь $h = x_{m+1} - x_m$ – шаг по переменной x, постоянный при всех m, a_1, a_2, b_1, b_2 – некоторые коэффициенты, которые надлежит выбрать. Обычно коэффициенты a_1, a_2, b_1, b_2 выбираются путем сравнения формул (6.9) по точности с формулами метода рядов Тейлора.

Получим формулы (6.9), имеющие ту же точность, что и формула рядов Тейлора (6.7). Для этого запишем формулы (6.9) в виде одной формулы:

$$y_{m+1} = y_m + h\left[a_1 f(x_m, y_m) + a_2 f\left(x_m + b_1 h, y_m + b_2 h f(x_m, y_m)\right)\right]. \qquad (6.10)$$

Далее представим правую часть нашего уравнения (6.2), т.е. функцию двух аргументов $f(x,y)$, линейной частью ряда Тейлора в окрестности точки (x_m, y_m):

$$f(x,y) = f(x_m, y_m) + f_x^{(1)}(x_m, y_m)(x - x_m) + f_y^{(1)}(x_m, y_m)(y - y_m) + \ldots$$

и положим здесь

$$x = x_m + b_1 h,$$

$$y = y_m + b_2 h f(x_m, y_m).$$

Получим

$$f(x_m + b_1 h, y_m + b_2 h f(x_m, y_m)) =$$

$$= f(x_m, y_m) + f_x^{(1)}(x_m, y_m) b_1 h + f_y^{(1)}(x_m, y_m) b_2 h f(x_m, y_m) + O(h^2). \qquad (6.11)$$

Подставляя (6.11) в правую часть выражения (6.10), будем иметь

$$y_{m+1} = y_m + h(a_1 + a_2) f(x_m, y_m) +$$

$$+ a_2 b_1 h^2 f_x^{(1)}(x_m, y_m) + a_2 b_2 h^2 f_y^{(1)}(x_m, y_m) f(x_m, y_m) + O(h^3). \qquad (6.12)$$

Приравняем формулы (6.7) и (6.12). Для совпадения членов с множителем $hf(x_m, y_m)$ необходимо выполнить равенство

$$a_1 + a_2 = 1. \qquad (6.13)$$

Для совпадения членов с множителем $h^2 f_x^{(1)}(x_m, y_m)$ необходимо, чтобы

$$a_2 b_1 = \frac{1}{2}, \qquad (6.14)$$

а для совпадения членов с множителем $h^2 f_y^{(1)}(x_m, y_m) f(x_m, y_m)$ необходимо, чтобы

$$a_2 b_2 = \frac{1}{2}. \qquad (6.15)$$

Мы получили 3 уравнения (6.13), (6.14), (6.15) с четырьмя неизвестными $a_1,\ a_2,\ b_1,\ b_2$. Следовательно, одно из этих неизвестных можно выбрать произвольно. Выберем, например,

$$a_2 = \lambda \neq 0,\ 0 < \lambda \leq 1.$$

Тогда

$$a_1 = 1 - \lambda,$$

$$b_1 = b_2 = \frac{1}{2\lambda}.$$

Соотношения (6.9) теперь примут вид

$$k_1 = f(x_m, y_m),$$

$$k_2 = f(x_m + \frac{h}{2\lambda}, y_m + \frac{h}{2\lambda}k_1), \tag{6.16}$$

$$y_{m+1} = y_m + h\big((1-\lambda)k_1 + \lambda k_2\big).$$

Формулы (6.16) обеспечивают погрешность порядка h^3 при любом $0 < \lambda \le 1$.

При $\lambda = \frac{1}{2}$ получаем следующую наиболее употребительную форму метода Рунге – Кутта 2-го порядка:

$$k_1 = f(x_m, y_m),$$

$$k_2 = f(x_m + h, y_m + hk_1), \tag{6.17}$$

$$y_{m+1} = y_m + \frac{h}{2}(k_1 + k_2).$$

Графически формулы (6.17) представлены на рис. 6.2. В точке (x_m, y_m) проводится касательная к интегральной кривой $y = y(x)$ (прямая L_1) и определяется тангенс угла наклона (угловой коэффициент) этой касательной k_1. Аналогично методу Эйлера определяется новая точка $(x_m + h, y_m + hk_1)$. В этой точке проводится касательная с угловым коэффициентом k_2 (прямая L_2). Новое значение функции y_{m+1} определяется как точка пересечения касательной с усредненным угловым коэффициентом

$$k_3 = \frac{k_1 + k_2}{2}$$

(прямая L, параллельная прямой $\bar{L}$) и вертикальной прямой $x = x_{m+1} = x_m + h$.

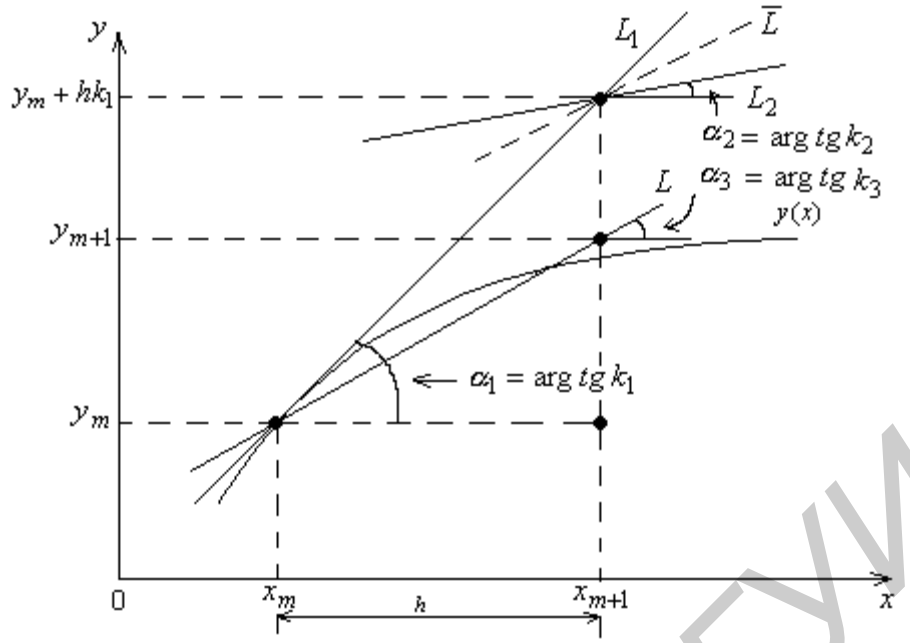

Рис. 6.2. Иллюстрация метода Рунге – Кутта 2-го порядка

При $\lambda = 1$ получаем следующую форму метода Рунге – Кутта 2-го порядка:

$$k_1 = f(x_m, y_m),$$

$$k_2 = f(x_m + \frac{h}{2}, y_m + \frac{h}{2}k_1),$$

$$y_{m+1} = y_m + hk_2.$$

При $\lambda = \frac{2}{3}$ будем иметь формулы

$$k_1 = f(x_m, y_m),$$

$$k_2 = f(x_m + \frac{3}{4}h, y_m + \frac{3}{4}hk_1),$$

$$y_{m+1} = y_m + \frac{h}{3}(k_1 + 2k_2),$$

которые обеспечивают наименьший верхний предел погрешности.

Отметим, что метод Рунге – Кутта 2-го порядка хотя и имеет ту же погрешность, что и метод рядов Тейлора (6.7), однако является более экономичным. Действительно, для расчета нового значения функции здесь необходимо 2 раза

вычислять значение функции $f(x, y)$, в то время как в формуле (6.7) нам приходилось для этих целей вычислять значения трех функций. Кроме того, две из этих функций являются производными функции $f(x, y)$, требующими своего отыскания.

6.5. Метод Рунге – Кутта 4-го порядка

Аналогично тому, как это было сделано в подразд. 6.4 для метода Рунге – Кутта 2-го порядка, можно вывести формулы для методов Рунге – Кутта 3-го и 4-го порядков. Без выводов приведем формулы для метода Рунге – Кутта 4-го порядка:

$$
\begin{aligned}
k_1 &= f(x_m, y_m), \\
k_2 &= f(x_m + \frac{h}{2}, y_m + \frac{h}{2} k_1), \\
k_3 &= f(x_m + \frac{h}{2}, y_m + \frac{h}{2} k_2), \\
k_4 &= f(x_m + h, y_m + h k_3), \\
y_{m+1} &= y_m + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
\tag{6.18}
$$

Метод Рунге – Кутта 4-го порядка (6.18) имеет погрешность порядка h^5. Среди методов Рунге – Кутта он наиболее употребителен.

Рассмотренные в данном разделе методы называются одношаговыми, так как для получения нового значения интегральной кривой достаточно знать лишь одно ее предыдущее значение.

7. РЕШЕНИЕ СИСТЕМ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

7.1. Постановка задачи

Системой обыкновенных дифференциальных уравнений первого порядка называется совокупность дифференциальных уравнений следующего вида:

$$\begin{cases} y_1'(x) = f_1(x, y_1, y_2, \ldots, y_n), \\ y_2'(x) = f_2(x, y_1, y_2, \ldots, y_n), \\ \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ y_n'(x) = f_n(x, y_1, y_2, \ldots, y_n). \end{cases} \tag{7.1}$$

Неизвестными здесь являются функции $y_1(x)$, $y_2(x)$, …, $y_n(x)$ независимой переменной x, а $y_1'(x)$, $y_2'(x)$, …, $y_n'(x)$ – их производные. Задача Коши для данной системы дифференциальных уравнений формулируется следующим образом: найти функции $y_1(x)$, $y_2(x)$, …, $y_n(x)$, удовлетворяющие равенствам (7.1) и начальным условиям

$$\begin{aligned} y_1(x_0) &= y_{1,0}, \\ y_2(x_0) &= y_{2,0}, \\ &\ldots \ldots \\ y_n(x_0) &= y_{n,0}. \end{aligned} \tag{7.2}$$

Обычно для записи системы дифференциальных уравнений (7.1) используется векторная форма, для чего данные организуются в виде векторов. Введем в рассмотрение векторную функцию – вектор-столбец

$$\bar{y}(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \ldots \\ y_n(x) \end{bmatrix}.$$

Тогда можно рассматривать также вектор-столбец производной

$$\bar{y}'(x) = \begin{bmatrix} y_1'(x) \\ y_2'(x) \\ \dots \\ y_n'(x) \end{bmatrix}$$

и вектор-столбец функций правой части системы (7.1)

$$\bar{f}(x, \bar{y}) = \begin{bmatrix} f_1(x, \bar{y}) \\ f_2(x, \bar{y}) \\ \dots \\ f_n(x, \bar{y}) \end{bmatrix}.$$

С использованием этих векторных обозначений система дифференциальных уравнений (7.1) запишется в виде

$$\bar{y}'(x) = \bar{f}(x, \bar{y}), \tag{7.3}$$

а начальные условия (7.2) – в виде

$$\bar{y}(x_0) = \bar{y}_0, \tag{7.4}$$

где

$$\bar{y}_0 = \begin{bmatrix} y_{1,0} \\ y_{2,0} \\ \dots \\ y_{n,0} \end{bmatrix}.$$

Мы видим, что векторная запись (7.3), (7.4) системы дифференциальных уравнений 1-го порядка (7.1), (7.2) имеет тот же вид, что и дифференциальное уравнение 1-го порядка (6.1), (6.2). Это внешнее сходство позволяет предположить, что методы решения дифференциального уравнения 1-го порядка (см. подразд. 6.3 – 6.5) можно распространить (обобщить) и на систему дифференциальных уравнений 1-го порядка вида (7.1), (7.2). Это предположение оказывается справедливым.

7.2. Приведение дифференциального уравнения n-го порядка к системе дифференциальных уравнений 1-го порядка

Пусть требуется найти решение дифференциального уравнения n-го порядка

$$\Phi(x,u,u',u'',\ldots,u^{(n)})=0, \tag{7.5}$$

удовлетворяющее начальным условиям

$$u(x_0)=u_0,\ u'(x_0)=u_0',\ \ldots,\ u^{(n-1)}(x_0)=u_0^{(n-1)}, \tag{7.6}$$

где $u_0,u_0',\ldots,u_0^{(n-1)}$ – некоторые числа.

Если уравнение (7.5) можно разрешить относительно старшей производной $u^{(n)}(x)$, то его можно представить в виде системы n дифференциальных уравнений 1-го порядка. Покажем, как это сделать. Пусть уравнение (7.5) представлено в виде

$$u^{(n)}(x)=F(x,u(x),u'(x),u''(x),\ldots,u^{(n-1)}(x)). \tag{7.7}$$

Для функции $u(x)$ и ее производных до $(n-1)$-го порядка введем обозначения

$$y_1(x)=u(x),$$
$$y_2(x)=u'(x),$$
$$y_3(x)=u''(x),$$
$$\ldots\ldots\ldots\ldots$$
$$y_n(x)=u^{(n-1)}(x).$$

Дифференцирование этих равенств с учетом выражения (7.7) дает нам следующую систему дифференциальных уравнений первого порядка:

$$y_1'(x)=y_2(x),$$
$$y_2'(x)=y_3(x),$$
$$y_3'(x)=y_4(x), \tag{7.8}$$
$$\ldots\ldots\ldots\ldots$$
$$y_n'(x)=F(x,y_1(x),y_2(x),y_3(x),\ldots,y_n(x)).$$

Начальные условия (7.6) приобретают теперь следующий вид:

$$y_1(x_0) = u_0 = y_{1,0},$$
$$y_2(x_0) = u_0' = y_{2,0},$$
$$\ldots\ldots\ldots\ldots\ldots \qquad (7.9)$$
$$y_n(x_0) = u_0^{(n-1)} = y_{n,0}.$$

7.3. Метод Эйлера

Этот метод решения векторного дифференциального уравнения (7.3) состоит в последовательных расчетах по формуле

$$\bar{y}_{m+1} = \bar{y}_m + h \cdot \bar{f}(x_m, \bar{y}_m), \qquad (7.10)$$

начиная с точки $(x_0, \bar{y}_0)$, заданной начальными условиями x_0, $\bar{y}(x_0) = \bar{y}_0$. Здесь h – шаг интегрирования по независимой переменной x.

Для системы из двух уравнений векторная формула (7.10) представляется в виде двух следующих скалярных формул:

$$y_{1,m+1} = y_{1,m} + h \cdot f_1(x_m, y_{1,m}, y_{2,m}),$$
$$y_{2,m+1} = y_{2,m} + h \cdot f_2(x_m, y_{1,m}, y_{2,m}).$$

7.4. Метод Рунге – Кутта 2-го порядка

Этот метод состоит в последовательных расчетах по формулам

$$\bar{k}_1 = \bar{f}(x_m, \bar{y}_m),$$
$$\bar{k}_2 = \bar{f}(x_m + h, \bar{y}_m + h\bar{k}_1), \qquad (7.11)$$
$$\bar{y}_{m+1} = \bar{y}_m + \frac{h}{2}(\bar{k}_1 + \bar{k}_2),$$

начиная с точки $(x_0, \bar{y}_0)$. Необходимо заметить, что здесь $\bar{k}_1$ и $\bar{k}_2$ – векторы.

Для системы из двух уравнений векторные формулы (7.11) представляются в виде следующих скалярных формул:

$$k_{1,1} = f_1(x_m, y_{1,m}, y_{2,m}),$$

$$k_{1,2} = f_2(x_m, y_{1,m}, y_{2,m}),$$

$$k_{2,1} = f_1(x_m + h, y_{1,m} + hk_{1,1}, y_{2,m} + hk_{1,2}),$$

$$k_{2,2} = f_2(x_m + h, y_{1,m} + hk_{1,1}, y_{2,m} + hk_{1,2}),$$

$$y_{1,m+1} = y_{1,m} + \frac{h}{2}(k_{1,1} + k_{2,1}),$$

$$y_{2,m+1} = y_{2,m} + \frac{h}{2}(k_{1,2} + k_{2,2}).$$

7.5. Метод Рунге – Кутта 4-го порядка

Этот метод состоит в последовательных расчетах по формулам

$$\bar{k}_1 = \bar{f}(x_m, \bar{y}_m),$$

$$\bar{k}_2 = \bar{f}(x_m + \frac{h}{2}, \bar{y}_m + \frac{h}{2}\bar{k}_1),$$

$$\bar{k}_3 = \bar{f}(x_m + \frac{h}{2}, \bar{y}_m + \frac{h}{2}\bar{k}_2), \qquad (7.12)$$

$$\bar{k}_4 = \bar{f}(x_m + h, \bar{y}_m + h\bar{k}_3),$$

$$\bar{y}_{m+1} = \bar{y}_m + \frac{h}{6}(\bar{k}_1 + 2\bar{k}_2 + 2\bar{k}_3 + \bar{k}_4),$$

начиная с точки $(x_0, \bar{y}_0)$.

Для системы из двух уравнений каждая из векторных формул (7.12) представляется в виде двух скалярных формул, так что вместо (7.12) будем иметь

$$k_{1,1} = f_1(x_m, y_{1,m}, y_{2,m}),$$

$$k_{1,2} = f_2(x_m, y_{1,m}, y_{2,m}),$$

$$k_{2,1} = f_1(x_m + \frac{h}{2}, y_{1,m} + \frac{h}{2}k_{1,1}, y_{2,m} + \frac{h}{2}k_{1,2}),$$

$$k_{2,2} = f_2(x_m + \frac{h}{2}, y_{1,m} + \frac{h}{2}k_{1,1}, y_{2,m} + \frac{h}{2}k_{1,2}),$$

$$k_{3,1} = f_1(x_m + \frac{h}{2}, y_{1,m} + \frac{h}{2}k_{2,1}, y_{2,m} + \frac{h}{2}k_{2,2}),$$

$$k_{3,2} = f_2(x_m + \frac{h}{2}, y_{1,m} + \frac{h}{2}k_{2,1}, y_{2,m} + \frac{h}{2}k_{2,2}),$$

$$k_{4,1} = f_1(x_m + h, y_{1,m} + hk_{3,1}, y_{2,m} + hk_{3,2}),$$

$$k_{4,2} = f_2(x_m + h, y_{1,m} + hk_{3,1}, y_{2,m} + hk_{3,2}),$$

$$y_{1,m+1} = y_{1,m} + \frac{h}{6}(k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1}),$$

$$y_{2,m+1} = y_{2,m} + \frac{h}{6}(k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2}).$$

8. ВЫПОЛНЕНИЕ СИМВОЛЬНЫХ ОПЕРАЦИЙ

8.1. Понятие символьных операций

Символьными (или аналитическими) операциями называются такие операции, исходные данные на выполнение которых, а также результаты их выполнения определяются в виде символьных (формульных) выражений. В настоящее время имеется возможность выполнять символьные операции на компьютере. Для этого разработаны различные программные системы, такие, как Reduce, Maple, Mathematica. Эти системы способны преобразовывать алгебраические выражения, находить аналитические решения систем линейных, нелинейных и дифференциальных уравнений, манипулировать полиномами, вычислять производные и интегралы, анализировать функции и находить их пределы и т.д. К символьным вычислениям относят также численные расчеты с произвольным числом цифр результатов и с отсутствующей погрешностью, поскольку это требует символьного представления чисел и особых алгоритмов выполнения операций с ними. Появление возможности выполнения символьных операций на компьютере привело к развитию нового научного направления – компьютерной математики (или компьютерной алгебры).

8.2. Выполнение символьных операций в Matlab

В систему Matlab 5.2.1 входит обновленная версия пакета расширения Symbolic Math Toolbox (Symbolic), которая базируется на ядре символьной математической системы Maple V R4, лидирующей в области автоматизации аналитических решений. Новейшая реализация системы символьной математики Maple V R5 в своем ядре и в расширениях имеет около 2700 функций. Система Matlab с пакетом Symbolic, включающим в себя чуть более сотни символьных команд и функций, намного уступает Maple V по количеству таких команд и функций.

В данный пакет включены наиболее важные и широко распространенные функции, так что возможности выполнения символьных операций в системе Matlab остаются весьма широкими. Помимо типовых аналитических вычислений (дифференцирование и интегрирование, упрощение математических выражений, подстановка и т.д.) пакет Symbolic позволяет реализовать арифметические операции с произвольной точностью.

8.3. Создание символьных переменных

Поскольку переменные системы Matlab по умолчанию не определены и задаются как векторные, матричные, числовые и т.д., т.е. не имеющие отношения к символьной математике, то для реализации символьных вычислений нужно прежде всего позаботиться о создании специальных символьных переменных. В простейшем случае их можно определить как строковые переменные, заключив имена в апострофы. Например, при вводе в окне управления команды

$$\sin('x')\text{^}2 + \cos('x')\text{^}2$$

и нажатии клавиши Enter мы получим результат

$$\text{ans} = 1.$$

Для создания символьных переменных или объектов используется функция **sym:**

– **x=sym('x')** – возвращает символьную переменную с именем **'x'** и записывает результат в **x**;

– **x=sym('x','real')** – возвращает символьную переменную вещественного типа с именем **'x'** и записывает результат в **x** (в общем случае символьные переменные рассматриваются как комплексные);

– **x=sym('x','unreal')** – возвращает символьную переменную мнимого типа с именем **'x'** и записывает результат в **x**.

Возможно создание числа или матрицы в символьном виде с помощью записи вида **eps=sym('0.001')**.

8.4. Создание группы символьных переменных

Для создания группы символьных переменных или объектов используется функция **syms**:

– **syms x1 x2 …** – создает группу символьных объектов, подобную выражениям **x1=sym('x1'); x2=sym('x2'); …**

– **syms x1 x2 … real** и **syms x1 x2 … unreal** – создают группы символьных объектов с вещественными (real) и невещественными (unreal) значениями. Последнюю функцию можно использовать для отмены задания вещественных объектов.

8.5. Создание списка символьных переменных

В математических выражениях могут использоваться как обычные, так и символьные переменные. Для выделения символьных переменных в некотором выражении используется функция **findsym**:

– **findsym(s)** – возвращает в алфавитном порядке список всех символьных переменных выражения **s**. При отсутствии таковых возвращается пустая строка;

– **findsym(s,n)** – возвращает список **n** символьных переменных, ближайших к 'x' в алфавитном порядке в выражении **s**.

Пример 8.1. В результате выполнения m-файла сценария

```
syms alpha x1 y b
a=1;
findsym(alpha+a+b)
findsym(cos(alpha)*b*x1 + 14*y,2)
findsym(y*(4+3*i) + 6*j)
```

будут выведены следующие результаты:

```
ans =
alpha, b
ans =
```

x1,y

ans =

y

Функция **findsym** позволяет упростить запись многих функций, поскольку она автоматически находит используемую в этих функциях символьную переменную.

8.6. Вывод символьного выражения

Система Matlab пока не способна выводить выражения и результаты их преобразований в естественной математической форме с использованием общепринятых спецзнаков для отображения интегралов, сумм, произведений и т.д. Тем не менее некоторые возможности близкого к математическому виду вывода обеспечивает функция **pretty**;

– **pretty(s)** – дает вывод выражения **s** в формате, приближенном к математическому;

– **pretty(s,n)** – аналогична предшествующей функции, но задает вывод выражения **s** в **n** позициях строки (по умолчанию равно 79).

Функция **latex(s)** возвращает выражение **s** в форме текстового редактора LaTeX. Это позволяет использовать это выражение в LaTeX для получения выражения в его обычной математической форме.

Пример 8.2. В результате выполнения *m*-файла сценария

```
syms x y
pretty(x^2/y^2)
z=latex(x^2/y^2)
```

будут выведены следующие результаты:

$$\frac{x^2}{y^2}$$

z =

{\frac {{x}^{2}}{{y}^{2}}}

8.7. Упрощение выражений

Функция **z = simplify(s)** поэлементно упрощает символьные выражения массива **s**. Если упрощение невозможно, то возвращается исходное выражение.

Пример 8.3. Программа

```
syms x
z=simplify(sin(x)^2+cos(x)^2)
```

возвращает результат z = 1.

Команда **simplify** в Symbolic не обладает в полной мере возможностями системы Maple V по упрощению выражений. Дополнительные возможности обеспечивает функция **simple(s)**, которая выполняет различные упрощения для элементов массива **s** и выводит как промежуточные результаты, так и самый короткий конечный результат. При обращении к функции **simple** в форме

[R,HOW]=simple(s)

промежуточные результаты не выводятся. Конечные результаты упрощений содержатся в векторе **R**, а в строковом векторе **HOW** указывается выполняемое преобразование.

Пример 8.4. Программа

```
syms x
[R1,HOW1]=simple(cos(x)^2-sin(x)^2)
[R2,HOW2]=simple(2*cos(x)^2-sin(x)^2)
```

возвращает следующие результаты:

```
R1 =
cos(2*x)
HOW1 =
```

```
combine(trig)
R2 =
3*cos(x)^2-1
HOW2=
simplify
```

8.8. Вычисление производных

Для вычисления в символьном виде производных от выражения **s** служит функция **diff**:

– **diff(s)** – возвращает символьное значение первой производной от символьного выражения или массива символьных выражений **s** по независимой переменной, определенной функцией **findsym**;

– **diff(s,n)** – возвращает символьное значение **n**-й производной от символьного выражения или массива символьных выражений **s** по независимой переменной, определенной функцией **findsym**;

– **diff(s,v)** – возвращает символьное значение первой производной от символьного выражения или массива выражений **s** по переменной **v**;

– **diff(s,v,n)** или **diff(s,n,v)** – возвращает символьное значение **n**-й производной от символьного выражения или массива символьных выражений **s** по переменной **v**.

Пример 8.5. Программа

```
syms x t
y1=diff(sin(x^2))
y2=diff(t^6,6)
```

возвращает следующие результаты:

```
y1 =
2*cos(x^2)*x
y2 =
720
```

8.9. Вычисление интегралов

Для вычисления определенных и неопределенных интегралов в символьном виде служит функция **int**:

– **int(s)** – возвращает символьное значение неопределенного интеграла от символьного выражения или массива символьных выражений **s** по переменной, которая автоматически определяется функцией **findsym**. Если **s** – константа, то вычисляется интеграл по переменной **'x'**;

– **int(s,a,b)** – возвращает символьное значение определенного интеграла на отрезке интегрирования **[a,b]** от символьного выражения или массива символьных выражений **s** по переменной, которая автоматически определяется функцией **findsym**. Пределы интегрирования **a, b** могут быть как символьными, так и числовыми;

– **int(s,v)** – возвращает символьное значение неопределенного интеграла от символьного выражения или массива выражений **s** по переменной **v**;

– **int(s,v,a,b)** – возвращает символьное значение определенного интеграла от символьного выражения или массива символьных выражений **s** по переменной **v** с пределами интегрирования **[a,b]**.

Пример 8.6. Программа

```
syms x alpha u x1
y1=int(1/(1+x^2))
y2=int(sin(alpha*u),alpha)
y3=int(x1*log(1+x1),0,1)
```

возвращает следующие результаты:

```
y1 =
atan(x)
y2 =
-cos(alpha*u)/u
y3 =
1/4
```

8.10. Вычисление сумм рядов

Для аналитического вычисления суммы

$$R = \sum_{i=a}^{b} s(i),$$

где i – переменная суммирования, $s(i)$ – общий член ряда, параметр b может быть конечным или бесконечном (inf), служит функция **symsum**:

– **symsum(s)** – возвращает символьное значение суммы бесконечного ряда по переменной суммирования, найденной автоматически с помощью функции **findsym**;

– **symsum(s,a,b)** – возвращает символьное значение суммы ряда по переменной суммирования, найденной автоматически с помощью функции **findsym**, при изменении этой переменной от **a** до **b**;

– **symsum(s,v)** – возвращает символьное значение суммы бесконечного ряда по переменной суммирования **v**;

– **symsum(s,v,a,b)** – возвращает символьное значение суммы ряда по переменной суммирования **v** при изменении этой переменной от **a** до **b**.

Пример 8.7. Программа

```
syms k n
y1=simple(symsum(k,0,n-1))
y2=simple(symsum(k,0,n))
y3=simple(symsum(k^2,0,n))
```

возвращает следующие результаты:

```
y1 =
1/2*n*(n-1)
y2 =
1/2*n*(n+1)
y3 =
1/6*n*(n+1)*(2*n+1)
```

8.11. Вычисление пределов

Для вычисления предела функции $f(x)$ служит функция **limit**;

– **limit(f,a)** – возвращает предел в точке **a** символьного выражения **f** по независимой переменной, найденной с помощью функции **findsym**;

– **limit(f)** – возвращает предел в нуле символьного выражения **f** по независимой переменной, найденной с помощью функции **findsym**;

– **limit(f,x,a)** – возвращает предел символьного выражения **f** в точке **x=a**;

– **limit(f,x,a,'right')** – возвращает предел символьного выражения **f** в точке **x=a+0** (справа);

– **limit(f,x,a,'left')** – возвращает предел символьного выражения **f** в точке **x=a-0** (слева).

Пример 8.8. В результате выполнения программы

```
syms x t a
y1=limit(sin(x)/x)
y2=limit((x-2)/(x^2-4),2)
y3=limit((1+2*t/x)^(3*x),x,inf)
y4=limit(1/x,x,0,'right')
v=[(1 + a/x)^x, exp(-x)];
y5=limit(v,x,inf,'left')
```

будет получен следующий результат:

```
y1 =
1
y2 =
1/4
y3 =
exp(6*t)
y4 =
inf
y5 =
[ exp(a),      0]
```

8.12. Разложение функции в ряд Тейлора

Ряд Тейлора для функции $f(x)$ имеет вид

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n .$$

При $a = 0$ этот ряд называется рядом Маклорена. Для получения разложений аналитических функций в ряд Тейлора служит функция **taylor**:

– **taylor(f)** – возвращает 6 членов ряда Маклорена;

– **taylor(f,n)** – возвращает члены ряда Маклорена до (n-1)-го порядка;

– **taylor(f,a)** – возвращает 6 членов ряда Тейлора в окрестности точки **a**;

– **taylor(f,a,n)** – возвращает члены ряда Тейлора до (n-1)-го порядка в окрестности точки **a**.

Пример 8.9. В результате выполнения программы

```
syms x t
y1=taylor(exp(-x))
y2=taylor(log(x),6,1)
y3=taylor(sin(x),pi/2,6)
y4=taylor(x^t,3,t)
```

будет получен следующий результат:

```
y1 =
1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
y2 =
x-1-1/2*(x-1)^2+1/3*(x-1)^3-1/4*(x-1)^4+1/5*(x-1)^5
y3 =
1-1/2*(x-1/2*pi)^2+1/24*(x-1/2*pi)^4
y4 =
1+log(x)*t+1/2*log(x)^2*t^2
```

8.13. Вычисление определителя матрицы, обращение матрицы

Функция **det(X)** вычисляет определитель (детерминант) квадратной матрицы X в символьном виде.

Для обращения (инвертирования) матрицы в символьном виде используется функция **inv(X)**.

Пример 8.10. В результате выполнения программы

```
syms a b c d t
A=[a b;c d];
y1=det(A)
B=[1/(2-t), 1/(3-t)
    1/(3-t), 1/(4-t)];
y2=inv(B)
```

будет получен следующий результат:

```
y1 =
a*d-b*c
y2 =
[    -(-3+t)^2*(-2+t), (-3+t)*(-2+t)*(-4+t)]
[ (-3+t)*(-2+t)*(-4+t),    -(-3+t)^2*(-4+t)]
```

9. ДОПОЛНЕНИЕ

9.1. Вычисление корней полиномов

Пусть требуется найти корень полинома m-й степени

$$P_m(x) = a_0 + a_1 x + a_2 x^2 + ... + a_m x^m . \quad (9.1)$$

Воспользуемся методом Ньютона – Рафсона:

$$x_n = x_{n-1} - \frac{P_m(x_{n-1})}{P'_m(x_{n-1})} .$$

Значение $P_m(x_{n-1})$ здесь можно вычислить по правилу Горнера, т.е. с помощью соотношений

$$b_m = a_m ,$$

$$b_j = a_j + x_{n-1} b_{j+1}, \; j = m-1, m-2, ..., 0 . \quad (9.2)$$

В результате получим

$$P_m(x_{n-1}) = b_0 .$$

Далее, согласно (3.18) имеем

$$P_m(x) = (x - x_{n-1}) G_{m-1}(x) + b_0 , \quad (9.3)$$

где

$$G_{m-1}(x) = b_1 + b_2 x + b_3 x^2 + ... + b_{m-1} x^{m-2} + b_m x^{m-1} .$$

Дифференцируя (9.3), получим

$$P'_m(x) = G_{m-1}(x) + (x - x_{n-1}) G'_{m-1}(x) .$$

Следовательно,

$$P'_m(x_{n-1}) = G_{m-1}(x_{n-1}) .$$

Но $G_{m-1}(x)$ является полиномом степени $m-1$, так что его также можно вычислить по правилу Горнера, воспользовавшись рекуррентными формулами

$$c_m = b_m ,$$

$$c_j = b_j + x_{n-1} c_{j+1}, \; j = m-1, m-2, ..., 2, 1 . \quad (9.4)$$

В результате получим

$$P'_m(x_{n-1}) = G_{m-1}(x_{n-1}) = c_1.$$

Подставляя найденные значения $P_m(x_{n-1}) = b_0$ и $P'_m(x_{n-1}) = c_1$ в формулу итераций Ньютона – Рафсона, получаем, что корни полинома (9.1) отыскиваются по итерационной формуле

$$x_n = x_{n-1} - \frac{b_0}{c_1}. \tag{9.5}$$

Итак, расчеты по отысканию корня полинома выполняются в следующем порядке: по формулам (9.2) находим b_0, по формулам (9.4) – c_1 и по формуле (9.5) – новое приближение корня. Этот метод нахождения корней полиномов часто называют методом Бирге – Виета.

9.2. Решение систем нелинейных уравнений. Метод Ньютона

Система нелинейных уравнений записывается в виде

$$\begin{aligned} f_1(x_1,...,x_n) &= 0, \\ f_2(x_1,...,x_n) &= 0, \\ &\cdots\cdots\cdots\cdots \\ f_n(x_1,...,x_n) &= 0. \end{aligned} \tag{9.6}$$

Такие системы уравнений решаются практически исключительно численными методами. Изложим здесь метод Ньютона. Формулы итераций по методу Ньютона можно получить следующим образом. Возьмем некоторую точку $(x_1^{(k)},...,x_n^{(k)})$, которую назовем начальным приближением к решению рассматриваемой системы нелинейных уравнений (9.6). Разложим функции $f_i(x_1,...,x_n)$ в системе (9.6) в ряд Тейлора в окрестности точки $(x_1^{(k)},...,x_n^{(k)})$ и удержим в разложении только линейные члены. Получим следующую систему уравнений:

$$f_1(x_1^{(k)},...,x_n^{(k)}) + f'_{1,x_1}(x_1^{(k)},...,x_n^{(k)})\Delta x_1^{(k)} + ... + f'_{1,x_n}(x_1^{(k)},...,x_n^{(k)})\Delta x_n^{(k)} = 0,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (9.7)$$

$$f_n(x_1^{(k)},...,x_n^{(k)}) + f'_{n,x_1}(x_1^{(k)},...,x_n^{(k)})\Delta x_1^{(k)} + ... + f'_{n,x_n}(x_1^{(k)},...,x_n^{(k)})\Delta x_n^{(k)} = 0.$$

Здесь $f'_{i,x_j}(x_1^{(k)},...,x_n^{(k)})$, $i, j = \overline{1,n}$, – частная производная функции $f_i(x_1,...,x_n)$ по аргументу x_j, вычисленная в точке $(x_1^{(k)},...,x_n^{(k)})$, и

$$\Delta x_j^{(k)} = x_j - x_j^{(k)},\ j = \overline{1,n}. \quad (9.8)$$

Это – система линейных алгебраических уравнений относительно переменных $\Delta x_1^{(k)},...,\Delta x_n^{(k)}$, которая может быть решена, например, методом Гаусса. Получив решение этой системы, из формулы (9.8) можем найти новое приближение:

$$x_j^{(k+1)} = x_j^{(k)} + \Delta x_j^{(k)},\ j = \overline{1,n}. \quad (9.9)$$

Далее решаем систему линейных уравнений (9.7) со значениями $x_j^{(k+1)}$ и по полученному решению находим

$$x_j^{(k+2)} = x_j^{(k+1)} + \Delta x_j^{(k+1)},\ j = \overline{1,n}.$$

Хорошим критерием остановки процесса является условие

$$\sum_{i=1}^{n}(x_i^{(k+1)} - x_i^{(k)})^2 \le \varepsilon, \quad (9.10)$$

где ε – некоторое малое число, характеризующее допустимую погрешность вычисления корней системы нелинейных уравнений.

Более компактной является векторно-матричная форма метода Ньютона, которая позволяет также провести аналогию с методом Ньютона для решения одного уравнения. Для получения векторно-матричной формы метода Ньютона систему уравнений (9.6) записывают в векторной форме

$$\bar{f}(\bar{x}) = 0,$$

где $\bar{x}^T = (x_1,...,x_n)$ – вектор-строка неизвестных переменных, $\bar{f}^T(\bar{x}) = (f_1(\bar{x}),...,f_n(\bar{x}))$ – вектор-строка функций в левой части системы урав-

нений (9.6). При таких обозначениях система линейных уравнений (9.7) примет вид

$$W^{(k)}\Delta\bar{x}^{(k)} = -\bar{f}(\bar{x}^{(k)}), \tag{9.11}$$

где

$$W^{(k)} = \left.\frac{d\bar{f}(\bar{x})}{d\bar{x}}\right|_{\bar{x}=\bar{x}^{(л)}} = \left.\left(\frac{\partial f_i(\bar{x})}{\partial x_j}\right)\right|_{\bar{x}=\bar{x}^{(л)}} -$$

матрица частных производных функций $f_i(\bar{x})$, вычисленная в точке $\bar{x} = \bar{x}^{(k)}$,

$$\Delta\bar{x}^{(k)} = \bar{x} - \bar{x}^{(k)}. \tag{9.12}$$

Получив решение $\Delta\bar{x}^{(k)}$ уравнения (9.11), по формуле (9.12) получим новое приближение:

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} + \Delta\bar{x}^{(k)}.$$

Критерий остановки процесса итераций (9.10) записывается теперь в виде

$$\| \bar{x}^{(k+1)} - \bar{x}^{(k)} \| \le \varepsilon,$$

где $\| \bar{x}^{(k+1)} - \bar{x}^{(k)} \|$ – евклидова норма вектора $\bar{x}^{(k+1)} - \bar{x}^{(k)}$.

Векторно-матричная форма позволяет записать итерацию метода Ньютона в виде формулы, аналогичной формуле (5.10) метода Ньютона для одного уравнения. Действительно, уравнение (9.11) можно записать в виде

$$\frac{d\bar{f}(\bar{x}^{(k)})}{d\bar{x}}(\bar{x} - \bar{x}^{(k)}) = -\bar{f}(\bar{x}^{(k)}), \tag{9.13}$$

откуда получаем следующую формулу итераций:

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - \left(\frac{d\bar{f}(\bar{x}^{(k)})}{d\bar{x}}\right)^{-1} \bar{f}(\bar{x}^{(k)}).$$

9.3. Решение систем линейных алгебраических уравнений с трехдиагональной матрицей (метод прогонки)

В ряде приложений приходится решать систему линейных алгебраических уравнений с матрицей, у которой равны нулю все элементы, кроме элементов, расположенных на главной диагонали, на диагоналях, расположенных над и под главной диагональю. Такие системы уравнений называются трехдиагональными. Трехдиагональная система уравнений может быть решена методом Гаусса. Однако в этом случае метод Гаусса упрощается. Метод Гаусса, модифицированный для решения трехдиагональных систем линейных алгебраических уравнений, получил название *метода прогонки*. Изложим его.

Трехдиагональная СЛАУ имеет следующий вид:

$$\begin{pmatrix} a_{1,1}, & a_{1,2}, & 0, & 0, & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 0 \\ a_{2,1}, & a_{2,2}, & a_{2,3}, & 0, & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 0 \\ 0, & \ldots\ldots\ldots\ldots & a_{i,i-1}, & a_{i,i}, & a_{i,i+1}, \ldots\ldots\ldots\ldots\ldots 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & a_{n-1,n-1}, & a_{n-1,n} & & \\ 0, & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & 0, & a_{n,n-1}, & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_i \\ x_{i+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ \vdots \\ d_{i-1} \\ d_i \\ d_{i+1} \\ \vdots \\ d_n \end{pmatrix}.$$

Эту систему уравнений можно записать в виде

$$a_{i,i-1}x_{i-1} + a_{i,i}x_i + a_{i,i+1}x_{i+1} = d_i,\ i = \overline{1,n}, \qquad (9.14)$$

$$a_{1,0} = 0,\ a_{n,n+1} = 0.$$

Удобнее для обозначения коэффициентов СЛАУ использовать один индекс. Обозначив

$$a_{i,i-1} = \alpha_i,$$

$$a_{i,i} = -\beta_i,$$

$$a_{i,i+1} = \gamma_i,$$

$$d_i = \delta_i,$$

вместо (9.14) получим

$$\alpha_i x_{i-1} - \beta_i x_i + \gamma_i x_{i+1} = \delta_i,\ i = \overline{1,n}, \qquad (9.15)$$

$$\alpha_1 = 0,\ \gamma_n = 0.$$

Формула (9.15) представляет собой разностное уравнение. Применяя к трехдиагональной системе (9.15) метод Гаусса, замечаем, что прямой ход метода сводится к исключению элементов α_i. После окончания прямого хода получается система уравнений, содержащая в каждом i-м уравнении только 2 неизвестных: x_i и x_{i+1}. Поэтому формулы обратного хода имеют вид

$$x_i = \eta_{i+1} + \xi_{i+1} x_{i+1},\ i = n, n-1, ..., 1, \qquad (9.16)$$

где η_{i+1}, ξ_{i+1} – некоторые числа.

Получим на основании формулы (9.16) формулы прямого хода. Для этого уменьшим в (9.16) индекс i на единицу,

$$x_{i-1} = \eta_i + \xi_i x_i, \qquad (9.17)$$

и подставим (9.16) и (9.17) в уравнение (9.15). Получим:

$$\alpha_i(\eta_i + \xi_i x_i) - \beta_i x_i + \gamma_i x_{i+1} = \delta_i.$$

Выражая отсюда x_i, будем иметь

$$x_i = \frac{\gamma_i}{\beta_i - \alpha_i \xi_i} x_{i+1} + \frac{-\delta_i + \alpha_i \eta_i}{\beta_i - \alpha_i \xi_i}. \qquad (9.18)$$

Чтобы выражение (9.18) совпало с (9.16), необходимо выполнение равенств

$$\xi_{i+1} = \frac{\gamma_i}{\beta_i - \alpha_i \xi_i},\quad \eta_{i+1} = \frac{\alpha_i \eta_i - \delta_i}{\beta_i - \alpha_i \xi_i},\ \ i = \overline{1,n}. \qquad (9.19)$$

Формулы (9.19) – это формулы прямого хода метода прогонки, а формула (9.16) – обратного хода. Осталось определить, с каких значений начинать расчеты по формулам прямого и обратного хода. Расчеты прямого хода по формуле (9.19) начинаются со значений ξ_1 и η_1, которые нам неизвестны. Однако в формулах (9.19) перед ξ_1 и η_1 стоит множитель α_1, который равен нулю. Поэтому можно брать $\xi_1 = \eta_1 = 0$. Расчеты обратного хода по формуле (9.16) начинаются со

значения x_{n+1}. Поскольку перед x_{n+1} в (9.16) стоит множитель $\xi_{n+1} = \gamma_n = 0$, то следует взять $x_{n+1} = 0$.

Вычисления по формулам (9.16), (9.19) метода прогонки требуют всего $3n$ ячеек памяти и $9n$ арифметических операций, что гораздо меньше, чем по общим формулам метода исключения Гаусса.

Попутно с решением системы уравнений можно найти определитель трехдиагональной матрицы системы уравнений по формуле

$$| A |= \prod_{i=1}^{n} (\alpha_i \xi_i - \beta_i).$$

9.4. Интерполирование функций сплайнами

Интерполяционным сплайном n-й степени для функции $f(x)$ называется полином n-й степени вида

$$s(x) = \sum_{k=0}^{n} a_{i,k} x^k, \; x_{i-1} \le x \le x_i, \tag{9.20}$$

со следующими свойствами: 1) коэффициенты $a_{i,k}$ кусочно-постоянны, т.е. постоянны для соседней пары точек (узлов) x_{i-1}, x_i; 2) полином $s(x)$ в узлах x_i принимает заданные значения $y_i = f(x_i)$, $i = 0,1,...,N$; 3) полином $s(x)$ непрерывен вместе со своими $(n-1)$ производными.

На практике наиболее употребительны сплайны 1-й степени ($n = 1$) и 3-й степени ($n = 3$). Интерполяция сплайном первой степени называется кусочно-линейной. В этом случае две соседние ординаты y_{i-1}, y_i соединяются прямой линией. Интерполяции сплайном третьей степени можно дать следующую интерпретацию. Пусть требуется провести график функции по известным значениям функции $y_i = f(x_i)$, $i = \overline{1, N}$, (рис. 9.1). Это можно сделать следующим образом. Необходимо взять гибкую металлическую линейку, поставить ее

на ребро и изгибать, придерживая линейку в нескольких местах так, чтобы ее ребро проходило сразу через все точки. Линейка опишет некоторую кривую, которая и будет сплайном 3-й степени. Действительно, гибкая линейка – это упругий брусок. Из курса сопротивления материалов известно, что уравнение его свободного равновесия есть $\varphi^{(4)}(x) = 0$. Значит, в промежутке между парой соседних точек интерполирующая функция является полиномом 3-й степени.

Сформулируем задачу интерполирования функции $f(x)$ сплайном 3-го порядка и получим ее решение. Постановку и решение задачи будем комментировать на примере с гибкой линейкой (см. рис. 9.1).

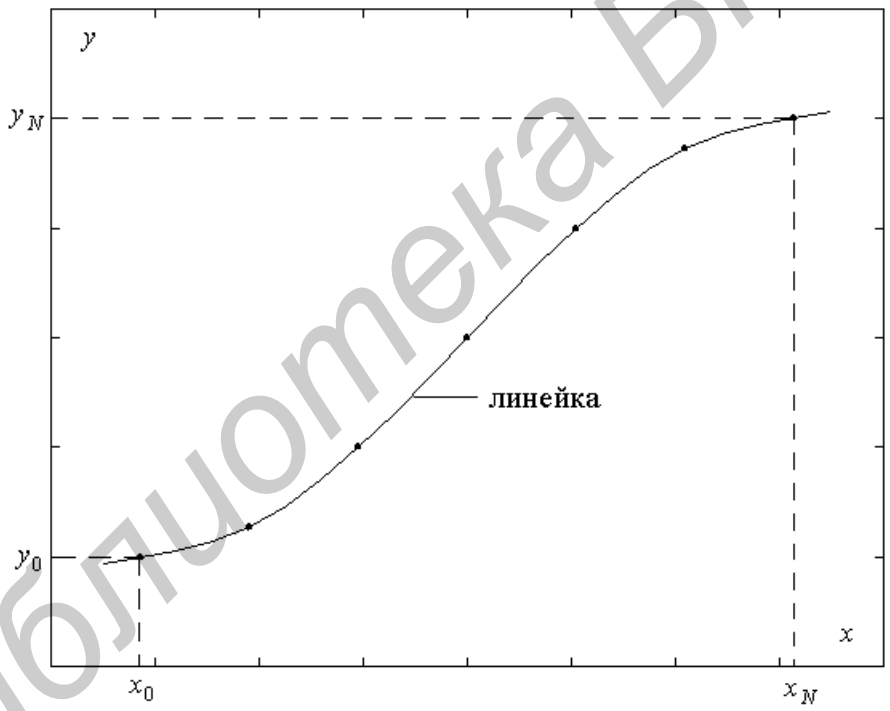

Рис. 9.1. Иллюстрация интерполирования сплайнами

Итак, известны узлы интерполирования x_0, x_1, ..., x_N из некоторого отрезка интерполирования (a, b) и значения функции в узлах $y_i = f(x_i)$, $i = 0,1,...,N$. Требуется получить сплайн вида (9.20) 3-й степени.

Полином 3-й степени на i-м отрезке интерполирования удобно записывать в виде

$$\varphi(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3,\ x_{i-1} \le x \le x_i,\ i = \overline{1, N}. \quad (9.21)$$

Этот полином должен иметь свои коэффициенты a_i, b_i, c_i, d_i для каждого отрезка $[x_{i-1}, x_i]$, которых всего N. Следовательно, требуется определить $4N$ коэффициентов a_i, b_i, c_i, d_i, $i = \overline{1, N}$, с тем чтобы воспользоваться формулой (9.21) на всем отрезке интерполирования (a, b). Для определения этих коэффициентов используются условия, которым должен удовлетворять сплайн.

Поскольку сплайн в узлах должен принимать заданные значения, то мы получаем $2N$ условий: N условий для левого конца отрезка $[x_{i-1}, x_i]$:

$$y_{i-1} = \varphi(x_{i-1}) = a_i, \quad (9.22)$$

и N условий для правого конца отрезка $[x_{i-1}, x_i]$:

$$y_i = \varphi(x_i) = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3, \quad (9.23)$$

где $h_i = x_i - x_{i-1}$, $i = \overline{1, N}$.

Дополнительные условия получим исходя из требований непрерывности производных полинома в узлах (гладкости металлической линейки). Для этого найдем первую и вторую производные полинома:

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

$$\varphi''(x) = 2c_i + 6d_i(x - x_{i-1}),\ x_{i-1} \le x \le x_i,\ i = \overline{1, N}.$$

Непрерывность производных в узлах означает равенство левых и правых пределов производных во внутренних узлах. Непрерывность первой производной $\varphi'(x)$ во внутренних узлах дает $N - 1$ условий вида

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2,\ i = \overline{1, N-1}. \quad (9.24)$$

Непрерывность второй производной $\varphi''(x)$ во внутренних узлах также дает $N - 1$ условий следующего вида:

$$c_{i+1} = c_i + 3d_i h_i,\ i = \overline{1, N-1}. \quad (9.25)$$

В равенствах (9.24), (9.25) справа записаны значения производных в точке $x = x_i$, т.е. в правой точке i-го отрезка $[x_{i-1}, x_i]$, а слева – значения производных в левой точке $(i+1)$-го отрезка.

Соотношения (9.24), (9.25) мы можем считать верными и при $i = N$, полагая $b_{N+1} = 0$, $c_{N+1} = 0$, поскольку из выражения (9.23) действительно следует, что

$$a_{N+1} = b_{N+1} = c_{N+1} = d_{N+1} = 0.$$

Последние два условия получают из предположения о нулевой кривизне функции (т.е. о равенстве нулю второй производной) в крайних точках, что соответствует свободно опущенным концам гибкой линейки. Итак, считая, что $\varphi''(x_0) = 0$, $\varphi''(x_N) = 0$, получаем

$$c_1 = 0, \tag{9.26}$$

$$c_N + 3d_N h_N = 0. \tag{9.27}$$

Уравнения (9.22) – (9.27) образуют систему линейных уравнений для определения $4N$ неизвестных коэффициентов a_i, b_i, c_i, d_i, $i = \overline{1, N}$. Эту систему можно решить методом исключения Гаусса. Но гораздо выгоднее привести ее к трехдиагональному виду и решить более экономичным методом прогонки.

К трехдиагональному виду (9.15) система уравнений (9.22) – (9.27) приводится следующим образом.

Уравнение (9.22) дает нам все коэффициенты a_i:

$$a_i = y_{i-1}, \; i = \overline{1, N}. \tag{9.28}$$

Из уравнений (9.25), (9.27) следует, что

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \; i = \overline{1, N-1}, \tag{9.29}$$

$$d_N = -\frac{c_N}{3h_N}. \tag{9.30}$$

Поскольку формула (9.30) следует из формулы (9.29) при $i = N$ и $c_{N+1} = 0$, то будем считать формулу (9.29) справедливой при любых $i = \overline{1, N}$, т.е. вместо (9.29), (9.30) считать, что

$$d_i = \frac{c_{i+1} - c_i}{3h_i},\ i = \overline{1, N}, \tag{9.31}$$

$$c_{N+1} = 0. \tag{9.32}$$

Из выражения (9.23) получим

$$b_i = \frac{y_i - a_i - c_i h_i^2 - d_i h_i^3}{h_i} = \frac{y_i - a_i}{h_i} - c_i h_i - d_i h_i^2,\ i = \overline{1, N},$$

или с учетом (9.28), (9.31)

$$b_i = \frac{y_i - y_{i-1}}{h_i} - c_i h_i - \frac{c_{i+1} - c_i}{3} h_i = \frac{y_i - y_{i-1}}{h_i} - \frac{1}{3} h_i (c_{i+1} + 2c_i),\ i = \overline{1, N}. \tag{9.33}$$

Исключим теперь из уравнения (9.24) величины d_i, b_i, b_{i+1} с помощью формул (9.31), (9.33). Вместо (9.24) получим

$$\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{1}{3} h_{i+1}(c_{i+2} + 2c_{i+1}) =$$

$$= \frac{y_i - y_{i-1}}{h_i} - \frac{1}{3} h_i (c_{i+1} + 2c_i) + 2c_i h_i + 3\frac{c_{i+1} - c_i}{3h_i} h_i^2,\ i = \overline{1, N}.$$

Приводя подобные члены, получим следующую систему линейных уравнений для коэффициентов c_i:

$$h_i c_i + 2(h_{i+1} + h_i) c_{i+1} + h_{i+1} c_{i+2} = 3\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right),\ i = \overline{1, N}.$$

Полагая здесь $j = i + 1$, перепишем эту систему в виде

$$h_{j-1} c_{j-1} + 2(h_j + h_{j-1}) c_j + h_j c_{j+1} = 3\left(\frac{y_j - y_{j-1}}{h_j} - \frac{y_{j-1} - y_{j-2}}{h_{j-1}} \right),\ j = \overline{2, N+1}.$$

Учитывая, что, $c_1 = 0$, $c_{N+1} = 0$, эту систему можно сократить на одну переменную:

$$h_{j-1}c_{j-1} + 2(h_j + h_{j-1})c_j + h_j c_{j+1} = 3\left(\frac{y_j - y_{j-1}}{h_j} - \frac{y_{j-1} - y_{j-2}}{h_{j-1}}\right), \quad j = \overline{2, N}. \quad (9.34)$$

Тем самым мы получили систему из $(N-1)$ уравнений вида (9.15) относительно неизвестных $c_2, c_3, ..., c_N$. Система уравнений (9.34) решается методом прогонки (см. разд. 9.3). После нахождения коэффициентов c_i остальные коэффициенты a_i, d_i, b_i легко вычисляются по формулам (9.28), (9.31), (9.32), (9.33), а затем по формуле (9.21) вычисляется значение интерполяционного полинома в нужной точке.

На практике для решения системы (9.34) используется готовая программа, реализующая метод прогонки. Такая программа выдает решение трехдиагональной системы уравнений (9.15) в виде массива $x = (x_1, x_2, ..., x_n)$. Поэтому для решения системы (9.34) с помощью готовой программы необходимо использовать коэффициенты

$$\alpha_j = h_{j-1},$$

$$\beta_j = -2(h_j + h_{j-1}),$$

$$\gamma_j = h_j,$$

$$\delta_j = 3\left(\frac{y_j - y_{j-1}}{h_j} - \frac{y_{j-1} - y_{j-2}}{h_{j-1}}\right), \; j = \overline{1, N-1}.$$

Затем, используя решение $x = (x_1, x_2, ..., x_{N-1})$ системы с такими коэффициентами, необходимо получить коэффициенты сплайна $c_1 = 0$, $c_i = x(i-1)$, $i = \overline{2, N}$.

ЛИТЕРАТУРА

1. Мак-Кракен, Д. Численные методы и программирование на Фортране / Д. Мак-Кракен, У. Дорн. – М. : Мир, 1978. – 584 с.

2. Гусак, А. А. Элементы методов вычислений / А. А. Гусак. – Минск : Университетское, 1982. – 519 с.

3. Бахвалов, Н. С. Численные методы / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. – М. : Наука, 1988. – 700 с.

4. Вержбицкий, В. М. Численные методы. Линейная алгебра и нелинейные уравнения: учеб. пособие для вузов / В. М. Вержбицкий. – М. : Высш. шк., 2000. – 266 с.

5. Вержбицкий, В. М. Численные методы. Математический анализ и обыкновенные дифференциальные уравнения: учеб. пособие для вузов / В. М. Вержбицкий. – М. : Высш. шк., 2001. – 382 с.

6. Дьяконов, В. П. Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ / В. П. Дьяконов. – М. : Наука, 1988. – 239 с.

7. Бахвалов, Н. С. Численные методы / Н. С. Бахвалов. – М. : Наука, 1975. – 700 с.

8. Крылов, В. И. Справочная книга по численному интегрированию / В. И. Крылов, Л. Т. Шульгина. – М. : Наука, 1966. – 370 с.

9. Корн, Г. Справочник по математике для научных работников и инженеров / Г. Корн, Т. Корн. – М. : Наука, 1973. – 832 с.

10. Муха, В. С. Введение в MATLAB : метод. пособие для выполнения лаб. работ по курсам «Статистические методы обработки данных» и «Теория автоматического управления» для спец. 53 01 02 «Автоматизированные системы обработки информации» / В. С. Муха, В. А. Птичкин. – Минск : БГУИР, 2002. – 40 с.

11. Дьяконов, В. П. MATLAB 5.0/5.3. Система символьной математики / В. П. Дьяконов, И. В. Абраменкова. – М. : Нолидж. – 1999. – 740 с.

12. Дьяконов, В. П. Mathematica 4 с пакетами расширений / В. П. Дьяконов. – М. : Нолидж, 2000. – 608 с.

13. Компьютерная алгебра. Символьные и алгебраические вычисления. – М. : Мир, 1986. – 391 с.

14. Грегори, Р. Безошибочные вычисления. Методы и приложения / Р. Грегори, Р. Е. Кришнамурти. – М. : Мир, 1988. – 207 с.

15. Кетков, Ю. Л. MATLAB 6.x: программирование численных методов / Ю. Л. Кетков, А. Ю. Кетков, М. М. Шульц. – СПб. : БХВ-Петербург, 2004. – 672 с.

16. Муха, В. С. Вычислительные методы и компьютерная алгебра : лаб. практикум для студ. спец. 53 01 02 «Автоматизированные системы обработки информации» / В. С. Муха, Т. В. Слуянова. – Минск : БГУИР, 2003. – 84 с.

СОДЕРЖАНИЕ

Учебное издание

Муха Владимир Степанович

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ И КОМПЬЮТЕРНАЯ АЛГЕБРА

Учебно-методическое пособие

2-е издание

исправленное и дополненное

Редактор *Т. Н. Крюкова*
Корректор *Е. Н. Батурчик*
Компьютерная верстка *Е. Г. Бабичева*

Подписано в печать 01.03.2010. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Times».
Отпечатано на ризографе. Усл. печ. л. 8,72. Уч.-изд. л. 6,3. Тираж 150 экз. Заказ 126.

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