Владимирский государственный университет

В.Н. ГОРЛОВ
Н.И. ЕРКОВА

МЕТОДЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ ДЛЯ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

УЧЕБНОЕ ПОСОБИЕ

АЛГОРИТМЫ И ПРОГРАММЫ

ВЛАДИМИР 2009 г.

Федеральное агенство по образованию
Государственное образовательное учреждение
высшего профессионального образования
Владимиский государственный университет

В.Н. ГОРЛОВ, Н.И. ЕРКОВА

МЕТОДЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ ДЛЯ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ. АЛГОРИТМЫ И ПРОГРАММЫ.

Учебное пособие

Владимир 2009

УДК 519.6
ББК 22.19
Г70

Рецензенты:
Доктор физико-математических наук профессор
Кафедры математического моделирования и информационных технологий в экономике
Камской государственной инженерно-экономической академии
А.Б. Евлюхин

Кандидат технических наук, профессор,
зав. кафедрой информатики и вычислительной техники
Владимирского государственного гуманитарного университета
Ю.А. Медведев
Печатается по решению редакционного совета
Владимирского государственного университета

Горлов В.Н.

Г70 Методы вычислительной математики для персональных компьютеров. Алгоритмы и программы. : учеб. пособие / В.Н. Горлов, Н.И. Еркова: Владим. Гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2009.- 148 с. – ISBN 978-5-9984-0010-0.

Изложены вычислительные методы решения нелинейных уравнений, систем линейных и нелинейных уравнений, задач приближения функций, методы численного интегрирования и одномерной оптимизации. Рассмотрение каждого метода сопровождается исследованием обусловленности и устойчивости, обсуждением особенностей реализации его на персональных компьютерах. Приведено большое количеств примеров и иллюстраций.

Предназначено для студентов 2 – 4 курсов всех специальностей очной и заочной форм обучения.

Главы 1 - 6 написаны В.Н. Горловым, главы 7 – 8 – Н.И. Ерковой.

УДК 519.6
ББК 22.19

ISBN 978-5-9984-0010-0. Владимирский государственный Университет

Введение

Применение вычислительной техники и, в частности, персональных компьютеров, позволяет решать многие трудоёмкие и сложные задачи. При этом широко используются программные средства различного назначения (пакеты прикладных программ и программные комплексы для решения математических задач, обработки результатов экспериментов, систем управления базами данных и т. д.). Однако наличие готовых программных средств не только не снимает проблемы обучения специалистов методам решения задач и обработки данных, но и делает подготовку в этом направление больше актуальной. Это объясняется тем, что использование готовых программных средств требует от специалиста грамотной математической постановки задачи, выбора эффективного метода решения и оценки погрешности полученного результата. В противном случае полученные с помощью ЭВМ результаты могут оказаться весьма далёкими от истинных.

В пособии рассматриваются основные понятия теории погрешностей, значительное внимание выделяется корректности и обусловленности вычислительных задач и вычислительных методов, а также грамотной оценке достоверности результатов.

На базе сформулированных понятий рассматриваются методы решения нелинейных уравнений, прямые и итерационные методы решения систем линейных уравнений, методы решения систем нелинейных уравнений, методы приближения функции, методы численного интегрирования и одномерной оптимизации функций. Изложение материала сопровождается примерами решения конкретных задач, приводится много геометрических иллюстраций. Рассмотрены особенности программной реализации вычисли-тельных методов на персональных ЭВМ. Приведены описания и листинги программ на языках Бейсик, Фортран и Паскаль. Параллельные тексты программ на трех языках будут полезны читателям (студентам), владеющим одним из них, для практического освоения двух других.

Материал, представленный в учебном пособии, предназначен для студентов всех факультетов, он будет также полезен аспирантам и слушателям ФПКП

ГЛАВА 1.
ВЫЧИСЛИТЕЛЬНЫЕ ЗАДАЧИ. МЕТОДЫ И АЛГОРИТМЫ. ОСНОВНЫЕ ПОНЯТИЯ.

§ 1.1. Корректность вычислительной задачи

Постановка вычислительной задачи. Под вычислительной задачей всюду будем понимать одну из следующих задач, которые возникают при анализе математических моделей: прямую задачу, обратную задачу или задачу идентификации. Прилагательное “вычислительная” показывает, что основные усилия будут направлены на то, чтобы найти (вычислить) решение задачи.

В дальнейшем будем считать, что постановка задачи включает в себя задание множества допустимых входных данных X и множества возможных решений Y. Цель вычислительной задачи состоит в нахождении решения $y \in Y$ по заданным входным данным $x \in X$. Для простоты понимания можно ограничиться рассмотрением задач, в которых входные данные и решение могут быть только числами или набором чисел (векторами, матрицами, последовательностями). Предположим, что для оценки величины погрешностей приближенных входных данных x^* и приближенного решения y^* введены абсолютные $\Delta(x^*), \Delta(y^*)$ и относительные $\delta(x^*), \delta(y^*)$ погрешности, а также их границы $\overline{\Delta}(x^*), \overline{\Delta}(y^*), \overline{\delta}(x^*), \overline{\delta}(y^*)$. Ниже будут даны определения этих величин.

Договоримся об обозначениях, которые дальше будут использоваться при сравнении величин. Кроме привычных знаков $=$, $\neq$, $\leq$, $<$, будут использоваться знаки приближенного равенства $\simeq$ и приближенного нера-венства $\underset{\sim}{<}$. Если положительные величины a и b одного порядка (т.е. $10^{-1} < a/b < 10$), будем использовать обозначение $a \sim \mathrm{b}$. Если же a меньше b, будем писать a«b, что эквивалентно соотношению a/b«1.

Абсолютная и относительная погрешность. Пусть x - точное (и, в общем случае, неизвестное) значение некоторой величины, x^* - известное приближенное значение той же величины (приближенное число). Ошибкой или погрешностью приближенного числа x^* называют разность $x - x^*$ между точным и приближенным значениями. Простейшей количественной мерой ошибки является абсолютная погрешность:

$$\Delta(x^*) = |x - x^*| \ . \qquad (1.1)$$

Следует отметить, что по величине абсолютной погрешности далеко не всегда можно сделать правильное заключение о качестве приближения. Например, если $\Delta(x^*) = 0,I$, то следует ли считать погрешность большой или нужно признать ее малой. Ответ существенным образом зависит от принятых единиц измерения и масштабов величин. Если $x \simeq 0,3$, то скорее всего точность приближения не велика; если же $x \simeq 0,3 \cdot 10^8$, то следует признать точность очень высокой. Таким образом, естественно соотнести погрешность величины и ее значение, т.е. ввести в рассмотрение относительную погрешность (при $x \neq 0$)

$$\delta(x^*) = \frac{|x - x^*|}{|x|} \ . \qquad (1.2)$$

Использование относительных погрешностей удобно, в частности, тем, что они не зависят от масштабов величин и единиц измерения. Отметим, что приведенном примере $\delta(x^*) \simeq 0,33 = 33\%$ в первом случае и $\delta(x^*) \simeq 0,33 \cdot 10^{-9} = 0,33 \cdot 10^{-7}\%$ - во втором.

Так как значение x неизвестно, то непосредственно вычисление вели-чин $\Delta(x^*)$ и $\delta(x^*)$ по формулам (1.1), (1.2) невозможно. Более реальная и часто поддающаяся решению задача состоит в получении оценок погрешности вида

$$|x - x^*| \le \overline{\Delta}(x^*); \qquad (1.3)$$

$$\frac{|x - x^*|}{|x|} \le \overline{\delta}(x^*), \qquad (1.4)$$

где $\overline{\Delta}(x^*)$ и $\overline{\delta}(x^*)$ - известные величины, которые мы будем называть верхними границами (или просто границами) абсолютной и относительной погреш-ностей.

Если величина $\overline{\Delta}(x^*)$ известна, то неравенство (*1.4*) будет выполнено при

$$\overline{\delta}\left(x^*\right)=\frac{\overline{\Delta}\left(x^*\right)}{|x|}\ . \qquad (1.5)$$

Точно так же, если величина $\overline{\delta}\left(x^*\right)$ известна, то

$$\overline{\Delta}\left(x^*\right)=|x|\cdot\overline{\delta}\left(x^*\right)\ . \qquad (1.6)$$

Так как значение *x* неизвестно, при практическом применении формулы (1.5), (1.6) заменяют приближенными равенствами

$$\bar{\delta}(x^*)\approx\frac{\bar{\Delta}(x^*)}{\left|x^*\right|}\ ,\ \bar{\Delta}(x^*)\approx\left|x^*\right|\cdot\bar{\delta}(x^*). \qquad (1.7)$$

Определение корректности задачи. Понятие корректности математической задачи было впервые сформулировано французским математиком Ж. Адамаром и развито затем академиком И.Г. Петровским. Вычислительная задача называется корректной (по Адамару – Петровскому), если выполнены следующие три требования: 1) ее решение $y\in Y$ существует при любых входных данных $x\in X$; 2) это решение единственно; 3) решение устойчиво по отношению к малым возмущениям входных данных. Если хотя бы одно из этих требований не выполнено, задача называется некорректной.

Существование решения вычислительной задачи – единственное к ней требование. Отсутствие решения может говорить, например, о непригодности математической модели либо о неправильной постановке задачи. Иногда отсутствие решения является следствием неправильного выбора множества допустимых входных данных *X* или множества *Y*.

Единственность. Существует класс вычислительных задач, для которых единственность является естественным свойством; для других же задач реше-ние не может быть единственным. Например, квадратное уравнение

$$ax^2+bx+c=0 \qquad (1.8)$$

имеет два корня

$$x_1=(-b-\sqrt{b^2-4ac})/2a\ ;\ x_2=(-b+\sqrt{b^2-4ac})/2a \qquad (1.9)$$

Как правило, если задача имеет реальное содержание, эта не единствен-ность может быть ликвидирована введением дополнительных ограничений на решение (т.е. сужением множества *Y*). В некоторых случаях проблема снимается тем, что признается целесообразным найти набор всех решений, отвечающих входным данным *x*, и тогда в качестве *y* объявляется этот набор. Например, для уравнения (1.8) решением можно назвать пару (x_1, x_2).

Устойчивость решения. Решение *y* вычислительной задачи называется устойчивым по входным данным *x*, если оно зависит от входных данных непрерывным образом. Это означает, что для любого $\varepsilon > 0$ существует $\delta(\varepsilon) > 0$ такое, что всяким исходным данным x^*, удовлетворяющим условию $\Delta(x^*) < \delta$, отвечает приближенное решение y^*, для которого $\Delta(y^*) < \varepsilon$. Таким образом, для устойчивой вычислительной задачи ее решение теоретически может быть найдено со сколь угодно высокой точностью ε, если обеспечена достаточно высокая точность δ выходных данных. Следует отметить, что одна и та же задача может оказаться и устойчивой и не устойчивой в зависимости от того, как выбран способ вычисления абсолютных погрешностей $\Delta(x^*)$ и $\Delta(y^*)$. В реальных задачах этот выбор определяется тем, в каком смысле должно быть близко приближенное решение к точному и малость какой из мер погрешности входных данных можно гарантировать.

Обусловленность вычислительной задачи. Пусть вычислительная задача корректна (ее решение существует, единственно и устойчиво по входным данным). Теоретически устойчивость задачи означает, что ее решение может быть найдено со сколь угодно малой погрешностью, если только гаранти-ровать малую погрешность входных данных. Однако на практике погреш-ности входных данных не могут быть сделаны сколь угодно малыми, точ-ность их ограничена.

Под обусловленностью вычислительной задачи понимают чувствитель-ность ее решения к малым погрешностям входных данных отвечают малые погрешности решения, и плохо обусловленной, если возможны сильные изменения решения. Количественной мерой степени обусловленности задачи является число обусловленности. Эта величина может интерпретироваться как коэффициент возможного возрастания

погрешностей в решении по отношению к вызвавшим их погрешностям входных данных.

Пусть между абсолютными погрешностями входных данных x и решения y установлено неравенство

$$\Delta(y^*) \le \nu_\Delta \cdot \Delta(x^*). \qquad (1.10)$$

Тогда величина ν_Δ называется абсолютным числом обусловленности. Если же установлено неравенство

$$\delta(y^*) \le \nu_\delta \cdot \delta(x^*) \qquad (1.11)$$

Между относительными ошибками данных и решения, то величину ν_δ называют относительным числом обусловленности. В неравенствах (1.10), (1.11) вместо погрешностей Δ и δ могут фигурировать их границы $\overline{\Delta}$ и $\overline{\delta}$. Обычно под числом обусловленности задачи ν понимают одну из величин ν_Δ или ν_δ, причем из смысла задачи бывает ясен выбор. Чаще всё же под числом обусловленности понимается относительное число обусловленности.

Для плохо обусловленной задачи $\nu >> 1$. В некотором смысле неустойчивость задачи – это крайнее проявление плохой обусловленности, отвечающее значению $\nu = \infty$. Конечно, ν - это максимальный коэффициент возрастания уровня ошибок, и для конкретных исходных данных действительный коэффициент возрастания может оказаться существенно меньше. Однако при выборе оценок (1.10), (1.11) стремятся к тому, чтобы не завышать значений ν_Δ и ν_δ и поэтому $\nu >> 1$ все же говорит о реальной возможности существенного роста ошибок.

§ 1.2 Численные методы

В данном параграфе будут рассмотрены некоторые особенности методов, которые используются в вычислительной математике для преобра-зования задач к виду , удобному для реализации на ЭВМ, и позволяют конструировать вычислительные алгоритмы. Мы будем называть эти методы численными и вычислительными. С некоторой

степенью условности можно разбить основные численные методы на следующие классы: 1) методы эквивалентных преобразований; 2) методы аппроксимации; 3) прямые (точные) методы; 4) итерационные методы; 5) методы статистических испытаний (метод Монте-Карло). Метод, осуществляющий вычисление решения конкретной задачи, может иметь довольно сложную структуру, но элементарными его шагами являются реализации указанных методов. Дадим о них общее представление.

Методы эквивалентных преобразований – это методы, позволяющие заменить исходную задачу другой задачей, имеющей то же решение. Выполнение эквивалентных преобразований оказывается полезным, если новая задача проще исходной или обладает лучшими свойствами или для неё существует известный метод, а может быть, и готовая программа.

Методы аппроксимации позволяют приблизить (аппроксимировать) исходную задачу другой задачей, решение которой в определенном смысле близко к решению исходной задачи. Погрешность, возникающая при такой замене, называется погрешностью аппроксимации. Как правило, аппроксимирующая задача содержит некоторые параметры, позволяющие регулировать величину погрешности аппроксимации или воздействовать на другие свойства задачи. Принято говорить, что метод аппроксимации сходится, если погрешность аппроксимации стремиться к нулю при стремлении параметров метода к некоторому предельному значению.

Прямые методы. Метод решения задачи называют прямым, если он позволяет получить решение за конечное число элементарных операций. Например, прямым является метод вычисления корней квадратного уравнения $x^2 + bx + c = 0$ по формулам

$$x_{1,2} = \left(-b \pm \sqrt{b^2 - 4ac}\right)/2. \qquad (1.12)$$

Элементарными здесь считаются четыре арифметические операции и операции вычисления квадратного корня.

Заметим, что элементарная операция прямого метода может быть довольно сложной (вычисление элементарной или специальной функции, решение системы линейных алгебраических уравнений, вычисление определенного интеграла и т.д.). То, что она принимается за элементарную, предполагает, во всяком случае, что выполнение этой операции существенно проще вычисления решения всей задачи.

При построении прямых метод существенное внимание уделяется минимизации числа элементарных операций.

Итерационные методы – это специальные методы построения последовательных приближений к решению задач. Применение метода начинается с выбора одного или нескольких начальных приближений. Для получения каждого из последующих приближений выполняется однотипный набор действий с использованием найденных ранее приближений – итераций (от латинского iteratio – повторение). Неограниченное продолжение этого итерационного процесса теоретически позволяет построить бесконечную последовательность приближений к решению. Если это последовательность сходится к решению задачи, то говорят, что итерационный метод сходится. Множество начальных приближений, для которых метод сходится, назы-вается областью сходимости метода.

Практическая реализация итерационных методов всегда связана с необходимостью выбора критерия окончания итерационного процесса. Вычисления не могут продолжаться бесконечно долго и должны быть прерваны в соответствии с некоторым критерием, связанным, например, с достижением заданной точности. Использование для этой цели, априорных оценок чаще всего оказывается невозможным или неэффективным. Качество верно описывая поведение метода, такие оценки являются завышенными и дают весьма недостоверную количественную информацию.

Для формирования критерия окончания по достижении заданной точности, как правило, используются так называемые апостериорные оценки (от латинского aposterior- после опыта)- неравенства, в которых величина погрешности оценивается через известные или получаемые в ходе вычислительного процесса величины. Хотя такими оценками нельзя воспользоваться до начала вычислений, в ходе вычислительного процесса они позволяютдавать конкретную количественную оценку погрешности.

§ 1.3. Корректность вычислительных алгоритмов

Численный метод, доведенный до степени детализации, позволявшей реализовать его на ЭВМ, принимает форму вычислительного алгоритма.

Определение 1.1. Вычислительным алгоритмом будем называть точное предписание действий над входными данными, задающее вычислительный процесс, направленный на преобразование произвольных входных данных *x* (из множества допустимых для данного алгоритма входных данных X) в полностью определяемых этими входными данными результат.

Реальный вычислительный алгоритм складывается из двух частей абстрактного вычислительного алгоритма, формулируемогов общепринятых математических терминах, и программы, записанной на одном из алго-ритмических языков и предназначенной для реализации алгоритма на ЭВМ. Как правило, в руководствах по методам вычислений излагаются именно абстрактные алгоритмы, но их обсуждение проводится так, чтобы выявить те или иные особенности алгоритмов, которые оказывают существенное влия-ние на эффективность программной реализации.

К вычислительным алгоритмам, предназначенным для широкого использования, предъявляется ряд весьма жестких требований. Первое из них - корректность алгоритма.

Определение 1.2. Будем называть вычислительный алгоритм корректным, если выполнены три условия: 1) он позволяет за конечное число элементарных для компьютера операций преобразовать входное данное x € X в результат *y;* 2) результат *y* устойчив по отношению к малым возмущениям входных данных; 3) результат *y* обладает вычислительной устойчивостью. Если хотя бы одно из условий 1-3 не выполнено, будем называть алгоритм некорректным.

Необходимость условия 1 понятна. Если для получения результата нужно выполнить бесконечное число операций либо требуются операции, не реализованные на компьютере, алгоритм следует признать некорректным.

Устойчивость результата *y* к малым возмущениям, входных данных (устойчивость по входным данным) означает, что результат непрерывным образом зависит от входных данных при условии, что отсутствует вычислительная погрешность. Это требование устойчивости аналогично требованию устойчивости вычислительной задачи. Отсутствие такой устойчивости делает алгоритм непригодным для использования на практике.

Отметим, что в формулировку устойчивости алгоритма по входным данным входит неявно одно очень важное предположение. Предполагается, что вместе с входным данным x в множество допустимых входных данных x входят и все близкие к x приближенные входные данные x^*.

Вычислительная устойчивость. Из-за наличия ошибок округления при вводе входных данных в компьютер и при выполнении арифметических операций неизбежно появление вычислительной погрешности. Ее величина будет разной на различных компьютерах из-за различий в разрядности и способах округления, но для фиксированного алгоритма в основном величи-на погрешности определяется машинной точностью

Определение 1.3. Назовем алгоритм вычислительно устойчивым, если вычислительная погрешность результата стремится к нулю при $\varepsilon_м \to 0$. Обычно вычислительный алгоритм называют устойчивым, если он устойчив по входным данным и вычислительно устойчив, и неустойчивым, если хотя бы одно из этих условий не выполнено.

Варианты практических заданий к главе 1

1) Определить, какое равенство точнее.
2) Округлить сомнительные цифры числа, оставив верные знаки а) в узком смысле; б) в широком смысле. Определить абсолютную погрешность результата.
3) Найти предельные абсолютные и относительные погрешности чисел, если они имеют только верные цифры: а) в узком смысле; б) в широком смысле.

№ 1

1) √44 = 6.63; 19 / 41 = 0.463
2) а) 22.563(±0.016);
   б) 2.8546; δ = 0.3%;
3) а) 0.2387; б) 42.884.

№ 2

1) 7/15=0.467; √30= 5.48;
2) а) 17.2834; δ = 0.3%;
   б) 6.4257 (±0.0024);
3) а) 3.751; б) 0.537.

№ 3

1) √10.5=3.24; 4 / 17 = 0.235.
2) а) 34.834; δ = 0.1%;
   б) 0.5748 (±0.0034);
3) а) 11.445; б) 2.043.

№ 4

1) 15/7=2.14; √10= 3.16;
2) а) 2.345 (±0.0042);
   б) 0.34484; δ = 0.4%;
3) а) 2.3445; б) 0.745.

№ 5

3) 6/7=0.857; √4.8 = 2.19;

№ 6

1) 12/11=1.091; √6.8= 2.61;

4) а) 5.435(±0.0028);
б) 10.8441; δ = 0.5%;
3) а) 8.345; б) 0.288.

№ 7

5) 2/21=0.095; √22 = 4.69;
6) а) 2.4543(±0.0032);
б) 24.5643; δ = 0.1%;
3) а) 0.374; б) 4.348.

№ 9

7) 8/11=0.545; √83 = 9.11;
8) а) 21.68563; δ = 0.3%;
б) 3.7834 (±0.0041);
3) а) 41.72; б) 0.678.

2) а) 8.24153; δ = 0.2%;
б) 0.12356 (±0.00036);
3) а) 12.45; б) 3.4453.

№ 8

1) 23/15=1.53; √9.8= 3.13;
2) а) 23.574; δ = 0.2%;
б) 8.3445 (±0.0022);
3) а) 20.43; б) 0.576.

№ 10

1) 17/19=0.889; √52= 7.21;
2) а) 13.537(±0.0026)
б) 7.521; δ = 0.12%;
3) а) 5.634; б) 0.0748.

ГЛАВА 2.
МЕТОДЫ ПОИСКА РЕШЕНИИ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

§ 2.1. Постановка задачи. Определения. Основные этапы решения

Задача поиска корней нелинейного уравнения с одним неизвестным вида

$$f(x)=0 \qquad (2.1)$$

часто возникает как элементарный шаг при решении различных научных и технических проблем.

Определение 2.1. Корнем (или решением) уравнения (2.1) называется значение $\bar{x}$ при котором $f(\bar{x})=0$.

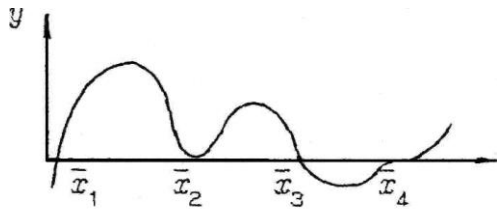


Рис.2.I Графическое изображение корней нелинейного уравнения

Корень $\bar{x}$ уравнения (2.1) называется простым, если $f'(\bar{x})\neq 0$. В противном случае, т.е.в случае $f'(\bar{x})=0$, корень называется кратным. Целое число m будем называть кратностью корня $\bar{x}$, если $f'^{(k)}(\bar{x})=0$ для $k=1,\quad 2,\ldots,m-1$ и $f^{(k)}(\bar{x})\neq 0$.

Геометрически корень $\bar{x}$ соответствует точке пересечения графика функции $y=f(x)$ с осью Ox. Корень x является простым, если график пересекает ось Ox под ненулевым углом, и является кратным, если пересечение происходит под нулевым углом.

Функция f, график которой изображен на рис. 2.1, имеет 4 корня. Корни $\bar{x}_1$ и $\bar{x}_3$ - простые, $\bar{x}_2$ и $\bar{x}_4$ - кратные.

Отметим, что задача поиска простых корней является существенно более простой (и чаще встречающейся), чем задача поиска кратных корней. В действительности большинство методов решения уравнения (2.1) ориентировано именно на вычисление простых корней.

Уточнение, постановки задачи. . При решении конкретной задачи часто представляют интерес не все корни уравнения, а лишь некоторые из них. Тогда постановку задачи уточняют, указывая на то, какие из корней подлежат определению (положительные корни, корни из заданного интервала, максимальный из корней и т.д.). В подавляющем большинстве случаев найти точное решение нелинейного уравнения (2.1) (представить решение в виде конечной формулы) оказывается невозможным. Даже для простейшего алгебраического уравнения степени *n*

$$x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 = 0 \qquad (2.2)$$

явные формулы, выражающие его корни через коэффициенты посредством конечного числа арифметических операций и операций извлечения корней степени 5 выше n, найдены лишь при n = 2,3,4. Однако уже для уравнения степени 5 и выше таких формул не существует. Этот замечательный факт, известный как теорема Абеля, был установлен в тридцатые года прошлого века Н.Абелем и Э.Галуа.

В реальных исследованиях зависимость $y = f(x)$ часто является лишь приближенным описанием, моделирующим истинную связь между параметрами x и y. Поэтому даже точное решение $\bar{x}$ уравнения (2.1) все равно является лишь приближенным значением того параметра x, который в действительности соответствует $y = 0$. Кроме того, даже если уравнение (2.1) допускает возможность найти решение в виде конечной формулы, то результат вычислений по этой формуле почти с неизбежностью содержит вычислительную погрешность и поэтому является приближенным.

В дальнейшем мы откажемся от попыток найти точные значения корней уравнения (2.1) и сосредоточим внимание на методах решения более реалистичной задачи приближенного вычисления корней с заданной точностью ε.

В данной главе под задачей поиска решений уравнения (2.1) будем понимать задачу вычисления с заданной точностью ε конечного числа подлежащих определению корней этого уравнения.

Основные этапы. Решение задачи поиска корней нелинейного уравнения осуществляется в два этапа. Первый этап называют этапом локализации или отделения корней, второй – этапом итерационного уточнения корней.

Локализация корней. Отрезок [a,b], содержащий только один корень $\bar{x}$ уравнения (2.1), называют отрезком локализации корня $\bar{x}$.

Цель этапа локализации считают достигнутой, если для каждого из подлежащих определению корней удалось указать отрезок локализации (его длину стараются по возможности сделать минимальной).

Способы локализации корней многообразны и указать универсальный метод не представляется возможным. В простых задачах хороший результат может давать графический метод (пример 2.1). Широко применяется построение таблиц значений функций f вида $y_1 = f(x_1)$, i=1,2,...,n. При этом способе локализации о наличии на отрезке $[x_{j-1}, x_j]$ корня судят по перемене знака функции на концах отрезка (пример 2.2). Основанием служит следующая хорошо известная теорема математического анализа.

Теорема 2.1. Пусть функция f непрерывна на отрезке [a,b] и принимает на его концах значения разных знаков, т.е. $f(a)\cdot f(b) < 0$. Тогда отрезок [a,b] содержит, по крайней мере, один корень уравнения $f(x) = 0$.

К сожалению, корень четной кратности не удается локализовать по перемене знака с помощью даже очень подробной таблицы. Дело в том, что в малой окрестности такого корня (например, корня $\bar{x}_2$ на рис. 2.1) функция f имеет постоянный знак.

Пример 2.1. Локализуем корни уравнения

$$4(1-x^2) - e^x = 0. \qquad (2.3)$$

Для этого преобразуем уравнение к виду $1-x^2 = 0{,}25\,e^x$ и построим графики функции $y = 1-x^2$ и $y = 0{,}25e^x$ (рис. 2.2).

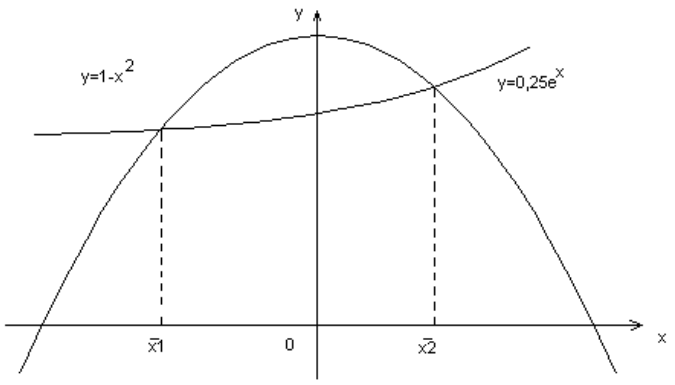


Рис. 2.2. Отделение корней уравнения (2.3)

Абсциссы точек пересечения этих графиков являются корнями исходного уравнения. Из рис. 2.2 видно, что уравнение имеет два корня $\bar{x}_1$ и $\bar{x}_2$, расположенные на отрезках [-1,0] и [0,1]. Убедимся, что функция $f(x) = 4(1-x^2) - e^x$ принимает на концах указанных отрезков значения разных знаков. Действительно, $f(-1) = -e^{-1} < 0$, $f(0) = 3 > 0$, $f(1) = -e < 0$.

Следовательно, в силу теоремы 2.1 на каждом из отрезков [-1,0], [0,1] находится, по крайней мере, один корень.

Пример 2.2. Локализуем корни уравнения

$x^3 - 1{,}1x^2 - 2{,}2x + 1{,}8 = 0.$

Для этого составим таблицу значений $f(x) = x^3 - 1{,}1x^2 - 2{,}2x + 1{,}8$ на отрезке [-2,2] с шагом 0,4.

Таблица 2.1

x	-2,0	-1,6	-1,2	-0,8	-0,4	0,0	0,4	0,8
f(x)	-6,200	-1,592	1,128	2,344	2,440	1,800	0,808	-0,152

x	1,2	1,6	2,0
f(x)	-0,696	-0,440	1,000

Из таблицы 2.1 видно, что функция f меняет свой знак на концах отрезков [-1,6;-1,2], [0,4;0,8], [1,6;2,0].

Теорема 2.1 дает основание утверждать, что каждый из этих отрезков содержит, по крайней мере, один корень. Учитывая, что в силу основной теоремы алгебры многочлен третьей степени не может иметь более трех корней, делаем вывод о том, что полученные три отрезка содержат ровно по одному корню. Таким образом, корни локализованы.

Рассмотрим программную реализацию табличного метода локализации корней уравнения

$$J_n(x) = 0,$$

где $J_n(x)$ – функция Бесселя первого рода n-го порядка. Функцию Бесселя будем вычислять с помощью ряда

$$J_n(x) = (x/2)^n \sum_{k=0}^{\infty} (-1)^k \frac{(x/2)^{2k}}{k!(k+n)!} \quad .$$

Параметрами левой части уравнения являются величины: p_1 =n - порядок функции Бесселя, p_2= ε – допустимая погрешность вычисления ряда. На языке Бейсик (программа 2.1В) основная программа занимает строки 10-90, а программа вычисления левой части уравнения – строки 100-190.

```
1     rem                                    программа 2.1В
10    dim   P(9)
20    print   "X0,x9,h";\input  x0,x9,h
30    print   "сколько параметров";\input  n
40    for  k=1 to n\print "P"; k;\ input P(k)\ next k
50    for x=x0 to x9 step h
60    gosub  100
70    print  x, F \ next x
90    goto  20
100   R=x/2  \  T=1
110   for k=1 to P(1) \ T=T*R/k  \  next k
120   k=1  \  F=T  \  R=-R*R
130   T=T*R/(k*(k+P(1)))\  F=F+T  \  k=k+1
140   if abs(T)>P(2)  then 130
190   return
```

```
                                                программа 2.1Р
var P:   array [1…9]  of  real; x, x0, x9, h: real;
n, k: integer;
function F(x:real): real;  var R, S, T: real;  n: integer;
begin  R:=x/2 ;  T:=1.0;  n:= round(P[1]);
for k:=1 to n do T:=T*R/k; k:=1; S:=T;
if abs(R)> 1.0E-18  then  begin R:=-R*R;
while abs(T)>P[2] do begin
T:=T*R/(k*(k+n)); S:=S+T;  k:=k+1  end
end;  F:=S  end:  begin
repeat write('x0,x9,h?'); readln(x0, x9, h);
write('сколько параметров?') ;  readln(n);
for k:=1 to n do begin
write('P(',k,')?'); read(P[k])  end;
x:=x0;
while x<=x9 do begin
writeln(x, '  ', f(x));
x:=x+h
end;
writeln
```

```
until false
end.

c                          программа 2.1F
c                          табличный метод отделения корней
      common  p(9)
1     type  *, 'x0, x9, h?'
      accept  *, x0, x9, h
      type  *, 'сколько параметров?'
      accept *,n
      if  (n.eq.0)  goto  3
      do 2  k=1, n
      type  5, k
2     accept  *, p(k)
3     n=(x9-x0)/ h+1.5
      x=x0
      do  4  k=1, n
      y=f(x)
      type  *, x, y
4     x=x+h
5     format  ('p(',12,')?')
      goto  1
      end
      function f(x)
      common  p, e
      n=p
      r=x/2
      t=1.0
      if (n.eq.0) goto 12
      do 11 k=1,n
 11   t=t*r/k
 12   k=1
      f=t
      r=-r*r
 13    t=t*r/(k*(k+n))
      f=f+1
      k=k+1
      if    (abs(t).gt.e)goto
 13
      return
      end
```

В строке 10 описывается массив P(9) для размещения параметров левой части решаемого уравнения. Затем в строке 20 находятся операторы диалогового ввода с клавиатуры интервала [x0,x9] и шага h изменения аргумента ***x***, а в строке 30 - операторы ввода количества параметров уравнения n. В строке 40 в цикле по переменной k последовательно

задаются значения параметров P(1),P(2),... . Заголовок цикла для изменения аргумента ***x*** левой части уравнения находится в строке 50. В этом цикле осуществляется обращение к подпрограмме вычисления левой части решаемого уравнения (строка 60) и выводятся на дисплей значения $(x_0,f(x_0)),...,(x_n,f(x_n))$(строка 70).

Для того, чтобы иметь возможность без повторных запусков программы изменять параметры задачи, осуществляется] в конце основной прог-раммы безусловная передача из строки 90 на ее начальные исполняемые операторы. Поэтому для окончания работы с программой необходимо пре-рвать ее выполнение с клавиатуры (CTRL/C). Функция Бесселя вычисляется с помощью подпрограммы (строки 100 - 190).

Аналогичную структуру имеют программы на языках Фортран и Паскаль (программы 2.1F и 2.1P). В программе 2.IF параметры передаются из основной программы в подпрограмму-функцию вычисления левой части уравнения F(x) через неименованный common-блок. Ввод исходных данных осуществляется также в диалоговом режиме .

В программе 2.IP параметры P[k] передаются в подпрограмму – функ-цию F(x) как глобальные переменные. Преобразование вещественной переменной в целую осуществляется с помощью стандартной функции round. Использование условного оператора, проверяющего переменную R, связано с тем, что при использовании одного из компиляторов с языка Паскаль создавалась программа, приводящая к прерыванию при малых аргументах X, связанному с исчезновением порядка при выполнении операции возведения в квадрат.

Итерационное уточнение корней. На этом этапе для вычисления каждого из корней с точностью $\varepsilon>0$ используется тот или иной итерационный метод, позволяющий построить последовательность $x^{(0)}, x^{(1)}, x^{(2)},..., x^{(n)},...$ приближений к корню $\bar{x}$.

Общее представление об итерационных методах и основные определения были даны в § 1.2. Введем дополнительно некоторые определения.

Определение 2.2. Итерационный метод называется одношаговым, если для вычисления очередного приближения $x^{(n+1)}$ используется только одно предыдущее приближение $x^{(n)}$. Если для вычисления $x^{(n+1)}$ используется k предыдущих приближений $x^{(n-k+1)}, x^{(n-k+2)},..., x^{(n)}$, то итерационный метод называют k-шаговым.

Заметим, что для построения итерационной последовательности одношаговым методом требуется задание только одного начального приближения $x^{(0)}$, в то время как при использовании k-шагового метода k начальных приближений $x^{(0)}, x^{(1)},..., x^{(k-1)}$.

Определение 2.3 Будем говорить, что итерационный метод сходится со скоростью геометрической прогрессии со знаменателем q<1, если для всех n верна оценка погрешности

$$\left|x^{(n)} - \bar{x}\right| \le C_0 q^n \qquad (2.4)$$

Пусть k-шаговый итерационный метод обладает следующим свойством. "Существует δ-окрестность корня $\bar{x}$ такая, что. если приближения $x^{(n-k+1)}, x^{(n-k+2)}, \ldots, x^{(n)}$ принадлежат этой окрестности, то справедлива оценка

$$\left|x^{(n+1)} - \bar{x}\right| \le C\left|x^{(n)} - \bar{x}\right|^P \qquad (2.5)$$

с постоянными C>0 и P>1. В этом случае число P называют порядком точности метода. Если P=1 и C<1, то говорят, что метод обладает линейной скоростью сходимости в указанной δ – окрестности корня. Если P>1, то принято говорить о сверхлинейной скорости сходимости. При P=2 скорость сходимости называют квадратичной, а при P=3 – кубической.

§ 2.2. Обусловленность задачи вычисления корня

Пусть $\bar{x}$ - корень уравнения (2.1), подлежащий определению. Будем считать, что входными данными для задачи вычисления корня $\bar{x}$ являются значения $f(x)$ функции f в малой окрестности корня. Так как значения $f(x)$ вычисляются на ЭВМ по некоторой программе, в действительности задаваемые значения будут приближенными и их следует обозначать через $f^*(x)$. Ошибки в $f^*(x)$ могут быть связаны как с ошибками округления, так и с использованием для вычисления f приближенных методов.

Интервал неопределенности. Допустим, что в достаточно малой окрестности выполняется неравенство $|f(x) - f^*(x)| < \bar{\Delta}$, где $\bar{\Delta} = \bar{\Delta}(f^*)$ - граница абсолютной погрешности. Вблизи корня погрешность ведет себя крайне нерегулярно и в первом приближении может восприниматься пользователем как некоторая случайная величина. На рис. 2.3 представлена идеальная ситуация, отвечающая исходной математической постановке задачи, а на рис. 2.3,б – реальная ситуация, соответствующая вычислениям функции на ЭВМ.

Если функция f непрерывна, то найдется такая малая окрестность $(\bar{x}-\bar{\varepsilon},\bar{x}+\bar{\varepsilon})$ корня $\bar{x}$ радиуса $\bar{\varepsilon}>0$, в которой выполняется неравенство

$$|f(x)|<\Delta. \qquad (2.6)$$

Для $x\,\epsilon\,(\bar{x}-\bar{\varepsilon},\bar{x}+\bar{\varepsilon})$, знак вычисленного значения $f^*(x)$, вообще говоря, не обязан совпадать со знаком $f(x)$, и, следовательно, становится невозможным определить, какое именно значение x из интервала $(\bar{x}-\bar{\varepsilon},\bar{x}+\bar{\varepsilon})$ обращает функцию f в нуль (рис. 2.3,б).

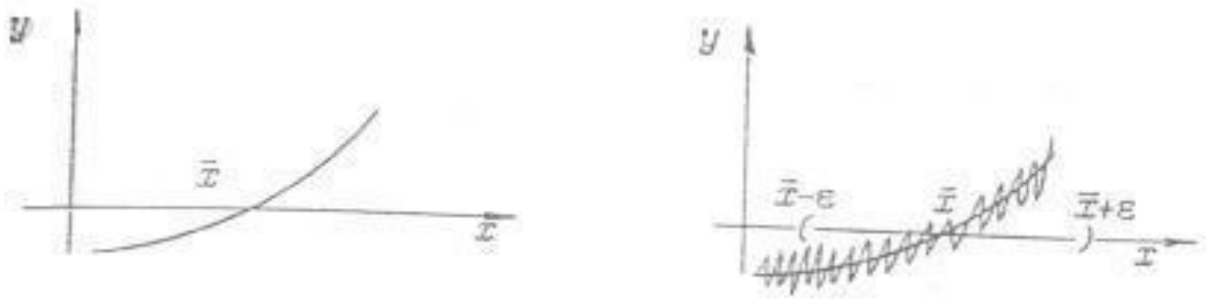
Интервал $(\bar{x}-\bar{\varepsilon},\bar{x}+\bar{\varepsilon})$ называется интервалом неопределенности корня $\bar{x}$. Сделаем оценку величины $\bar{\varepsilon}$. Пусть корень $\bar{x}$ простой. Для x, близких к $\bar{x}$, справедливо приближенное равенство

$$f(x)\cong f(\bar{x})+f'(\bar{x})(x-\bar{x})=f'(\bar{x})(x-\bar{x}).$$

Тогда неравенство (2.6) принимает вид $|f'(\bar{x})(x-\bar{x})|\lesssim\Delta$, а отсюда получаем

$$\bar{x}-\bar{\Delta}/|f'(\bar{x})|\lesssim x\lesssim\bar{x}+\bar{\Delta}/|f'(\bar{x})|.$$

Считая, что приближенное значение x^* принадлежит интервалу неопределен-ности, в этом неравенстве можем заменить x на x^*.



а) б)

Рис. 2.3. Изображения графика функции $f(x)$ вблизи корней: а – истинное изображение, б – изображение, построенное по приближенно вычисленным значениям f

Следовательно, для простого корня $\bar{x}$ справедлива оценка

$$\bar{\Delta}(x^*) = \varepsilon^* \cong \nu_\Delta \bar{\Delta}(f^*). \qquad (2.7)$$

Число $\nu_\Delta = 1/|f'(\bar{x})|$ играет здесь роль абсолютного числа обусловленности. Радиус интервала неопределенности оказывается прямо пропорциональным погрешности $\bar{\Delta}$ вычисления значения f. Кроме того, значение ε возрастает (обусловленность задачи ухудшается) с уменьшением $|f'(\bar{x})|$, т.е. уменьшением модуля тангенса угла наклона, под которым график функции пересекает ось Ox (рис. 2.4).

На практике оценить величину радиуса интервала неопределенности даже по порядку довольно сложно. Однако знать о его существовании нужно, по крайней мере, по двум причинам. Во-первых, не имеет смысла ставить задачу о вычислении корня $\bar{x}$ с точностью $\varepsilon < \bar{\varepsilon}$. В условиях неопределенности, вызванных приближенным заданием функции, любое значение $\bar{x}^* \in (\bar{x} - \bar{\varepsilon}, \bar{x} + \bar{\varepsilon})$ может быть с одной и той же достоверностью принято за решение уравнения. Во-вторых, нельзя требовать от алгоритмов поиска корня достоверной информации после того, как очередное приближении попало в интервал неопределенности или оказалось очень близко от него; в этой ситуации вычисления следует прекратить и считать, что получен максимум действительно возможного.

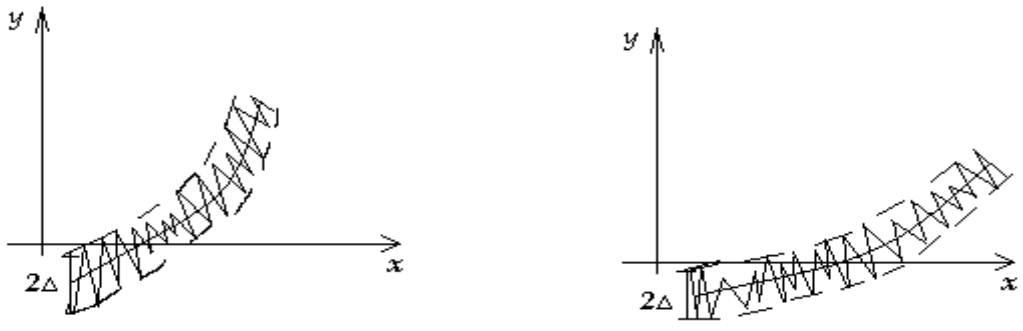


Рис. 2.4. Зависимость $\bar{\varepsilon}$ от величины f ‘(x)

Для большинства итерационных методов определить этот момент можно, так как, начиная с него, поведение приближений становится крайне нерегулярным. Если вдалеке от интервала неопределенности величина

$$q^{(n)} = \left|x^{(n)} - x^{(n-1)}\right| / \left|x^{(n-1)} - x^{(n-2)}\right| \qquad (2.8)$$

обычно бывает меньше единицы $\left(\left|x^{(n)} - x^{(n-1)}\right| < \left|x^{(n-1)} - x^{(n-2)}\right|\right)$, то появление при некотором n значение $q^{(n)} > 1$, свидетельствует, скорее всего, о начале "раз-болтки" – хаотического поведения итерационной последовательности.

В этой ситуации вычисления имеет смысл прервать, чтобы выяснить причину явления и принять правильное решение. Использование для контро-ля вычислений величины (2.8) называют часто Гарвика.

§ 2.3. Метод бисекции

Описание метода. Пусть требуется с заданной точностью $\varepsilon > 0$ найти корень $\bar{x}$ уравнения (2.1). Отрезок локализации [a,b], т.е. отрезок, содержащий только один корень $\bar{x}$, будем считать заданным.

Предположим, что функция f непрерывна на отрезке [a,b] и на его концах принимает значения различных знаков:

$$f(a)\,f(b) < 0 \qquad (2.9)$$

На рис. 2.5 изображен случай, когда $f(a) < 0$ и $f(b) > 0$. В дальнейшем изложении будем обозначать отрезок [a,b] через $[a^{(0)},b^{(0)}]$. Примем за исходное значение корня середину отрезка – точку $x^{(0)} = (a^{(0)}+b^{(0)})/2$. Так как положение корня $\bar{x}$ на отрезке $[a^{(0)},b^{(0)}]$ неизвестно, то утверждать можно лишь то, что погрешность этого приближения не превышает половины длины отрезка (рис. 2.6.):

$$\left|x^{(0)} - \bar{x}\right| \le \left(b^{(0)} - a^{(0)}\right)/2 \qquad (2.10)$$

Погрешность можно уменьшить путем уточнения отрезка локализации, т.е. замены начального отрезка $[a^{(0)},b^{(0)}]$ отрезком $[a^{(1)},b^{(1)}]$ меньшей длины. Согласно методу бисекции (половинного деления), в

качестве $[a^{(1)},b^{(1)}]$ берут тот из отрезков $[a^{(0)},x^{(0)}]$ и $[x^{(0)},b^{(0)}]$, на концах которого выполняется условие $f(a^{(1)})f(b^{(1)})<0$.

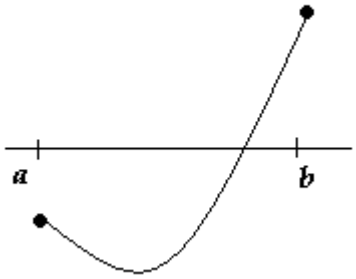


Рис. 2.5. График функции *f(x)*

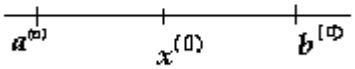


Рис.2.6. Точка $x^{(0)}$ – середина отрезка $[a^{(0)};b^{(0)}]$

Этот отрезок содержит искомый корень. Действительно, если $f(a^{(1)})f(b^{(1)})<0$, то наличие корня следует из теоремы 2.1; если же $f(a^{(1)})f(b^{(1)})=0$, то корнем является один из концов отрезка. Середина полученного отрезка $x^{(1)} = (a^{(1)} + b^{(1)})/2$ дает приближение к корню с оценкой погрешности

$$\left|x^{(1)}-\bar{x}\right|\le\left(b^{(1)}-a^{(1)}\right)/2=(b-a)/2^2.$$

За очередное уточнение отрезка локализации $[a^{(2)}, b^{(2)}]$ снова берется тот из отрезков $[a^{(1)}, x^{(1)}]$, $[x^{(1)}, b^{(1)}]$, на концах которого выполняется условие $f(a^{(2)})f(b^{(2)})\le 0$.

Рассмотрим $(n+1)$ итерацию метода. Пусть отрезок $[a^{(n)},b^{(n)}]$ уже найден и вычислены значения $x^{(n)}, f(a^{(n)}), f(b^{(n)})$. Тогда выполняются следую-щие действия:

1. Вычисляется $f(x^{(n)})$.

2. Если $f(a^{(n)})f(b^{(n)})<0$, то за отрезок локализации $[a^{(n+1)},b^{(n+1)}]$ принимается отрезок $[a^{(n)},x^{(n)}]$. В противном случае $f(x^{(n)})f(b^{(n)})<0$ и за $[a^{(n+1)},b^{(n+1)}]$ принимается отрезок $[x^{(n)},b^{(n)}]$.

3. Вычисляется $x^{(n+1)}=(a^{(n+1)},b^{(n+1)})/2$.

Неограниченное продолжение итерационного процесса даёт последо-вательность отрезков $[a^{(0)},b^{(0)}],[a^{(1)},b^{(1)}],\ldots,[a^{(n)},b^{(n)}]$, содержащих искомый

корень. Каждый из них (за исключением начального) получен делением пополам предыдущего отрезка.

Скорость сходимости. Середина n-го отрезка $x^{(n)} = (a^{(n)}, \mathrm{b}^{(n)})/2$ даёт приближение к корню $\bar{x}$ с оценкой погрешности

$$\left| x^{(n)} - \bar{x} \right| \le (b^{(n)} - a^{(n)})/2 = (b-a)/2^{(n+1)}. \qquad (2.11)$$

Из (2.11) видно, что метод бисекции сходится со скоростью геометрической прогрессии со знаменателем $q = 1/2$. По сравнению с другими методами метод бисекции сходится довольно медленно. Однако следует заметить, что для достижения разумной точности ε число итераций не будет очень большим. Например, для уменьшения первоначального отрезка локализации в 10^6 раз нужно 19 итераций.

Критерий окончания. Итерации следует вести до тех пор, пока не выполнится неравенство $b^{(n)} - a^{(n)} < 2\varepsilon$. При его выполнении, согласно оценке (2.11), можно принять $x^{(n)}$ за приближение к корню с точностью ε.

Пример 2.3. Найдём методом бисекции с точностью $\varepsilon = 10^{-2}$ положительный корень уравнения $4(1-x^2) - e^x = 0$. В примере 2.1 этот корень был локализован на отрезке $[0,1]$, причём $f(0) > 0, f(1) < 0$. Допустим, что $a^{(0)} = 0, b^{(0)} = 1, x^{(0)} = (a^{(0)} + b^{(0)}0/2 = 0.5$.

Первая итерация. Вычислим $f(x^{(0)}) \cong 1{,}3512$. Так как $f(a^{(0)}) \cdot f(x^{(0)}) > 0$, то $[a^{(1)}, \mathrm{b}^{(1)}] = [0.5,1]$. Вычисляем $x^{(1)} = 1/2(a^{(1)} + b^{(1)}) = 0{,}75$.

Вторая итерация. Вычисляем $f(x^{(1)}) \cong -0{,}3670$. Так как выполняется неравенство $f(a^{(1)}) f(x^{(1)}) < 0$, то $[a^{(2)}, \mathrm{b}^{(2)}] = [0.5,0.75]$ и $x^{(2)} = (a^{(2)} + b^{(2)})/2 = 0{,}625$.

Результаты следующих итераций (с четырьмя цифрами после десятичной точки) приведены в табл. 2.2. При n=6 выполняется неравенство $b^{(6)} - a^{(6)} < 2 \cdot 10^{-2}$. Следовательно, заданная точность достигнута и можно принять $\bar{x} \cong x^{(6)}$. В результате $\bar{x} = 0.71 \pm 0.01$.

Программу метода бисекции реализуем в виде трёх блоков (рис. 2.7). В главной программе (блок 0) в диалоговом режиме зададим интервал [A,B], где расположен корень, погрешность вычисления корня E и левой части уравнения E1, а также параметры решаемого уравнения P1, P2,… и

обратимся к программе метода (блок 1). В блоке 2 вычисляется левая часть уравнения.

Таблица 2.2

Номер итерации n	$a^{(n)}$	$b^{(n)}$	$x^{(n)}$	$f(x^{(n)})$	$b^{(n)} - a^{(n)}$
0	0,0000	1,0000	0,5000	1,3513	1,0000
1	0,5000	1,0000	0,7500	-0,3670	0,5000
2	0,5000	0,7500	0,6250	0,5693	0,2500
3	0,6250	0,7500	0,6875	0,1206	0,1250
4	0,6875	0,7500	0,7187	-0,1182	0,0625
5	0,6875	0,7187	0,7031	0,0222	0,0312
6	0,7031	0,7187	0,7109		0,0156

Программы 2.2B, 2.2P, 2.2F ориентированы на решение уравнения $K(x) - P_1 = 0$, где $P_1 \in [\pi/2; \infty]$ - параметр уравнения;

$K(x) = \int_0^{\pi/2} (1 - x \cdot \sin^2 t)^{-1/2} dt$ - полный эллиптический интеграл первого рода.

Решить данное уравнение означает найти такую величину x, при которой интеграл принимает заданное значение P_1. Интеграл $K(x)$ вычислим по методу Кинга.

На языке Бейсик (программа 2.2B) из главной программы (строки 10-90) после задания исходных данных происходит обращение к подпрограмме метод бисекции (строки 100 - 190). Левая часть уравнения вычисляется в строках 200 – 290. Такая структура программы позволяет легко переходить к решению другого уравнения без изменения главной программы и подпрограммы метода.

Знаки функции $f(x)$ в точке A и B в программах метода бисекции на Бейсике и Фортране определяются с помощью стандартных функций SGN и $SIGN$ соответственно, на Паскале используется подпрограмма-функция $SGN(X)$. Так как при уменьшении интервала $[A, B]$ в процессе половинного деления точка A всегда остается слева от искомого корня, то не смотря на ее перемещение, знак $f(A)$ не будет изменяться. Поэтому знак

$f(A)$ определяет ся один раз, а в процессе делении исходного интервала знак функции $f(x)$

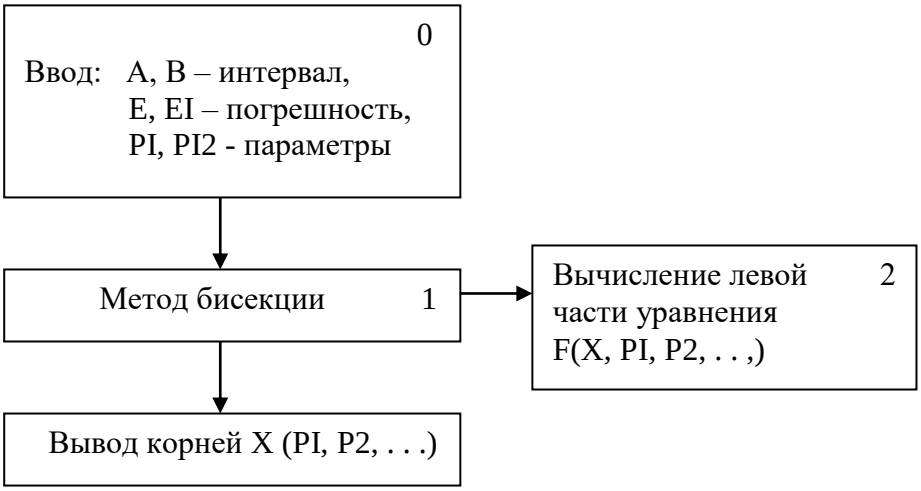


Рис. 2.7. Алгоритм поиска корня методом бисекции

определяется только в средней точке $\bar{x} = (A + B)/2$ и сравнивается на совпадение или различие со знаком функции $f(a)$. При совпадении выполняется оператор присваивания $A = X$, т. е. точка A передвигается в середину отрезка, в противном случае передвигается точка B (выполняется оператор $B = X$).

В пределах программы не предусмотрена первоначальная проверка знаков левой части уравнения на концах интервала $[A, B]$. Предполагается, что эта задача выполняется с помощью программ 2.1 локализации корней.

```
1 rem                                                    программа 2.2B
2 rem                                                    метод бисекции
10 dim p(9)
20 print "a,b,e"; \ input a,b,e
30 print "сколько параметров?"; \ input n
40 for k=1 to n \ print "p"; k; \ input p(k)\next k
50 gosub 100
60 print "x=";x
90 goto 20
100 x=a \ gosub 200
110 s=sgn(f)
120 x=(a+b)/2 \ gosub 200
```

```
130 if abs(f)<p(2) then return
140 if sgn(f)=s then a=x \ goto 160
150 b=x
160 if b-a>e then 120
190 return
200 r=1\r1=sqr(1-x)
210 r2=(r+r1)/2\r1=sqr(r*r1)\r=r2
220 if r-r1>p(2) then 210
230 f=r*p(1)-pi/2
290 return
```

программа 2.2F
метод бисекции

```
      external f
      common p(9)
1     type *, 'a, b, e?'
      accept *, a, b, e
      accept *, 'сколько параметров'
      accept *, n
      if (n.eq.0) goto 3
      do 2 k=1, n
      type 4, k
2     accept *, p(k)
3     call dich (a, b, x, e, p(2), f)
      type *, 'x=', x
      goto 1
4     format ('p'(',i1,') ?')
      end
    subroutine dich(a,b,x,e,e1,f) s=sign(1.0,f(a))
11  x=(a+b)/2
    r=f(x)
    if (abs(r).le.e1) return
    if (sign(1.0,r).eq.s) a=x
    If (a.ne.x) b=x
    if ((b-a).gt.e) goto 11
    return
    end
    function f(z)
    common c,e
    r=1.
    r1=sqrt(1.-x)
21  r2=(r+r!)/2
```

```
r1=sqrt(r*r!)
r=r2
if((r-r1).gt.e) goto 21
f=c*(r+r1)-3.14159265
return
end
 (программа 2.2Р метод бисекции )
type
fr=function (x:real):real;
var p: array[1..9] of real; a,b,x,e: real;
n,k: integer;
function f(x: real): real;far;
var r,r1,r2: real;
begin r:=1.0; r1:=sqrt(1.0-x);
while r-r1>p[2] do begin
r2:=(r+r2)/2; r1:=sqrt(r*r1); r:=r2
end; f:=(r+r1)*p[1]-3.14159265 end;
function sgn(x:real): integer;
begin sgn:=0;
if x<0.0 then sgn:=-1;
if x>0.0 then sgn:=1;
end;

procedure dich(var a,b,x,e,e1:real;f:fr);
var i: integer; r: real; begin i:=sgn(f(a));
while b-a>e do begin
x:=(a+b)/2; r:=f(x); if abs(r)<e1 then exit;
if sgn(r)=i then a:=x else b:=x
end
end;
begin
repeat write('a,b,e?'); readln(a,b,e);
write('Skolko parametrov?'); readln(n);
for k:=1 to n do begin
write('p(',k,')?'); readln(p[k]);
end;
```

```
dich(a,b,X,e,p[2],f); writeln('x=',x);
until false
end.
```

Влияние вычислителъной погрешности. Для правильной работы метода бисекции принципиально важным является правильное определение знака функции f. В случае, когда $x^{(n)}$ попадает в интервал неопределенности корня (см. §2.2), вычисленное значение $f(x^{(h)})$ не обязательно должно быть верным по знаку и последующие итерации не имеют смысла. Однако метод следует признать очень надежным; он гарантирует точность приближения, примерно равную радиусу интервала неопределенности ε.

§ 2.4. Метод простой итерации

Описание метода. Применение метода простой итерации для решения нелинейного уравнения (2.I) связано с необходимостью преобразования этого уравнения к виду

$$x=\varphi(x). \qquad (2.12)$$

Преобразование (2.I2) (приведение уравнения к виду, удобному для итерации) можно выполнить различными способами; некоторые из них будут указаны ниже. Функцию φ далее будем называть итерационной функцией.

Допустим, что каким-либо образом выбрано приближенное значение корня $x^{(0)}$. Подставим $x^{(0)}$ в правую часть уравнения (2.I2) и вычислим значение $x^{(1)} = \varphi(x^{(0)})$. Подставляя теперь $x^{(1)}$ в правую часть уравнения (2.12), получим $x^{(2)} = \varphi(x^{(1)})$. Продолжая этот процесс неограниченно, получим последовательность приближений к корню, вычисляемых по формуле

$$x^{(n+1)} = \varphi(x^{(n)}), \quad n \geq 0 \qquad (2.13)$$

Заметим, что метод простой итерации – одношаговый (см. § 2.1).

Если существует предел построенной последовательности $x = \lim\limits_{n\to\infty} x^{(n)}$, то переходя к пределу в равенстве (2.13) и предполагая функцию φ непрерыв-ной, получим равенство

$$\bar{x} = \varphi(\bar{x}). \qquad (2.14)$$

Это означает, что $\bar{x}$ - корень уравнения (2.12).

Геометрическая иллюстрация метода простой итерации. Построим график функции $y = x$ и $y = \varphi(x)$. На рис. 2.8 корень $\bar{x}$ уравнения (2.12) является абсциссой точки пересечения графиков функций $y = x$ и $y = \varphi(x)$. Выберем начальное приближение $x^{(0)}$, которому отвечает расположенная на кривой $y = \varphi(x^{(0)})$ точка $M^{(0)}$ с координатами $(x^{(0)}, x^{(1)})$ (напомним, что $x^{(1)} = \varphi(x^{(0)})$). Соединим точку $M^{(0)}$ отрезком прямой $y = x^{(1)}$ с расположенной на прямой $y = x^{(1)}$ точкой $N^{(1)}$ с координатами $(x^{(1)}, x^{(1)})$. Проведем теперь через точку $N^{(1)}$ прямую $x = x^{(1)}$ до пересечения с кривой $y = \varphi(x)$ в точке $M^{(1)}$ с координатами $(x^{(1)}, x^{(2)})$. Продолжая этот процесс, получаем ломанную линию $M^{(0)}\, N^{(1)}\, M^{(1)}\, N^{(2)}$..., для которой абсциссы точек $M^{(n)}$ представляют собой последовательные приближения $x^{(n)}$ к решению $\bar{x}$.

Сходимость метода простой итерации. Сходимость метода простой итерации связана с выполнением условия $|\varphi'(x)| < 1$.

Лемма 2.1. Пусть одношаговый итерационный метод обладает линейной скоростью сходимости в некоторой δ-окрестности корня $\bar{x}$. Тогда независимо от выбора начального приближения $x^{(0)} \in (\bar{x} - \delta, \bar{x} + \delta)$ последова-тельность приближений $x^{(n)}$ не выходит за пределы этой окрестности, метод сходится со скоростью геометрической прогрессии со знаменателем $q = c$ и верна оценка погрешности

$$\left|x^{(n)} - \bar{x}\right| \le q^n \left|x^{(0)} - \bar{x}\right|, \; n \ge 0$$

Теорема 2.2. Пусть в некоторой δ-окрестности корня $\bar{x}$ функция φ дифференцируема и удовлетворяет неравенству

$$|\varphi'(x)| \le q \qquad (2.15)$$

где $0 \le q < 1$ - постоянная.

Тогда независимо от выбора начального приближения $x^{(0)}$ из указанной δ-окрестности корня итерационная последовательность не выходит из этой окрестности, метод сходится со скоростью геометрической прогрессии и справедлива оценка погрешности

$$\left|x^{(n)} - \bar{x}\right| \le q^n \left|x^{(0)} - \bar{x}\right| \qquad (2.16)$$

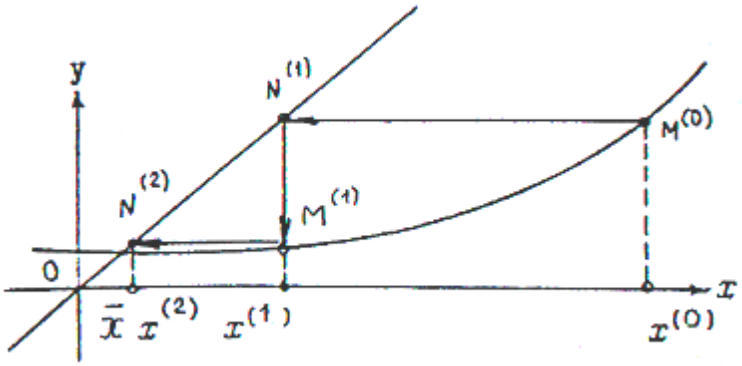


Рис. 2.8. Процесс вычисления приближений
метода простой итерации

Доказательство. Вычитая из равенства (2.13) равенство (2.14) и используя формулу конечных приращений Лагранжа, получим

$$x^{(n+1)} - \bar{x} = \varphi(x^{(n)}) - \varphi(\bar{x}) = \alpha^{(n+1)}(x^{(n)} - \bar{x}). \qquad (2.17)$$

Здесь $\alpha^{(n+1)} = \varphi'(\xi^{(n)})$, где $\xi^{(n)}$-некоторая точка, расположенная между $x^{(n)}$ и $\bar{x}$. Если $x^{(n)} \in (\bar{x} - \delta, \bar{x} + \delta)$, то $\left|\alpha^{(n+1)}\right| \le q$ в силу условия (2.13). Тогда на основании (2.15) получаем

$$\left|x^{(n+1)} - \bar{x}\right| \le q\left|x^{(n)} - \bar{x}\right|$$

Это означает, что метод простой итерации обладает линейной скоростью сходимости и поэтому доказательство теоремы завершается применением леммы 2.1.

Оценка погрешности (2.16) является априорной. Следует отметить, что неравенство (2.16), как правило, не используется для практической оценки погрешности. Одна из причин в том, что значение $\bar{x}$ неизвестно, кроме того, использование неравенства (2.16) приводит к существенно завышенной оценке погрешности.

Критерий окончания. Введем апостериорную оценку погрешности, пригодную для практического применения.

Теорема 2.3. Пусть выполнены условия теоремы 2.2 и $x^{(0)} \in (\bar{x}-\delta, \bar{x}+\delta)$. Тогда верна апостериорная оценка погрешности

$$\left|x^{(n)}-\bar{x}\right| \le \left(q/(1-q)\right)\left|x^{(n)}-x^{(n-1)}\right|, n \ge 1. \qquad (2.18)$$

Доказательство. В силу равенства (2.17) имеем $x^{(n)}-\bar{x} = \alpha^{(n)}\left(x^{(n-1)}-\bar{x}\right) = \alpha^{(n)}\left(x^{(n-1)}-x^{(n)}\right)+\alpha^{(n)}\left(x^{(n)}-\bar{x}\right)$. Отсюда

$$x^{(n)}-\bar{x} = \left(\alpha^{(n)}/\left(1-\alpha^{(n)}\right)\right)\left(x^{(n-1)}-x^{(n)}\right). \qquad (2.19)$$

Взяв модуль от левой и правой частей этого неравенства и воспользовавшись неравенством $\left|\alpha^{(n)}/\left(1-\alpha^{(n)}\right)\right| \le q(1-q)$, получим (2.17). Если величина q известна, то неравенство (2.18) дает эффективный способ контроля погреш-ности и можно сформулировать следующий критерий окончания итера-ционного процесса. Вычисления следует вести до выполнения неравенства

$$\left|x^{(n)}-x^{(n-1)}\right| < \left((1-q)/q\right)\varepsilon. \qquad (2.20)$$

Если это условие выполнено, можно считать, что $x^{(n)}$ является приближением к $\bar{x}$ с точностью до ε.

Пример 2.4. Используем метод простой итерации для вычисления положительного корня x уравнения $4\left(1-x^2\right)-e^x=0$ с точностью $\varepsilon = 10^{-4}$. Результат примера 2.3 дает для корня отрезок локализации [a,b]= [0.70,0.72].

Преобразуем уравнение к виду (2.12), где $\varphi(x) = \sqrt{1-e^x/4}$. Заметим, что $\varphi'(x) = -e^x/\left(8\sqrt{1-e^x/4}\right)$, $\varphi''(x) = -e^x\left(1-e^x/8\right)\Big/\left(8\sqrt{1-e^x/4}\right)^3$. Так как $\varphi'' < 0$ на

отрезке [a,b], производная φ' монотонно убывает и $q = \max\left|\varphi'(x)\right| = \varphi'(b) \approx 0.37$. Следовательно, условие сходимости (2.15) выполнено. Возьмем $x^{(0)} = 0.7$ и будем вести итерации до выполнения критерия (2 19). В табл. 2.3 соответствующие приближения приведены с десятью знаками мантиссы. Критерий окончания выполняется при $n = 5$. После округления значения $x^{(5)}$ до 4 значащих цифр получим $\bar{x} = 0.7035 \pm 0.0001$

Таблица 2.3

n	$x^{(n)}$	$\left(q/(1-q)\right)\left\|x^{(n)} - x^{(n-1)}\right\|$
0	0.7000000000	
1	0.7046714292	$3\cdot10^{-3}$
2	0.7029968319	$1\cdot10^{-3}$
3	0.7035984939	$4\cdot10^{-4}$
4	0.7033824994	$2\cdot10^{-4}$
5	0.7034600632	$5\cdot10^{-5}$

При решении практических задач часто вместо критерия (2.19) используется привлекательный своей простотой критерий

$$\left|x^{(n)} - x^{(n-1)}\right| < \varepsilon. \qquad (2.21)$$

Действительно, если $0 < q \le 1/2$, то $(1-q)/q > 1$, и выполнение неравенства (2.21) влечет за собой выполнение неравенства (2.20) и использование критерия (2.21) оправдано. В то же время в случае $1/2 \le q <$ использо-вание критерия (2.20) может привести к преждевременному прекращению итераций. Дело в том, что когда величина q близка к единице, итерационный процесс сходится медленно и расстояние между двумя последовательными приближениями $x^{(n)}$ и $x^{(n+1)}$ не характеризует расстояние от $x^{(n)}$ до решения $\bar{x}$.

Приведение уравнения к виду, удобному для итерации. Важным моментом в применении метода простой итерации является эквивалентное

преобразо-вание уравнения (2.1) к виду (2.12). Рассмотрим один из простых способов такого преобразования.

Предположим, что производная f' на отрезке [a,b] непрерывна и положительна. Тогда существуют положительные постоянные m, N такие, что $0 < m \le f`(x) \le M$, $x \in [a,b]$. Приведем уравнение (2.1) к виду

$$x = x - \alpha f(x), \qquad (2.22)$$

Где $\alpha > 0$. В этом случае итерационная функция φ имеет вид $\varphi(x) = x - \alpha f(x)$. Требуется выбрать α так, чтобы выполнялось условия (2.15), причем q должно быть по возможности минимальным.

Заметим,что $\mathrm{I} - \alpha \mathrm{M} \le \varphi`(x) = \mathrm{I} - \alpha f`(x) \le \mathrm{I} - \alpha m$, и поэтому $|\varphi`(x)| \le q(\alpha) = \max\{|1 - \alpha M|, |1 - \alpha m|\}$. Для того чтобы было выполнено неравенство $q(\alpha) > \mathrm{I}$, достаточно взять любое $\alpha \in (0, 2/\mathrm{M})$. Конкретный выбор параметра α зависит от наличия информации о числах m и M.

Если известны обе величины, то лучшим является выбор $\alpha = \alpha_0 = 2/(\mathrm{M} + m)$. В этом случае $q(\alpha_0) = (\mathrm{M} - m)/(\mathrm{M} + m)$. Если же известно только M, то можно положить $\alpha = \alpha_1 = \mathrm{I}/\mathrm{M}$. В этом случае $q(\alpha_1) = 1 - m/\mathrm{M}$.Кроме того, при $\alpha = \alpha_1$ производная φ`(x) неотрицательна, и в силу теоремы 2.3 сходимость будет монотонной.

Примером программной реализации метода простой итерации являются программы 2.3B, 2.3F, 2.3P. Эти программы предназначены для решения уравнения

$$\Phi(x) = \mathrm{P}_1,$$

где

$$\Phi(x) = 2/\sqrt{\pi} \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{K!(2k+1)}$$

Входными данными являются погрешность E, начальное приближение X, число итераций M, параметры уравнения PI. Если за M итераций не будет найден корень с заданной точностью, то на дисплее выведется соответствующее сообщение.

```
1 rem                                                    программа 2.3B
2 rem                                                метод простой итерации
10   dim p(9)
20   print "x, e, m";\ input x, e, m\ b=-6
30   print "Сколько параметров?";\ input n
40   for k=1 to n \ print "p"; k;\ input p(k)\ next k
50   gosub 100
60   print "x=";x
90   goto 20
100 for i=1 to m
110 x=f\gosub 200
120 if abs (x-f),e then return
130 next i
140 print "Итерации все"
190 return
200 t=2*x/sqrt(pi)\ f=t-p(1)\ r=-x*x\ k=1
210 g=f\ t=t*r/k\ f=f+t/(2*k+1)\ k=k+1
220 if <>g then 210
230 f=x+b*f
240 return
```

```
c                                                        программа 2.3F
c                                                    метод простой итерации
      external F
      common b, p(9)
      b=-6
      type *, 'x, e, m?'
      accept *, x, e, m?
      type *,  'сколько параметров?'
      accept *, n
      if (n.eq.0) goto 3
      do 2 k=1, n
      type 4,k
2     accept *, p(k)
```

```
c                                                        программа 2.3F
c                                                    метод простой итерации
      external F
      common b, p(9)
      b=-6
```

```
      type *, 'x, e, m?'
      accept *, x, e, m?
      type *,  'сколько параметров?'
      accept *, n
      if (n.eq.0) goto 3
      do 2 k=1, n
      type 4,k
2     accept *, p(k)
3     call iter (b,x,e,m,f)
      type *,'x=',x
      goto 1
4     format ('p(',i2,')?')
      end
      subroutine iter (b,x,e,m,f)
      do 11 i=1,m
        x1=x
      x=f(x)
11    if (abs(x-x1).lt.e) return
      type *,'итерации все'
      return
      end
      function f(x)
      common b,p1
      data pi/3.14159265/
      t=2*x/sqrt(pi)
      F=t-p1
      r=-x*x
      k=1
21    q=f
      t=t*r/k
      f=f+t/(2*k+1)
      k=k+1
      if (f.ne.q) goto 21
      f=x+b*f
      return
      end
```

программа 2.3Р
метод простой итерации

```
var p:array [1…9] of real; x,e:real;
m,n,k:integer;
```

```
function f(x:real):real;
const pi=3.14159265;
var q,r,s,t:real; k:integer;
begin
t:=2*x/sqrt(pi); s:=t-p[1]; r:=-x*x; k:=1;
repeat q:=s; t:=t*r/k;
s:=s+t/(2*k+1); k:=k+1;
until s=q;
f:=x+b*s;
end;
procedure iter (var b,x,e:real; m:integer; function f:real);
var x1,r:real; i:integer;
begin
for i:=1 to m do begin
x1:x; x:=f(x);
if abs(x-x1)<e then exit;
if i=m then writeln (‘итерации все’)
end; begin
repeat write(‘b,x,e,m?’); readln(b,x,e,m);
write(‘сколько параметров?’); readln(n);
for k:=1 to n do begin
 write (‘p(‘,k,’)?’); readln (p[k]);
end;
iter (b,x,e,m,f); writeln(‘x=’,x);
until false
end.
```

§2.5. Метод Ньютона.

Метод Ньютона является одним из наиболее эффективных методов решения самых разнообразных нелинейных задач. Расчетные формулы метода можно получить, используя различные подходы. Рассмотрим некоторые из них.

Метод касательных. Расчетную формулу для решения нелинейного уравнения (2.1) можно получить, используя простые геометрические соображения. Соответствующая иллюстрация приведена на рис. 2.9

Пусть x - заданное начальное значение приближение к корню $\bar{x}$. В точке $M^{(0)}$ с координатами $(x^{(0)}, f(x^{(0)}))$ проведем касательную к графику функции $y = f(x)$ и за новое приближение $x^{(1)}$ примем абсциссу точки пересечения этой касательной с осью OX. Аналогично за приближение $x^{(2)}$

примем абсциссу точки пересечения с осью OX касательной , проведенной к графику в точке $M^{(1)}$ с координатами $(x^{(1)}, f(x^{(1)}))$.

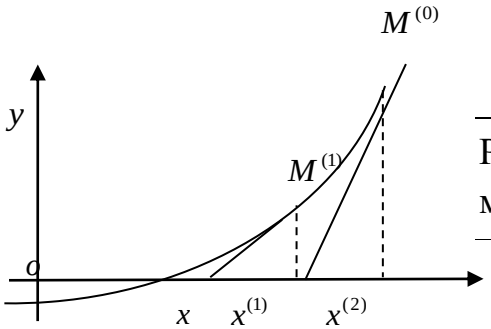


Рис. 2.9 Иллюстрация метода Ньютона

Продолжая этот процесс, получим последовательность $x^{(0)}, x^{(1)}, x^{(2)},..., x^{(n)}...$ приближений к корню $\bar{x}$.

Напомним, что уравнение касательной, проведенной к графику функции $y = f(x)$ в точке $(x^{(n)}, f(x^{(n)}))$, имеет вид

$$y = f(x^{(n)}) + f'(x^{(n)})(x - x^{(n)}). \qquad (2.23)$$

Положим в выражении (2.23) $y = 0$. Тогда при условии $f'(x^{(n)}) \neq 0$ абсцисса $x^{(n+1)}$ точки пересечения с осью OX удовлетворяет равенству

$$0 = f(x^{(n)}) + f'(x^{(n)})(x^{(n+1)} - x^{(n)}). \qquad (2.24)$$

Выражая из (2.24) $x^{(n+1)}$, получаем расчетную формулу метода Ньютона

$$x^{(n+1)} = x^{(n)} - f(x^{(n)}) / f'(x^{(n)}), n \geq 0. \qquad (2.25)$$

Метод линеаризации. В более общем случае метод Ньютона можно рассматривать кА итерационный метод, использующий специальную линеаризацию задачи и позволяющий свести решение исходного нелинейного уравнения к решению последовательности линейных уравнений.

Пусть приближение $x^{(n)}$ уже получено. Представим функцию $f(x)$ в окрестности точки $x^{(n)}$ по формуле Тейлора

$$f(x) = f(x^{(n)}) + f'(x^{(n)})(x - x^{(n)}) + f''(\xi)/2)(x - x^{(n)})^2, \qquad (2.26)$$

где ξ - некоторая точка , расположенная между x и $x^{(n)}$.

Заменяя в уравнении $f(x)=0$ функцию $f(x)$ главной линейной частью разложения (2.26), получим линейное уравнение

$$f(x^{(n)})+f'(x^{(n)})(x-x^{(n)})=0. \qquad (2.27)$$

Принимая решение уравнения (2.27) за новое приближение $x^{(n+1)}$, при-ходим к формуле (2.25). Приведем основную теорему о сходимости метода Ньютона.

Теорема 2.4. Пусть $\bar{x}-$ простой корень уравнения $f(x)=0,$ в некоторой окрестности которого функции f дважды непрерывно дифференцируема. Тогда найдется такая малая $\delta-$окрестность корня $\bar{x}$, что при произвольном выборе начального приближения $x^{(0)}$ из этой окрестности итерационная последовательность метода Ньютона не выходила за пределы окрестности и справедлива оценка

$$\Delta a_k \Delta a_r = \left(\sum_{j=0}^{m}\sum_{i=0}^{n}\gamma_{kj}^{(-1)}\varphi_j(x_l)\varepsilon_l\right)\left(\sum_{p=0}^{m}\sum_{s=1}^{n}\gamma_{rp}^{(-1)}\varphi_p(x_s)\varepsilon_s\right).$$

то

$$M\{\Delta a_k \Delta a_r\}=\sum_{j=0}^{m}\sum_{i=0}^{n}\gamma_{kj}^{(-1)}\gamma_{rp}^{(-1)}\left(\sum_{l=1}^{n}\sum_{s=1}^{n}\varphi_j(x_l)\varphi_p(x_s)M\{\varepsilon_l\varepsilon_l\}\right).$$

Учитывая, что $M\{\varepsilon_l\varepsilon_s\}=\delta_{ls}\sigma^2$, где $\delta_{ls}=0$ при $l\neq s,$ $\delta_{ls}=1$ при l=$s,$ *а*

$$\sum_{l=1}^{n}\varphi_j(x_l)\varphi_p(x_l)=\gamma_{pj},$$

а также справедливо равенство

$$\sum_{l=1}^{n}\gamma_{rp}^{(-1)}\gamma_{pj}=\delta_{rj},$$

Эквивалентное соотношение $\Gamma^{-1}\Gamma=E$, получим

$$M\{\Delta q_k \Delta q_r\}=\sum_{j=0}^{m}\sum_{i=0}^{n}\gamma_{kj}^{(-1)}\gamma_{rp}^{(-1)}\sum_{l=1}^{n}\varphi_j(x_l)\varphi_p(x_l)\sigma^2=$$

$$= \sum_{j=0}^{m}\sum_{p=0}^{m}\gamma_{kj}^{(-1)}\gamma_{rp}^{(-1)}\gamma_{pj}\sigma^2 = \sum_{j=0}^{m}\gamma_{kj}^{(-1)}\delta_{rj}\sigma^2 = \gamma_{kr}^{(-1)}\sigma^2.$$

При $k=r$ получаем формулу для дисперсии:

$$D(\Delta a_k)=M\{(\Delta a_k)^2\}=\gamma_{kk}^{(-1)}\sigma^2.$$

Для коэффицента корреляции имеем:

$$p_{kr}=\frac{M\{\Delta a_k \Delta a_r\}}{\sqrt{D(\Delta a_k)D(\Delta a_r)}}=\frac{\gamma_{kr}^{(-1)}}{\sqrt{\left(\gamma_{kk}^{(-1)}\right)\left(\gamma_{kk}^{(-1)}\right)}}.$$

Лемма 7.1. Случайная ошибка $\Delta\phi_m(x)$ имеет нулевое среднее и дисперсию $D\{\Delta\phi_m(x)\}=\sum_{k=0}^{m}\sum_{r=0}^{m}\varphi_k(x)\varphi_r(x)\gamma_{rk}^{(-1)}\sigma^2$.

Доказательство. Среднее значение ошибки равно

$$M\{\Delta\phi_m(x)\}=\sum_{k=0}^{m}M\{\Delta a_k\}\varphi_k(x)=0.$$

Запишем выражение для дисперсии:

$$D\{\Delta\phi_m(x)\}=M\left\{(\Delta\phi_m(x))^2\right\}=\left\{\left(\sum_{k=0}^{m}\Delta a_k\varphi_k(x)\right)\times\left(\sum_{r=0}^{m}\Delta a_r\varphi_r(x)\right)\right\}=\sum_{k=0}^{m}\sum_{r=0}^{m}\varphi_k(x)\varphi_r(x)M\{\Delta a_r\Delta a_k\}.$$

Учитывая, что $M\{\Delta a_r\Delta a_k\}=\gamma_{rk}^{(-1)}\sigma^2$, получим:

$$D\{\Delta\phi_m(x)\}=\sum_{k=0}^{m}\sum_{r=0}^{m}\varphi_k(x)\varphi_r(x)\gamma_{rk}^{(-1)}\sigma^2.$$

Среднеквадратическое значение ошибки $\Delta\phi_m(x)$ определим по формуле:

$$\phi=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\Delta\phi_m(x_1))^2}.$$

Теорема 7.3. Среднеквадратическое значение дисперсии ошибки $\Delta\phi_m(x)$ равно

$$M\{x^2\} = \frac{m+1}{n}\sigma^2.$$

Теорема. 2.6. Пусть [*a, b*] – отрезок локализации простого корня $\bar{x}$ уравнения (2.1). Предположим, что на этом отрезке функция *f* дважды непрерывно дифференцируема, а её производные *f'(x)* и *f''(x)* законопостоянны. Пусть $x^{(0)} \in [a,b]$ - произвольное начальное приближение и в случае $f(x^{(0)})f''(x^{(0)}) < 0$ выполнено дополнительное условие $x^{(1)} \in [a,b]$ (первое приближение не выходит за пределы отрезка локализации). Тогда, начиная с n=1, итерационная последовательность метода Ньютона $x^{(n)}$ сходится к $\bar{x}$ монотонно с той стороны отрезка [a, b], где *f'(x)f''(x)>0*.

Следствие 2.1. Пусть уравнение *f(x)* имеет корень $\bar{x}$, функция *f(x)* дважды непрерывно дифференцируема на всей числовой оси, а её производные *f'* и *f''* знакопостоянны. Тогда метод Ньютона сходится при любом начальном приближении $x^{(0)}$ (т.е. является глобально сходящимся), причем, начиная с n=1, последовательность $x^{(n)}$ сходится монотонно с той стороны от корня, где *f(x)f''(x)>0*.

В заключении отметим, что простота и высокая сходимость делают метод Ньютона весьма привлекательным. Однако практическое его применение связано с преодолением двух существенных трудностей. Одна из них заключается в необходимости вычисления производной *f'(x)*.

Часто найти аналитическое выражение для *f'* невозможно, а определить приближенное значение с высокой точностью очень трудно. Более существенно то, что метод Ньютона обладает, вообще говоря, только локальной сходимостью. Это означает, что областью его сходимости является некоторая малая δ- окрестность корня и для гарантии сходимости необходимо выбирать хорошее начальное приближение, попадающее в эту окрестность. Неудачный выбор начального приближения может дать расходящуюся последовательность и даже привести к аварийному останову (если на очередной итерации $f'(x^{(n)}) \cong 0$).для преодоления этой трудности метод Ньютона используют часто в сочетании с каким-либо медленно, но гарантированно сходящимся методом типа бисекции. Такие гибридные методы находят в последнее время широкое практическое применение.

В качестве примера программной реализации метода Ньютона рассмотрим программы (2.4B, 2.4F, 2.4P) решения алгебраического уравнения произвольной степени *n+1*.

$$p_1 x^{n+1} + p_2 x^n + \ldots + p_n = 0.$$

Левую часть уравнения (многочлен степени n+1) и производную полинома будем вычислять по схеме Горнера.

В основном блоке программы 2.4B (строки I0-90) осуществляется ввод начального приближения к корню (переменная X) и допустимой погрешности (переменная E). Затем задается количество параметров $P_к$ (переменная N). В конкретной задаче решения алгебраического уравнения переменная N будет на единицу больше старшей степени аргумента *x*. Ввод коэффициентов уравнения осуществляется циклически, при этом обязательно должны быть коэффициенты при всех степенях *x*.

При отсутствии слагаемых с каким-либо степенями *x* соответствующие коэффициенты задаются нулевыми. После задания всех исходных данных осуществляется обращение к подпрограмме метода Ньютона и вывод результата на дисплей (строки 50, 60).

В подпрограмме метода (строки I00-I90) происходит обращение к подпрограмме вычисления отношения *f(x)/f'(x)* (строка I00), затем вычисляется очередное приближение к корню (строка I00). Для предотвращения возможного «зацикливания» в подпрограмме метода в случае неудачно выбранного начального приближения к корню или неправильно заданных параметров рекомендуется добавить операторы, реализующие счетчик итераций.

В строке 200 находятся операторы инициализации переменных для функции (FI) и её производной (P). В цикле (строка 2I0) оператор вычисления производной предшествует оператору вычисления функции. Формально оператор можно получить путем дифференцирования правой части второго оператора по x по обычным правилам дифференцирования. Такой же способ применим и к оператору инициализации производной.

После вычисления функций и производной определяется их отношение (строка 220). Вывод на дисплей текущих значений аргумента и погрешности осуществляется для наблюдения за процессом поиска корня (строка 280).

В программе 2.4F количество параметров N и сами параметры передаются в программу-функцию F(x) вычисления отношения *f(x)/f'(x)* через неименованный COMMON-блок. В подпрограмме реализован метод Ньютона по тем же принципам, что и в соответствующем блоке программы 2.4B.

В программе 2.4P в процедуре NEWTON для организации итерацион-ного процесса используется цикл типа repeat-until, который будет повто-ряться до тех пор, пока логическое выражение после until не примет значение true.

```
1 rem                                                    программа 2.4B
2 rem                                                    метод Нютона
10 dim p(9)
20 print “x,e”;
25 input x,e
30 print “сколько параметров”;
35 input n
40 for k=1 to n
45 print "p";k;
47 input p(k) \ next k
50 gosub 100
60 print ″x=″;x
90 goto 20
100 gosub 200
110 x=x-f1
120 if abs(f1)>e then 100
190 return
200 f1=p(1)\ r=0
210 for k=2 to n \ r=f1+x*r \f1=p(k)+x*f1 \ next k
220 f1=f1/r
280 print x,f1
290 return
```

```
c                                                        программа 2.4F
c                                                        метод Ньютона
      external f
      common n,p(9)
1     type *,’x,e?’
      accept *,x,e
      type *,’сколько параметров’
```

```
      accept *,n
      if (n.eq.0) goto 3
      do 2 k=1,n
      type 4,k
2     accept *,p(k)
3     call newton(x,e,f)
      type *,'x=',x
      goto 1
4     format ('p(',12,')?')
      end
      subroutine newton(x,e,f)
11    f1=f(x)
      x=x-f1
      if(abs(f1).gt.e) goto 11
      return
      end
      function f(x)
      common x,p(9)
      f=P(1)
      r=0.0
      do 21 k=2,n
      R=f+x*r
      21      F=p(k)+x*f
      f=f/r
      return
      end
```

(программа 2.4P метод Ньютона)

```
Var p:array [1..9] of real;
X,e: real;
K,n:integer;
Function f(x:real):real;
Var q,r:real;
Begin q:=p[1];
R:=0.0;
For k:=2 to n do begin
R:=q+x*r;  q:=p[k]+x*q;
End;
F:=q/r;  end;
Procedure Newton(var x,e:real; function f:real);
Var f1:real;
Begin
```

```
Repeat f1:=f(x); x:=x-f1;  until abs(f1)<E;
End; begin
Repeat write('x,e ?'); readln (x,e);
Write('p(',k,')?'); readln(p[k]); end;
Newton (x,e,f); writeln ('x=',x);
Until false;
End.
```

Варианты практических заданий к главе 2

Отделить корни уравнения графически и уточнить один из них методом простых итераций и методом Ньютона с точностью до 0.001

1.	$x - \sin x = 0.25$	$x^3 - 3x^2 + 9x - 8 = 0$
2.	$tg\ (0/58x + 0.1) = x^2$	$x^3 - 6x - 8 = 0$
3.	$\sqrt{\ }\ x\ cox\ (0.387x) = 0$	$x^3 - 3x^2 + 6x + 3 = 0$
4.	$tg\ (0.4x + 0.4) = x^2$	$x^3 - 0.1x^2 + 0.4x - 1.5 = 0$
5.	$3x - \cos x - 1 = 0$	$x^3 - 3x^2 + 6x + 3 = 0$
6.	$x + lg\ x = 0.5$	$x^3 + 3x + 1 = 0$
7.	$x^2 + 4\sin x = 0$	$x^3 - 3x^2 + 12x - 9 = 0$
8.	$tg\ (0.3x + 0.4) = x^2$	$x^3 + 4x - 6 = 0$
9.	$ctg\ x - x/3 = 0$	$x^3 - 3x^2 + 12x - 12 = 0$
10.	$x^2 + 4\sin x = 0$	$x^3 - 2x + 4 = 0$

ГЛАВА 3.
ПРЯМЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ.

3.1. Постановка задачи

Данная глава посвящена одной из основных задач вычислительной алгебры – прямым методам решения систем линейных алгебраических уравнений с вещественными коэффициентами:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1m}x_m = b_1; \\ &a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... + a_{2m}x_m = b_2; \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + ... + a_{mm}x_m = b_m; \end{aligned} \qquad (3.1)$$

В матричной форме записи эта система принимает вид

$$AX = B, \qquad (3.2)$$

где

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} & . & . & . & a_{1m} \\ a_{21} & a_{22} & a_{23} & . & . & . & a_{2m} \\ a_{31} & a_{32} & a_{33} & . & . & . & a_{3m} \\ . & . & . & . & . & . & . \\ a_{m1} & a_{m2} & a_{m3} & . & . & . & a_{mm} \end{vmatrix}, \quad X = \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ \ldots \\ x_m \end{vmatrix}, \quad B = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \\ \ldots \\ b_m \end{vmatrix}$$

Задача решения системы (3.1) сравнительно редко представляет самостоятельный интерес для приложений, в то же время от умения эффективно решать такие системы часто зависит сама возможность математического моделирования самых разнообразных процессов с примене-нием компьютеров. Значительная часть численных методов решения различ-ных (а особенно нелинейных) задач включает в себя решение систем (3.1) как элементарный шаг соответствующего алгоритма. Многие из представляющих практике интерес научно-технических задач описываются математическими моделями, при реализации которых на

компьютере возникает необходимость решения систем линейных уравнений.

Пусть матрица A заданна и является невырожденной. В этом случае решение системы (*3.1*) существует, единственно и устойчиво по входным данным. Это означает, что рассматриваемая задача корректна.

§ 3.2. Метод Гаусса

Одним из самых распространенных метода решения систем линейных алгебраических уравнений является метод Гаусса (метод последовательного исключения неизвестных).

Вычислительная процедура метода Гаусса состоит из двух основных этапов: прямого хода и обратного хода (обратной подстановки). Прямой ход метода Гаусса заключается в последовательном исключении неизвестных из системы (*3.1*) для преобразования ее к эквивалентной системе с верхней треугольной матрицей. Вычисление значения неизвестной производится на этапе обратного хода.

Рассмотрим простейший вариант метода Гаусса, называемый схемой единственного деления.

Шаг 1 . Предположим, что коэффициент $a_{11} \neq 0$. Будем называть его главным элементом 1-го шага.

Найдем величины

$$\mu_{i1} = a_{i1} / a_{11} \quad , \quad i = 2,3,..., m \, , \tag{3.3}$$

называемые множителями 1-го шага. Вычтем последовательно из 2-го, 3-го,…,m-го уравнения системы (*3.1*) 1-е уравнение системы, умноженное соответственно на $\mu_{21}, \mu_{31},..., \mu_{m1}$. Это позволит обратить в нуль коэффициенты при x_1 во всех уравнениях кроме первого.

В результате получим эквивалентную систему

$$\begin{aligned}
&a_{11}x_1 + a_{12}x + a_{13}x_3 + ... + a_{1m}x_m = b_1 \; ; \\
&a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + ... + a_{2m}^{(1)}x_m = b_2^{(1)} \; ; \\
&a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + ... + a_{3m}^{(1)}x_m = b_3^{(1)} \; ; \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&a_{m2}^{(1)}x_2 + a_{m3}^{(1)}x_3 + ... + a_{mm}^{(1)}x_m = b_m^{(1)} \; ,
\end{aligned} \tag{3.4}$$

в которой коэффициенты $a_{ij}^{(1)}$ и $b_i^{(1)}$ $(i, j = 2,3,..., m)$ вычисляется по формулам

$$a_{ij}^{(1)} = a_{ij} - \mu_{i1} a_{1j}, \quad b_i^{(1)} = b_i - \mu_{i1} b_1 \ . \tag{3.5}$$

Шаг 2. Пусть $a_{22}^{(1)} \neq 0$, где $a_{22}^{(1)}$- коэффициент, называемый главным элементом второго шага. Вычислим множитель 2-го шага

$$\mu_{i2} = a_{i2}^{(1)} / a_{22}^{(1)}, \quad i = 3,4,...,m$$

и вычтем последовательно из 3-го, 4-го,…, m-го уравнения системы (*3.4*) 2-е уравнение, умноженное соответственно на $\mu_{32},..., \mu_{m2}$.

В результате будет получена система:

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1m}x_m &= b_1 \ ; \\
a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + ... + a_{2m}^{(1)}x_m &= b_2^{(1)} \ ; \\
a_{33}^{(2)}x_3 + ... + a_{3m}^{(2)}x_m &= b_3^{(2)} \ ; \\
&\cdots\cdots\cdots\cdots \\
a_{m3}^{(2)}x_3 + ... + a_{mm}^{(2)}x_m &= b_m^{(2)} \ ,
\end{aligned} \tag{3.6}$$

где коэффициенты $a_{ij}^{(2)}$ и $b_i^{(2)}$ $(i = 3,4,...,m)$ вычисляются по формулам

$$a_{ij}^{(2)} = a_{ij}^{(1)} - \mu_{i2} a_{2j}^{(1)}, \quad b_i^{(2)} = b_i^{(2)} = b_i^{(1)} - \mu_{i2} b_2^{(1)}.$$

Аналогичным образом проводятся остальные шаги. Опишем очередной k-й шаг.

Шаг k. В предположении, что главный элемент k-го шага отличен от нуля, вычислим множитель k-го шага

$$\mu_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \qquad i = k+1,...,m$$

и вычтем последовательно из $(k+1)$- го, $(k+2)$- го,…, m- го уравнения полученной на предыдущем шаге системы k- е уравнение, умноженное

соответственно на $\mu_{k+1.k}$, $\mu_{k+2.k},\ldots,\mu_{mk}$. После $(m-1)$- го шага исключения будет получена система уравнений

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1m}x_m = b_1; \\ &a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2m}^{(1)}x_m = b_2^{(1)}; \\ &a_{33}^{(2)}x_3 + \ldots + a_{3m}^{(2)}x_m = b_3^{(2)}; \\ &\ldots\ \ldots\ \ldots\ \ldots \\ &a_{mm}^{(m-1)}x_m = b_m^{(m-1)}. \end{aligned} \qquad (3.7)$$

На этом вычисления прямого хода заканчиваются.

Обратный ход. Подставляя найденное значение x_m в предпоследнее уравнение, получим x_{m-1}. Осуществляя обратную подстановку, далее последовательно находим x_{m-2}, $x_{m-3},\ldots,x_1$. Вычисления неизвестных здесь проводятся по формулам

$$\begin{aligned} &x_m = b_m^{(m-1)} / a_{mm}^{(m-1)}; \\ &x_k = (b_k^{(k-1)} - a_{k.k+1}^{(k+1)}x_{k+1} - \ldots - a_{km}^{(k-1)}x_m) / a_{kk}^{(k-1)}; \\ &k = m-1,\ldots,\ 1. \end{aligned} \qquad (3.8)$$

Пример 3.1. Методом Гаусса решить систему

$$\begin{aligned} &10x_1 + 6x_2 + 2x_3 = 25; \\ &5x_1 + x_2 - 2x_3 + 4x_4 = 14; \\ &3x_1 + 5x_2 + x_3 - x_4 = 10; \\ &6x_2 - 2x_3 + 2x_4 = 8. \end{aligned} \qquad (3.9)$$

Решение. 1-й шаг прямого хода. Вычислим множители $\mu_{21} = a_{21}/a_{11} = 5/10 = 0.5$, $\mu_{31} = a_{31}/a_{11} = 3/10 = 0.3$, $\mu_{41} = a_{41}/a_{11} = 0/10 = 0$. Вычитая из 2, 3 и 4- го уравнений системы (3.9) первое уравнения, умноженное на $\mu_{21}, \mu_{31}, \mu_{41}$ соответственно, получим

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 &= 25;\\ -2x_1 - 3x_3 + 4x_4 &= 1.5;\\ 3.2x_2 + 0.4x_3 - x_4 &= 2.5;\\ 6x_2 - 2x_3 + 2x_4 &= 8. \end{aligned}$$

2-й шаг прямого хода. Вычислим $\mu_{32} = a_{32}^{(1)} / a_{22}^{(1)} = 3.2/(-2) = -1.6$, $\mu_{42} = a_{42}^{(1)} / a_{22}^{(1)} = 6/(-2) = -3$. Вычитая из 3- го и 4- го уравнений системы (3.10) второе уравнение, умноженное на μ_{32} и μ_{42} соответственно, приходим к системе

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 &= 25;\\ -2x_1 - 3x_3 + 4x_4 &= 1.5;\\ -4.4x_3 + 5.4x_4 &= 4.9;\\ -11x_3 + 14x_4 &= 12.5. \end{aligned} \qquad (3.11)$$

3-й шаг прямого хода. Вычисляя множитель $\mu_{43} = (-11)/(4.4) = 2.5$ и вычитая из 4-го уравнения системы (3.11) 3-е уравнение, умноженное на μ_{43}, приводим систему к треугольному виду

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 &= 25;\\ -2x_1 - 3x_3 + 4x_4 &= 1.5;\\ -4.4x_3 + 5.4x_4 &= 4.9;\\ 0.5x_4 &= 0.25. \end{aligned} \qquad (3.12)$$

Обратный ход. Из последнего уравнения системы (3.12) находим $x_4 = 0{,}5$. Подставляя значение x_4 в 3-е уравнение, находим $x_3 = (4{,}9 - 5{,}4x_4)/(-4{,}4) = -0{,}5$. Продолжая далее обратную подстановку, получаем

$$x_2 = (1{,}5 + 3x_3 - 4x_4)/(-2) = 1;$$
$$x_1 = (25 - 6x_2 - 2x_3)/10 = 2.$$

Ответ: $x_1 = 2,\ x_2 = 1,\ x_3 = -0{,}5,\ x_4 = 0{,}5.$

Результаты вычислений можно свести в следующую таблицу (табл.3.1).

Заметим, что вычисление множителей, а также обратная подстановка требуют деления на главные элементы $a_{kk}^{(k-1)}$. Поэтому, если один из главных элементов оказывается равным нулю, схема единственного деления не может быть реализована.

Рассмотренный выше вариант метода Гаусса (схема единственного деления) может оказаться некорректным и, следовательно, непригодным для вычислений на ЭВМ и в этом случае, когда все главные элементы отличны от нуля, но среди них есть близкие к нулю.

Метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора). В этом варианте метода Гаусса допускается нарушение естественного порядка исключения неизвестных.

На 1-м шаге метода среди элементов a_{ij} определяется максимальный по модулю элемент

$a_{i_1 \cdot j_1}$. Первое уравнение системы и уравнения с номером i_1 меняются местами. Далее стандартным образом производится исключение неизвестного x_{j_1} из всех уравнений кроме первого.

На k-м шаге метода среди коэффициентов при неизвестных в уравнениях системы с номерами $i = k,\ldots,m$ выбирается максимальный по модулю коэффициент $a_{i_1 \cdot j_k}^{(k-1)}$. Затем *k-е* уравнение и уравнение, содержащее
найденный коэффициент, меняются местами и неизвестные x_{j_k} исключаются из уравнения с номерами $i = k+1,\ldots,m$.

На этапе обратного хода неизвестные вычисляются в следующем порядке: $x_{j_m}, x_{j_{m-1}},\ldots,x_{j_1}$.

Для некоторых классов матриц при использовании схемы единственного деления главные элементы располагаются на главной диагонали и поэтому в применении стратегии частичного выбора нет необходимости. Это относится, например, к системам с положительно определенными матрицами, а так же к матрицам, обладающим свойством диагонального преобладания:

$$\sum_{\substack{j=1 \\ j \neq i}}^{m} |a_{ij}| < |a_{ii}|, i = 1,2,\ldots,m. \qquad (3.13)$$

Метод Гаусса с выбором главного элемента (схема частичного выбора). На k *–м* шаге прямого хода преобразуются коэффициенты в уравнениях системы с номерами $i = k+1,\ldots,m$ по формулам

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \mu_{ik} a_{kj}^{(k-1)}, b_i^{(k)} = b_i^{(k-1)} - \mu_{ik} b_i^{(k-1)}, \quad (3.14)$$

$i = k+1,\ldots,m$.

Таблица 3.1

	a_{i1}	a_{i2}	a_{i3}	a_{i4}	b_i	μ_{i1}
Исходная система	10 5 3 0	6 1 5 6	2 -2 1 -2	0 4 -1 2	25 14 10 8	
1-ый шаг прямого хода	10 0 0 0	6 -2 3,2 6	2 -3 0,4 -2	0 4 -1 2	25 1,5 2,5 8	 0,5 0,3 0
2-ой шаг прямого хода	10 0 0 0	6 -2 0 0	2 -3 -4,4 -11	0 4 5,4 14	25 1,5 4,9 7,5	 -1,6 -3
3-ий шаг прямого хода и обратный ход	10 0 0 0	6 -2 0 0	2 -3 -4,4 0	0 4 5,4 0,5	25 1,5 4,9 0,25	2 1 -0,5 0,5

Следует отметить, что появление больших множителей $м_{ik}$ может привести к сильному росту коэффициентов системы и связанных с этим ошибок, В методе Гаусса с выбором главного элемента по столбцу обеспечивается выполнение неравенства I$м_{ik}$I<=1 для всех $k=1,2,\ldots,m-1$ и –

$i=k+1,...,m$. Отличие этого варианта метода Гауса от схемы единственного деления состоит в том, что на k-м шаге исключения в качестве ведущего элемента выбирается максимальный m модулю коэффициент a_{jk}при x_k в уравнениях с номерами $i=k,k+1,...,m$. Затем соответствующее выбранному коэффициенту кранение с номером i_k меняется местами с k-м уравнением системы, с тем что бы ведущий элемент занял место коэффициента $a_{kk}^{(k-1)}$. После этой перестановки исключение неизвестного x проводится, как в схеме единственного деления. Матрицы, удовлетворяющие условию (3.13), таковы, что в каждой из строк модуль элемента a_{ii}, расположенного на главной диагонали, больше суммы модулей элементов всех остальных элементов строки.

Масштабирование.. Существуют два способа масштабирования системы AX=B. Первый состоит в умножении каждого из уравнений на некоторый множитель $м_1$, такой, чтобы коэффициенты системы были величинами порядка единицы. Второй способ заключается в умножении на масштабирующий множитель α_j каждого j-го столбца матрицы, что соответствует замене переменных $x_j^{'} = \alpha_j^{-1} x_j$. На практике обычно масштаби-рование производят делением каждого уравнения на его наибольший по модулю коэффициент. Это вполне удовлетворительный способ для большинства реально встречающихся задач.

Программа решения системы линейных уравнений методом Гаусса состоит из трех блоков. В первом блоке (главная программа) задается порядок системы и выполняется обращение к блоку 2, в котором на основе матрицы A и вектора B определяются элементы расширенной матрицы. Затем вызывается подпрограмма, реализующая метод Гаусса (блок 3) и выводятся результата решения системы либо сообщение том, что решения не существует в случав вырожденной матриц (det=0).

Для общности блок 2 оформлен в виде отдельной подпрограммы. Здесь элементы расширенной матрицы задаются по строкам в диалоговом режиме с клавиатуры компьютера. Хотя для других конкретных задач матрицы могут определяться путем зачисления по заданному алгоритму.

В программе 3.1B прямой ход занимает строки 200-320, обратный ход 330-390, выбор главных элементов - 220-270.

Особенностью программа 3.1P на языке Паскаль является введение новых типов переменных MAT и VEC дли матрицы A и вектора

результатов X. Такое введение необходимо, потому что переменные A и X являются формальными и фактическими параметрами процедур.

Программа 3.1B метод Гауса

```
1       rem
5       rem
1       Dim a(20,21),x(20)

2       Print “n”;\ input n

3       Gosub 100

4       Gosub 200

5       If s=0 then print “del=0”\ goto 20

6       For i=1 to n\ print “x”;i;”=”;x(i)\ next i

9       Goto 20

1       For i=1 to n

1       For j=1 to n+1\ print “a”;i;j;\ input a(i,j)\ next j

1       Next i

1       Return

2       n1=n+1

2       For k=1 to n\k1=k+1\ s=a(k,k)\ j=k

2       For i=k1 to n\ r=a(i,k)

2       If abs(r)>abs(s) then s+r\ j=i

2       Next i

2       If s=0 then return

2       If j=k then 280
```

```
   2          For i=k to n1\ r=a(k,i)\ a(k,i)=a(j,i)\ a(j,i)=r\ next i
   2          For j=k1 to n1\ a(k,j)=a(k,j)/s\ next j
   2          For i=k1 to n\ r=a(i,k)
   3          For j=k1 to n1\ a(i,j)=a(i,j)-a(k,j)*r\ next j
   3          Next i
 320  next k
 330  for i=n to 1 step -1\s=a(i,n1)
 340  for j=i+1 to n\s=s-a(i,j)*x(j)\next j
 350  x(i)=s\next 1
 390  return
 c                                                          программа 3.1Р
 c                                                            метод Гаусса
      real a(20,21),x(20)
 1    type *,'n?'
      accept *,n
      call matr(n,a)
```



```
      call gauss(n,a,x,s)
      if (s.ne.O) goto 2
      type *,'det=0'
      goto 1
 2    do 3 i=1 ,n
 3    type 4,i,x(i)
      goto 1
 4    format (1x, 'x',12,'=',E13.6)
     end
     subroutine   matr(n,a)
     read   a(20,21)
     do   11    i=1,n
     do   11    j=1,n+1
     type    12,i,j
 11  accept *,a(i,j)
 12  format ('a',212,'?')
     return
     subroutine   gauss(n,a,x,s)
     read   a(20,21),x(20)
```

```
      n1=n+1
      do   25    k=1,n
      к1=к+1
      s=a(k,k)
      j=k
      do 21 i=k1,n
      r=a(i,k)
      if (abs(r).le.abs(s)) goto 21
      s=r
21    continue
      if (s.eq.0) return
      if (j.eq.k) goto 23
      do 22 I=k,n1
      r=a(k,i)
      a(k,i)=a(j,i)
22    a(j,i)=r
23    do 24 j=k1,n1
24    a(k,j)=a(k,j)/s
      do 25 i=k1,n
      r=a(i,k)
      do 25 j=k1,n1
25    a(i,j)=a(i,j)-a(k,j)*r
      x(n)=a(n,n1)
```



```
      do 27 i=n-1,1,-1
      s=a(i,n1)
      do 26 j=i+1,n
26    s=s-a(i,j)*x(j)
27    x(i)=s
      return
      end
```

```
программа   3.IP
{                                          метод Гаусса            }
tupe mat=array[1..20,1..21]  of real;
       vec=array[1..20]  of real;
var   a: mat;
       x: vec;
i,n:integer;
s:real;
procedure matr(n:integer;var a:mat);
var  i,j:integer;
```

```
begin for i:=1 to n do
for j:=1 to n+1 do begin
write('a',i,j,'?'); readln(a[i,j])
end
end;
procedure gauss(n:integer;var a:mat;var x:vec;var s:real);
var i,j,k,l,k1,n1:integer;
r:real;
begin n1:=n+1;
for k:=1 to n do begin k1:=k+1; s:=a[k,k]; j:=k;
for i:=k1 to n do begin r:=a[i,k];
if abs(r)>abs(s) then begin s:=r; j:=i end
end;
if s=0.0 then exit;
if j<>k then for i:=k to n1 do begin
r:=a[k,i]; a[k,i]:=a[j,i]; a[j,i]:=r end;
for j:=k1 to n1 do a[k,j]:=a[k,j]/s;
for i:=k1 to n do begin r:=a[i,k];
for j:=k1 to n1 do a[i,j]:=a[i,j]-a[k,j]*r
end
end;
if s<>0.0 then for i:=n downto 1 do
begin s:=a[i,n1];
```



```
for j:=i+1 to n do s:=s-a[i,j]*x[j];
x[i]:=s
end
end;
begin
repeat write('n?'); readln(n); matr(n,a);
gauss(n,a,x,s);
if s<>0.0 then for i:=1 to n do
writeln('x',i,'=',x[i])
else writeln('det=0')
until false
end.
```

Варианты практических заданий к главе 3

Используя схему Гауссу, решить систему уравнений

1. $3.21x_1 - 4.25x_2 + 2.23x_3 = 5.06$
   $7.09x_1 + 1.17x_2 - 2.23x_3 = 4.75$

$0.43x_1 - 1.4x_2 - 0.62x_3 = -1.05$

2. $0.42x_1 - 1.13x_2 + 7.05x_3 = 6.15$
$1.14x_1 - 2.15x_2 + 5.11x_3 = -4.16$
$-0.71x_1 + 0.81x_2 - 0.02x_3 = -0.17$

3. $2.5x_1 - 3.12x_2 - 4.08x_3 = -7.5$
$0.61x_1 + 0.71x_2 - 0.05x_3 = 0.44$
$-1.03x_1 - 2.05x_2 + 0.87x_3 = -1.16$

4. $7.09x_1 + 1.17x_2 - 2.23x_3 = -4.75$
$0.43x_1 - 1.4x_2 - 0.62x_3 = -1.05$
$3.21x_1 - 4.25x_2 + 2.13x_3 = 5.06$

5. $1.14x_1 - 2.15x_2 - 5.11x_3 = -4.16$
$-0.71x_1 + 0.81x_2 - 0.02x_3 = -0.17$
$0.42x_1 - 1.13x_2 + 7.05x_3 = 6.15$

6. $0.61x_1 + 0.71x_2 - 0.05x_3 = 0.44$
$-1.03x_1 - 2.05x_2 + 0.87x_3 = -1.18$
$2.5x_1 - 3.13x_2 - 5.03x_3 = -7.5$

7. $3.11x_1 - 1.66x_2 - 0.60x_3 = -0.92$
$-1.65x_1 + 3.51x_2 - 0.78x_3 = 2.57$
$0.60x_1 + 0.78x_2 - 1.87x_3 = 1.65$

8. $0.10x_1 + 1.2x_2 - 0.13x_3 = 0.10$
$0.12x_1 + 0.71x_2 + 0.15x_3 = 0.26$
$-0.13x_1 + 0.15x_2 + 0.63x_3 = 0.38$

9. $0.71x_1 + 0.10x_2 + 0.12x_3 = 0.29$
$0.10x_1 + 0.34x_2 - 0.04x_3 = 0.32$
$0.12x_1 - 0.04x_2 + 0.10x_3 = -0.10$

10. $0.34x_1 - 0.04x_2 + 0.10x_3 = 0.33$
$-0.04x_1 + 0.10x_2 + 0.12x_3 = -0.05$
$0.10x_1 + 0.12x_2 + 0.71x_3 = 0.28$

ГЛАВА 4.
ИТЕРАЦИОННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Системы линейных алгебраических уравнений можно решать как с помощью прямых, так и итерационных методов. Для систем уравнения средней размерности чаще используются прямые методы. Итерационные методы применяются главным образом для решения задач большой погрешности, для которых использование прямых методов невозможно из-за ограничений в доступной оперативной памяти компьютера или необходи-мости выполнения чрезмерно большого числа арифметических операций.

Применение итерационных методов для качественного решения задачи требует серьезного использования ее структуры, специальных знаний и определенного опыта. Именно по этой причине разработано большое число различных итерационных методов, каждый из которых ориентирован на решение сравнительно узкого класса задач, и существует

довольно мало стандартных программ, реализующих итерационные методы.

В этой главе будут рассмотрены наиболее простые и известные итерационные методы, позволяющие решать достаточно широкий класс систем.

§ 4.1 Основные определения

Решением системы линейных алгебраических уравнений является вектор $x = (x_1, x_2, ..., x_m)^T$, который в дальнейшем будем рассматривать как элемент векторного пространства R^m. Приближенное решение $x^* = (x_1^*, x_2^*, ..., x_m^*)^T$ и погрешность $x - x^* = (x_1 - x_1^*, ..., x_m - x_m^*)$ также являются элементами пространства R^m. Для того чтобы анализировать итерационные методы решения систем, необходимо уметь количественно оценивать «величину» векторов x^* и $x - x^*$. С этой целью введем понятие нормы векторов.25_01

Определение 4.1. Будем говорить, что в R^m задана норма, если каждому вектору x из R^m поставлено в соответствие вещественное число $\|x\|$, называемое нормой вектора x и обладающее следующими тремя свойствами:

1) $\|x\| > 0$. причём $\|x\| = 0$, тогда и только тогда, когда x=0;
2) $\|a_x\| = |a| \, \|x\|$ для любого вектора x и любого числа *a;*
3) $\|x+y\| \le \|x\| + \|y\|$ для любых векторов x и y.

Существуют множество различных способов введения норм. В числен-ных методах наиболее часто применяются следующие нормы:

$$\|x\|_1 = \sum_{1=1}^{m} |x_1| \cdot \|x\|_2 = \left(\sum_{1=1}^{m} x_1^{\,2} \right)^{1/2} \cdot \|x\|_\infty = \max |x_1|. \qquad (4.1)$$

Первые две из них являются частными случаями более общей нормы

$$\|x\|_p = \left(\sum_{1=1}^{m} |x_1|^p \right)^{1/p} . p \ge 1 \qquad (4.2)$$

(при p=1 и p=2), а последняя, как можно показать, получается из нормы (4.2) предельным переходом при $p \to \infty$.

Абсолютная и относительная погрешности. Будем всюду далее считать, что в пространстве m-мерных векторов R^m введена и фиксирована некоторая норма (напрмиер, одна из норм $\left|x\right|_p$, $I \le P \le \infty$). В этом случае в качестве меры степени близости векторов ч и x* естественно использовать величину $\left|x-x^*\right|$, являющуюся аналогом расстояния между точками x и x*. Введём абсолютную и относительную погрешности вектора

$$\Delta(x^*) = \|x - x^*\| \cdot \delta(x^*) = \frac{\|x - x^*\|}{\|x\|} \cdot \qquad (4.3)$$

Выбор конкретной нормы зависит от требований, которые предъявляются к точности x*. Норма $\left|x\right|_1$ используется в тех случаях, когда малой должна быть суммарная абсолютная ошибка в компоненте решения; выбор $\left|x\right|_2$ соответствует критерию малости среднеквадратичной ошибки , а применение в качестве нормы $\left|x\right|_\infty$ означает , что малой должна быть максимальная из относительных ошибок в компонентах решения.

Определение 4.2. Пусть $\{x^{(n)}\}_{n=1}^{\infty}$ - последовательность векторов $x^n = (x_1^{(n)}, x_2^{(n)}, ..., x_m^{(n)})$. Говорят, что последовательность векторов $x^{(n)}$

сходится к вектору x при $n \to \infty$ ($x^{(n)} \to x$ при $n \to \infty$), если $\Delta\left(x^{(n)}\right) = \left\|x^{(n)} - x\right\| \to 0$ при $n \to \infty$.

Определение 4.3. Величина

$$\|A\| = \max_{x \neq 0} \frac{\|A_x\|}{\|x\|} \qquad (4.4)$$

называется нормой матрицы A, подчинённой норме векторов, введённой в R^m. Как следует из определения (4.4), каждой из векторных норм $\left|x\right|$ соответствует своя подчинённая норма матрицы $\left|A\right|$. Известно, в частности, что нормам $\left|x\right|_1$, $\left|x\right|_2$ и $\left|x\right|_\infty$ подчинены нормы $\left|A\right|_1$, $\left|A\right|_2$ и $\left|A\right|_\infty$, вычисляемые по формулам

$$\|A\|_1 = \max_{1 \le j \le m} \cdot \sum_{i=1}^{m} |a_{ij}| \qquad (4.5)$$

$$\|A\|_2 = \max\sqrt{\lambda_j(A^{\chi}A)} \tag{4.6}$$

$$\|A\|_{\infty} = \max_{1\le i\le m}\cdot\sum_{j=1}^{m}|a_{ij}| \tag{4.7}$$

где $\lambda_j(A^TA)$ - собственные числа матрицы A^TA.

Нормы $|A|_1$ и $|A|_{\infty}$ вычисляются просто. Для получения значения первой из них нужно найти сумму модулей элементов каждого из столбцов матрицы A, а затем выбрать максимальную из этих сумм. Для получения значения $\|A\|_{\infty}$ нужно аналогичным образом поступить со строками матрицы А. Вычислить значение нормы $\|A\|_2$, как правило, бывает трудно, так как для этого следует определять собственные числа λ_j . Для оценки величины $\|A\|_2$ можно, например, использовать неравенство $\|A\|_2 \le \|A\|_E$.

$$\|A\|_E = \sqrt{\sum_{i,j=1}^{m} a_{ij}^2}$$

где - величина, называемая евклидовой нормой матрицы А.

§ 4.2. Метод простой итерации

Пусть дана система линейных алгебраических уравнений

$$Ax=b, \tag{4.9}$$

с квадратной невырожденной матрицей А.

Для того,чтобы применить метод простой итерации к решению системы (4.9), эту систему необходимо предварительно преобразовать к виду

$$x=bx+c, \tag{4.10}$$

где b - квадратная матрица с элементами b_{ij} (i,;j=i ,2,...,m); c - вектор-столбец с элементами c_i (i=i,2,...,m).

Систему (4.10) можно записать в развернутой форме

$$
\begin{aligned}
x_1 &= b_{11}x_1 + b_{12}x_2 + b_{13}x_3 + \ldots + b_{1m}x_m + c \\
x_2 &= b_{21}x_1 + b_{22}x_2 + b_{23}x_3 + \ldots + b_{2m}x_m + c_2 \\
\ldots &\qquad\qquad \ldots \qquad\qquad \ldots \\
x_m &= b_{m1}x_1 + b_{m2}x_2 + b_{m3}x_3 + \ldots + b_{mm}x_m + c_m
\end{aligned}
\tag{4.11}
$$

Для приведения системы к виду (4.11) воспользуемся следующим способом. Из первого уравнения системы (4.10) выразим неизвестное x_1:

$$x_1 = a_{11}^{-1}(b_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1m}x_m ,$$

из второго уравнения - неизвестное x_2:\

$$x_2 = a_{22}^{-1}(b_2 - a_{21}x_1 - a_{23}x_3 - \ldots - a_{2m}x_m .$$

Из т-го уравнения выразим x_m:

$$x_m = a_{mm}^{-1}(b_m - a_{m1}x_1 - a_{m3}x_3 - \ldots - a_{mm-1}x_{m-1} .$$

В результате получим систему уравнений, в которой матрица В имеет на главной диагонали нулевые элементы. Остальные коэффициенты выражаются по формулам

$$b_{ij} = -(a_{ij} / a_{ii}), c_i = b_i / a_{ii}, i, j = 1,2,\ldots,m, j \neq i \tag{4.12}$$

Конечно, для возможности выполнения указанного преобразования необходимо, чтобы диагональные элементы матрицы А были ненулевыми.

Поиск решения системы уравнений начинается с выбора начального приближения $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_m^{(0)})$. За начальное приближение примем вектор-столбец свободных членов системы (4.11), т.е. . $x^{(0)} = c$. Подставляя $x^{(0)}$ в правую часть системы (4.11), получим первое приближение

$$x^{(0)}\ x^{(1)} = bx^{(0)} + c . \tag{4.13}$$

На основе вектора $x^{(1)}$ получим второе приближение

$$x^{(2)} = bx^{(1)} + c. \tag{4.14}$$

Продолжая этот процесс, получим последовательность

$$x^{(0)}, x^{(1)}, x^{(2)}, ..., x^{(n)} ...,$$

приближений, вычисляемых по формуле

$$x^{(k+1)} = (bx^{(k)} + c.) \qquad k=0,1,2,... \tag{4.15}$$

Напишем формулы приближений в развернутом виде:

$$\begin{gathered} x_1^{(0)} = c_1, \\ x_1^{(k+1)} = \sum_{j=1}^{m} b_{ij} x_j^{(k)} + c_1, \\ (b_{ii} = 0; i = 1,2,\dots,m; k = 0,1,2,\dots). \end{gathered} \tag{4.16}$$

Когда для итерации используется система (4.11) с коэффициентами, вычисленными по формулам (4.12), метод простых итераций принято называть методом Якоби.

Сходимость метода простой итерации. Теорема 4.1. Пусть выполнено условие

$$\|b\|<1. \tag{4.17}$$

Тогда: 1) решение x системы (4.10) существует и единственно; 2) при произвольном начальном приближении $x^{(0)}$ метод простой итерации сходит-ся и справедлива оценка погрешности

$$\left\|x^{(n)} - x\right\| \leq \|b\|^n \left\|x^{(0)} - x\right\|. \tag{4.18}$$

Теорема 4.1 дает простое достаточное условие (4.17) сходимости метода простой итерации. В упрощенном виде это условии можно

интерпретировать как условие достаточной малости коэффициентов матрицы B в системе (4.10).
Заметим, что если $\|b\| = \|b\|_\infty$, то условие (4.17) принимает вид

$$\|b\|_\infty = max_{1\le j\le m} \sum_{j=1}^{m} |b_{ij}| < 1.$$

В силу равенства $b_{ii} = 0. b_{ij} = {-a_{ij}}/{a_{ii}}$ для метода Якоби это условие эквивалентно условию диагонального преобладания (3.14). Если $\|\mathrm{B}\| = \|\mathrm{B}\|_1$, то для метода Якоби можно получить другое условие диагонального преобладания

$$\sum_{j=1;\, i\ne j}^{m} |a_{ij}| < |a_{jj}|. j = 1,2,\dots,m. \qquad (4.19)$$

Таким образом, при выполнении условия (4.17) метод простой итерации сходится со скоростью геометрической прогрессии со знаменателем $q = \|B\|$. При уменьшении величины $\|\mathrm{B}\|$ скорость сходимости возрастает. Хотя метод сходится при любом начальном приближении $x^{(0)}$, но из оценки (4.18) следует, что изначальное приближение желательно выбирать близким к решению.

Замечание 4.1. Оценка погрешности (4.18) является априорной. Ее использование в качестве критерия окончания итерационного процесса может привести к завышению необходимого числа итераций, так как $\|x^{(0)} - x\|$ неизвестно.

Апостериорная оценка погрешности. *Теорема 4.2*. Пусть выполнено условие $\|\mathrm{B}\| < 1$. Тогда справедлива апостериорная оценка погрешности

$$\left\|x^{(n)} - x\right\| \leq \frac{\|B\|}{1-\|B\|}\left\|x^{(n)} - x^{(n-1)}\right\|. \qquad (4.20)$$

Оценку погрешности (4.20) можно использовать для окончания итераций, так как величина, стоящая в правой части (4.20), легко вычисляется после нахождения очередного приближения $x^{(n)}$. Если требуется найти решение системы с точностью ε, то итерации следует вести до выполнения неравенства

$$\frac{\|B\|}{1-\|B\|}\left\|x^{(n)} - x^{(n-1)}\right\| < \varepsilon. \qquad (4.21)$$

При решении практических задач иногда используется упрощенный критерий окончания итерационного процесса

$$\left\|x^{(n)} - x^{(n-1)}\right\| < \varepsilon. \qquad (4.22)$$

Однако следует иметь ввиду, что применение критерия (4.22) может привести к существенно преждевременному окончанию итераций. Для метода простой итерации применение этого критерия обосновано только тогда, когда $\|B\| \leq \frac{1}{2}$.

Пример 4.1. Решить систему

$$\begin{aligned} 6{,}25\, x_1 - x_2 + 0{,}5\, x_3 &= 7{,}5; \\ -x_1 - 5\, x_2 + 2{,}12\, x_3 &= -8{,}68; \\ 0{,}5\, x_1 - 2{,}12 x_2 + 3{,}6\, x_3 &= -0{,}24 \end{aligned} \qquad (4.23)$$

методом простой итерации в форме Якоби с точностью $\varepsilon = 10^{-3}$ в норме $\|.\|_\infty$.

Решение. Приведем систему уравнений к виду (4.11), вычисляя коэффициенты по формулам (4.12)

$$\begin{aligned} x_1 &= 0{,}16\, x_2 - 0{,}08\, x_3 + 1{,}2; \\ x_2 &= 0{,}21 x_1 - 0{,}424\, x_3 + 1{,}736; \end{aligned} \qquad (4.24)$$

$x_3 = -0{,}1389\, x_1 - 0{,}5889 x_2 - 0{,}0667$

В трете уравнении (4.24) коэффициенты даны с точностью до погрешности округления. Здесь

$$B = \begin{bmatrix} 0 & 0{,}16 & -0{,}08 \\ 0{,}2 & 0 & -0{,}424 \\ -0{,}1389 & -0{,}5889 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 1{,}2 \\ -1{,}736 \\ -0{,}0667 \end{bmatrix}$$

Проверим выполнение достаточного условия сходимости метода простой итерации. Для этого вычислим $\|B\|_\infty = \max\{0.24,\ 0.624,\ 0.7278\} = 0.7278.$ Достаточное условие сходимости выполнено, так как $\|B\|_\infty < 1$.

Примем за начальное приближение к решению вектор $x^{(0)} = (0, 0, 0)^T$. Итерационный процесс будем продолжать до выполнения критерия

$$\left\|x^{(n)} - x^{(n-1)}\right\| < (1 - \|B\|_\infty)\varepsilon / \|B\|_\infty = \varepsilon_1 \qquad (4.25)$$

В данном случае $\varepsilon_1 \cong 0{,}37 * 10^{-3}$. Значения приближений с четырьмя цифрами после десятичной точки приведены в таблице 4.1.

Из таблицы видно, что при n=15 условие (4.25) выполняется и можно положить $x_1 = 0{,}800 \pm 0{,}001$; $x_2 = -2.00 \pm 0.001$; $x_3 = 1.000 \pm 0.001$.

Таблица 4.1

n	0	1	2	3	4
$x_1^{(n)}$	0.0000	1.2000	0.9276	0.9020	0.8449
$x_2^{(n)}$	0.0000	-1.7360	-1.4677	-1.8850	-1.8392
$x_3^{(n)}$	0.0000	-0.0667	0.7890	0.6688	0.9181
$\left\|x^{(n)} - x^{(n-1)}\right\|_\infty$		1.7360	0.8577	0.4173	0.2493
n	…	12	13	14	15
$x_1^{(n)}$	…	0.8006	0.8003	0.8002	0.8001
$x_2^{(n)}$	…	-1.9985	-1.9993	-1.9995	-1.9998
$x_3^{(n)}$	…	0.9987	0.9990	0.9995	0.9997

$\left\|x^{(n)} - x^{(n-1)}\right\|_\infty$	…	0.0018	0.0008	0.0005	0.0003

§ 4.3. Метод Зейделя.

Итерационные формулы метода. Основное отличие метода Зейделя от метода простой итерации в форме Якоби состоит в том, что при вычислении (k-1)-го приближения к неизвестному x_1, x_2, … , x_{i-1}. Пусть система (4.9) приведена к виду (4.11) с коэффициентами, которые вычислены по формулам (4.12). На (k+1)-й итерации компоненты приближения $x^{(k+1)}$ вычисляются по формулам

$$
\begin{aligned}
x_1^{(k+1)} &= \sum_{j=1}^{m} b_{1j} x_j^{(k)} + c_1; \\
x_2^{(k+1)} &= b_{21} x_1^{(k+1)} + \sum_{j=2}^{m} b_{2j} x_j^{(k)} + c_2;; \\
&\dots \quad \dots \qquad \dots \\
x_1^{(k+1)} &= \sum_{j=1}^{i-1} b_{ij} x_j^{(k+1)} + \sum_{j=i}^{m} b_{ij} x_j^{(k)} + c_i; \\
\dots &\qquad \dots \qquad \dots \qquad \dots \\
x_m^{(k+1)} &= \sum_{j=1}^{m-1} b_{nj} x_j^{(k+1)} + b_{mm} x_m^{(k)} + c_m .
\end{aligned}
\tag{4.26}
$$

Введём в рассмотрение нижнюю и верхнюю треугольные матрицы:

$$
B_1 = \begin{bmatrix} 0 & 0 & 0\ldots\ldots\ldots & 0 \\ b_{21} & 0 & 0\ldots\ldots\ldots & 0 \\ b_{31} & b_{32} & 0\ldots\ldots\ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ b_{m1} & b_{m2} & b_{m3}\ldots\ldots & 0 \end{bmatrix}, B_2 = \begin{bmatrix} 0 & b_{12} & b_{13}\ldots. & b_{1m} \\ 0 & 0 & b_{23}\ldots. & b_{2m} \\ 0 & 0 & 0\ldots\ldots & b_{3m} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0\ldots. & 0 \end{bmatrix} \tag{4.27}
$$

Тогда расчётные формулы метода Зейделя принимают компактный вид

$$
x^{(k+1)} = B_1 x^{(k+1)} + B_2 x^{(k)} + c . \tag{4.28}
$$

Заметим, что $B = B_1 + B_2$, и поэтому решение исходной системы удовлетворяет равенству

$$x = B_1 x + B_2 x + c\,. \qquad (4.29)$$

Достаточные условия сходимости.

Теорема 4.3. Пусть $\|B\| < 1$, где $\|B\|$ - одна из норм $\|B\|_\infty, \|B\|_1$. Тогда при любом выборе начального приближения $x^{(0)}$ метод Зейделя сходится со скоростью геометрической прогрессии со знаменателем $q \le \|B\|$.

Теорема 4.4. Пусть выполнено условие

$$\|B_1\| + \|B_2\| < 1. \qquad (4.30)$$

Тогда при любом выборе приближения метод Зейделя сходится и верна оценка погрешности

$$\|x^{(n)} - x\| \le q^n \|x^{(0)} - x\|\,. \qquad (4.31)$$

где $q = \|B_2\| / (1 - \|B_1\|) < 1$.

Апостериорная оценка погрешности.

Предложение 4.1. Пусть выполнено условие $\|B\| < 1$. Тогда для метода Зейделя справедлива апостериорная оценка погрешности

$$\|x^{(n)} - x\| \le (\|B_2\| / (1 - \|B\|)) \|x^{(n)} - x^{(n-1)}\|, \quad n \ge 1. \qquad (4.32)$$

Доказательство. Вычитая из равенства (4.28) равенство (4.29), поучим

$$x^{(k+1)} - x = B_1 (x^{(k+1)} - x) + B_2 (x^{(k)} - x)\,. \qquad (4.33)$$

Возьмём $k = n - 1$ и запишем равенство (4.33) в следующем виде:

$$x^{(n)} - x = B(x^{(n)} - x) + B_2 (x^{(n-1)} - x^{(n)})\,. \qquad (4.34)$$

Тогда

$$\|x^{(n)} - x\| \le \|B\| \|x^{(n)} - x\| + \|B_2\| \|x^{(n-1)} - x^{(n)}\|\,. \qquad (4.35)$$

Неравенство (4.35) позволяет сформулировать простой критерий окончания итерационного процесса. Если решение требуется найти с точностью $\varepsilon > 0$, то итерации метода Зейделя следует вести до выполнения

неравенства $\| x^{(n)} - x^{(n-1)} \| \| B_2 \| /(1- \| B \|) < \varepsilon$ или эквивалентного ему неравенства

$$\| x^{(n)} - x^{(n-1)} \| < \varepsilon_2 . \qquad (4.36)$$

где $\varepsilon_2 = \varepsilon(1- \| B \|) / \| B_2 \|$.

Геометрическая интерпретация метода. Приведём геометрическую интерпретацию метода Зейделя на примере решения системы

$$a_{11}x_1 + a_{12}x_2 = b_1 ,$$
$$a_{21}x_1 + a_{22}x_2 = b_2 .$$

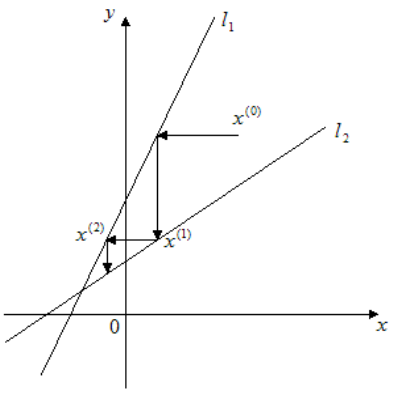
Первое уравнение задаёт на плоскости x_1Ox_2 прямую l_1, второе – прямую l_2 (рис.4.1). Расчётные формулы метода принимают вид

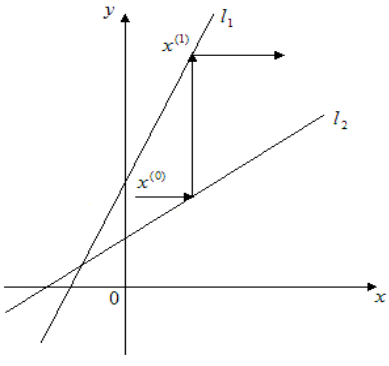
$$x_1^{(k+1)} = \quad b_{12}x_2^k + c_1 ,$$
$$x_2^{(k+1)} = b_{21}x_1^{(k+1)} \quad + c_2 ,$$

где $b_{12} = -a_{12}/a_{11}$; $c_1 = b_1/a_{11}$; $b_{21} = -a_{21}/a_{22}$; $c_2 = b_2/a_{22}$.

Пусть приближение $x^{(k)}$ уже найдено. Тогда для определения $x_1^{(k+1)}$ координата $x_2 = x_2^{(k)}$ фиксируется, и точка x перемещается параллельно оси ox_1 до пересечения с прямой l_1. Координата x_1 точки пересечения принимается за $x_1^{(k+1)}$. Затем точка x перемещается вдоль прямой $x_1 = x_1^{(k+1)}$

до пересечения с прямой l_2. Координата x_2 точки пересечения принимается за $x_2^{(k+1)}$. Далее процесс повторяется.





a) b)

Рис. 4. 1. Иллюстрация метода Зейделя: a – сходящийся итерационный процесс; b – расходящийся итерационный процесс

Пример 4.2. Решить систему

$$\begin{cases} 6.25x_1 - x_2 + 0.5x_3 = 7.5; \\ -x_1 + 5x_2 + 2.12x_3 = -8.68; \\ 0.5x_1 + 2.12x_2 + 3.6x_3 = -0.24 \end{cases}$$

методом Зейделя с точностью $\varepsilon = 10^{-3}$ в норме $\| \|_\infty$.

Решение: Приведем систему к виду (4.24) и вычислим $\|B\|_\infty = \max\{0.24, 0.624, 0.7278\} = 0.7278$. Следовательно, в силе теоремы 4.3 метод Зейделя сходится.

Последовательные приближения будем вычислять по формулам:

$$\begin{aligned} x_1^{(k+1)} &= 0.16x_2^{(k)} - 0.08x_3^{(k)} + 1.2; \\ x_2^{(k+1)} &= 0.21x_1^{(k+1)} - 0.424x_3^{(k)} - 1.76; \\ x_3^{(k+1)} &= -0.1389x_1^{(k+1)} - 0.5889x_2^{(k+1)} - 0.0667 \end{aligned}$$

в предложении, что $x^{(0)} = (0,0,0)^T$. Для заданной системы

$$B_2 = \begin{bmatrix} 0 & 0.16 & -0.08 \\ 0.0 & 0.0 & -0.424 \\ 0 & 0 & 0 \end{bmatrix}$$

и $\|B\|_\infty = 0.424$. Итерационный процесс будем продолжать до выполнения неравенства (4.36), где $\varepsilon = 10^{-3} \cdot (1 - 0.7278)/0.424 \approx 0.64 \cdot 10^{-3}$. Значения приближений с четырьмя десятичными цифрами после десятичной точки приведены в табл. 4.2.

n	0	1	2	3	4

Таблица 4.2

$x_1^{(n)}$ $x_2^{(n)}$ $x_3^{(n)}$	0.0000 0.000 0.000	1.20000 -14960 0.4960	0.9088 -1.8288 0.8841	0.8367 -1.9435 0.9616	0.8121 -1.9813 0.9873	0.8040 -1.9938 0.9958
$\left\|x^{(n)} - x^{(n-1)}\right\|_\infty$	-	1.4960	0.3328	0.1147	0.0376	0.0125

n	6	7	8
$x_1^{(n)}$ $x_2^{(n)}$ $x_3^{(n)}$	0.8013 -1.9980 0.9986	0.8004 1.9993 0.9995	0.8001 -1.9998 0.9998
$\left\|x^{(n)} - x^{(n-1)}\right\|_\infty$	0.0041	0.0014	0.0005

Варианты практических заданий к главе 4

Решить систему линейных уравнений с точностью ε = 0.001

1. $x_1 = 0.23x_1 - 0.04x_2 + 0.21x_3 - 0.18x_4 + 1.24$
   $x_2 = 0.45x_1 - 0.23x_2 + 0.06x_3 - 0.88$
   $x_3 = 0.26x_1 + 0.34x_2 - 0.11x_3 + 0.62$
   $x_4 = 0.05x_1 - 0.26x_2 + 0.34x_3 - 0.12x_4 - 1.17$

2. $x_1 = 0.21x_1 + 0.12x_2 - 0.34x_3 - 0.16x_4 - 0.64$
   $x_2 = 0.34x_1 - 0.08x_2 + 0.17x_3 - 0.18x_4 + 1.42$
   $x_3 = 0.16x_1 + 0.34x_2 + 0.15x_3 - 0.31x_4 - 0.42$
   $x_4 = 0.12x_1 - 0.26x_2 - 0.08x_3 + 0.25x_4 + 0.83$

3. $x_1 = 0.32x_1 - 0.18x_2 + 0.02x_3 + 0.21x_4 + 1.83$
   $x_2 = 0.34x_1 - 0.08x_2 + 0.17x_3 - 0.18x_4 + 1.42$
   $x_3 = 0.16x_1 + 0.34x_2 + 0.15x_3 - 0.31x_4 - 0.42$
   $x_4 = 0.12x_1 - 0.26x_2 - 0.08x_3 + 0.25x_4 + 0.83$

4. $x_1 = 0.42x_1 - 0.32x_2 + 0.03x_3 + 0.44$
   $x_2 = 0.11x_1 - 0.26x_2 - 0.36x_3 + 1.42$
   $x_3 = 0.12x_1 + 0.08x_2 - 0.14x_3 - 0.24x4 - 0.83$
   $x_4 = 0.15x_1 - 0.35x_2 - 0.18x3 - 1.42$

5. $x_1 = 0.18x_1 - 0.34x_2 - 0.12x_3 + 0.15x_4 - 1.33$
   $x_2 = 0.34x_1 - 0.08x_2 + 0.17x_3 - 0.18x_4 + 1.42$
   $x_3 = 0.16x_1 + 0.34x_2 + 0.15x_3 - 0.31x_4 - 0.42$

$$x_4 = 0.12x_1 - 0.26x_2 - 0.08x_3 - 0.25x_4 + 0.83$$

ГЛАВА 5
МЕТОДЫ ПОИСКА РЕШЕНИЙ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

В этой главе рассматривается задача поиска решений системы нелинейных уравнений существенно более сложная, чем задача поиска решения одного нелинейного уравнения или системы линейных алгебраических уравнений. В дальнейшее изложении будем считать, что множестве m - мерных векторов введена некоторая норма, порождающая соответствующую норму для квадратных матриц порядка m (см. § 4.1).

§ 5.1. Постановка задачи и основные этапы решения

Постановка задачи. Задача поиска решения системы m нелинейных уравнений с *m* неизвестными вида

$$f_1(x_1, x_2, \dots, x_m) = 0$$
$$f_2(x_1, x_2, \dots, x_m) = 0$$
$$\dots \quad \dots \quad \dots$$
$$f_m(x_1, x_2, \dots, x_m) = 0$$

на практике встречается значительно чаще, чем рассмотренная в гл. 2 задача поиска решения уравнения с одним неизвестным, так как в реальных исследованиях интерес представляет, как правило, определение не одного, а нескольких параметров. Найти точное решение системы, т.е. вектор $\vec{\bar{x}} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_m)^T$, удовлетворяющий уравнениям (5.1), практически невозможно. В отличие от случая решения систем линейных алгебраических уравнений использование прямых методов здесь исключается. Единственно реальный путь решения системы (5.1) состоит в использовании итерационных методов для получения приближенного решения $\overrightarrow{x^*} = (x^*{}_1, x^*{}_2, \dots, x^*{}_m)^T$, удовлетворяющего при заданном ε>0 неравенству $||\vec{x}^* - \bar{x}|| < \varepsilon$.

Следует отметить тот факт, что задача решения системы (5.1) может не иметь решения, а в случае, когда решения есть, их число может быть произвольным.

Пример 5.1. Рассмотрим систему уравнений

$$4x_1^2 + x_2^2 = 4;$$
$$x_1 - x_2^2 + t = 0,$$

где x_1, x_2 – неизвестные; t – параметр. Первое уравнение задает на плоскости x_1Ox_2 эллипс, второе уравнение – параболу. Координаты точек пересечения этих кривых дают решения системы. Если значение параметра t меняется от -2 до +2 то возможны следующие ситуации; а) t=-2 – решений нет; б) t=-1 – одно решение; в) t=0 – два решения;t=1 – три решения; б) t=2 – четыре решения.

Обозначения. Введем сокращенную форму записи систем. Наряду с вектором неизвестных $\vec{x} = (x_1, x_2, \dots, x_m)^T$ в дальнейшем будем использовать вектор-функцию $f = (f_1, f_2, \dots, f_m)^T$. В этих обозначениях система (5.1.)примет вид

$$\vec{f}(\vec{x}) = 0 \qquad (5.2)$$

Будем считать функции $f_1(x)$ непрерывно дифференцируемыми в некоторой окрестности решения $\vec{\bar{x}}$. Введем для системы функций $f_1, f_2, \dots f_m$ матрицу Якоби

$$\vec{f}'(\vec{x}) = \begin{bmatrix} \frac{\sigma f_1(x)}{a x_1} & \frac{\sigma f_1(x)}{a x_2} & \dots & \frac{\sigma f_1(x)}{a x_m} \\ \frac{\sigma f_2(x)}{a x_1} & \frac{\sigma f_2(x)}{a x_2} & \dots & \frac{\sigma f_2(x)}{a x_m} \\ \dots & \dots & \dots & \dots \\ \frac{\sigma f_m(x)}{a x_1} & \frac{\sigma f_m(x)}{a x_2} & \dots & \frac{\sigma f_m(x)}{a x_m} \end{bmatrix} \qquad (5.3)$$

Основные этапы решения. Поиск решения системы (5.1) начинается с этапа локализации. Для каждого из искомых решений $\vec{\bar{x}}$ указывается множество, которое содержит только одно это решение и расположено в достаточно малой его окрестности. Часто в качестве такого множества выступает параллелепипед или шар в m-мерном пространстве.

В некоторых случаях этап локализации не вызывает затруднений; соответствующие множества могут быть заданными, определяться из физических соображений, из смысла параметров x_j либо быть известными из опыта решения подобных задач.

Однако чаще всего задача локализации (особенно при больших m) представляет собой сложную проблему, от успешного решения которой в основном и зависит возможность вычисления решений системы (5.1). Во многих случаях полное решение задачи локализации невозможно. Задача локализации может считаться решенной удовлетворительно, если для $\vec{\bar{x}}$ удается найти хорошее начальное приближение. В простейших случаях (например, для системы двух уравнений с двумя неизвестными) могут быть использованы графические методы.

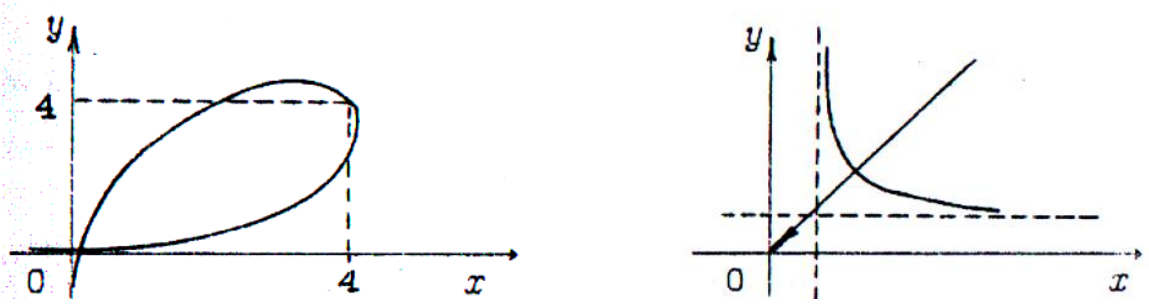
На втором этапе для вычисления решения с заданной точностью используется один из итерационных методов решения нелинейных систем.

При рассмотрении этих методов будут использованы определения § 2.1, связанные со сходимостью итерационных методов, считая, что в неравенствах (2.4) и (2.5) следует заменить знак модуля на знак нормы, а δ-окрестность решения $\vec{\bar{x}}$ следует понимать как множество точек x удовлетворяющих условию $\left\|\vec{x}-\vec{\bar{x}}\right\|<\delta$.

Пример 5.2. Локализуем решения системы

$$\begin{aligned} x_1^3 + x_2^3 &= 8x_1x_2, \\ x_1 \ln x_2 &= x_2 \ln x_1. \end{aligned} \qquad (5.4)$$

Решение. На плоскости x_1Ox_2 построим графики уравнений системы. График первого уравнения рассматриваемой системы приведен на рис. 5.1,а. Графиком этого уравнения является лист Декарта. График второго уравнения состоит из луча-биссектрисы первого координатного угла и пересекающей эту биссектрису в точке (e,e) кривой (рис. 5.1,б). Из рис. 5.2 видно, что эти кривые пересекаются в трёх точках A, B, C, т.е. система имеет три решения. Графики симметричны относительно прямой $x_1 = x_2$, поэтому координаты точки B равны и легко вычисляются: $x_1 = 4$; $x_2 = 4$. По этой же причине достаточно определить только координаты $\vec{x}_1$ и $\vec{x}_2$ точки C, так как точка A имеет координаты $x_1 = \vec{x}_2$ и $x_2 = \vec{x}_1$. Рис. 5.2 показывает, что точка C содержится в прямоугольнике $\Pi = \{(x, y) : 3.5 \le x_1 \le 4, \quad 1.5 \le x_2 \le 2.5\}$ и $x_1 \approx 3.8 \quad x_2 \approx 2$



а) б)

Рис. 5.1. Графики уравнений (5.4) : а - график уравнения $x_1^3 + x_2^3 = 8x_1x_2$;

б - график уравнения $x_1 \ln x_2 = x_2 \ln x_1$

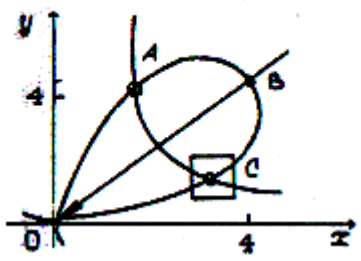


Рис. 5.2. Изображение решений системы (5.4)

Корректность и обусловленность задачи. Будем считать, что система (5.1) имеет решение $\vec{\bar{x}}$, причем в некоторой окрестности этого решения матрица Якоби $\vec{f}'(\vec{x})$ невырождена. Выполнение этого условия гарантирует, что в указанной окрестности нет других решений системы (5.1). Заметим, что погрешности в вычислении вектора-функции $\vec{f}$ приводят к тому, что существует область неопределенности D, содержащая решение системы $\vec{\bar{x}}$, такая, что для всех $\vec{x} \in D$ векторное уравнение $\vec{f}(\vec{x}) = 0$ удовлетворяется с точностью до погрешности. Область D может иметь довольно сложную геометрическую структуру. Мы ограничимся только лишь оценкой радиуса $\bar{\varepsilon}$ этой области.

Если известно, что для близких к $\vec{\bar{x}}$ значений $\vec{x}$ вычисляемые значения $\vec{f}^*(\vec{x})$ удовлетворяют неравенству $\left\| \vec{f}(\vec{x}) - \vec{f}^*(\vec{x}) \right\| \le \Box \left(\vec{f}^* \right)$, то ε можно приближенно оценить с помощью неравенства $\bar{\varepsilon} \le \left\| \left(\vec{f}^* \left(\vec{\bar{x}} \right) \right)^{-1} \right\| \Box \left(\vec{f}^* \right)$.

Так что в рассматриваемой задаче роль абсолютного числа обусловленности играет норма матрицы, обратной матрице Якоби $\vec{f}^*(\vec{x})$.

§ 5.2. Метод простой итерации

Пусть требуется с заданной точностью $\varepsilon > 0$ найти решение $\bar{x} = (\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_m)$ системы (5.1). Преобразуем систему (5.1) к следующему эквивалентному виду (к виду, удобному для итераций):

$$\begin{aligned} x_1 &= \varphi_1\left(x_1, x_2, \ldots, x_m\right); \\ x_2 &= \varphi_2\left(x_1, x_2, \ldots, x_m\right); \\ &\ldots \\ x_m &= \varphi_m\left(x_1, x_2, \ldots, x_m\right). \end{aligned} \tag{5.5}$$

Если ввести вектор-функцию $\vec{\varphi} = \left(\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_m\right)^T$, то (5.5) запишется следующим образом:

$$\vec{x} = \vec{\varphi}\left(\vec{x}\right). \tag{5.6}$$

Предположим, что начальное приближение $\vec{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, \ldots, x_m^{(0)}\right)^T$ задано. Подставляя его в правую часть системы (5.6), получим $\vec{x}^{(1)} = \vec{\varphi}\left(\vec{x}^{(0)}\right)$. Если $\vec{x}^{(1)}$ подставить в правую часть (5.6), то можно получить $\vec{x}^{(2)} = \vec{\varphi}\left(\vec{x}^{(1)}\right)$. Продолжая вычисления по формулам

$$\vec{x}^{(k+1)} = \vec{\varphi}\left(\vec{x}^{(k)}\right), \tag{5.7}$$

получим последовательность $\vec{x}^{(0)}, \vec{x}^{(1)}, \ldots, \vec{x}^{(n)}, \ldots$ приближений к решению $\vec{\bar{x}}$. Это означает, что очередное приближение $\vec{x}^{(k+1)}$ вычисляется через предыдущее приближение $\vec{x}^{(k)}$ следующим образом:

$$\begin{aligned} x_1^{(k+1)} &= \varphi_1\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)}\right); \\ x_2^{(k+1)} &= \varphi_2\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)}\right); \\ &\ldots \\ x_m^{(k+1)} &= \varphi_m\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)}\right). \end{aligned}$$

Пусть $\vec{\varphi}'\left(\vec{x}\right)$- матрица Якоби, отвечающая вектору-функции $\vec{\varphi}\left(\vec{x}\right)$ (определение см. в § 5.1). Сформулируем теорему о сходимости метода простых итераций.

Теорема 5.1. Пусть в некоторой σ-окрестности решения $\vec{x}$ функции $\varphi_i\left(\vec{x}\right), i = 1, 2, \ldots, m$ дифференцируемы и выполнено неравенство

$$\left\|\vec{\varphi}'\left(\vec{x}\right)\right\| \le q < 1, \tag{5.8}$$

где $0 \le q < 1$, q - постоянная.

Тогда независимо от выбора начального приближения $\vec{x}^{(0)}$ из указанной σ-окрестности корня итерационная последовательность не выходит из этой окрестности, метод сходится со скоростью геометрической прогрессии и справедлива оценка погрешности

$$\left\| \vec{x}^{(n)} - \vec{\bar{x}} \right\| \le q^n \left\| \vec{x}^{(0)} - \vec{\bar{x}} \right\|. \qquad (5.9)$$

Условие (5.8) означает, что в окрестности решения производные $\frac{\delta \varphi_1}{\delta \varphi_j}$ для всех i и j должны быть "достаточно малы по модулю". Иначе говоря, систему (5.1) следует преобразовать к такому виду (5.5), чтобы функции φ_i слабо менялись при изменении аргументов, т.е. были почти "постоянными". Каких-либо общих правил, как это сделать, в общем случае нет.

В условиях теоремы 5.1 верна апостериорная оценка погрешности

$$\left\| \vec{x}^{(n)} - \vec{\bar{x}} \right\| \le \frac{q}{1-q} \left\| \vec{x}^{(n)} - \vec{x}^{(n-1)} \right\|, \qquad (5.10)$$

которая удобна для формирования критерия окончания итераций, если известна величина q. Однако найти значение q, удовлетворяющее неравенству (5.8) для всех $\vec{x}$ из некоторой σ–окрестностью корня, далеко не просто. В ряде случаев при наличии достаточно хорошего начального

приближения $\vec{x}^{(0)}$ можно, считая, что $q \approx q_0 = \left\| \vec{\varphi}'(\vec{x}^{(0)}) \right\|$, использовать следующий практический критерий окончания итерационного процесса:

$$\left\| \vec{x}^{(n)} - \vec{x}^{(n-1)} \right\| \le \varepsilon_1 = \varepsilon(1 - q_0)/q_0.$$

Пример 5.3. Используя метод просто итерации, найдем с точностью $\varepsilon = 10^{-3}$ решение $\bar{x}_1, \bar{x}_2$ системы (5.3).

Решение. Приведем систему к следующему виду, удобную для итераций:

$$x_1 = \sqrt[3]{8x_1x_2 - x_2^3}\ ,$$

$$x_2 = x_2 + x_2 / \ln(x_2) - x_1 / \ln(x_1)\ ,$$

Здесь $\varphi_1(x_1, x_2) = \sqrt[3]{8x_1x_2 - x_2^3}$, $\varphi_2(x_1, x_2) = x_2 + x_2 / \ln(x_2) - x_1 / \ln(x_1)$. Проверим выполнение условия сходимости в окрестности точки C. Вычислим матрицу Якоби

$$\overline{\varphi}`(x_1, x_2) = \begin{bmatrix} \frac{\sigma\varphi_1}{\sigma x_1} & \frac{\sigma\varphi_1}{\sigma x_2} \\ \frac{\sigma\varphi_2}{\sigma x_1} & \frac{\sigma\varphi_2}{\sigma x_2} \end{bmatrix} = \begin{bmatrix} \frac{8x_2}{3(8x_1x_2 - x_2^3)^{2/3}} & \frac{8x_2 - 3x_2^2}{3(8x_1x_2 - x_2^3)^{2/3}} \\ \ln^{-2} x_1 - \ln^{-1} x_1 & \mathrm{I} + \ln^{-1} x_2 - \ln^{-2} x_2 \end{bmatrix}$$

Так как $\bar{x}_1 \approx 3.8$ и $\bar{x}_2 \approx 2$, то для $\vec{x} \approx \vec{\bar{x}}$ имеем

$$\overline{\varphi}`(x_1, x_2) \approx \overline{\varphi}`(3.8,2) = \begin{vmatrix} 0.379 & 0.436 \\ -0.188 & 0/361 \end{vmatrix}.$$

Тогда $\left\|\varphi`(x_1, x_2)\right\|_\infty \approx \left\|\varphi`(3.8,2)\right\|_\infty \approx 0.379 + 0.436 \approx 0.815$.

Следовательно, можно сделать вывод о том, что метод простой итерации

$$x_1^{(k+1)} = \sqrt[3]{8x_1^{(k)}x_2^{(k)} - (x_2^{(k)})^3}\ ,$$

$$x_2^{(k+1)} = x_2^{(k)} + x_2^{(k)} / \ln(x_2^{(k)}) - x_1^{(k)} / \ln(x_1^{(k)}) \qquad (5.11)$$

По-видимому будет сходиться со скоростью геометрической прогрессии со знаменателем $q \approx 0.815$, если начальное приближение брать в достаточно малой окрестности решения.

Возьмем $x_1^{(0)} \approx 3.8$; и $x_2^{(0)} \approx 2$ и буде выполнять итерации по формулам (5.11), используя критерий окончания (5.10), в котором $\varepsilon = 10^{-3}$, $q = 0.815$. Результаты вычисление с шестью знаками мантиссы приведены в таблице 5.1

Таблица 5.1

k	1	2	3	4	5
$x_1^{(k)}$	3.80000	3.75155	3.74960	3.75892	3.76720
$x_2^{(k)}$	2.00000	2.03895	2.06347	2.07498	2.07883

k	5	6	7	8	9
$x_1^{(k)}$	3.77198	3.77399	3.77450	3.77440	3.77418
$x_2^{(k)}$	2.07920	2.07850	2.07778	2.07732	2.07712

При k=9 критерий окончания выполнения и можно положить $\bar{x}_1 = 3.774 \pm 0.001$; $\bar{x}_2 = 2.077 \pm 0.001$.

Модификации метода простой итерации. В некоторых случаях для ускорения сходимости можно использовать следующий аналог метода Зейделя (см. § 4.3):

$$x_1^{(k+1)} = \varphi_1\left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \dots, x_m^{(k)}\right);$$

$$x_2^{(k+1)} = \varphi_2\left(x_1^{(k+1)}, x_2^{(k)}, x_3^{(k)}, \dots, x_m^{(k)}\right);$$

$$x_3^{(k+1)} = \varphi_3\left(x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k)}, \dots, x_m^{(k)}\right);$$

. . . .

$$x_i^{(k+1)} = \varphi_i\left(x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x_i^{(k)}, \dots, x_m^{(k)}\right);$$

. . . . . .

$$x_m^{(k+1)} = \varphi_m\left(x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k+1)}, \dots, x_m^{(k)}\right).$$

Более общий вариант метода Зейделя состоит в следующем: -я компонента решения на $(k+1)$ -й итерации метода определяется как решение нелинейного уравнения

$$F\left(x_i^{(k+1)}\right) = 0 \ ,$$

где $F(x_i) = f_i(x_1^{(k-1)}, \dots, x_{i-1}^{(k-1)}, x_i, x_{i+1}^{(k)}, \dots, x_m^{(k)})$.

Преимущества этого подхода состоят в возможности использовать методы решения уравнений с одним неизвестным и в отсутствии необходимости приведения системы уравнений к виду, удобному для итераций. Указанный метод будет сходиться, если матрица Якоби $\vec{f}'(\vec{x})$ обладает свойством диагонального преобладания.

5.3. Метод Ньютона для решения систем нелинейных уравнений.

Предположим, что исходя из начального приближения $\vec{x}^{(0)}$ к решению $\vec{\bar{x}}$ построены приближения $\vec{x}^{(1)}, \vec{x}^{(2)}, \dots, \vec{x}^{(n)}$. Заменим в системе (5.1) каждую из функций $f_i(i = 1, 2, \dots, m)$ линейной частью ее разложения по формуле Тейлора в точке $x^{(n)}$:

В результате придем к системе линейных алгебраических уравнений

$$f_1\left(\vec{x}^{(n)}\right) + \sum_{j=1}^{m} \frac{\partial f_1\left(\vec{x}^{(n)}\right)}{\partial x_j}\left(x_j - x_j^{(n)}\right) = 0;$$

$$f_2\left(\vec{x}^{(n)}\right) + \sum_{j=1}^{m} \frac{\partial f_2\left(\vec{x}^{(n)}\right)}{\partial x_j}\left(x_j - x_j^{(n)}\right) = 0;$$

. . . . . .

$$f_m\left(\vec{x}^{(n)}\right) + \sum_{j=1}^{m} \frac{\partial f_m\left(\vec{x}^{(n)}\right)}{\partial x_j}\left(x_j - x_j^{(n)}\right) = 0;$$

имеющей в матричной форме записи следующий вид:

$$\vec{f}'\left(\vec{x}^{(n)}\right) + \vec{f}'\left(\vec{x}^{(n)}\right)\left(\vec{x} - \vec{x}^{(n)}\right) = 0, \qquad (5.12)$$

где $\vec{f}'$ - матрица Якоби (5.3).

Предположим, что матрица $\vec{f}'(\vec{x}^{(n)})$ невырожденная, т.е. существует обратная матрица $(\vec{f}'\left(\vec{x}^{(n)}\right))^{-1}$. Тогда система (5.12) имеет единственное решение, которое и принимается за очередное приближение $\vec{x}^{(n+1)}$ к решению $\vec{\vec{x}}^{(n)}$. Таким образом, $\vec{x}^{(n+1)}$ удовлетворяет равенству

$$\vec{f}\left(\vec{x}^{(n)}\right) + \vec{f}'\left(\vec{x}^{(n)}\right)\left(\vec{x}^{(n+1)} - \vec{x}^{(n)}\right) = 0. \qquad (5.13)$$

Выражая из (5.13) $\vec{x}^{(n+1)}$, получим итерационную формулу метода Ньютона

$$\vec{x}^{(n+1)} = \vec{x}^{(n)} - (\vec{f}'(\vec{x}^{(n)}))^{-1}\,\vec{f}(\vec{x}^{(n)}) \qquad (5.14)$$

Формула (5.14) предполагает использование трудоемкой операции обращения матрицы, поэтому непосредственное ее использование для вычисления $\vec{x}^{(n+1)}$ в большинстве случаев нецелесообразно. Обычно вместо этого решают эквивалентную системе (5.13) систему линейных уравнений

$$\vec{f}'(\vec{x}^{(n)})\Delta\vec{x}^{(n+1)} = -\vec{f}(\vec{x}^{(n)}) \qquad (5.15)$$

относительно поправки $\Delta\vec{x}^{(n+1)} = \vec{x}^{(n+1)} - \vec{x}^{(n)}$. Затем полагают

$$\vec{x}^{(n+1)} = \vec{x}^{(n)} + \Delta\vec{x}^{(n+1)} \qquad (5.16)$$

Сформулируем основную теорему о сходимости метода Ньютона.

Теорема 5.2. Пусть в некоторой окрестности решения $\vec{x}^{(n)}$ системы (5.1) функции $f_i(i=1,2,...,m)$ дважды непрерывно дифференцируемы и матрица $\vec{f}'(\vec{x}^{(n)})$ невырождена. Тогда найдется такая малая $\delta-$окрестность решения $\vec{x}$, что при произвольном выборе начального приближения из этой окрестности итерационная последовательность метода Ньютона не выходит за пределы окрестности и справедлива оценка

$$\left\|\vec{x}^{(n+1)}-\vec{x}^{(n)}\right\|<\delta^{-1}\left\|\vec{x}^{(n)}-\vec{x}\right\|^2, n\geq 0.$$

Эта оценка означает, что метод сходится с квадратичной скоростью. Квадратичная скорость сходимости метода Ньютона позволяет использовать простой практический критерий окончания

$$\left\|\vec{x}^{(n+1)}-\vec{x}^{(n)}\right\|<\varepsilon. \qquad (5.17)$$

Пример 5.4. Используя метод Ньютона, найдем с точностью $\varepsilon=10^{-4}$ решение x_1, x_2 системы (5.4).

Решение. Возьмем $x_1^{(0)}=3.8; x_2^{(0)}=2$ и будем вести вычисления по формулам (5.15),(5.16), в которых

$$\vec{f}=\begin{vmatrix} x_1^3+x_2^3-8x_1x_2 \\ x_1\ln x_2-x_2\ln x_1 \end{vmatrix}, \vec{f}'=\begin{vmatrix} 3x_1^2-8x_2 & 3x_2^2-8x_1 \\ \ln x_2-x_2/x_1 & x_1/x_2-\ln x_1 \end{vmatrix}$$

Результаты вычисления с шестью знаками мантиссы приведены в таб. 5.2

Таблица 5.2

k	0	1	2	3
$x_1^{(k)}$	3.80000	3.77258	3.77388	3.77389
$x_2^{(k)}$	2.00000	2.07189	2.07708	2.07710

При k=3 критерий окончания $\left\|\vec{x}^{(k)}-\vec{x}^{(k-1)}\right\|_\infty<\varepsilon=10^{-4}$ выполняется и можно положить $x_1=3.7739\pm0.0001; x_2=2.0771\pm0.0001$.

Варианты практических заданий к главе 5

1) Используя метод итераций, решить систему нелинейных уравнений уравнений с точностью 0.001

2) Используя метод Ньютона, решить систему нелинейных уравнений с точностью 0.001

1. $\sin(x+1) - y = 1.2$ $2x + \cos y = 2$	1. $\operatorname{tg}(xy + 0.4) = x^2$ $0.6x^2 + 2y^2 = 1,\ x>0, y>0$
2. $\cos(x - 1) + y = 0.5$ $x - \cos y = 3$	2. $\sin(x+y) - 1.6x = 0$ $x^2 + y^2 = 1,\ x>0, y>0$
3. $\sin x + 2y = 2$ $\cos(y-1) + x = 0.7$	3. $\operatorname{tg}(xy+0.1) = x^3$ $x^2 + 2y^2 = 1$
4. $\cos x + y = 1.5$ $2x + \sin(y-0.5) = 1$	4. $\sin(x+y) - 1.2x = 0.2$ $x^2 + y^2 = 1$
5. $\sin(x+0.5) - y = 1$ $\cos(y-2) + x = 0$	5. $\operatorname{tg}(xy + 0.3) = x^2$ $0.9x^2 + 2y^2 = 1$
6. $\cos(x+0.5) + y = 0.5$ $\sin y - 2x = 1.6$	6. $\sin(x+y) - 1.3x = 0$ $x^2 + y^2 = 1$
7. $\sin(x-1) = 1.3$ $x - \sin(y+1) = 0.8$	7. $\operatorname{tg} xy = x^2$ $0.8x^2 + 2y^2 = 1$
8. $2y - \cos(x+1) = 0$ $x + \sin y = -0.4$	8. $\sin(x+y) - 1.5x = 0.1$ $x^2 + y^2 = 1$
9. $\cos(x+0.5) - y = 2$ $\sin y - 2x = 1$	9. $\operatorname{tg} xy = x^2$ $0.7x^2 + 2y^2 = 1$
10. $\sin(x+2) - y = 1.5$ $x + \cos(y-2) = 0.5$	10. $\sin(x+y) - 1.2x = 0.1$ $x^2 + y^2 = 1$

ГЛАВА 6.
ПРИБЛИЖЕНИЕ ФУНКЦИИ

Постановка задачи. Одной из важнейших задач в процессе математического моделирования является вычисление значений функции, входящих в математическое описание модели. Для сложных моделей подобные вычисления могут быть трудоемкими даже при использовании ЭВМ. При выполнении программ , реализующие основные вычислительные методы, большая часть времени также затрачивается на вычисление функций.

Используемые в математических моделях функции задаются как аналитическим способом, так и табличным, при котором функции известны только при дискретных значениях аргумента. Ограниченный объем памяти ЭВМ не позволяет хранить подобные таблицы функций, желательно иметь возможность "сгущать" таблицы, заданные с крупным шагом аргументов.

Поставленные проблемы решаются путем приближенной замены функции *f(x)* на более простую функцию $\varphi(x)$, которую нетрудно вычислять при любом значении аргумента x в заданном интервале его изменения. Введенную функцию $\varphi(x)$ можно использовать не только для приближен-ного определения численных значений *f(x),* но и для проведения аналити-ческих выкладок при теоретическом исследовании модели. Приближение функции *f*(x) более простой функцией $\varphi(x)$ называется аппроксимацией (от латинского *approximo* - приближаюсь).

Таким образом, задача о приближении (аппроксимации) функции состоит в следующем. Функцию *f(x),* заданную таблично, требуется приближенно заменить (аппроксимировать) некоторой функцией $\varphi(x)$ так чтобы отклонение $\varphi(x)$ от *f(x)* в некоторой области удовлетворял заданному условию. Функция $\varphi(x)$ называется аппроксимирующей функцией. Близости функций *f(x)* и $\varphi(x)$ добиваются введением в аппроксимирующую функцию $\varphi(x)$ свободных параметров $C_0, C_1, C_2, \ldots, C_n$ и соответствующим их выбором.

§ 6.1. Интерполяция зависимостей

Частным случаем задачи аппроксимации таблично заданной функции является интерполирование. Пусть функция *f(x)* задана таблицей значений, полученной из эксперимента или путем вычисления в

последовательности значений аргумента $x_0, x_1, x_2, \ldots, x_n$ (табл. 6.1). Выбранные значения аргумента x называются узлами таблицы. Будем считать что узлы в общем случае не являются равноотстоящими. Для функции *y=f(x)* требуется найти аппроксимирующую функцию φ(x, C_1, … ,C_n) такую, чтобы она совпадала с табличными значениями функции *f*(x) во всех узлах x_j:

$$\varphi(x_j, C_0, C_1, C_2, \ldots, C_n) = f_j,\ 0 \le j \le n\ .$$

Свободные параметры C_j определяются из системы (6.1).

Таблица 6.1

x	$f(x)$
x_0	f_0
x_1	f_1
...	...
x_n	f_n

Подобный способ определения аппроксимирующей функции называется лагранжевой интерполяцией, а условие (6.1) - условием Лагранжа.

Задачей интерполяции в узком смысле считают нахождение приближенных значений табличной функции при аргументах x, не совпадающих с узловыми. Если значение аргумента x расположено между узлами $x_0 \le x \le x_n$, то нахождение приближенного значения функции *f(x)* называют интерполяцией; если аппроксимирующую функцию вычисляют вне интервала $[x_0, x_n]$, то процесс называют экстраполяцией. Происхождение этих терминов связано с латинскими словами inter - между, внутри, *pole* - узел, *extra* - вне. В более общем плане с помощью интерполяции решают широкий круг задач численного анализа - дифференцирование и интегрирование функций, нахождение нулей и экстремумов функций, решение дифференциальных уравнений и т.д. Возможность решения подобных задач обусловлена достаточно простым видом аппроксимирующей функции $\varphi(x)$.

§ 6.1.I. Интерполяция каноническим полиномом

Выберем в качестве аппроксимирующей функции $\varphi(x)$ полином $P_n(x)$ степени *n* в каноническом виде:

$$\varphi(x) = P_n(x) = C_0 + C_1 x + C_2 x^2 + \ldots + C_n x^n. \qquad (6.2)$$

Свободными параметрами интерполяции C_j являются коэффициенты полинома (6.2). Интерполяция полиномами обладает такими преимуществами как простота вычислений их значений, дифференцирования и интегрирования.Коэффициенты C_j определим из условий Лагранжа $P_n(x_j) = f.$, $0 \le j \le n$. Учитывая (6.2), получим систему $(n+1)$ линейных уравнений:

$$
\begin{aligned}
&c_0 + c_1 x_0 + c_2 x_0^2 + \ldots + c_n x_0^n = f_0;\\
&c_0 + c_1 x_1 + c_2 x_1^2 + \ldots + c_n x_1^n = f_1;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&c_0 + c_1 x_n + c_2 x_n^2 + \ldots + c_n x_n^n = f_n;
\end{aligned}
\qquad (6.3)
$$

Решение полученной системы (n+1) уравнений относительно свободных параметров $c_0,c_1,\ldots,c_n$ позволяет найти коэффициенты интерполирующего полинома (6.2).

Пример 6.1. Пусть функция *y=f(x)* задана табл.6.2. Найдем интерполирующую функцию в виде алгебраического многочлена второго порядка. Для этого запишем систему уравнений (6.3), требуя выполнения в каждом узле условия (6.1):

$$
\begin{aligned}
&c_0 + c_1(-1) + c_2(-1)^2 = 3{,}5;\\
&c_0 + c_1(0) + c_2(0)^2 = 0.5;\\
&c_0 + c_1(2) + c_2(2)^2 = 6.5;
\end{aligned}
$$

Таблица 6.2

x	*f(x)*
-1	3.5
0	0.5
2	6.5

Решая эту систему линейных уравнений, получаем значения свободных параметров $c_0=0.5$, $c_1=-1$, $c_2=2$. Таким образом, найден интерполирующий полином:

$$\varphi(x) = 0.5 - x + 2x^2. \qquad (6.4)$$

Геометрической интерпретацией задачи интерполяции является нахождение функции, график которой проходит через заданную систему точек (x_i,f_i), $i=0,1,\ldots,n$ (рис. 6.1). Если вид функции $\varphi(x)$ не оговорен, то задача имеет бесконечно много решений. Однако если в качестве интерполирующей функции используется полином (6.2) степени n, то задача имеет единственное решение.

Покажем единственность интерполирующего полинома (6.2). Будем исходить из противного: предположим, что для функции *y=f(x)*, заданной в (n+1) узле x_i, $i=0,1,\ldots,n$, существуют два многочлена $P_n(x)$ и $P_n'(x)$ степени не выше n, удовлетворяющие условиям Лагранжа:

$$P_n(x_j) = f_j \text{ и } P_n'(x_j) = f_j,\ 0 \le j \le n \qquad (6.5)$$

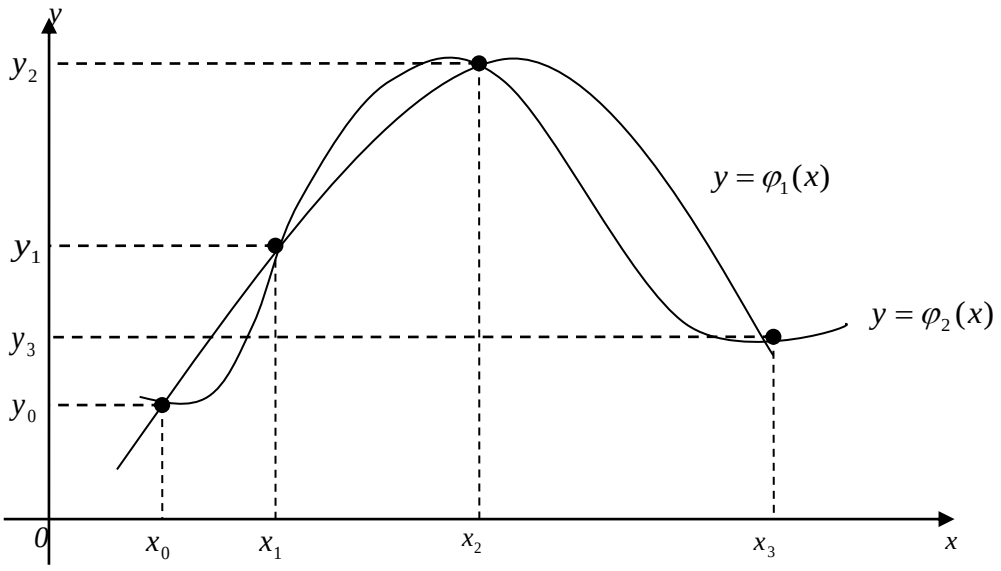


Рис.6.1. Геометрическая иллюстрация задачи интерполяции

Тогда разность $R_n(x) = P_n(x) - P_n^{'}(x)$ так же является алгебраическим многочленом степени не выше *n*. Исходя из (6.5), $R_n(x)$ обращается в нуль в (*n*+1) узлах интерполяции:

$$R_n(x_j) = P_n(x_j) - P_n^{'}(x_j) = 0,\ 0 \le j \le n$$

и, следовательно, имеет (*n*+1) корень. Поскольку многочлен степени *n* не может иметь более *n* корней, то из этого следует, что $R_n(x)$ тождественно равен нулю при любых *x* и, следовательно, интерполяционный многочлен единственен $P_n(x_j) = P_n^{'}(x_j)$.

Таким образом, среди бесконечного множества функций $\varphi(x)$, удовлетворяющих условию Лагранжа (6.1), графики которых проходят

через заданную систему точек (рис.6.1), лишь одна функция является алгебраическим многочленом степени не выше *n*.

Для вычисления значений коэффициентов интерполяционного полинома (6.2) можно использовать программы 3.1, в которых следует изменить подпрограмму для формирования расширенной матрицы системы уравнений. Численные значения полинома определяют по схеме Горнера.

Пронумерованные блоки программы (рис. 6.2) оформлены в виде подпрограмм, остальные блоки размещаются в основной программе.

Массивы X и F введены для размещения узлов и значений интерполируемой функции, массив C – для коэффициентов полинома, двумерный массив A – для элементов расширенной матрицы системы (6.3). Расширенная матрица формируется в блоке 2. последний столбец матрицы определяется во внешнем цикле (строка 200 программы 6.1B).

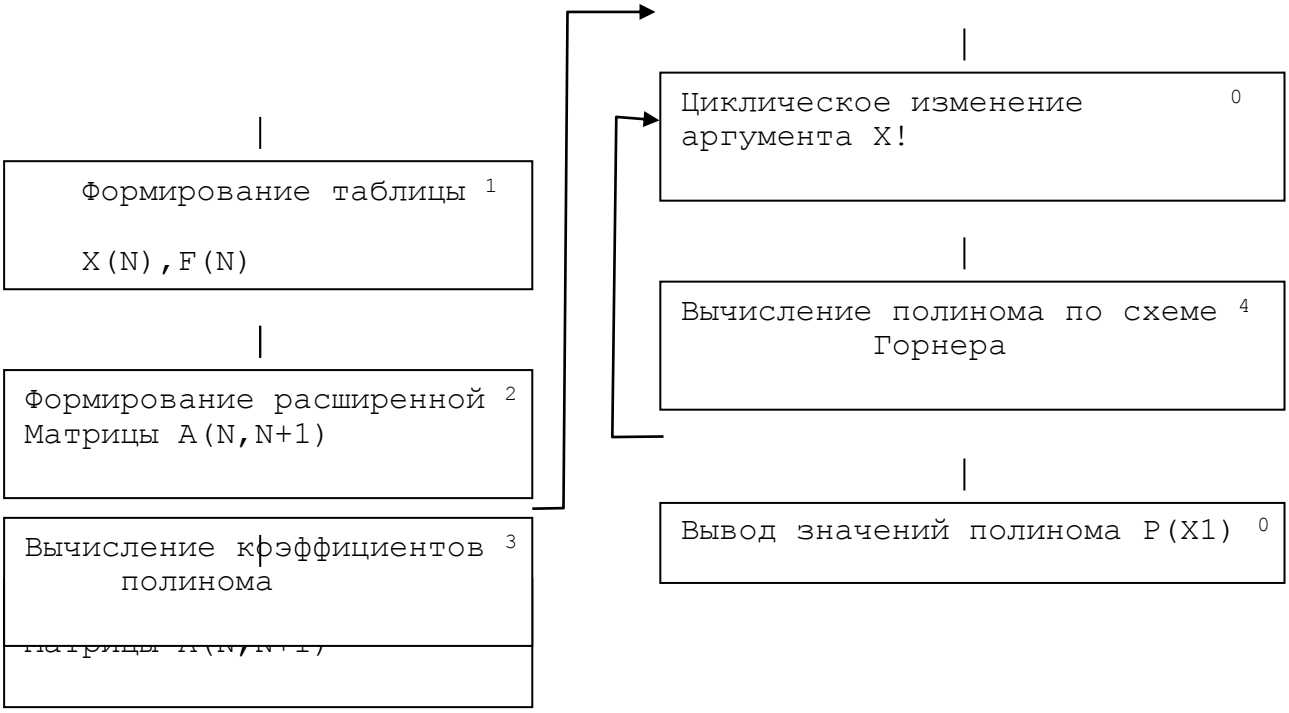


Рис. 6.2. Блок-схема программы интерполяции
каноническим полиномом

Другие элементы матрицы, являющиеся степенями узловых значений аргумента *x*, получаются путем последовательного умножения во внутреннем цикле (строка 210).

В подпрограмме Гаусса на языках Бейсик и Паскаль проведены изменения (по сравнению с программами 3.1) с учетом того, что расширенная матрица A имеет строку и столбец с нулевым номером, а также исключены операторы, проверяющие определитель основной

матрицы на нуль, так как определитель Вандермонда отличен от нуля, если отсутствуют совпадающие узлы (определитель системы (6.3) называется определителем Вандермонда).

Блок 4 вычисления полинома по схеме Горнера может быть дополнен при необходимости операторами для вычисления производных от интерполяционного полинома. Учитывая, что начальные имена массивов на языке Фортран начинаются с единицы, переменную в программе 6.1F необходимо задавать на единицу больше степени аппроксимирующего полинома.

Следует отметить, что рассмотренный способ вычисления интерполяционного полинома не является эффективным по затратам времени и объему памяти ЭВМ. Существуют более экономичные формы представления и вычисления полиномов.

```
1     rem                                                  программа 6.1в
2     rem                          интерполяция каноническим полиномом
10    dim x(20),f(20),c(20),a(20,21)
20    print "n,x0,x9,h"; \ input n,xO,x9,h
30    gosub 100 \ rem  Формирование таблицы
40    gosub 200 \ rem  расширенная матрица
50    gosub 300 \ rem  коэффициенты полинома
60    for i=0 to n \ print “c”;i;“=”;c(i) \ next i
70    for x1=x0 to x9 step h
80    gosub 500 \ rem  вычисление полинома
85    print x1,p \ next x1
90    goto 20
100   for i=0 to n \ print "x";i;"f";i;
110   input x(i),f(i) \ next i
190   return
200   for i=0 to n \ r=1 \ s=x(i) \ a(i,n+1)=f(i)
210   for j=0 to n \ a(i,j)=r \ r=r*s \ next j
```



```
220   next i
290   return
300   n1=n+1
310   for k=0 to n \ k1=k+1 \ s=a(k,k) \ j=k
320   for i=k1 to n \ r=a(i,k)
330   if abs(r) > abs(s) then s=r \ j=i
340   next i
350   if j=k then 370
```

```
360    for i=k to n1 \ r=a(k,i) \ a(k,i)=a(j,i) \ a(j,i)=r \ next i
370    for j=k1 to n1 \ a(k,j)=a(k,j)/s \ next j
380    for i=k1 to n \ r=a(i,k)
390    for j=k1 to n1 \ a(i,j)=a(i,j)-a(k,j)*r \ next j
400    next i
410   next k
420   for i=n to 0 step -1 \ s=a(1,n1)
430   for j=j+1 to n \ s=s-a(i,j)*c(j) \ next j
440   c(1)=s \ next 1
490   return
500   p=c(n)
510   for i=n-1  to  0  step -1  \ p=c(1)+p*x1  \  next 1
590   return
```

(программа 6.1f интерполяция каноническим полиномом)

```
      Real x(20),f(20),c(20),a(20,21)
1     Type *,'n,x0,x9,h?'
      Accept *,n,x0,x9,h
      Call tab(n,x,f)
      Call matr(n,x,f,a)
      Call gauss(n,a,c)
      Do 2  i=1,n
2     Type 3,1,c(1)
3     Format (1x,'x',12,'=',e13.6)
      K=(x9-x0)/h+1.5
      X1=x0
      Do 4   i=1,k
      Call pol(n,c,x1,p)
      Type *,x1,p
4     X1=x1+h
      Goto 1
      End
      Subroutine tab(n,x,f)
```



```
      Real x(20),f(20)
      Do 11 i=1,n
      Type 12,i,1
11    Accept *,x(1),f(1)
12    Format (3x,'x',12,',f',12,'?')
      return
      End
      Subroutine  matr(n,x,f,a)
```

```
      Real x(20),f(20),a(20,21)
      Do 21  i=1,n
      R=1
      S=x(1)
      A(1,n+1)=f(1)
      Do 21 j=1,n
      A(i,j)=r
21    r=r*s
      return
      end
      subroutine  gauss(n,a,x)
      real a(20,21),x(20)
      n1=n+1
      do 35 k=1,n
      k1=k+1
      s=a(k,k)
      j=k
      do 31 i=k1,n
      r=a(i,k)
      if  (abs(r).1e.abs(s)) goto 31
      s=r
      j=1
31    continue
      if (j.eq.k) goto 33
      do 32  i=k,n1
      r=A(K,i)
      A(k,i)=A(j,i)
      32    A(j,i)=r
      33    Do 34 j=K1,N1
      34    A(k,j)=A(k,j)/s
      Do 35  i=k1,n
      R=A(i,k)
      Do 35 j=k1,n1
```



```
35      a(i,j)-a(i,j)-a(k,j)*r x(n)=a(n,n1)
        d0 37 i=n-1,1,-1
        s=a(i,nl)
        do 36 j=i+1,n
36      s=s-a(i,j)*x(j)
37      x(i)=s
        return
        end
```

```
      subroutine   pol(n,c,x1,p)
      real c(20)
      p=c(n)
      do 41  i=n-1,i ,-1
41    p=c(i)+p*x1
      return
      end
```

программа 6.1р

интерполяция каноническим полиномом

```
type mat=array[0..20,0..21 ] of  real;
vec=array[0..20] of  real;
var i,k,n: integer; x1,x0,x9,h,p: real;
x,f,c: vec; a: mat;
procedure tab (n:integer;var  x,f:vec);
var i: integer;
begin
for i:=0 to n do begin
wrlte('x',i,'f',i,'?');readln(x[i],f[i])
end end;
procedure matr(n:integer;var x,f:yec;var a:mat);
var 1,j:integer;r,s:real;
begin
for i:=0 to n do begin r:=1.0;
s:=x[i]; a[i,n+1]:=f[i];
for j:=0 to n do begin a[i,j]:=r; r:=r*s end
end  end;
procedure   gauss(n:integer; var a:mat; var x:vec);
var i,j,k,k1,n1:integer; r,s:real;
begin n1:=n+l
for k:=0 to n do begin k1:=k+l; s:=[k,k];   j:=k;
```



```
for i:=k1  to n do begin r:=a[i,k];
if abs(r)>abs(s)  then begin s:=r;  j:=l end end;
if  j<>k  then for i:=k to n1 do begin
r:=a[k,i]; a[k,i]:=a[j,i]; a[j,i]:=r end;
for j:=k1 to nl do a[k,j]:=a[k,j]/s;
for i:=k1  to n do begin r:=a[i,k];
for  j:=k1 to nl do a[i,j]:=a[i,j]-a[k,j]*r
end end;
```

```
for i:=n downto 0 do begin s:=a[i,n1];
for j:=i+1 to n do s:=s-a[i,j]*x[j];
x[i];=s    end end;
procedure   pol(n:integer; var c:vec; var xl ,p:real);
var i:integer;
begin  p:= c[n];
for i:=n-1 downto 0 do p:=c[i]+p*xl end;
begin
repeat write ('n,x0,x9,h?');  readln(n,x0,x9,h);
tab(n,x,f); matr(n,x,f,a); gauss(n,a,c);
for i:=0 to n do writeln('c',i,' =',c[i]);
k:=round((x9-x0)/h+1.0); x1:=x0;
for  i:=1  to k do begin pol(n,c,x1,p);
writeln(x1,'  ' ,p);  x1:=xl+h end
until false
end.
```

§ 6.1.2. Интерполяционный полином Лагранжа

Рассмотрим задачу интерполяции с помощью алгебраического полинома (*6.2*). Пусть функция $f(x)$ задана в $(n+1)$ узлах, произвольно расположенных на отрезке $[a,b]$

$$f_0 = f(x_0),\ f_1 = f(x_1),...,\ f_n = f(x_n)$$

Требуется найти интерполирующий алгебраический многочлен $L_n(x)$ степени не выше n, удовлетворяющий условиям (6.1)

$$L_n(x_0) = f_0, L_n(x_1) = f_1,..., L_n(x_n) = f_n. \qquad (6.6)$$

Будем искать $L_n(x)$ в виде

$$L_n(x) = Q_0(x) f_0 + Q_1(x) f_1 + ... + Q_n(x) f_n, \qquad (6.7)$$

где $Q_i(x)$ - коэффициенты, зависящие только от узлов $x_i, i = 0,1,..., n$ и текущего значения x.

Для того чтобы выполнялись условия (6.6), требуется, чтобы коэффициенты $Q_i(x)$ удовлетворяли условию

$$Q_i(x_j) = \begin{cases} 0, \ \textit{если } i \neq j, \\ 1, \ \textit{если } i = j, \end{cases} \tag{6.8}$$

Действительно, чтобы $L_n(x_0) = f_0$, необходимо, чтобы в (6.7) $Q_0(x_0) = 1$; $Q_1(x_0) = 0, \ldots, Q_n(x_n) = 0$. В то же время в других узлах интерполяции первое слагаемое формулы (6.7), связанное с f_i, должно быть равно нулю, т.е. $Q_0(x_i) = 0, \ i = 1, 2, \ldots, n$.

Этим требованиям отвечает коэффициент вида

$$Q_0(x) = \frac{(x - x_1)(x - x_2)(x - x_3) \ldots (x - x_n)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3) \ldots (x_0 - x_n)}. \tag{6.9}$$

Поскольку в числителе $Q_0(x)$ записано произведение разностей со всеми узлами, кроме x_0, то $Q_0(x)$ обращается в нуль при $x = x_i$, $i = 1, 2, \ldots, n$. В то же время при $x = x_0$ числитель и знаменатель дроби взаимно сокращаются и $Q(x_0) = 1$. Для того чтобы $L_n(x) = f_1$, коэффициенты в (6.7) должны принять следующие значения: $Q_1(x_1) = 1$, $Q_0(x_1) = Q_2(x_1) = \ldots Q_n(x_1) = 0$. Чтобы в других узлах коэффициент $Q_1(x)$, связанный с f_i, принял значение нуль $(Q_1(x_i) = 0, \ i = 1, 2, \ldots, n)$, он должен иметь вид

$$Q_1(x) = \frac{(x - x_0)(x - x_2)(x - x_3) \ldots (x - x_n)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3) \ldots (x_1 - x_n)}.$$

Произведение разностей в числителе (6.10) обращает его в нуль во всех узлах, кроме x_1, а при $x = x_1$ коэффициент равен 1. Обобщая (6.9) и (6.10), получим выражение для $Q_i(x)$

$$Q_i(x) = \prod_{\substack{j=0 \\ i \neq j}}^{n} \frac{x - x_j}{x_i - x_j} \tag{6.11}$$

и для интерполяционного полинома Лагранжа

$$Q_i(x) = \sum_{i=0}^{n} f_i \prod_{\substack{j=0 \\ i \neq j}}^{n} \frac{x - x_j}{x_i - x_j}. \qquad (6.12)$$

В отличие от канонического интерполяционного полинома для вычисления значений полинома Лагранжа не требуется предварительного определения коэффициентов полинома путем решения системы уравнений, его можно записать непосредственно по заданной таблице значений функции. Для этого следует учесть следующие правила: полином Лагранжа содержит столько слагаемых, сколько узлов в таблице; каждое слагаемое – это произведение коэффициента полинома на соответствующее значение f_i; числитель коэффициента при f_i содержит произведение разностей x со всеми узлами кроме x_i, знаменатель полностью повторяет числитель при $x = x_i$.

Однако следует отметить, что для каждого значения x полином (6.12) приходится пересчитывать вновь, коэффициенты же канонического полинома вычисляются только один раз. С известными коэффициентами для вычисления значений канонического полинома требуется значительно меньшее количество арифметических операций по сравнению с полиномом Лагранжа. Поэтому практическое применение полинома Лагранжа оправдано только в случае, когда интерполяционная функция вычисляется в сравнительно небольшом количестве точек x.

Пример 6.2. Функция $y = f(x)$ задана табл. 6.2. Требуется найти интерполирующую функцию в виде полинома Лагранжа.

Решение. Используя приведенные правила, запишем интерполяционный полином

$$L_2(x) = \frac{(x-0)(x-2)}{(-1-0)(-1-2)} 3.5 + \frac{(x+1)(x-2)}{(0+1)(0-2)} 0.5 + \frac{(x+1)(x-0)}{(2+1)(2-0)} 6.5 =$$

$$= x(x-2)\frac{3.5}{3} - (x+1)(x-2)\frac{0.5}{2} + (x+1)\frac{6.5}{6}. \qquad (6.13)$$

После упрощений многочлен (6.13) преобразуется к виду $L_2(x) = 0.5 - x + 2x^2$, совпадающему с (6.4), что подтверждает единственность интерполяционного многочлена для заданной таблицы интерполируемой функции.

Приведенный пример показывает основное достоинство полинома Лагранжа – возможность его применения для таблицы с неравноотстоящими узлами интерполяции. Это свойство обеспечивает универсальность полинома Лагранжа с точки зрения возможности его применения к любой таблице значений функции.

Блок-схема алгоритма интерполяции полинома Лагранжа отличается от блок-схемы рис.6.2 только отсутствием блока 3 для вычислений коэф-фициентов.

В программах 6.2 наряду со значениями интерполяционного полинома Лагранжа предусмотрено вычисление первой и второй производных. Операторы для производных $P1$ и $P2$ (строка 270 программы 6.2В) получены путем почленного дифференцирования правой части оператора для накопления суммы (6.12). Произведение в формуле (6.12) вычисляется путем последовательного умножения с помощью внутреннего цикла по переменной J (строки 220-270), вне цикла переменная R для произведения инициализи-руется оператором $R = 1$ (строка 210). Переменные $P1$ и $P2$ введены для получения произведения для первой и второй производных.

Программа 6.2В

```
1    rem
2    rem                полином Лагранжа и его производные
10   dim  x(20), f(20)
20   print  “n,x0,x9,h”;\input n,x0,x9,h
30   gosub 100\rem формирование таблицы
```



```
40   for  x1=x0 to x9 step h
50   gosub  200\rem вычисление полинома
60   print  x1,p,p1,p2
70   next  x1
90   goto  20
100   for  i=0 to n\print  “x”;1;”f”;1
110   input  x(i),f(i) \ next 1
190   return
```

```
      200   p=0\p1=0\p2=0
      210   for  i=0 to n \ r=1\r1=0\r2=0
      220   for  j=0 to n
      230   if  i=j then 260
      240   q=x(i)-x(j)\s=x1-x(j)\r2=(r2*s+2*r1)/q
      250   r1=(r1*s+r)/q\r=r*s/q
      260   next  j
      270   p2=p2+r2*f(i) \ p1=p1+r1*f(i)\p=p+r*f(i)
      280   next  i
      290   return

      c                                                    программа 6.2F
      c                                          полином Лагранжа и его производные
            real  x(20),f(20)
        type  *,'n,x0,x9,h?'
              Accept  *,n,x0,x9,h
              Call tab(n,x,f)
              k=(x9-x0)/h+1.5
              x1=x0
              do 2 i=1,k
              call pl(n,x,f,x1,p,p1,p2)
              type *,x1,p,p1,p2
2             x1=x1+h
              goto 1
              end
              subroutine tab(n,x,f)
              real x(20),f(20)
              do 11 i=1,n
              type 12, i,i
11            Accept *,x(i),f(i)

r1=0.0
r2=r1
```



```
do 21 j=1,n
if(i.eq.j) goto 21
q=x(i)-x(j)
s=x1-x(j)
r2+(r2*s+2*r1)/q
r1=(r1*s+r)/q
r=r*s/q
```

```
21    continue
p2=p2+r2*f(i)
p1+p1+r1*f(i)
22   p=p+t*f(i)
return
end
```

программа 6.2Р

полином Лагранжа и его производные

```
type vec=array [0..20] of real;
var i,k,n:integer; x1,x0,x9,h.,p,pi ,p2: real; x,f:vec;
procedure tab(n:integer; var x,f:vec);
var i: integer; begin
for i:=0 to n do begin
write ('x',i,',f',i,'?'); readln(x[i],f[i])
end end;
procedure pl(n:integer; var x,f:vec;
var x1,p,p1,p2:real);
var i;j:integer; r,r1,r2,q,s:real;
begin p:=0.0; p1:=p; p2:=p
for i:=0 to n do begin r:=1.0; r1:=0.0; r2:=rl;
for j:=0 to n do
if i<>j then begin
s:=x1-x[j];
q:=x[i]-x[j]; r2:=(r2*s+2*r1)/q;
r1:=(r1*s+r)/q; r:=r*s/q end; p2:=p2+r2*f[i];
p1:=p1+r1*f[i]; p:=p+r*f[i] end end; begin
repeat write ('n,x0,x0,h?'); readln (n,x0,x9,h); tab(n,x,f);
k:=round((x9-x0)/h+1.0); x1:=x0;
fot i to k do begin pl(n,x,f,x1,p,p1,p2);
writeln (x1,' ', p,' ',p1,' ',p2); x1:=x1+h end
until false end.
```


§ 6.1 .3. Интерполяционный полином Ньютона

На практике в большинстве случаев интерполируемая функ-ция $y=f(x)$ задается в равноотстоящих узлах так, что $x_i=x_o+ih$ где h - шаг интерполяции, а $i=1,...,n$. В этом случае для нахождения интер-поляционного полинома может применятся полином Ньютона, который использует конечные разности. Конечной разностью первого

порядка в точке x_i называется разность $\Delta f_i = f_{i+1} - f_i$, где $f_{i+1} = f(x_i + h)$ и $f_i = f(x_i)$.

Для функции, заданной таблично в $(n-1)$ узле x_i, $i = 0,1,\ldots,n$, конечные разности первого порядка могут быть вычислены в точках $0,1,2,\ldots,n-1$:

$$\Delta f_0 = f_1 - f_0,$$
$$\Delta f_1 = f_2 - f_1,$$
$$\ldots\ldots\ldots\ldots$$
$$\Delta f_{n-1} = f_n - f_{n-1}$$

Используя конечные разности первого порядка, можно получить конечные разности второго порядка $\Delta^2 f_i = \Delta f_{i+1} - \Delta f_i$, которые вычисляются в точках $0,1,2,\ldots,n-2$:

$$\Delta^2 f_0 = \Delta f_1 - \Delta f_0,$$
$$\Delta^2 f_1 = \Delta f_2 - \Delta f_1,$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\Delta^2 f_{n-2} = \Delta f_{n-1} - \Delta f_{n-2}$$

Отметим, что

$$\Delta^2 f_0 = \Delta f_1 - \Delta f_0 = (f_2 - f_1) - (f_1 - f_0) = f_2 - 2f_1 + f_0. \qquad (6.14)$$

Для конечной разности k-го порядка в узле с номером i справедлива рекуррентная формула, связывающая ее с разностями $(k-1)$-го порядка $\Delta^k f_i = \Delta^{k-1} f_{i+1} - \Delta^{k-1} f_i$. Эта Формула позволяет просто вычислить конечные

разности с помощью таблицы конечных разностей (*табл.6.3*). При этом для вычисления конечной разности любого порядка в точке x_i вычисляется разность значений, размещенных в $(i+1)$-й строке и i-й строке столбца, расположенного слева от текущего.

Пусть функция $y = f(x)$ задана в равноотстоящих с шагом h в узлах $x_i \quad i = 1,2,\ldots,n$. Требуется найти интерполяционный полином $P_n(x)$, удовлетворяющий условию

$$P_n(x_i) = f_i, \qquad i = 0,1,2,\ldots,n. \tag{6.15}$$

Будем искать интерполяционный полином в виде

$$P_n(x) = a_0 + a_1(x - x_0) + \ldots + a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}). \tag{6.16}$$

где a_i, $i = 0,1,2,\ldots,n$ - коэффициенты, не зависящие от узлов интерполяции

x_i	f_i	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$	$\Delta^5 f$
x_0	f_0	Δf_0	$\Delta^2 f_0$	$\Delta^3 f_0$	$\Delta^4 f_0$	$\Delta^5 f_0$
$x_1 = x_0 + h$	f_1	Δf_1	$\Delta^2 f_1$	$\Delta^3 f_1$	$\Delta^4 f_1$	
$x_2 = x_0 + 2h$	f_2	Δf_2	$\Delta^2 f_2$	$\Delta^3 f_2$		
$x_3 = x_0 + 3h$	f_3	Δf_3	$\Delta^2 f_3$			
$x_4 = x_0 + 4h$	f_4	Δf_4				
$x_5 = x_0 + 5h$	f_5					

Полином вида (*6.16*) позволяет учитывать дополнительные узлы интерполяции добавлением слагаемых, уточняющих предыдущий

результат. При этом первые слагаемые (*6.16*) не изменяются. В этом, в частности, состоит преимущество интерполяционного полинома (*6.16*) перед полиномом Лагранжа. Напомним, что в силу единственности интерполяционного полинома формула (*6.16*) является наряду с

полиномом Лагранжа лишь одной из возможных форм записи интерполяционного полинома, удовлетворяющего условию интерполяции (6.15) для функции, заданной таблично.

Для нахождения коэффициентов полинома Ньютона a_j будем подставлять в (6.16) значения x, совпадающие с узлами интерполяции, требуя выполнения (6.15). Пусть $x = x_0$, согласно (6.15) $P_n(x_0) = a_0 = f_0$. Следователь-но, $a_0 = f_0$. Пусть $x = x_1$. Тогда

$$P_n(x_1) = a_0 + a_1(x_1 - x_0) = f_0 + a_1(x_1 - x_0) = f_1. \qquad (6.17)$$

Из равенства (6.17) следует

$$a_1 = \frac{f_1 - f_0}{x_1 - x_0} = \frac{\Delta f_0}{h}.$$

Пусть $x = x_2$. Тогда

$$P_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = f_0 + \frac{\Delta f_0}{h} \quad 2h + a_2 \quad 2h^2 = f_2.$$

Выражая неизвестный коэффициент и учитывая (6.14), получим

$$a_2 = \frac{f_2 - 2\Delta f_0 - f_0}{2h^2} = \frac{f_2 - 2(f_1 - f_0) - f_0}{2h^2} = \frac{\Delta^2 f_0}{2h^2}.$$

Продолжая подстановку, можно получить выражение для любого коэффициента с номером i:

$$a_i = \frac{\Delta^i f_0}{i! h^t}, \quad i = 0,1,..., n.$$

Подставим найденные значения коэффициентов в (6.16), получим интерполяционный полином Ньютона

$$P_n(x) = f_0 + \frac{\Delta f_0}{1! h}(x - x_0) + \frac{\Delta^2 f_0}{2! h^2}(x - x_0)(x - x_1) + ... + \frac{\Delta^n f_0}{n! h^n}(x - x_0)(x - x_1)...(x - x_{n-1}). \qquad (6.18)$$

Воспользуемся полиномом (6.18) как одной из возможных форм записи интерполяционного полинома для вычисления значения функции, заданной табл. 6.4, при x=1,0:

$$P_3(1) = 2{,}0 + \frac{1{,}0}{1\cdot 1{,}5}(1-0) + \frac{1{,}5}{2!1{,}5^2}(1-0)(1-1{,}5) - \frac{1{,}2}{3!1{,}5^3}(1-0)\times(1-1{,}5)\times(1-3) = 1{,}707.$$

В силу единственности интерполяционного полинома такой же результат будет получен с помощью полинома Лагранжа при использовании одних и тех же узлов интерполяции.

Если интерполируемая функция задана большой таблицей, то возникает проблема выбора количества и расположения узлов, используемых при построении интерполяционного полинома (6.18).

Рассмотрим функцию $y = \sqrt{x}$, значения и соответствующие конечные разности которой записаны в табл. 6.4.

Пусть требуется вычислить значение функции при x=2,3. Воспользуемся интерполяционным полиномом Ньютона. Выберем x_0=1 и ограничимся четырьмя узлами интерполяции.

$$P(2{,}3) = 1 + \frac{0{,}225}{0{,}5}(2{,}3-1) - \frac{0{,}036}{2\cdot 0{,}25}(2{,}3-1)\times(2{,}3-1{,}5) + \frac{0{,}014}{6\cdot 0{,}125}\times(2{,}3-1)\times$$
$$\times(2{,}3-1{,}5)\times(2{,}3-2{,}5) == 1 + 0{,}585 - 0{,}0749 + 0{,}0058 = 1{,}5159.$$

Для вычисления значения функции в той же точке x=2,3 выберем $x_0 = 2$:

$$P(2{,}3) = 1{,}414 + \frac{0{,}167}{0{,}5}(2{,}3-2) - \frac{0{,}016}{2\cdot 0{,}25}(2{,}3-2)(2{,}3-2{,}5) + \frac{0{,}004}{6\cdot 0{,}125}\times$$
$$\times(2{,}3-2)(2{,}3-2{,}5)(2{,}3-3) = 1{,}414 + 0{,}1002 + 0{,}00192 + 0{,}000224 = 1{,}5163$$

Таблица 6.4

x_i	f_i	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$	$\Delta^5 f$
1,0	1,000	0,225	-0,036	0,014	-0,008	0,006
1,5	1,225	0,189	-0,022	0,006	-,002	
2,0	1,414	0,167	-0,016	0,004		
2,5	1,581	0,151	-0,012			
3,0	1,732	0,139				
3,5	1,871					

Отметим, что при выборе x_0 близко к интересующей нас точке x=2,3 слагаемые интерполяционного полинома Ньютона убывают значительно быстрее, чем в случае, когда x_0 не примыкает к точке интерполяции. В силу этого второй результат является более точным(значение $\sqrt{2,3}$, вычисленное с пятью значащими цифрами, равно 1,5166).

Приведённый пример показывает, что если имеется возможность выбора узлов интерполяции, то при использовании интерполяционного полинома Ньютона целесообразно выбрать x_0 близко к точке интерполяции x(интерполирование вперёд). Это обеспечивает более высокую точность при фиксированном числе узлов интерполяции или наименьшее число необхо-димых узлов при заданной точности результата. Запись интерполяционного полинома в виде полинома Ньютона позволяет учитывать дополнительные узлы в первой части таблицы, уточняя ранее полученный результат вновь вычисленными слагаемыми.

Равносильный вариант полинома Ньютона можно записать при симметричной перенумерации узлов исходной таблицы узлов и значений функции

$$P_n(x) = b_n + (x - x_{n-1})b_{n-1} \ldots + b_0(x - x_n)(x - x_{n-1})\ldots(x - x_1) \qquad (6.19)$$

Для определения коэффициентов b_i будем поочерёдно подставлять (6.19) узлы интерполяции, учитывая условия Лагранжа. При $x = x_n$ $P_n(x_n) = b_n = f_n$. Следовательно, b_n=f_n.

При $x = x_{n-1}$ $P_n(x_{n-1}) = b_n + b_{n-1}(x_{n-1} - x_n) = f_n + b_{n-1}(x_{n-1} - x_n) = f_{n-1}$,

откуда

$$b_{n-1} = \frac{f_{n-1} - f_n}{x_{n-1} - x_n} = \frac{f_n - f_{n-1}}{x_n - x_{n-1}} = \frac{\Delta f_{n-1}}{h} .$$

Продолжая подстановку, получим выражения для всех новых коэффициентов полинома (6.19) и запишем интерполяционный полином Ньютона

$$P_n(x) = f_n + \frac{\Delta f_{n-1}(x - x_n)}{11h} + \frac{\Delta^2 f_{n-2}}{21h^2}(x - x_n)(x - x_{n-1}) + \ldots + \\ + \frac{\Delta^n f_0}{n1h^n}(x - x_n)(x - x_{n-1})\ldots(x - x_1). \qquad (6.20)$$

Рассмотрим схему вычисления коэффициентов интерполяционного полинома, которую удобно использовать для программной реализации алгоритма (6.16), которую удобно использовать для программной реализации алгоритма расчёта коэффициентов полинома Ньютона. Полагаем $x=x_0$, тогда по условию (6.15)

$$f_0 + a_1(x_1 - x_0) = f_1 \ ,$$

откуда находим коэффициент

$$a_1 = \frac{f_0 - f_1}{x_0 - x_1} = f_{01} \ , \qquad (6.21)$$

который называется разделенной разностью первого порядка. Величина f_{01} близка к первой производной функции $f(x)$ при малом расстоянии между узлами x_0 и x_1. При $x = x_2$ полином (6.16) принимает значение

$$P_n(x_2) = f_0 + f_{01}(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) .$$

Из условия Лагранжа (6.15) определяем искомый коэффициент

$$a_2 = \frac{f_{01} - f_{02}}{x_1 - x_0} = f_{012} , \qquad (6.22)$$

где

$$_1 = \frac{f_0 - f_1}{x_0 - x_1} = f_{01}.$$

Величина f_{012} называется разделенной разностью второго порядка, которая при близком расположении $x_0, x_1, x_2,$ будет пропорциональна второй производной функции f(x).

Аналогичным образом при $x = x_3$ находим коэффициент полинома Ньютона

$$a_3 = \frac{f_{012} - f_{013}}{x_2 - x_3} = f_{0123} \qquad (6.23)$$

где $$f_{013} = \frac{f_{01} - f_{03}}{x_1 - x_3}, f_{03} = \frac{f_0 - f_0}{x_0 - x_3}$$

Для коэффициента a_k методом математической индукции запишем следующее выражение:

$$a_k = \frac{f_{01}..k-1 - f_{01}..k}{x_{k-1} - x_k} \qquad (6.24)$$

Полученные результаты сведем в табл. 6.5. Для построения интерполя-ционного полинома Ньютона используются только диагональные элементы приведенной таблицы, остальные элементы являются промежуточными данными. Поэтому в программе, реализующей вычисление коэффициентов поли-нома, разделенные разности для экономии памяти размещаются в массиве, где пе-рвоначально хранились значения функции *f(x)* в узлах. Этот массив будет частич-но обновляться при вычислении разделенных разностей очередного порядка.

Рассматриваемая схема вычисления коэффициентов интерполяцион-ного полинома Ньютона согласно табл.6.5 обладает рядом преимуществ по сравнению с классической схемой. Во-первых, обеспечивается меньшая погрешность вычисления разделенных разностей при близко расположенных узлах за счет меньшего количества вычитания близких чисел. Во-вторых, сокращается количество обра-щений к элементам массивов узлов и значений функции *f(x),* так как в формулах для

разделенных разностей уменьшаемые в числителе и знаменателе остаются неизменными для разности каждого порядка. В-третьих, аналитические выражения для коэффициентов полинома Ньютона получаются более простым способом.

После определения коэффициентов полинома Ньютона вычисление его значений при конкретном аргументе x наиболее экономично проводить по схеме Горнера, получаемой путем последовательного вынесения за скобки множителя $(x - x_i)$ в формуле (6.16).

При интерполяции полиномом Ньютона удается разделить задачи определе-ния коэффициентов и вычисления значений полинома при различных значениях аргумента *x*. Аналогичное разделение задач происходит при интерполяции каноническим полиномом (§6.I.I). Поэтому структура программы, реализующей алгоритм интерполяции полином Ньютона, определяется блок схемой рис 6.2. Здесь в блоке 3 размещается подпрограм-ма определения коэффициентов полинома Ньютона путем вычисления разделенных разностей по формулам табл. 6.5.

Таблица 6.5.

x	$f(x)$	**1**	**2**	**3**	**4**
x_0	$\underline{f_0}$				
		$x_1 f_1 f_{01} = \frac{f_0 - f_1}{x_0 - x_1}$			
		$x_2 f_2 f_{02} = \frac{f_0 - f_2}{x_0 - x_2}$	$\underline{f_{012}} = \frac{f_{01} - f_{02}}{x_1 - x_2}$		
		$x_3 f_3 f_{03} = \frac{f_0 - f_3}{x_0 - x_3}$	$f_{013} = \frac{f_{01} - f_{03}}{x_1 - x_3}$	$\underline{f_{0123}} = \frac{f_{012} - f_{013}}{x_2 - x_3}$	
		$x_4 f_4 f_{04} = \frac{f_0 - f_4}{x_0 - x_4}$	$f_{014} = \frac{f_{01} - f_{04}}{x_1 - x_4}$	$f_{0124} = \frac{f_{012} - f_{014}}{x_2 - x_4}$	$\underline{f_{01234}} = \frac{f_{0123} - f_{012}}{x_3 - x_4}$

Вычисление коэффициентов полинома Ньютона (строки 200-290 программы 6.3B, подпрограммы CPN в программах 6.3F и 6.3P) осуществля-ется с помощью двух вложенных циклов по строкам

(переменная I) и столб-цам (переменная J) табл. 6.5. Во внешнем цикле по столбцам инициализи-руются значения уменьшаемых в числителе и знаменателе разделенных разностей (переменные A и B), так как эти величины не изменяются для всех разностей одного порядка.

Значения полинома Ньютона при циклическом изменении значений аргумента (переменной X1) определяются с помощью отдельной полпрограммы, реализующей схему Горнера (строки 300-390 программы 6.3B, подпрограммы PN программы 6.3F и 6.3P). Наряду со значениями полинома вычисляются первая и вторая производные при соответствующих аргументах (переменные P1 и P2).

При работе с программой 6.3F необходимо учитывать, что нумерация элементов массивов на языке Фортран начинается с единицы, поэтому значение переменной N надо задавать на единицу больше, чем в программах на Бейсике и Паскале.

```
1 rem                                                    программа 6.3B
2 rem                                  полином Ньютона и его производные
10 dim x(20), f(20)
20 print “n,x0,x9,h”; \ input n,x0,x9,h
30 gosub 100 \ rem формирование таблицы
40 gosub 200 \ rem коэффициенты полинома
50 for i=0 to n \ print “a”; i; ”=”; f(i)\ next 1
60 for x1=x0 to x9 step h
70 gosub 300 \ rem вычисление полинома
80 print x1,p,p1,p2, \ next x1
90 goto 20
100 for i=0 to n \ print “x”; i; “f”; i
110 input x(i), f(i) \ next 1
190 return
200 for j=1 to n \ a=f(j-1) \ b=x(j-1)
210 for i=j to n \ f(i)=(a-f(i)/(b-x(i)) \ next 1
220 next j
290 return
300 p=f(n) \ p1=0 \ p2=0
310 for i=n-1 to 0 step -1 \ r=x1-x(i)
320 p2=2*pi+r*p2 \ p1=p+r*p1 \ p=f(i)+r*p \ next 1
390
return
```

```
c                                                    программа 6.3F
c                                          полином Ньютона и его производные
      real x(20), f(20)
1     type*, 'n,x0,x9,h?'
         accept *, n,x0,x9,h
         call tab(n,x,f)
         call cpn(n,x,f)
   do  2 i=1,n
2  type 3,i,f(i)
3  format ('a',12,'=',e14.7)
   k=(x9-x0)/h+1.5
   x1=x0
   do 4 i=1,k
   call pn(n,x,f,x1,p,p1,p2)
   type *,x1,p,p1,p2
4  x1=x1+h
   go to 1
   end
   subroutine tab(n,x,f)
   real x(20), f(20)
   do 11 i=1,n
   type 12,i,i
11 accept   *,x(i),f(i)
12 format(3x,'x',12,'f',12,'?')
   return
   end
   subroutine cpn(n,x,f)
   real x(20),f(20)
   do 21 j=2,n
   a=f(j-1)
   b=x(j-1)
   do 21 i=j,n
21 f(i)=(a-f(i))/(b-x(i))
   return
   end
   subroutine pn(n,x,f,x1,p,p1,p2)
   real x(20), f(20)
   p=f(n)
   p=0.0
   p2=p1
   do 31 i=n-1,1,-1
```

```
      r=x1-x(i)
      p2=2*p1+r*p2
      p1=p+r*p1
31    p=f(i)+r*p
      return
      end
```

программа 6.3 p
полином Ньютона и его производные

```
Type vec=array[0..20] or real;
var i,k,n: integer; x1,x0,x9,h,p,p1,p2:real; x,f: vec;
procedure tab(n:integer; var x,f:vec);
var i:integer; begin
for i:=0 to  do begin
write('x',i,'f',i,'?'); readln(x[i],f[i]);
end; end;
procedure cpn(n:integer; var x,f:vec);
var I,j: integer; a,b: real;
begin
for j:=1 to n do begin a:=f[j-1]; b:=x[j-1];
for i:=j to n do f[i]:=(a-f[i])/(b-x[i]);
end; end;
procedure pn(n:integer; var x,f:vec;
var x1,p,p1,p2: real);
var I,j: integer; r: real;
begin p:=f[n]; p1:=0.0; p2:=p1;
for i:=n-1 downto 0 do begin r:=x1-x[i];
p2:=2*p1+r*p2; p1:=p+r*p1; p:=f[i]+r*p;
end; end; begin
repeat write ('n,x0,x9,h?'); readln(n,x0,x9,h);
tab (n,x,f); cpn(n,x,f); for i:=0 to n do
writeln('a',i,'=',f[i]); k:=round((x9-x0)/h+1.0);
x1:=x0; for i:=1 to k do begin pn(n,x,f,x1,p,p1,p2);
writeln(x1,' ',p,' ',p2); x1:=x1+h; end;
until false end.
```

§ 6.2. Подбор эмпирических формул

Если таблично заданная функция $y = f(x)$ отражает результаты эксперимента, то нецелесообразно, чтобы аппроксимирующая функция $\varphi(x)$ в точности повторяла значения f_i, $i = 0, 1, ..., n$, содержащие погрешности измерений. В этом случае следует отказаться от критерия приближения функции (6.1). Также следует поступить, если задано и требуется учесть слишком большое количество узлов интерполяции. Использование условия (6.1) привело бы в этом случае к очень большой степени интерполяционного полинома и, следовательно, к трудностям в его использовании.

В рассматриваемых случаях может быть применён эмпирический подбор аппроксимирующих функций, который состоит в выборе аппроксимирующей функции и последующем определении коэффициентов согласно некоторому критерию. Например, для аппроксимации зависимости, заданной таблицей $f_i = f(x_i)$, $i = 0, 1, ..., n$, может быть выбран многочлен $\varphi_m(x)$ степени $m < n$. Для определения коэффициентов a_i, $i = 0, 1, ..., n$ этого многочлена может быть применён критерий, состоящий в минимизации функции

$$E = F(e_0, e_1, ..., e_n), \qquad (6.25)$$

где e_i, $i = 0, 1, ..., n$ - отклонения аппроксимирующей функции от заданных значений $e_i = \varphi_m(x_i) - f_i$, $0 \le i \le n$.

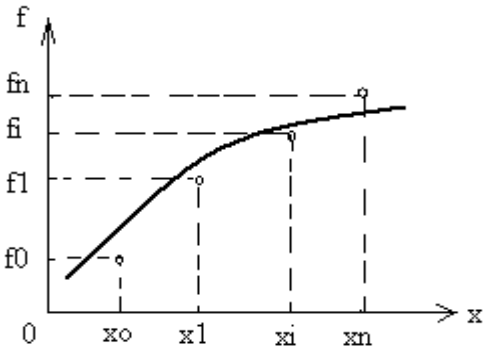


Рис. 6.3 График аппроксимирующей функции.

Так как $m < n$ и условие (6.1) не может быть выполнено во всех заданных точках, то график аппроксимирующей функции $\varphi_m(x)$ (рис. 6.3) не пройдёт

через все заданные точки $(x_i, f_i), 0 \le i \le n$. Этот подход используется при эмпирическом подборе формул.

§ 6.2.1. Метод средних

Подбор эмпирических формул состоит из двух этапов: выбор вида формулы; определение содержащихся в формуле коэффициентов. Если из физических соображений неизвестен вид аппроксимирующей зависимости, то в качестве эмпирической формулы обычно выбирают один из известных видов функции: алгебраический многочлен, показательную, логарифмичес-кую или другую функцию. Вид функции выбирают из геометрических соображений, наносят заданные точки на координатную плоскость. Поскольку аппроксимирующая функция, полученная эмпирическим путём, в ходе последующих исследований, как правило, подвергается преобразова-ниям, то стараются выбирать наиболее простую формулу, удовлетворяющую требованиям точности. Часто в качестве эмпирической формулы выбирают линейную, квадратическую или другую зависимость, описываемую алгебраическим многочленом невысокого порядка.

Если вид формулы выбран, то её можно представить в виде $y = \varphi(x, a_0, a_1, ..., a_m)$, где φ - функция известного вида; $a_0, a_1, ..., a_m$ - неизвестные коэффициенты (параметры). Задача определения коэффициентов эмпирической зависимости состоит в том, чтобы определить такие их значения, при которых эмпирическая формула даёт удовлетворительное в некотором смысле приближение к заданным значениям (x_i, f_i), $0 \le i \le n$; минимизация отклонений между $f(x)$ и $\varphi(x)$

$$e_i = \varphi(x_i, a_0, a_1, ..., a_m) - f_i, \qquad 0 \le i \le n$$

позволяет определить коэффициенты $a_0, a_1, ..., a_m$. Методы определения коэффициентов выбранной эмпирической функции различаются критерием минимизации отклонений e_i, $0 \le i \le n$.

Одним из способов определения параметров эмпирической формулы является метод средних. В этом методе параметры определяются из условия равенства нулю суммы отклонений

$$\sum_{i=0}^{n} e_i = \sum_{i=0}^{n} [\varphi_m(x_1, a_0, a_1, \ldots a_m) - f_1] = 0 \qquad (6.26)$$

Равенство (6.26) позволяет определить лишь единственный параметр эмпирической формулы. Поэтому, если формула содержит m параметров $(m < n)$, равенство (6.26) разбивают на систему уравнений путем группировки отклонений

$$\begin{aligned} &\sum_{i=0}^{k_1} e_1 = 0. \\ &\sum_{i=k_1+1}^{k_2} e_1 = 0. \\ &\ldots \\ &\sum_{i=k_m-1}^{n} e_1 = 0. \end{aligned} \qquad (6.27)$$

Систему уравнений (6.27) решают относительно неизвестных параметров $a_0, a_1, \ldots a_m$.

Пример 6.3. Результаты эксперимента приведены в табл. 6.6. Определить параметры эмпирической формулы.

Решение. Из геометрических соображений выберем в качестве аппроксимирующей функции прямую вида $y = a_0 + a_1 x$.

Таблица 6.6

x_1	f_1
1	3.33
2	2.80
3	1.50
4	1.40
5	0.50

В каждой из заданных точек x_1 (табл. 6.6) можно вычислить отклонение $e_i = a_0 + a_1 x_i - f_i$.

Для определения двух параметров эмпирической формулы $y = a_0 + a_1 x$ сгруппируем отклонения и запишем систему уравнений вида (6.27)

$$a_0 + a_1 \cdot 1 - 3.38 + a_0 + a_1 \cdot 3 - 2.8 + a_0 + a_1 \cdot 4 - 1.5 = 0,$$
$$a_0 + a_1 \cdot 6 - 1.4 + a_0 + a_1 \cdot 7 - 0.5 = 0$$

После преобразований получим

$$3a_0 + 8a_1 = 7.68;$$
$$2a_0 + 13a_1 = 1.9.$$

Решая эту систему, найдем значения коэффициентов $a_0 = 3.68$ и $a_1 = -0.42$. Таким образом, эмпирическая формула $y = 3.68 - 0.42x$ приближенно описывает зависимость величин x и y, отраженную табл. 6.6.

Отметим, что различные варианты группировки отклонений, при формировании системы уравнений вида (6.27), приводят к получению различных значений параметров эмпирической формулы.

Варианты практических заданий к главе 6

Найти приближенное значение функции при данном значении аргумента с помощью интерполяционного многочлена Лагранжа (задание 1) и интерполяционной формулы Ньютона (задание 2), если функция задана в нераноотстоящих узлах таблицы и в равноотстоящих узлах таблицы

Варианты к заданию № 1

Таблица 1

x	y
0.43	1.63597
0.48	1.73234
0.55	1.87686
0.62	2.03345
0.70	2.22846
0.75	2.35973

No варианта	x
1	0.702
	0.512
	0.645
2	0.738
	0.606
	0.713

Таблица 2

x	y
0.02	1.02316
0.08	1.09590
0.12	1.14725
0.17	1.21483
0.23	1.30120
0.30	1.40976

No варианта	x
3	0.102
	0.114
	0.125
4	0.203
	0.154
	0.237

Таблица 3

x	y	No варианта	x
0.35	2.73951	5	0.526
0.41	2.30080		0.453
0.47	1.96864		0.482
0.51	1.78776	6	0.552
0.56	1.59502		0.436
0.64	1.34310		0.602

Таблица 4

x	y	No варианта	x
0.41	2.57418	7	0.616
0.46	2.32513		0.478
0.52	2.09336		0.665
0.60	1.86203	8	0.573
0.65	1.74926		0.673
0.72	1.62098		0.517

Таблица 5

x	y	No варианта	x
0.68	0.80866	9	0.896
0.73	0.89492		0.812
0.80	1.02964		0.774
0.88	1.20966	10	0.955
0.93	1.34087		0.715
0.99	1.52369		0.918

Варианты к заданию № 2

Таблица 1

x	y	No варианта	x
1.375	5.04192	1	1.3832
1.380	5.17744		1.3926
1.385	5.32016		1.3862
1.390	5470695	2	1.3934
1.395	5629686		1.3866
1.400	5.79788		1.3945

Таблица 2

x	y	No варианта	x
0.115	8.65729	3	0.1264
0.120	8.29329		0.1315
0.125	7.95829		0.1232
0.130	7.64893	4	0.1334
0.135	7.36235		0.1285
0.140	7.09613		0.1176

Таблица 3

x	y	No варианта	x
1.150	6.61659	5	0.1521
1.155	6.39989		0.1611
1.160	6.19658		0.1662
1.165	6.00551	6	0.1542
1.170	5.82558		0.1625
1.175	5.65583		0.1548

Таблица 4

x	y	No варианта	x
0.180	5.61543	7	0.1838
0.185	5.46693		0.1875
0.190	5.32634		0.1944
0.195	5.19304	8	0.1976
0.200	5.06649		0.2038
0.205	4.94619		0.1057

Таблица 5

x	y	No варианта	x
0.210	4.83170	9	0.2121
0.215	4.72261		0.2165
0.220	4.61855		0.2232
0.225	4.51919	10	0.2263
0.230	4.42422		0.2244
0.235	4.33337		0.2137

Варианты к заданию № 3

Аппроксимировать многочленом второй степени по методу наименьших квадратов функцию, заданную таблицей

Значения x_i = ix0.1, i=1,2,…,20 одинаковые для всех вариантов

i	Значения $y_i = y(x)$				
	Вариант 1	Вариант 2	Вариант 3	Вариант 4	Вариант 5
1	2.05	2.09	2.02	1.99	2.07
2	1.94	2.05	1.98	2.03	2.17
3	1.92	2.19	1.67	2.20	2.21
4	1.87	2.18	1.65	2.39	2.31
5	1.77	2.17	1.57	2.19	2.10
6	1.88	2.27	1.42	2.61	2.09
7	1.71	2.58	1.37	2.35	2.12
8	1.60	2.73	1.07	2.60	3.49
9	1.56	2.82	0.85	2.55	3.82
10	1.40	3.04	0.48	2.49	3.95
11	1.50	3.03	0.35	2.50	4.22
12	1.26	3.45	-0.30	2.52	4.48
13	0.99	3.62	-0.61	2.44	5.06
14	0.97	3.85	-1.20	2.35	5.50
15	0.91	4.19	-1.39	2.26	5.68
16	0.71	4.45	-1.76	2.19	6.19
17	0.43	4.89	-2.28	2.24	6.42
18	0.54	5.06	-2.81	2.34	7.04
19	0.19	5.63	-3.57	1.96	7.57
20	0.01	5.91	-4.06	2.19	8.10

i	Значения $y_i = y(x)$				
	Вариант 6	Вариант 7	Вариант 8	Вариант 9	Вариант 10
1	2.18	0.16	2.09	2.15	0.04
2	2.43	0 01	2.31	2.41	0.47
3	2.40	0.10	2.72	2.58	0.78
4	2.43	0.05	2.77	2.84	1.01
5	2.65	0.35	2.78	3.28	1.19
6	2.75	0.19	2.97	3.46	1.60
7	2.67	0.50	3.00	4.02	1.93
8	2.66	0.74	3.51	4.11	2.22
9	2.63	1.03	3.43	4.61	2.50

10	2.75	1.06	3.58	5.03	3.01
11	2.41	1.49	3.57	5.34	3.22
12	2.42	1.79	3.54	5.86	3.71
13	2.12	2.03	3.82	6.33	4.23
14	1.74	2.22	3.90	6.81	4.78
15	1.57	2.50	3.77	7.21	5.27
16	1.17	2.88	3.81	7.67	5.75
17	0.96	3.21	4.00	8.23	6.16
18	0.63	3.63	3.97	8.68	6.76
19	0.25	3.90	4.08	9.35	7.30
20	-0.01	3.87	4.08	9.93	8.00

ГЛАВА 7.
ЧИСЛЕННЫЕ МЕТОДЫ ИНТЕГРИРОВАНИЯ ФУНКЦИЙ

Многие практические задачи геометрии, физики, электротехники и др. сводятся к вычислению определенного интеграла вида

$$\Im = \int_a^b f(x)dx, \qquad (9.1)$$

где a и b - нижний и верхний пределы интегрирования; $f(x)$ -непрерывная функция на отрезке $[a,b]$.

Если подынтегральная функция задана в аналитическом виде, то определенный интеграл (9.1) можно вычислить по формуле Ньютона-Лейбница

$$\int_a^b f(x)dx = F(b) - F(a), \qquad (9.2)$$

где $F(x)$ - первообразная подынтегральной функции.

Однако часто не удается воспользоваться формулой (9.2) по следующим причинам: трудно или невозможно найти первообразную подынтегральной функции; подынтегральная функция $f(x)$ задана таблично на конечном множестве точек. x_i , $0 \le i \le n$. В этих случаях обращаются к методам численного интегрирования, которые применительно к одно-кратному интегралу дают квадратурные формулы.

§ 7.1. Классификация методов

Сущность большинства методов вычисления определенных интегралов состоит в замене подынтегральной функции $f(x)$

аппроксимирующей функцией $\phi(x)$, для которой можно легко записать первообразную в элементарных функциях, т.е.

$$\int_a^b f(x)dx = \int_a^b \phi(x)dx + R = S + R, \qquad (9.3)$$

где S - приближенное значение интеграла; R - погрешность вычисления интеграла.

Используемые на практике методы численного интегрирования можно сгруппировать в зависимости от способа аппроксимации подынтегральной функции. Дадим краткую характеристику групп наиболее распространенных методов.

Методы Ньютона-Котеса основаны на полиномиальной аппроксимации подынтегральной функции. Методы этого класса отличаются друг от друга степенью используемого полинома, от которой зависит количество узлов, где необходимо вычислить функцию $f(x)$. Алгоритмы методов просты и легко поддаются программной реализации.

Онлайновые методы базируются на аппроксимации подынтегральной функции сплайнами, представляющими собой кусочный полином. Методы различаются по типу выбранных сплайнов. Такие метода имеет смысл использовать в задачах, где алгоритмы сплайновой аппроксимации применяются для обработки данных.

В методах наивысшей алгебраической точности (методы Гаусса-Кристоффеля и другие) используют неравноотстоящие узлы, расположенные по алгоритму, обеспечивающему минимальную погрешность интегрирования для наиболее сложных функций с заданным количеством узлов. Методы различаются способами выбора узлов и широко используются для интегрирования, в том числе они применимы и для несобственных интегралов.

В методах Монте-Карло узлы выбираются с помощью датчика случайных чисел, ответ носит вероятностный характер. Методы оказываются эффективными при вычислении интегралов большой кратности.

В класс специальных группируются методы, алгоритмы которых разра-батываются на основе учета особенностей конкретных

подынтегральных функций, что позволяет существенно сократить время и уменьшить погрешность вычисления интегралов. Независимо от выбранного метода в процессе численного интегрирования необходимо вычислить приближен-ное значение S интеграла (9.1) и оценить погрешность R в (9.3). Погрешность будет уменьшаться при увеличении количества разбиений N интервала интегрирования [a,b] за счет более точной аппроксимации подынтегральной функции, однако при этом будет возрастать погрешность за счет суммирования частичных интегралов. Это обстоятельство приводит к необходимости разработки способа оценки погрешности выбранного метода интегрирования.

§ 7.2. Методы прямоугольников

Рассмотрим простейшие методы из класса методов Ньютона-Котеса, когда подынтегральную функцию f(x) на интервале интегрирования заменяют полиномом нулевой степени, т.е. константой. Интервал интегрирования разбивается на n отрезков длиной h=(b-a)/n и задача вычисления интеграла на [a,b] преобразуется к виду

$$\int_a^b f(x)dx = \sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} f(x)dx, \qquad (9.4)$$

где $x_o = a, x_n = b.$

Заменим подынтегральную функцию f(x) в пределах элементарного отрезка $[x_i, x_{i+1}]$ интерполяционным полиномом нулевой степени $f(x) \approx f(\xi)$, где f(ξ) – значение подынтегральной функции в точке $\xi \in [x_i, x_{i+1}]$. Приближенное значение интеграла определяется как площадь прямоугольника, одна из сторон которого равна h, а другая есть аппроксимирующая константа. Отсюда происходит и название методов.

Выберем $\xi = (x_i + x_{i+1})/2$ в центре элементарного отрезка (рис. 9.1) и, производя замену подынтегральной функции в (9.4), получим формулу средних прямоугольников.

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} f(\frac{x_i + x_{i+1}}{2})dx = \sum_{i=0}^{n} f(\frac{x_i + x_{i+1}}{2})(x_{i+1} - x_i) = h\sum_{i=0}^{n-1} f(\frac{x_i + x_{i+1}}{2}) =$$
$$= \frac{b-a}{n}\sum_{i=0}^{n-1} f(\frac{x_i + x_{i+1}}{2}). \qquad (9.5)$$

При использовании (9.5) интеграл в пределах элементарного отрезка $[x_i, x_{i+1}]$ приближенно вычисляется как площадь прямоугольника с основанием h и высотой $f(\frac{x_i + x_{i+1}}{2})$ (рис. 9.1). В методах левых (рис.9.2) и правых прямоугольников (рис.9.3)

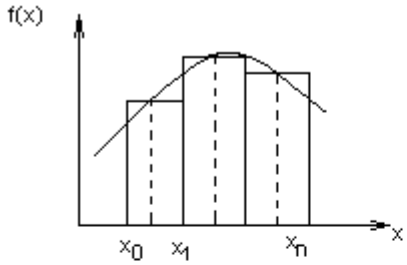


Рис.9.1. Метод средних прямоугольников

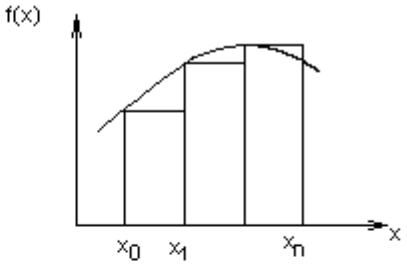


Рис 9.2. Метод левых прямоугольников

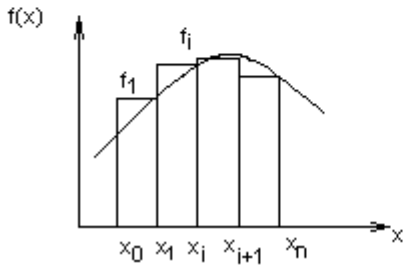


Рис. 9.3. Метод правых прямоугольников

также используется замена подынтегральной функции f(x) интерполирующим полиномом нулевой степени, но положение точки ξ выбирается в одном из концов элементарного отрезка.

Так, выбирая положение точки ξ в левом конце элементарного отрезка $\xi = x1$ (рис. 9.2), получим формулу левых прямоугольников

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x_i)dx = \sum_{i=0}^{n-1} f(x_1)(x_{i+1} - x_i) = f(x_0)h +$$

$$+ f(x_1)h + \ldots + f(x_{n-1})h = \frac{b-a}{n} \sum_{i=0}^{n-1} f_i$$

(9.6)

Если для интерполяции подынтегральной функции в пределах $[x_i,x_{i+1}]$ использовать её значение на правом конце отрезка $\xi = x_{i+1}$,то получим форму-лу правых прямоугольников

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x_{i+1})dx = \sum_{i=0}^{n-1} f(x_{i+1})(x_{i+1} - x_i) = f(x_1)h +$$

$$+ f(x_2)h + \ldots + f(x_n)h = \frac{b-a}{n} \sum_{i=1}^{n} f_i \qquad (9.7)$$

Определим погрешность вычисления определённого интеграла методом средних прямоугольников. Для этого запишем выражение для интеграла на интервале $[x_i,x_i+h]$, полученное методом средних прямоугольников

$$\int_{x_i}^{x_i+h} f(x)dx = hf(\bar{x}) + R, \qquad (9.8)$$

где $\bar{x} = x_1 + h/2, R = \Im_{точн.} - \Im_{прибл.}$, и оценим погрешность R. Для этого разложим подынтегральную функцию в ряд Тейлора около средней точки

$$f(x) = f(\bar{x}) + (x - \bar{x}) f'(\bar{x}) + \frac{(x-\bar{x})^2}{2!} f''(\bar{x}) + \ldots. \qquad (9.9)$$

В малой окрестности точки $\bar{x}$ этот ряд с высокой точностью представляет функцию *f(x)* при небольшом количестве членов разложения. Поэтому, подставляя под интеграл вместо функции *f(x)* её тейлоровское

разложение (9.9) и интегрируя его почленно, можно вычислить интеграл с любой наперёд заданной точностью

$$\Im_{точн.} = hf(\bar{x}) + \frac{(x-\bar{x})^2}{2}\Big|_{x_i}^{x_i+h} f'(\bar{x}) + \frac{(x-\bar{x})^3}{2\cdot 3}\Big|_{x_i}^{x_i+h} f''(\bar{x}) + \ldots =$$
$$= hf(\bar{x}) + \frac{h^3}{24} f''(\bar{x}) + \ldots \qquad (9.10)$$

При интегрировании и подстановке пределов получаем, что все интегра-лы от членов ряда (9.9), содержащих начальные степени $(x-\bar{x})$, обращаются в нуль.

Сравнивая соотношения (9.8) и (9.10), можно записать выражение для погрешности R. При малой величине шага интегрирования h основной вклад в погрешность R будет вносить первое слагаемое, которое называется главным членом погрешности R_{Oi} вычисления интеграла на интервале $[x_i, x_i+h]$

$$R_{Oi} = \frac{h^3}{24} f''(\overline{x_1}). \qquad (9.11)$$

Главный член полной погрешности для интеграла на всём интервале $[x_0, x_n]$ определяется путём суммирования погрешностей на каждом частичном интервале

$$R_0 = \sum_{i=1}^{n} R_{0i} = \frac{h^2}{24}\sum_{i=1}^{n} f''(x) = \frac{h^2}{24}\int_{x_0}^{x_n} f''(x)dx \qquad (9.12)$$

К последнему интегралу мы перешли, используя метод средних прямоугольников для функции *f''(x).*

Формула (9.12) представляет собой теоретическую оценку погрешности вычисления' интеграла методом средних прямоугольников. Эта оценка явля-ется априорной, т.к. не требует знания вычисляемого интеграла. Оценка (9.12) не удобна для практического вычисления погрешности, но полезна для установления структуры главного члена погрешности. Степень шага h, кото-рой пропорциональна величина R , называется порядком метода интегрирования. Метод средних прямоугольников имеет второй порядок.

Аналогично проведем априорную оценку метода левых прямоугольников. Разложим подынтегральную функцию в ряд Тейлора около точки

$x = x_i$

$$f(x) = f(x_i) + (x - x_i) f'(x_i) + \ldots \qquad (9.13)$$

Интегрируя разложение (9.13) почленно на интервале $[x_i, x_i + h]$,получим

$$\Im_{точн} = f(x_{i)h} + \frac{(x - x_i)^2}{2}\Big|$$

где первое слагаемое есть приближенное значение интеграла, вычисленное по методу левых прямоугольников, второе слагаемое является главным членом погрешности

^.1= ~F" /'(*!>■ (9-14)

На интервале [xQ,xnl главный член погрешности интегрирования получим суммированием частичных погрешностей (9.14)

(9.15)

Таким образом, метод левых прямоугольников имеет первый порядок. Кроме того, погрешность будет больше по сравнению с методом средних и за счет интеграла от производной $f'(x)$ и коэффициента в знаменателе (9.15). Обычно для большинства функций выполняется неравенство

$$\left| \int_{x_0}^{x_n} f'(x)dx \right| \geq \left| \int_{x_0}^{x_n} f''(x)dx \right|.$$

Однако, если подынтегральная функция f{x) определяется из эксперимента в дискретном" "наборв_узлов; то метод средних

прямоуголнкков применять нельзя из-за отсутствия значений f(x) в средних точках x_i .

Программу вычисления определенных интегралов любым методом составим в соответствии с блок-схемой рис. 9.4. В качестве примера рассмот-рим интеграл Бесселя

$$J_p(z) = \frac{1}{\pi}\int_0^{\pi} \cos(z\sin x - px)dx \qquad (9.16)$$

функции Бесселя первого рода порядка p от аргумента z. В программе 9.IB в диалоговом режиме (строка 10) задаются значения переменных :N - число разбиений интервала интегрирования; P - порядок функций Бесселя ;Z0,Z9,H1 - границы и шаг изменения аргумента Z. Пределы интегрирования A,B и множитель перед интегралом (9.16) задаются в строке 20 операциями присваивания. В цикле по переменной Z (строки 30-50)^выполняется обращение" к подпрограмме метода численного интегрирования и печать таблицы результатов. В подпрограмме метода средних прямоугольников в строке 100 вычисляется шаг интегрирования H, значение аргумента X полагается равным среднему на первом интервале интегрирования и инициализируется сумма S. В цикле по переменной I (строки ПО- 130) накапливается сумма S значений подынтегральной функции F в средних точках каждого частичного интервала. Так как шаг *H* не изменяется в процессе интегрирования, то на шаг умножается вся накопленная сумма *S* вне цикла (строка 140).

Подынтегральная функция (9.16) вычисляется в подпрограмме, расположенной в строках 200-290.

В программе 9.1F метод средних прямоугольников оформлен в виде подпрограммы RECT, имеющей входные параметры: A,B – пределы интегрирования; N – число разбиений интервала интегрирования; F – имя подпрограммы для вычисления подынтегральной функции и выходной параметр; S – приближенное значение интеграла. Введение имени F в число формальных параметров позволяет составить программы, включающие вычисление интегралов от нескольких различных функций. В вызывающей программе имена всех подпрограмм всех подынтегральных

функций должны быть включены в оператор EXTERNAL. Порядок P и аргумент функции Бесселя передаются в подпрограмму-функцию F(X) как

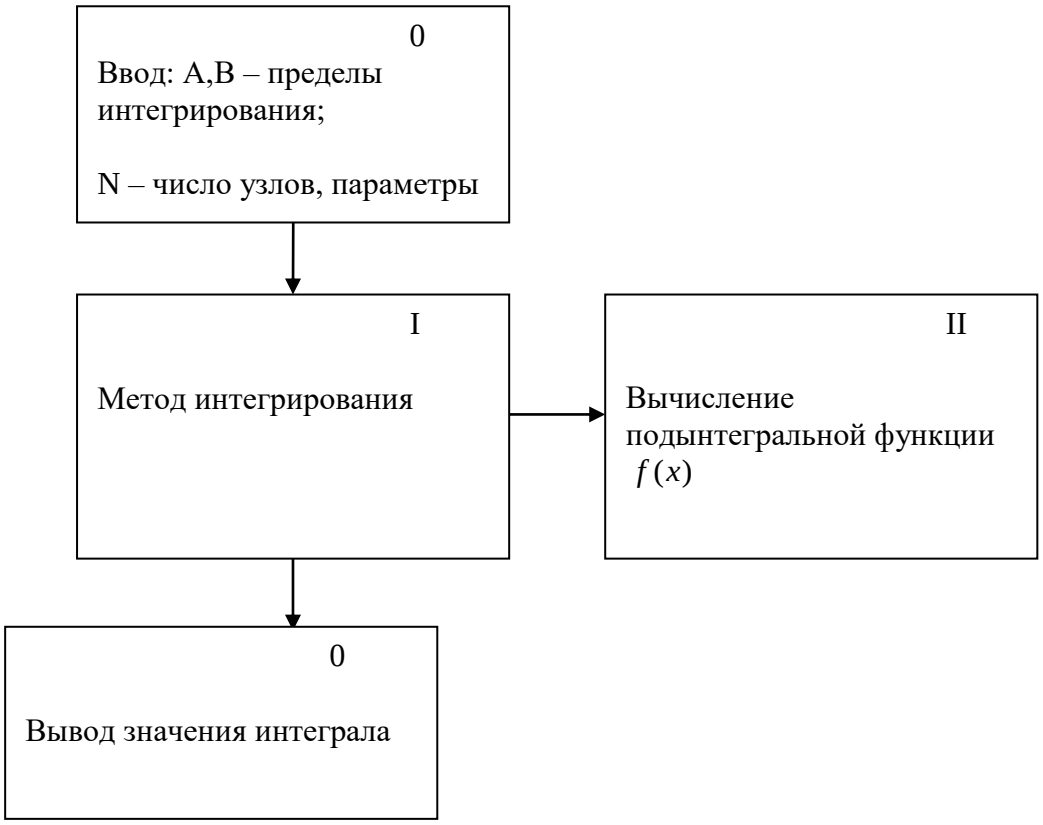


Рис. 9.4. Блок-схема программы численного интегрирования

глобальные из основного блока. Подпрограмма метода средних прямоугольников оформлена в виде процедуры RECT, не имеющей глобальных параметров, формальные параметры имеют тот же смысл, что и в программе 9.1F.

```
1                                                    программа 9.1B
2 rem                                          метод средних прямоугольников
10 print “n,p,z0,z9,h1”; \ input n,p,z0,z9,h1
20 a=0 \ b=pi \ c=1/b
30 for z=z0 to z9 step h1
40 gosub 100 \ rem метод интегрирования
50 print z,c*s \ next z
90 goto 10
100 h=(b-a)/n \ x=a+h/2 \ s=0 \ rem метод прямоугольников
```

```
110 for i=1 to n
120 gosub 200
130 s=s+f \ x=x+h \ next 1
140 s=s*h
190 return
200 f=cos(z*sin(x)-p*x) \ rem подынтегральная функция
290 return
```

программа 9.1F

```
c                                                    метод средних прямоугольников
      external f
      common p,z
1     type *, 'n,p,z0,z9,h?'
      accept *, n,p,z0,z9,h
      b=3.14159265
      c=1./b
      k=(z9-z0)/h+1.5
  z = z0
  do 2 i= 1, k
  call rect (0.,b,n,f,s)      !метод интегрирования
  type  *,z,c*s
2     z = z + h
  go to 1
  end
  subroutine rect (a,b,n,f,s)   ! метод прямоугольников
  h = (b-a)/n
  x = a+h/2
  s = 0.
  do 11 i = 1, n
  s=s + f(x)
11    x = x + h
  s =  s*h
  return
  end
  function f(x)         ! подынтегральная функция
  common p,z
  f = cos(z*sin(x) – p*x)
  return
  end
```

программа 9.IP

(метод средних прямоугольников)

```
Const b = 3.14159265;
var n,i,k: integer; z,z0,z9,h,p,c,s: real;
function f(x:real):real;
begin f:=cos(z*sin(x)-p*x) end;
procedure rect(a,b: real; n:integer;
function f:real; var s:real);
var  f: integer; h,x:real;
begin h:=(b-a)/n; x:=a+h/2; s:=0.0;
for i:=1 to n do begin
s:=s+f(x); x:=x+h; end;
s:=s*h; end;
begin c:=1.0/b;                    (*основная программа*)
repeat wite('n,p,z0,z9,h?');
readln(n,p,z0,z9,h);
k:=round ((z9-z0)/h+1.0); z:=z0;
for i:=1 to k do begin
rect (0.0,b,n,f,s);
writeln(z, ' ',c*s); z:=z+h;
end; until false end.
```

§ 7.3. Метод трапеций

Заменим подынтегральную функцию f(x) на отрезке $[x_1, x_1+h]$ полино-мом первой степени $P_1(x)$

$$f(x) \approx f_i + \frac{\Delta f_i}{h}(x - x_1), \tag{9.17}$$

где f_i и Δf_i - значение и конечная разность первого порядка функции f(x) в точке x_i. Как и в методах прямоугольников, такая аппроксимация неоднозначна. Одним из возможных способов является проведение прямой через значения функции на границах интервала интегрирования (рис. 9.5).

В этом случае приближённое значение интеграла определяется площадью трапеции

$$\int_{x_i}^{x_i+h} f(x)dx = \int_{x_i}^{x_i+h}[(f_i + \frac{\Delta f_i}{n}(x - x_i)]\mathrm{dx} = \mathrm{h}[\mathrm{f}(\mathrm{x_i}) + f(x_i + h)]/2\,. \tag{9.18}$$

На основе (9.18) можно получить приближённую формулу для вычисления интеграла

$$\int_a^b f(x)dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_i+h} f(x)dx \approx \sum_{i=0}^{n-1} h[f(x_i) + f(x_i+h)]/2 = \frac{h}{2}[f_0 + 2(f_1 + f_2 + \ldots + f_{n-1}) + f_n].$$

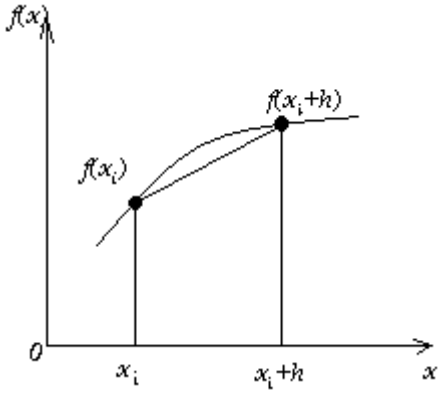


Рис. 9.5. Метод трапеций

В этом случае приближённое значение интеграла определяется площадью трапеции

$$\int_{x_i}^{x_i+h} f(x)dx = \int_{x_i}^{x_i+h} [(f_i + \frac{\Delta f_i}{n}(x - x_i)]\mathrm{dx} = \mathrm{h}[\mathrm{f}(\mathrm{x_i}) + f(x_i+h)]/2. \qquad (9.18)$$

На основе (9.18) можно получить приближённую формулу для вычисления интеграла

$$\int_a^b f(x)dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_i+h} f(x)dx \approx \sum_{i=0}^{n-1} h[f(x_i) + f(x_i+h)]/2 = \frac{h}{2}[f_0 + 2(f_1 + f_2 + \ldots + f_{n-1}) + f_n].$$

(9.19)

Определим погрешность метода трапеций. Запишем выражение для интеграла на интервале $[x_i, x_i + h]$, полученное методом трапеций

$$\int_{x_i}^{x_i+h} f(x)dx = h[f(x_i) + f(x_i + h)]/2 + R. \qquad (9.20)$$

Априорную погрешность R метода трапеций получим путём интегрирования тейлоровского разложения подынтегральной функции около точки x_i:

$$f(x) = f(x_i) + (x - x_i) f'(x_i) + \frac{(x - x_i)^2}{2} f''(x_i) + ... \qquad (9.21)$$

Используя (9.21), определим значение интеграла на интервале $[x_i, x_i + h]$

$$\int_{x_i}^{x_i+h} f(x)dx = hf(x_i) + \frac{h^2}{2} f'(x_i) - \frac{h^3}{2} f''(x_i) + ... \qquad (9.22)$$

С помощью разложения (9.21) вычислим подынтегральную функцию в точке $x_i + h$

$$f(x_i + h) = f(x_i) + hf'(x_i) + \frac{h^3}{2} f''(x_i)...,$$

откуда

$$hf'(x_i) = f(x_i + h) - f(x_i) - h^2 f''(x_i)/2 - ... \qquad (9.23)$$

Подставляя произведение (9.23) в выражение (9.22), получим

$$\int_{x_i}^{x_i+h} f(x)dx = h[f(x_i) + f(x_i + h)]/2 - h^3 f''(x_i)/12 + ...$$

Следовательно, главный член погрешности метода трапеций на одном интервале будет

$$R_{0i} = -h^3 f''(x_i)/12. \qquad (9.24)$$

Если интегрирование проводится путём разбиения отрезка $[x_0, x_n]$ на несколько интервалов, то общую погрешность получим суммированием частичных погрешностей (9.24)

$$R_0 = -\frac{h^2}{12} \int_{x_0}^{x_n} f''(x)dx. \qquad (9.25)$$

Как видно из выражения (9.25), метод трапеций, как и метод средних прямоугольников, имеет второй порядок. Если подынтегральная функция задана аналитически, то предпочтительнее из методов второго порядка применять метод средних прямоугольников вследствие его меньшей погрешности.

Программы 9.2, реализующие метод трапеций, составлены в соответствии с блок-схемой рис. 9.4. и предназначены для вычисления интеграла

$$\Phi(b) = 2/\sqrt{\pi}\int_{o}^{b}\exp(-x^2)dx \qquad (9.26)$$

В основном блоке программы 9.2B в стоке 10 определяется подынтегральная функция с помощью оператора *DEF*, затем в диалоговом режиме (строка 20) вводятся значения переменных: *N* - количество разбиений интервала интегрирования; *B*0, *B*9, *HI* - границы и шаг изменения аргумента *b* интеграла. В строке 30 задаются нижний предел *A*, постоянный множитель *C* интеграла (9.26) и инициализируется переменная *SI* для накопления значений интеграла. В цикле по верхнему пределу интегрирования *B* (сроки 30 - 70) осуществляется обращение к подпрограмме метода численного интегрирования, вычисляется значение интеграла *SI* с переменным верхнем пределом, изменяется нижний предел *A* и выводится на дисплей таблица результатов.

В программах 9.2F и 9.2B подпрограмма метода трапеций *TRAP* имеют только формальные и локальные параметры. Входными параметрами являются переменные *A*, *B* – пределы интегрирования; *N* количество разбиений интервала интегрирования; *F* – имя подпрограммы для вычисления подынтегральной функции. Выходным параметром является переменная *S* – значение интеграла.

```
1                                                    программа 9.2B
2 rem                                                метод трапеций
10 def fna(x)=exp(-x*x) \ rem подынтегральная функция
20 print "n, b0 ,b9 ,h1 "; \ input n, b0, b9, h1
30 c=2/sqr(pi) \ a=0 \ s1=0
40 for b=b0 to b9  step h1
50 gosob 100 \ rem метод интегрирования
60 s1=s1+s \ a=b
70 print b, c*s1 \ next b
```

```
90 goto 10
100 h=(b-a)/n \ s=(fna(a)+fna(b))/2
105 rem метод трапеций
110 for i=1 to n-1 \ s=s+fna(a+i*h) \ next i
120 s=s*h
190 return
```

программа 9.2F
метод трапеций

```
      external f
      c=2/sqrt(3.14159265)
1     type *,'n, b0, b9, h'
      accept *,n, b0, b9, h
      a=0.
      s1=0.
      k=(b9-b0)/h+1.5
      b=b0
      do 2 i=1, k
       call trap(a, b, n, f, s)
      s1=s1+s
      a=b
      type *,b, c*s1
2     b=b+h
      goto 1
      end
      subroutine trap(a, b, n, f, s)
      h=(b-a)/n
      s=(f(a)-f(b))/2
      do 11 i=1, n-1
11    s=s+f(a+i*h)
      s=s*h
      return
      end
      function f(x)
      f=exp(-x*x)
      return
      end
function f(x)
f=exp(-x*x)
return
end
```

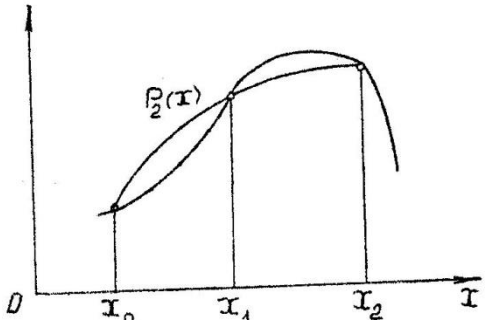
программа 9.2P
(метод трапеций)

```
type
fr=function (x:real):real;
var n,i,k:integer; a,b,b0,b9,h,c,s,s1 :real;
function f(x:real):real;far;
begin f:=exp(-x*x) end;
procedure trap (a,b:real; n:integer;f:fr; var s:real);
var i: Integer; h:real;
begin h:=(b-a)/n; s:=(f(a)+f(b))/2;
for i:=1 to n-1 do s:=s+f(a+i*h);
s:=s*h  end;
begin c:=2/sqrt(3.14159265);
repeat write ('n,b0,b9,h?');
readln(n,b0,b9,h);
k:=round((b9-b0)/h+1.0); b:=b0; a:=0.0; s1:=0.0;
for i:=1 to k do begin trap(a,b,n,f,s); s1:=s1+s; a:=b;
writeln(b,' ' ,c*s1); b:=b+h end
until false
end.
```

7.4. Метод Симпсона

Заменим подынтегральную функцию f(x) интерполяционным полиномом второй степени P_2 (x) - параболой, проходящей через узлы x_o, x_1, x_2 (рис. 9.6), тогда

$$\int_{x_0}^{x_2} f(x)dx = \int_{x_0}^{x_2} P_2(x)dx + R \qquad (9.27)$$



где R - погрешность вычисления интеграла.

Рис. 9.6. Метод Симпсона

Для записи полинома P_2 воспользуемся интерполяционной формулой Ньютона (6.18) для трех узлов (x_i,x_{i+1},x_{i+2}):

$$P_2(x) = f_i + \frac{\Delta f_i}{h}(x - x_i) + \frac{\Delta^2 f_i}{2h^2}(x - x_i)(x - x_{i+1}) \qquad (9.28)$$

Где f_i, Δf_i, $\Delta^2 f_i$ – значение и конечные разности первого и второго порядков функции f(x) в точке x_i.

В пределах отрезка $[x_i, x_{i+2}]$, на котором подынтегральная функция аппроксимирована полиномом (9.28), получим формулу Симпсона

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx \int_{x_i}^{x_{i+2}} \left(f_i + \frac{\Delta f_i}{2h^2}(x - x_i)(x - x_{i+1}) \right) dx = \frac{h}{3}(f_i + +4f_{i+1} + f_{i+2})$$

. (9.29)

Искомый интеграл

$$+2f_2 + 2f_4 + \ldots + 2f_{n-2} + 4f_{n-1} + f_n = \frac{h}{3}[f_0 + 4(f_1 + f_3 + \ldots + f_{n-1}) + \\ + 2(f_2 + f_4 + \ldots + f_{n-2}) + f_n]. \quad (9.30)$$

Формулу Симпсона можно получить и с помощью полинома Ньютона, записанного в виде

$$P_2(x) = f_0 + f_{01}(x - x_0) + f_{012}(x - x_0)(x - x_1), \quad (9.31)$$

где

$f_{01} = (f_0 - f_1)/(x_0 - x_1) = (f_1 - f_0)/h$, $f_{012} = (f_{01} - f_{02})/(x_1 - x_2) = (f_0 - 2f_1 + f_2)/2h^2$ - разделенные разности; h - расстояние между узлами.

Введем новую переменную $z = x - x_0$ тогда $x = z + x_0$ и полином (9.31) принимает вид

$$P_2(z) = f_0 + (f_{01} - f_{012}h)z + f_{012}z^2 \quad (9.32)$$

Теперь вычислим интеграл от полинома (9.32)

$$\int_{x_0}^{x_2} P_2(x)dx = \int_0^{2h} P_2(z)dz = \frac{h}{3}(f_0 + 4f_1 + f_2) \quad (9.33)$$

Для оценки погрешности формулы Симпсона разложим подынтегральную функцию $f(x)$ в ряд Тейлора около точки x_1 и проинтегрируем разложение почленно в интервале $[x_0, x_2]$

$$\int_{x_0}^{x_2} f(x)dx = 2hf_1 + \frac{h^3}{3} f''(x_1) + \frac{h^5}{3 \cdot 4 \cdot 5} f^{IV}(x_1) + O(h^7) \qquad (9.34)$$

Суммируя разложение около точки x_1 для функции $f(x)$ в узлах x_0 и x_1, получим, что

$$h^2 f''(x_1) = f_0 - 2f_1 + f_2 - \frac{h^4}{3 \cdot 4} f^{IV}(x_1)$$

тогда интеграл (9.34) принимает вид

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(f_0 + 4f_1 + f_2) - \frac{2h^5}{180} f^{IV}(x_1) + \ldots \qquad (9.35)$$

Первое слагаемое в правой части формулы (9.35) совпадает с формулой Симпсона, значит, второе слагаемое является главным членом погрешности для интеграла на интервале $[x_0, x_2]$

$$R_{01} = -\frac{2h^5}{180} f^{IV}(x_1) \qquad (9.36)$$

Если интеграл вычисляется на интервале $[x_0, x_n]$ путем разбиения его на четное число подынтегралов $[x_{i-1}, x_i]$, на каждой паре которых применяется формула Симпсона для узлов x_{i-1}, x_i, x_{i+1} то полная погрешность будет суммой правых частей соотношения (9.36). При малой величине шага h на основании метода средних прямоугольников получим

$$\sum_{i=0}^{n} 2hf^{IV}(x_{2i+1}) \approx \int_{x_0}^{x_n} f^{IV}(x)dx$$

тогда полная погрешность запишется в виде

$$R_0 = -\frac{h^4}{180}\int_{x_0}^{x_n} f^{IV}(x)dx \qquad (9.37)$$

Следовательно, формула Симпсона имеет четвертый порядок точности с очень малым численным коэффициентом в остаточном члене. Формула Симпсона позволяет получить высокую точность, если четвертая производная подынтегральной функции не слишком велика. В противном случае методы второго порядка могут дать большую точность, чем метод Симпсона.

В программах 9.3, которые составлены в соответствии с блок-схемой рис. 9.4, реализован метод Симпсона. В качестве примера применения этого метода вычислим таблицу значений полного эллиптического интеграла второго рода.

$$E(z) = \int_0^{\pi/2} \left(1 - z\sin^2 x\right)^{1/2} dx\,. \qquad (9.38)$$

В основном блоке программы 9.3B в диалоговом режиме (строка 10) вводятся переменные: N - количество разбиений интервала интегрирования; z0, z9, HI - граничные значения и шаг изменения аргумента z интеграла (9.38). Пределы интегрирования A и B задаются с помощью операций присваивания (строка 20). В цикле по аргументу z происходит обращение к подпрограмме метода Симпсона и вывод на дисплей таблицы результатов.

В подпрограмме метода Симпсона вычисляется шаг интегрирования H, инициализируется переменная S половинным значением подынтегральной функции на левой границе (строки 100-110). В цикле: по переменной I (строки 120-150) накапливаются значения узловых значений подынтеграль-ной функции с коэффициентами формулы Симпсона. Подпрограмма вычис-ления подынтегральной функции размещается в строках 200-290.

В программах 9.3F и 9.3P подпрограммы метода Симпсона SIMP имеют входные параметры: A,B - пределы интегрирования; N - количество разбиений интервала интегрирования; F - имя подпрограммы вычисления подынтегральной функции и выходной параметр, S - значение интеграла. В программе 9.3F параметр z подынтегральной функции передается из

основной программы через неименованный COMMON-блок. В программе 9.3Р несколько видоизменена реализация алгоритма Симпсона по сравнению с программами на языках Бейсик и Фортран с целью максимального уменьшения операторов в теле цикла.

```
1                                                    программа 9.3В
2    rem                                              метод Симпсона
10  PRINT "n,z0,z9,h1"
20  INPUT n, z0, z9, h1
30  a = 0 \ b = pi / 2
40  FOR z = z0 TO z9 STEP h1
50  GOSUB 100
60  REM метод интегрирования
70  PRINT z, s
80  NEXT z
90  GOTO 10
100 h = (b - a) / (2 * n) \ x = a
110 GOSUB 200
120 REM метод Симпсона
130 s = f / 2
140 FOR i = 1 TO n
150 x = x + h
160 GOSUB 200
170 s = s + 2 * f \ x = x + h
180 GOSUB 200
190 s = s + f
200 NEXT i
210 s = (2 * s - f) * h / 3
220 RETURN
230 f = SIN(x) \ f = SQR(1 - z * f * f)
240 REM подынтегральная функция
250 RETURN
```

```
c                                          программа 9.3 F
c                                          метод Симпсона


      external f
      common z
      1 type *,'n,z0,z9,h?'
      accept *,n,z0,z9,h
      k=(z9-z0)/h+1.5
```

```
      z=z0
      do 2 i=1,k
      call simp(0.0, 3.1415925/2,n,f,s)
      type *,z,s
      2 z=z+h
      goto 1
      end
      subroutine simp (a,b,n,f,s)
      h=(b-a)/(2*n)
      s=f(a)/2
      x=a
      do 11 i=1, n
      x=x+h
      s=s+2*f(x)
      x=x+h
      f1=f(x)
11    s=s+f1
      s=(2*s-f1)*h/3.
      return
      end
      function f(x)
      common z
      f=sin (x)
      f=sqrt (1-z*f*f)
      return
      end
```

программа 9.3 Р
метод Симпсона

```
var n, i, k: integer; z, z0, z9, h, s: real;
function f(x: real): real;
var s: real;
begin s:=sin(x); f:=sqrt(1.0-z*s*s)  end;
procedure simp (a, b: real; n: integer; function f: real; var s: real);
var i: integer; h, h2, x: real;
begin h:=(b-a)/n; h2:=h/2; s:=(f(a)-f(b))/2+2*f(a+h2); x:=a;
for i:=1 to n-1 do begin x:=x+h; s:=s+2*f(x+h2)+f(x) end;
s:=s*h/3.0 end;
begin
repite write ( ‘n, z0, z9, h?‘ );
```

```
readln (n,z0,z9,h);
k=round ((z9-z0)/h+1.0); z:=z0;
for i:=1 to k begin
simpc (0.0, 3.1415925/2,n,f,s);
writeln (z, ' ', s);
z:=z+h;
end;
until false;
end.
```

Варианты практических заданий к главе 7

Вычислить интеграл по формуле трапеций с тремя десятичными знаками.
Вычислить интеграл по формуле Симпсона при $n = 8$

1. $\int_{0.8}^{1.6} \frac{dx}{\sqrt{2x^2+1}}$ 1. $\int_{1.2}^{12} \frac{\lg(x+2)}{x} dx$
2. $\int_{1.2}^{2.7} \frac{dx}{\sqrt{x^2+3.2}}$ 2. $\int_{1.6}^{2.4} (x+1)\sin x dx$
3. $\int_{1}^{2} \frac{dx}{\sqrt{2x^2+1.3}}$ 3. $\int_{0.2}^{1} \frac{tg(x^2)}{x^2+1} dx$
4. $\int_{0.2}^{1.2} \frac{dx}{\sqrt{x^2+1}}$ 4. $\int_{0.6}^{1.4} \frac{\cos x}{x+1} dx$
5. $\int_{0.8}^{1.4} \frac{dx}{\sqrt{2x^2+3}}$ 5. $\int_{0.4}^{1.5} \sqrt{x\cos(x^2)dx}$
6. $\int_{0.4}^{1.2} \frac{dx}{\sqrt{2+0.5x^2}}$ 6. $\int_{0.8}^{1.2} \frac{\sin(2x)}{x^2} dx$
7. $\int_{1.4}^{2.1} \frac{dx}{\sqrt{3x^2-1}}$ 7. $\int_{0.8}^{1.6} \frac{\lg(x^2+1)}{x} dx$
8. $\int_{1.2}^{2.4} \frac{dx}{\sqrt{0.5+x^2}}$ 8. $\int_{0.4}^{1.2} \frac{\cos x}{x+2} dx$
9. $\int_{0.4}^{1.2} \frac{dx}{\sqrt{3+x^2}}$ 9. $\int_{0.4}^{1.2} (2x+0.5)\sin x dx$
10. $\int_{0.6}^{1.5} \frac{dx}{\sqrt{+2x^2}}$ 10. $\int_{0.4}^{0.8} \frac{tg(x^2+0.5)}{1+2x^2} dx$

ГЛАВА 8.
МЕТОДЫ ОДНОМЕРНОЙ ОПТИМИЗАЦИИ.

Важным направлением в проектировании и эксплуатации технологических процессов является оптимизация (минимизация и максимизация) некоторые характеристики $f(x)$. Функцию $f(x)$ часто называют целевой функцией . Заметим, что задача минимизации очеревой функции $f(x)$ может быть сведена к задачи максимизации с помощью введения новой очеревой функции $\bar{f}(x) = -f(x)$. случай, когда варируеся один скалярный параметр x, возникает задача одномерной минимизации.

Необходимость изучения методов решения задачи одномерной минимизации определяется не только тем , что эта задача может иметь самостоятельное значение, но и в значительной мере тем, что алгоритмы одномерной минимизации являются существенной составной частью алгоритмов решения задач многомерной минимизации (см. гл), а также других вычислительных задач.

4.1. Задача одномерной минимизации

Пусть $f(x)$ – действительная функция одного переменного, определенная на множестве $X \in (-\infty, \infty)$. Напомним, что точка $\bar{x} \in x$ называется точ-кой глобального минимума функции f на множестве X, если для всех $x \in X$ выполняется неравенство $f(\bar{x}) \le f(x)$. В этом случае значение $f(\bar{x})$ называет-ся минимальным значением функции *f* на *x*.

Точка $\bar{x} \in X$ называется точкой локального минимума функции *f*, если существует такая δ-окрестность этой точки, что для всех x из множества *X*, содержащихся в указанной δ-окресности, выполняется неравенство $f(\bar{x}) \le f(x)$. Если же для всех таких *x*, не совпадающих с $\bar{x}$, выполняется неравенство $f(\bar{x}) < f(x)$, то $\bar{x}$ называется точкой строгого локального минимума.

Пример 4.1. Для функции, график которой изображен на рис.4.1, точки x_3 и x_4 являются точками строгого локального минимума, а в точках x удовлетворяющих неравенству $x_1 \le x \le x_2$, реализуется нестрогий локальный минимум. Известно, что необходимым условием того, чтобы внутренняя для множества *X* точка $\bar{x}$ была точкой локального минимума дифференцируемой функции *f*, является выполнение равенства

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

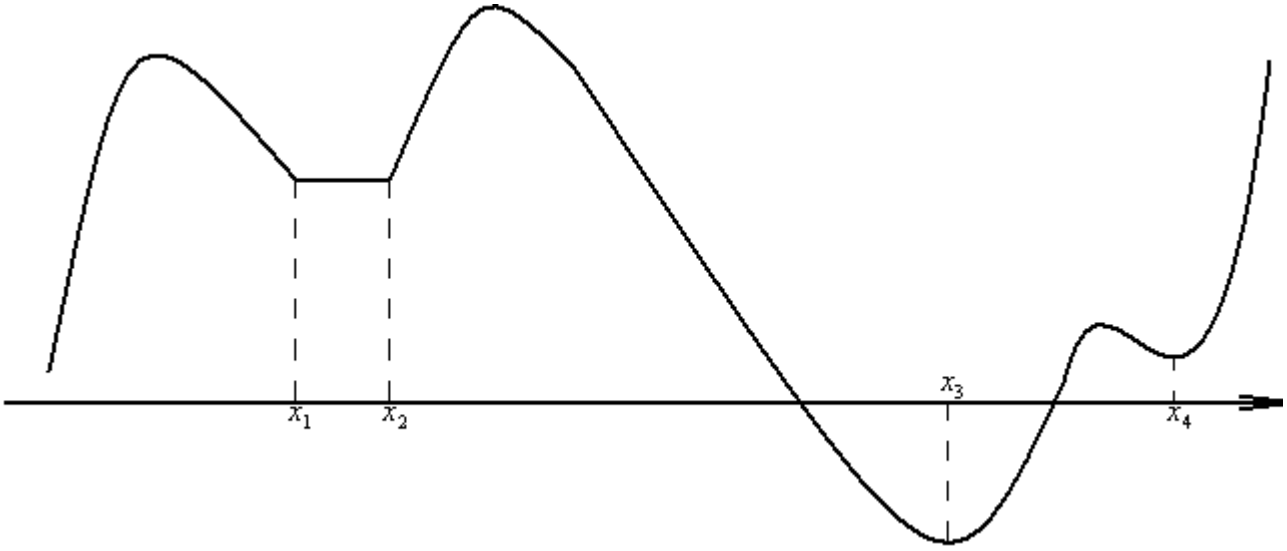


Рис. 4.1. Точки локального минимума

Число $\bar{x}$, удовлетворяющее (4.1), называется стационарной точкой функции *f*. Конечно не всякая стационарная точка $\bar{x}$ обязана быть точкой локального минимума. Для дважды непрерывно дифференцируемой функции достаточным условием того чтобы стационарная точка $\bar{x}$ была точкой строго локального минимума, является выполнение неравенства

$$f''(x) = 0$$

Существуют различные постановки задачи минимизации. В самой широкой постановке требуется найти все точки локального минимума и отвечающие им значения функции *f*. Чаще всё же в приложениях интересуются конкретной точкой локального минимума или точкой глобального минимума. Иногда представляет интерес только минимальное значение функции *f* независимо от того, в какой именно точке оно достигается.

Точно так же методы решения нелинейных уравнений (см.[2, гл.4]) настроены на поиск одного изолированного корня, большинство алгоритмов линеаризации в действительности осуществляют лишь поиск точки локального минимума функции *f*. Для того, чтобы применить один из алгоритмов линеаризации, нужно предварительно найти содержащий

точку $\bar{x}$ отрезка [*a*,*b*], на котором она является единственной точкой локального минимума. Этот отрезок в дальнейшем будет называется отрезком $\bar{x}_2$ - отрезок [0,1].

Докажем теперь строго, что на отрезке [0,1] действительно содержатся точка локального минимума. Для этого заметим, что $f'(x) = 3x^2 - 1 - e^{-x}$ и $f'(0) = -2 < 0$, $f'(1) = 3 - 1 - e^{-1} > 0$. Так как значение $f'(0)$ и $f'(1)$ разных знаков, то на отрезке [0,1] содержится точка $\bar{x}$, для которой $f'(x) = 0$. Но $f''(x) = 6x + e^{-x} > 0$ для всех $x \in [0,1]$. Следовательно, $f''(x) > 0$ и точка $\bar{x}$ на отрезке [0,1] является единственной точкой локального минимума. Аналогичным образом доказывается, что отрезок [-4,-3] так же является отрезком локализации.

Унимодальные функции. Пусть f – определенная на отрезке [a,b] функция. Предположим, что на этом отрезке содержится единственная точка $\bar{x}$ локального минимума функции *f*, причем функция строго убывает при $x \le \bar{x}$ и строго возрастает при $x \ge \bar{x}$. Такая функция называется унимодальной*. Возможны три случая расположения точки $\bar{x}$ на отрезке [a,b]: точка $\bar{x}$ является внутренней для отрезка, $\bar{x}$ совпадает с левым концом отрезка и $\bar{x}$ совпадает с правым концом отрезка. Совершенство и график унимодальной функции может иметь одну из форм локализации* точки $\bar{x}$. К сожалению, не существует каких-либо общих рецептов относительно того, как найти отрезок локализации.

В одномерном случае полезным может быть табулирование функции с достаточно мелким шагом и (или) построение графика. Отрезок [a,b] может быть известен из физических соображений, из опыта решения аналогичных задач и т.д. Для некоторых алгоритмов (например, для метода Ньютона) достаточно имеешь на отрезок локализации, а хорошее начальное приближение $x^{(0)}$ к $\bar{x}$.

Пример 4.2. Для функции $f(x) = x^3 - x + e^{-x}$ произведем локализацию точек локального минимума (рис. 4.2)

Решение. Из графика функции, приведенного на рис. 4.2, ясно видно, что функция $f(x)$ имеет две точки локального минимума $\bar{x}_1$ и $\bar{x}_2$, первая из

.

которых является и точкой глобального минимума. Для точки $\bar{x}_1$ за отрезок локализации можно принять отрезок [-4,-3].

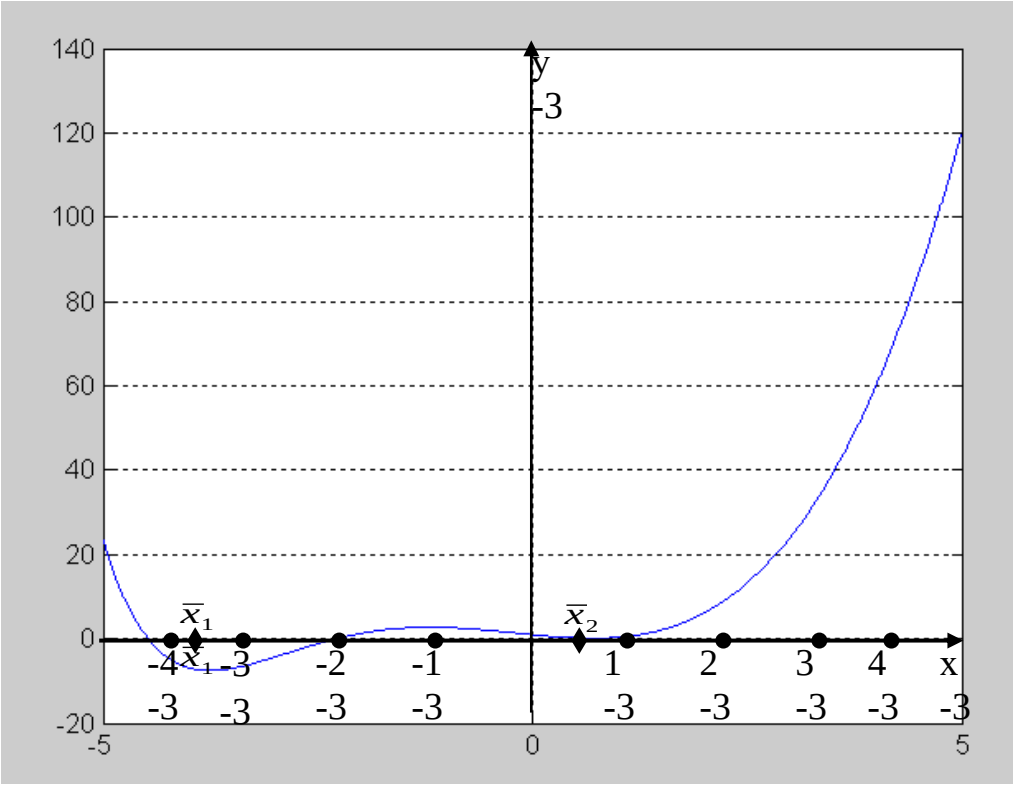


Рис. 4.2. График функции $f(x) = x^3 - x + e^{-x}$

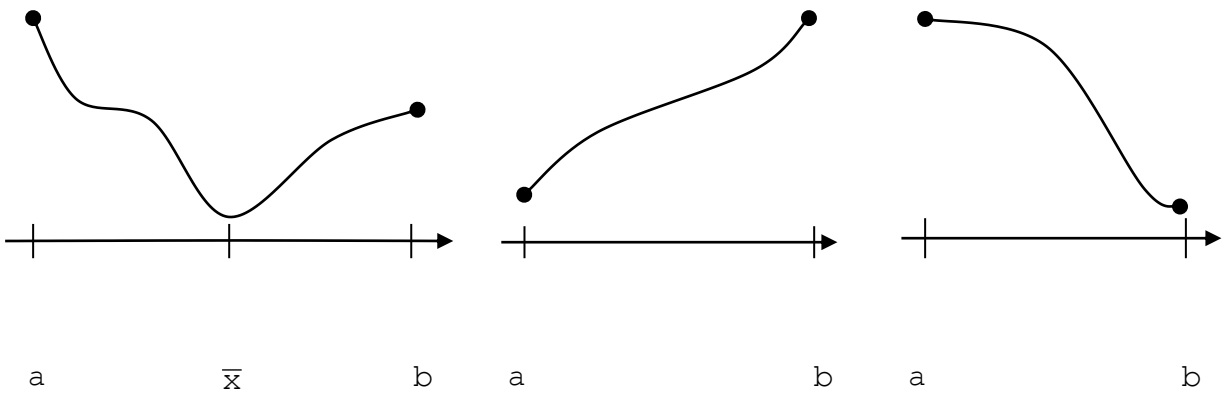


Рис. 4.3. а – унимодальная функция. Случай `a < x̄ < b`; б – унимодальная функция. Случай `x̄ = a`; в – унимодальная функция. Случай `x̄ = b`.

Замечание 4.1. Унимодальная функция, вообще говоря, не обязана быть непрерывной. Например, функция, изображенная на рис 4.4, унимодальна и имеет три точки разрыва.

Приведем достаточное условие унимодальности функции f на отрезке [a,b].

Предложение 4.1. Если для всех $x \in [a, b]$ выполнено условие f''(x)>0, то функция f унимодальна на отрезке [a,b].

Пример 4.3. Функция $f(x) = x^3 - x + e^{-x}$ унимодальна на каждом из отрезков [-4,-3] и [0,1]. Для того чтобы убедиться в этом, достаточно заметить, что $f''(x) = 6x + e^{-x} > 0$ для всех $x \in [-4,-3]$, $x \in [0,1]$, и воспользоваться предложением 4.1.

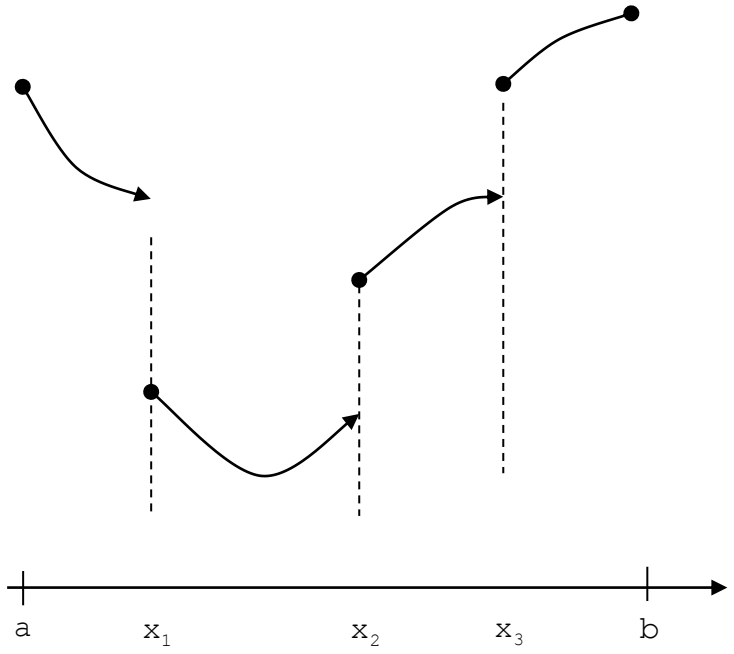


Рис. 4.4. Унимодальная функция имеющая точки разрыва

Приведем достаточное условие унимодальности функции f на отрезке [a,b].

Предложение 4.1. Если для всех $x \in [a, b]$ выполнено условие f''(x)>0, то функция f унимодальна на отрезке [a,b].

Пример 4.3. Функция $f(x) = x^3 - x + e^{-x}$ унимодальна на каждом из отрезков [-4,-3] и [0,1]. Для того чтобы убедиться в этом, достаточно заметить, что $f''(x) = 6x + e^{-x} > 0$ для всех $x \in [-4,-3]$, $x \in [0,1]$, и воспользоваться предложением 4.1.

Для сужения отрезка локализации точки $\bar{x}$ минимума унимодальной функции полезно использовать следующее утверждение.

Предложение 4.2. Пусть f - унимодальна на отрезке [a,b]функция и a ≤ α < γ < β ≤ b. Справедливы следующие утверждения:

а) если f(α) ≤ f(β), то $\bar{x} \in [a, \beta]$

б) если f(α) ≥ f(β), то $\bar{x} \in [\alpha, b]$

в) если f(α) ≥ f(β) и f(γ) ≤ f(β) то $\bar{x} \in [\alpha, \beta]$.

Доказательство. а) Предположим противное: $\bar{x} > \beta$. Тогда в силу уни-модальности f(α) > f(β), что противоречит условию.

б) Предположим противное $\bar{x} < \alpha$. Тогда в силу унимодальности f(α) < f(β), что противоречит условию.

в) По п.а $\bar{x} \in [a, \beta]$, а по п.б $\bar{x} \in [\alpha, b]$. Следовательно, $\bar{x} \in [\alpha, \beta]$.

Предложение доказано Геометрическая иллюстрация п.а и б приведена на рис. 4.5.

Многие алгоритмы минимизации построены в расчете на то, что на отрезке локализации целевая функция унимодальна. В частности, такими являются алгоритмы, рассматриваемые в § 4.3.

4.3. Метод прямого поиска. Оптимальный пассивный поиск.

Метод деления отрезка пополам. Метод Фибоначчи.

Ряд методов минимизации основан на сравнении значений функции f, вычисляемых в точках $x_1, x_2, ..., x_N$. Эти методы часто называют методами прямого поиска, а точки x_i - пробными точками.

Прежде чем перейти к изложению некоторых из наиболее известных методов прямого поиска, уточним постановку задачи минимизации. Будем считать, что требуется найти приближение x^* к точке минимума $\bar{x}$ унимодальной на отрезке $[a,b]$ функции f. Предположим, что разрешается вычислить значение функции в фиксированном числе N пробных точек $x_1, x_2, ..., x_N$ и затем за x^* принять одну из этих точек.

Оптимальный пассивный поиск. Метод решения поставленной задачи, в которой задается правило вычисления сразу всех пробных точек $x_1, x_2, ..., x_N$ и за x^* принимается та точка x_k, для которой $f(x_k) = \min\limits_{1 \le i \le N} f(x_i)$, называется методом пассивного поиска. Соответствующая геометрическая иллюстрация приведена на рис. 4.8.

Оценим погрешность этого метода. Для удобства положим x_0=a, x_{n-1}=b. На основании выбора точки x*=$x_{\hat{e}}$ справедливы неравенства $f(x_{\hat{e}-1}) \geq f(x_{\hat{e}})$ и $f(x_{\hat{e}}) \leq f(x_{\hat{e}+1})$. Поэтому из предложения 1.2в следует, что $\bar{\tilde{o}} \in [x_{\hat{e}-1}, x_{\hat{e}+1}]$.

Следовательно,

$$|\bar{\tilde{o}} - x_i| \leq \max\{x_{\hat{e}} - x_{\hat{e}-1}, x_{\hat{e}+1} - x_{\hat{e}}\},$$

Так как положение точки минимума $\bar{\tilde{o}}$ на отрезке [a,b] заранее неизвестно, то для x*= $x_{\hat{e}}$ справедливо лишь следующая гарантированная оценка погрешности

$$|\bar{\tilde{o}} - x^*| \leq \max_{1 \leq i \leq n+1} |x_i - x_{i-1}|, \qquad (4.7)$$

Можно показать, что величина, стоящая в правой части неравенства (4.7), станет минимальной, если точки $x_1, x_2, x_3, \ldots, x_n$ расположишь на отрезке [a,b] равномерно в соответствии с формулой x_i=a+ih, где h=$\frac{\Delta}{N+1}$, Δ=b-a. Метод с таким выбором пробных точек называется оптимальным пас-сивным поиском. Гарантированная оценка погрешности для него выглядит так:

$$|\bar{\tilde{o}} - x^*| \leq \frac{b-a}{N+1} = \frac{\Delta}{N+1} \qquad (4.8)$$

Пример 4.7. используем оптимальный пассивный поиск для того, чтобы найти с точностью E=0,1 точку $\bar{\tilde{o}}$ локального минимума функции $f(x)=x^3-x+e^{-x}$, локализованную на отрезке [0,1].

Так как минимальное значение достигается в точке $x_7 = 0.7$, то $\bar{x} = 0.7 \pm 0.1$.

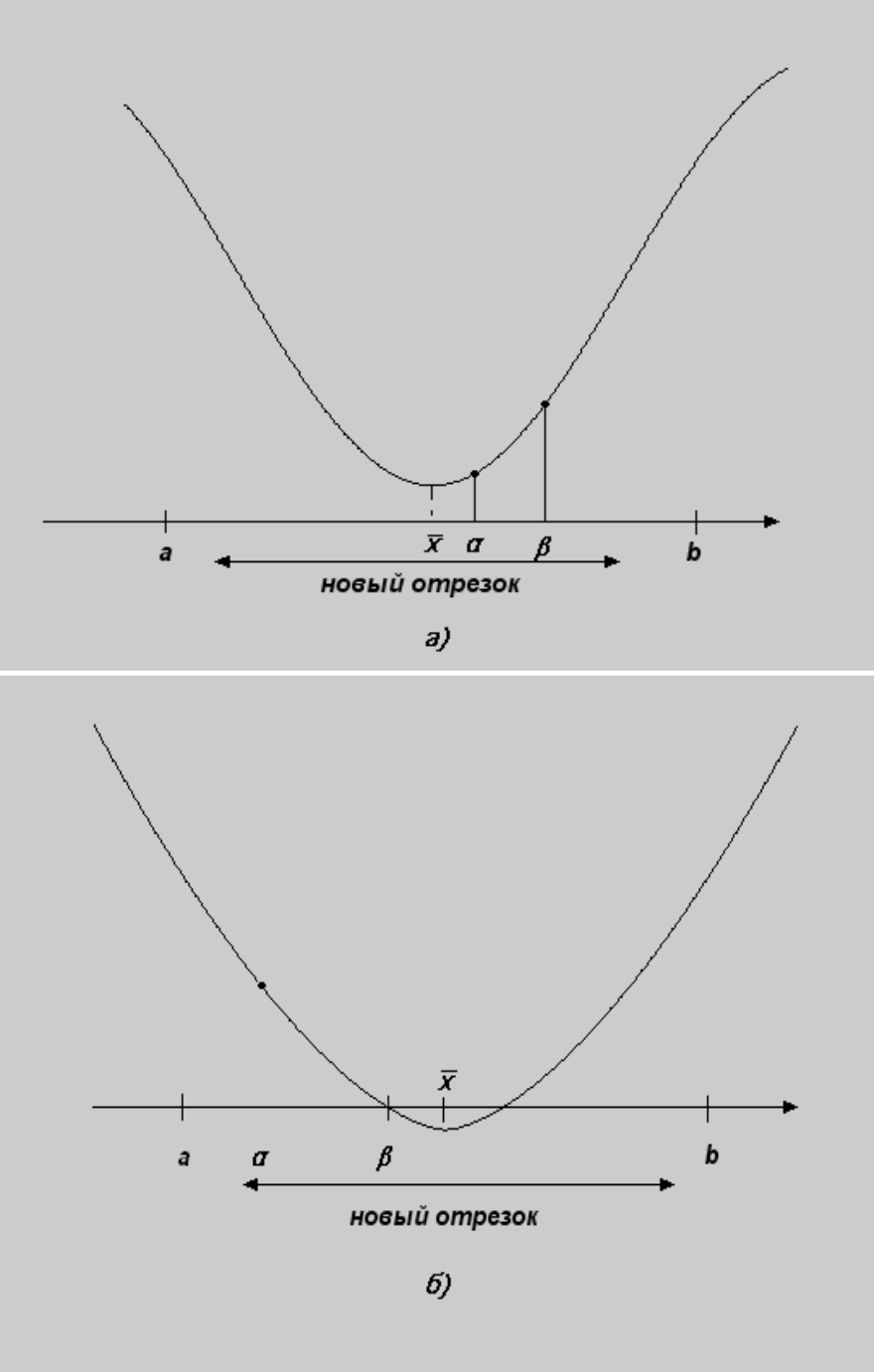


Рис. 4.5: *а* – сокращение отрезка локализации унимодальной функции: $f(\alpha) \leq f(\beta) \Rightarrow \bar{x} \in [a, b]$; *б* – сокращение отрезка локализации функции: $f(\alpha) \geq f(\beta) \Rightarrow \bar{x} \in [a, b]$.

Если бы мы пытались найти $\bar{x}$ с точностью $\varepsilon = 10^{-2}$, то оптимальный пассивный поиск потребовал бы вычисления значений функции уже в 99 точках

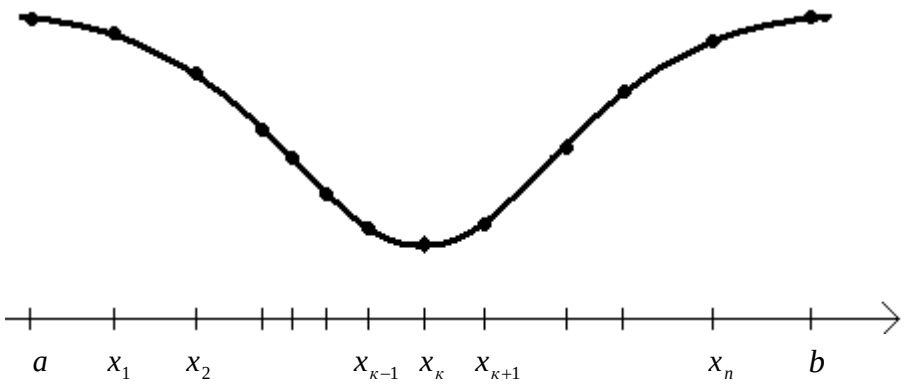


Рис. 4.8. Оптимальный пассивный поиск: $x^* = x_k$

Из формулы (4.8) следует, что для решения задачи потребуется вычислить значения функций в девяти пробных точках вида $x_i = 0.1i$, где $i = 1,2,...9$ (табл. 4.1)

Таблица 4.1

.

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
y	0.81	0.63	0.47	0.33	0.23	0.17	0.14	0.16	0.24

Метод деления отрезка пополам. Пусть для решения поставленной задачи разрешается последовательно вычислять значения функции f в N пробных точках $x_1, x_2, ..., x_N$. Причём для определения каждой из точек x_k можно использовать информацию о значениях функции во всех предыдущих точках $x_1, x_2, ..., x_{k-1}$. Соответствующие методы называют методами последова-тельного поиска. Рассмотрим простейший из методов этого семейства – метод деления отрезков пополам. Он, как и два других из рассматриваемых в этом параграфе методов (методы Фибоначчи и золотого сечения), работает по принципу последовательного сокращения отрезка локализации, основан на предложении 4.2 и опирается на следующий простой факт.

Предложение 4.3. Если функция унимодальна на отрезке $[a,b]$, то она унимодальна и на любом отрезке $[c,d]\subset[a,b]$.

Для удобства изложения обозначим отрезок [a,b] через $\left[a^{(0)},b^{(0)}\right]$. Поиск минимума начинается с выбора на отрезке $\left[a^{(0)},b^{(0)}\right]$ двух симметрично расположенных точек

$\alpha^{(0)}=(a^{(0)}+b^{(0)})/2-\delta,\quad \beta^{(0)}=(a^{(0)}+b^{(0)})/2+\delta.$

Здесь $0<\delta<(b-a)/2;\delta$-параметр метода. Далее вычисляются значения $f(\alpha^{(0)})$ и $f(\beta^{(0)})$. Сравнение этих значений позволяет на основе предложения 4.2 следующим образом сократив отрезок локализации:

1) если $f(\alpha^{(0)})\le f(\beta^{(0)})$, то $\bar{x}\in[a^{(1)},b^{(1)}]=[a^{(0)},\beta^{(0)}]$;
2) если $f(\alpha^{(0)})> f(\beta^{(0)})$, то $\bar{x}\in[a^{(1)},b^{(1)}]=[\alpha^{(0)},b^{(0)}]$.

Если описанную процедуру принять за одну итерацию метода и продолжить аналогичные операции для получения последовательности сокращающихся отрезков локализации, то получится итерационный метод. Опишем очередную его итерацию, исходя из того, что отрезок локализации $[a^{(n)},b^{(n)}]$ уже найден выполняются следующие действия:

1. вычисляются $\alpha^{(n)}=(a^{(n)}+b^{(n)})/2-\delta,\quad \beta^{(n)}=(a^{(n)}+b^{(n)})/2+\delta.$
2. Определяющее значение $f\left(\alpha^{(n)}\right)$ и $f\left(\beta^{(n)}\right)$.
3. Новый отрезок локализации определяется по правилу:

1) если $f\left(\alpha^{(n)}\right)\le f\left(\beta^{(n)}\right)$, то $\left[a^{(n+1)},b^{(n+1)}\right]=\left[a^{(n)},\beta^{(n)}\right]$;
2) если $f\left(\alpha^{(n)}\right)> f\left(\beta^{(n)}\right)$,то $\left[a^{(n+1)},b^{(n+1)}\right]=\left[\alpha^{(n)},\beta^{(n)}\right]$.

В первом случае за очередное приближение к точке минимума принимается $\bar{x}^{(n+1)}=\alpha^{(n)}$, а во втором случае $\bar{x}^{(n+1)}=\beta^{(n)}$. Обозначим через $\Delta^{(n)}=b^{(n)}-a^{(n)}$ длину $\left[a^{(n)},b^{(n)}\right]$. Как нетрудно заметить, справедливо равенство

$$\Delta^{(n+1)}=\Delta^{(n)}/2+\delta. \qquad (4.9)$$

Поэтому, если параметр δ достаточно мал $\left(\delta<<\Delta^{(n)}\right)$, то длина вновь полученного отрезка почти вдвое меньше длины предыдущего отрезка. Отсюда и название метода.

Используя равенство (4.9), можно показать с помощью метода математической индукции, что

$$\Delta^{(n)}=\left(\Delta-2\delta\right)/2^n+2\delta.$$

Заметим, что $\Delta^{(n)}$,монотонно убывая, стремится при $n \to \infty$ к величине 2δ, оставаясь при каждом значении n больше этой величины. Поэтому сделать при некотором n длину $\Delta^{(n)}$ отрезка локализации $\left[a^{(n)}, b^{(n)}\right]$ меньше заданного $\varepsilon > 0$ можно лишь выбрав $\delta < \varepsilon / 2$. В этом случае из очевидной оценки погрешности

$$\left|\bar{x}^{(n)} - \bar{x}\right| < \Delta^{(n)}$$

Следует, что значение $\bar{x}$ можно найти с точностью ε и справедлив следующий критерий окончания итерационного процесса. Вычисления следует прекратить, как только окажется

Таблица 4.2

n	$a^{(n)}$	$b^{(n)}$	$\alpha^{(n)}, f(\alpha^{(n)})$	$\beta^{(n)}, f(\beta^{(n)})$	
0	0.000000	1.000000	0.499000	0.501000	1.000
1	0.499000	1.000000	0.232389	0.230676	0.501
2	0.499000	0.750500	0.748500	0.750500	0.252
3	0.623750	0.750500	0.143924	0.144350	0.125
4	0.687125	0.750500	0.623750	0.625750	0.063
5	0.686125	0.719088	0.154860	0.154131	0.033
6	0.701607	0.719088	0.686125	0.688125	0.017
7	0.701607	0.711348	0.140403	0.140230	0.010
			0.717088	0.719088	
			0.139230	0.139940	
			0.711348	0.703607	
			0.139549	0.139520	
			0.708348	0.711348	
			0.139543	0.139587	

Так как $\Delta^{(7)} \le \xi$, то при $n = 7$ итерации прекращаются и полагаются $\bar{x} \approx \beta^{(n)} = 0.711348$. Таким образом $\bar{x} = 0.71 \pm 0.01$. Заметим, что для достижения точности $\xi = 10^{-2}$ потребовалось 14 вычислений функции.

Метод Фибоначчи Заметим, что метод деления отрезка пополам требует на каждой итерации вычисление двух новых значений, так как найденные на предыдущей итерации в точках $\alpha^{(n)}$ и $\beta^{(n)}$ значения далее не используются. Обратим теперь внимание на то, что одна из этих точек (обозначенная в предыдущем параграфе через $x^{(n)}$) является внутренней для отрезка $[a^{(n)}, b^{(n)}]$ и поэтому дальнейшее сокращение отрезка можно произвести, вычислив дополнительно значение функции лишь в одной новой точке. Это наблюдение приводит к методам, требующим на каждой итерации, кроме первой, расчета лишь одного нового значения функции f. Два наиболее известных среди них – методы Фибоначчи и золотого сечения**.**

Метод Фибоначчи является оптимальным последовательным методом, то есть методом, обеспечивающим максимальное гарантированное сокращение отрезка локализации при заданном числе N вычислений функции. Поиск Фибоначчи базируется на использовании чисел Фибоначчи F_n, задаваемых рекурсивной формулой

$$F_n = F_{n-1} + F_{n-2},\ n \geq 2$$

и начальными значениями $F_0 = 1; F_1 = 1$. Укажем несколько первых чисел: $F_0 = 1;\ \ F_1 = 1;\ \ F_2 = 2;\ \ F_3 = 3;\ \ F_4 = 5;\ \ F_5 = 8;\ \ F_6 = 13;\ \ F_7 = 21;\ \ F_8 = 34;\ \ F_9 = 55;$ $F_{10} = 89; F_{11} = 144$.

Метод Фибоначчи состоит из $N-1$ шагов. Очередной $(n+1)$-й шаг выполняется здесь вполне аналогично $(n+1)$-й итерации метода деления отрезка пополам. В отличие от него точки α^n, β^n вычисляются по формулам

$$\alpha^n = a^n + \frac{F_{N-n-1}}{F_{N-n+1}} \cdot \Delta^{(n)},\quad \beta^n = a^n + \frac{F_{N-n}}{F_{N-n+1}} \Delta^{(n)}.$$

Новый отрезок локализации определяется по тому же правилу:

1.) если $f(\alpha^n) \leq f(\beta^n)$, то $\left[a^{(n+1)}, b^{(n+1)}\right] = \left[a^{(n)}, \beta^{(n)}\right]$;

2.) если $f(\alpha^n) > f(\beta^n)$, то $\left[a^{(n+1)}, b^{(n+1)}\right] = \left[\alpha^{(n)}, b^{(n)}\right]$.

В первом случае за очередное приближение к точке минимума принимается $x^{(n+1)}=\alpha^{(n)}$, а во втором случае $x^{(n+1)}=\beta^{(n)}$.

Важно то, что в любом случае точка $x^{(n+1)}$ совпадает с одной из точек:

$$\alpha^{(n+1)}=a^{(n+1)}+\frac{F_{N-n-2}}{F_{N-n}}\Delta^{(n+1)};\quad \beta^{(n+1)}=a^{(n+1)}+\frac{F_{N-n-1}}{F_{N-n}}\Delta^{(n+1)}.$$

Поэтому на очередном шаге достаточно вычислить значение функции лишь в одной недостающей точке (рис. 4.10).

В результате выполнения $n-1$ шагов отрезок локализации уменьшается в $\frac{F_{N+1}}{2}$ раз, а точка $x^{(n+1)}$ оказывается центральной для последнего отрезка локализации $\left[a^{(N-1)},b^{(n-1)}\right]$, поэтому для $x^{(N-1)}$ справедлива следующая оценка погрешности:

$$\left|\bar{x}-x^{(N-1)}\right|\le\frac{1}{F_{N+1}}(b-a). \qquad (4.11)$$

Варианты практических заданий к главе 8

Графически отделить точку экстремума функции *f(x)*, т.е. найти отрезок *[a,b]*, на котором лежит точка экстремума. Оптимизировать функции, представленные в таблице, методом деления отрезка пополам

Номер варианта	*f(x)*	Характер Экстремума
1	$\frac{x^2}{2}+5\cos x$	min
2	$2e^x-\frac{5}{2}x^2$	min
3	$e^{-x}+x^2$	min
4	$\sqrt{(1+x)x}+\frac{1}{1+x}$	min
5	$\frac{1}{(\sin x+\cos x)^2}+x^2$	min
6	$e^{\lvert\sin x-\cos x\rvert}x$	max
7	$\sqrt{(x-1)}(x+2)^2+x^4$	max
8	$\frac{9+6x-3x^2}{x^2-2x+13}+\frac{1}{1+x^2}$	max
9	$\sqrt{(x-1)(x+2)^2+x^2}$	min
10	$\sin x^2+\sin^2 x$	max

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Амосов А.А., Дубинский Ю.А., Копченова Н.В.* Вычислительные методы решения инженерных задач. М.: МЭИ, 1991.- 236 с.
2. *Бабенко К.И.* Основы вычислительного анализа. – М.: Наука, 1986.- 744 с.
3. *Васильев В.К., Семенова Т.И.* Численные методы решения задач на ЭВМ.- М.: МТУСИ, 1992.- 43 с.
4. *Волков Е.А.* Численные методы.- М.: Наука, 1982.- 256 с.
5. *Данилина Н.И., Дубровская Н.С., Кваша О.П. и др.* Численные методы.- М.: Высш.шк.,1976.- 368 с.
6. *Демидович Б.П., Марон И.А.* Основы вычислительной математики. – М.: Наука, 1970.- 670 с.
7. *Дьяков В.П.* Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ.- М.: Наука, 1987.- 240 с.
8. *Калиткин Н.П.* Численные методы.- М.: Наука, 1987.- 512 с.
9. *Мак-Кракен Д., Дорн У.* Численные методы и программирование на Фортране.- 2-е изд.: Пер. с англ./Под ред. Б.М.Наймарка. – М.: Мир, 1977.- 584 с.
10. *Мудров А.Е.* Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль.- Томск: МП “Раско”, 1991.- 270 с.
11. *Нуссбаумер Г.* Быстрое преобразование Фурье и алгоритмы вычисления сверток: Пер. с англ.- М.: Радио и связь, 1985.- 248 с.
12. *Ортега Д., Рейнболдт В.* Информационные методы решения нелинейных систем со многими неизвестными.- М.: Мир, 1975.- 560 с.
13. *Плис А.И., Сливина Н.А.* Лабораторный практикум по высшей математике.- М.: Высш.шк.,1983.- 208 с.
14. *Малафеев С.И.* Статистический анализ экспериментальных данных/ Владим. политехн. ин-т, Владимир, 1989.- 56 с.
15. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений.- М.: Мир, 1980.- 280 с.
16. *Форсайт Дж., Молер Х.* Численное решение систем линейных алгебраических уравнений. – М.: Мир,1969.- 168 с.
17. *Neri F., Saitta G., Chiofalo S.* An accurate and straightforward approach to line regression analysis of erroraffected experimental da-ta//”Phis. E. Sei. Iustrum.”. 1989, 22, №4, p.215-217.

ОГЛАВЛЕНИЕ

Учебное пособие

ГОРЛОВ Виктор Николаевич
ЕРКОВА Нина Ивановна

МЕТОДЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ ДЛЯ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ. АЛГОРИТМЫ И ПРОГРАММЫ.

Учебное пособие

Подписано в печать 14.12.09
Формат 60x84/16. Усл. печ. л. 8.60. Тираж 100 экз.
Заказ 341-2009 г.

Издательство
Владимирского государственного университета
600000, Владимир, ул. Горького, 87.