МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ТЕОРИЯ И РЕАЛИЗАЦИЯ ЗАДАЧ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ В ПАКЕТЕ MathCad

Рекомендовано Учебно-методическим объединением вузов Российской Федерации по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки 230100 «Информатика и вычислительная техника»

Составители
А.И.Кочегуров, **Е.А. Кочегурова**

Издательство
Томского политехнического университета
2013

УДК 519.6 (075.8)
ББК 22.193я73
В94

В94 **Теория и реализация задач вычислительной математики в пакете MathCad** : учебное пособие / сост. А.И.Кочегуров, Е.А. Кочегурова; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2013. – 135 с.

В учебно-методическом пособии приведены основные теоретические сведения по постановке и методам решения основных задач вычислительной математики.

Пособие подготовлено на кафедре автоматики и компьютерных систем и предназначено для студентов ИДО, обучающихся по направлению 230100 «Информатика и вычислительная техника».

УДК 519.6(075.8)
ББК 22.193я73

Рецензенты

Кандидат технических наук, доцент кафедры
автоматизированных систем управления ТУСУР
А.А. Шелестов

Доктор технических наук, доцент кафедры
вычислительных систем ТУСУР
Р.В.Мещеряков

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Все методы решения математических задач можно разделить на два класса: точные и приближенные. В точных методах решение можно получить в виде аналитического выражения (формулы) но эти методы применимы только для решения ограниченного круга задач.

На практике иногда трудно найти точное решение математической задачи. Поэтому большое значение при решении инженерных задач приобрели *численные методы*, особенно с возрастанием роли ЭВМ. Современная вычислительная техника требует от инженера знаний основ вычислительной математики и применения этих знаний к решению научно-технических задач.

Развитие вычислительной техники позволили исследовать сложные проблемы и явления с помощью соответствующей математической модели. Такой метод исследования назван *вычислительным экспериментом*.

Схема вычислительного эксперимента выглядит следующим образом: формулируются основные законы, управляющие данным объектом исследования, и строится математическая модель. Модель представляет собой запись законов управления объектом в форме системы уравнений (алгебраических, дифференциальных, интегральных и др.).

После получения математической модели необходимо найти ее решение. Но в явном виде это удается сделать лишь в исключительных случаях. Именно на этом этапе вычислительного эксперимента требуется привлечение ЭВМ и, как следствие, использование численных методов. Под численным методом понимается такая интерпретация математической модели (называемая иногда дискретной моделью), которая доступна для реализации на ЭВМ. Иначе говоря, численный метод – это метод (алгоритм) решения задачи, сводящийся к тем действиям, которые выполняет ЭВМ, т.е. к арифметическим и логическим действиям над числами. Таким образом, результатом реализации численного метода является число или массив чисел.

Чтобы реализовать численный метод, нужно составить программу для ЭВМ на том или ином алгоритмическом языке. После этапа отладки программы проводятся собственно вычисления и анализ результатов. Полученные результаты изучаются с точки зрения их соответствия исследуемому явлению. При необходимости вносятся уточнения в математическую модель и изменения в численный метод, после чего описанные выше этапы повторяются.

Таким образом, основу вычислительного эксперимента составляет:

- математическая модель;
- численный метод (алгоритм);
- программа для ЭВМ.

Предметом данного пособия является изучение вопросов, связанных лишь с одним разделом – *численные методы*. Рассматриваются некоторые численные методы решения основных задач линейной алгебры и математического анализа.

Вычислительная математика является одной из интенсивно развивающихся отраслей науки, но при этом она постоянно обогащается новыми практическими применениями.

С развитием вычислительной математики неразрывно связано развитие программирования, которое позволяет упростить способы взаимодействия человека и ЭВМ.

За многие годы накоплены обширные библиотеки научных подпрограмм на различных алгоритмических языках, предназначенных для решения типовых задач вычислительной математики. Кроме того, имеется целый ряд различных математических пакетов, реализующих разнообразные численные методы и производящих аналитические математические преобразования. Наиболее известными сегодня являются пакеты прикладных программ (ППП) и математические библиотеки: MatLab (фирма The MathWorks), Maple (фирма Waterloo Maple Inc), Mathematica (фирма Wolfram Research), MathCAD (фирма MathSoft Inc). Пакет Maple также популярен в научных кругах. Кроме аналитических преобразований, пакет решает задачи численно. Характерной особенностью пакета является то, что он позволяет конвертировать документы в формат LaTeX – стандартный формат подавляющего большинства научных издательств. Кроме того, ряд других программных продуктов используют интегрированный символический процессор Maple. Например, пакет подготовки научных публикаций Scientific WorkPlace (фирма TCI Software Research) позволяет обращаться к символическому процессору Maple, производить аналитические преобразования и встраивать полученные результаты в документ.

Пакет MatLab фактически представляет собой своеобразный язык программирования высокого уровня, ориентированный на решение научных задач. Характерной особенностью пакета является то, что он позволяет сохранять документы в формате языка программирования C.

Пакет Mathematica является сегодня наиболее популярным в научных кругах, особенно среди теоретиков. Пакет предоставляет широкие возможности в проведении символических (аналитических) преобразований, однако требует значительных ресурсов компьютера. Система ко-

манд пакета во многом напоминает обычный язык программирования.

Пакет MathCAD популярен, пожалуй, более в инженерной, чем в научной, среде. Характерной особенностью пакета является использование привычных стандартных математических обозначений, т.е. документ на экране выглядит точно так же, как обычный математический расчет. Для использования пакета не требуется изучать какую-либо систему команд, как, например, в случае пакетов Mathematica или Maple. Пакет ориентирован в первую очередь на проведение численных расчетов, но имеет встроенный символический процессор Maple, что позволяет выполнять аналитические преобразования. В последних версиях предусмотрена возможность создавать связки документов MathCAD с документами MathLab. В отличие от упомянутых выше пакетов, MathCAD является средой визуального программирования, т.е. не требует знания специфического набора команд.

В последнее время просматривается тенденция к сближению и интеграции различных пакетов. Например, последние выпуски пакетов Mathematica и Maple имеют хорошие возможности для визуального программирования; в MatLab включена библиотека аналитических преобразований Maple; MathCAD позволяет работать совместно с MatLab.

Пакет MathCAD наиболее отвечает требованиям вычислительной среды при изучении вычислительной математики: привычные стандартные математические обозначения, среда визуального программирования, возможность символьных вычислений, дружественный интерфейс, относительная непритязательность к возможностям компьютера. Поэтому именно пакет MathCAD был выбран для обучения студентов численным методам.

1. ПОГРЕШНОСТИ ВЫЧИСЛЕНИЙ И АЛГОРИТМОВ

1.1. Задачи вычислительной математики

Решение большинства математических задач возможно в двух видах: аналитическом и численном.

Аналитическими решениями занимается классическая математика (математический анализ, линейная алгебра). Основная задача классической математики – установить существование и единственность решения.

Многие математические задачи невозможно решить, используя аналитические методы; либо решения настолько громоздкие, что их практическое использование невозможно. Численное решение любой задачи, как правило, осуществляется приближенно. Главная задача численных методов (или вычислительной математики) – нахождение решения с требуемой точностью.

В широком смысле вычислительную математику определяют как раздел математики, занимающийся разработкой и исследованием вычислительных алгоритмов и их применением к решению конкретных задач. С развитием возможностей вычислительной техники увеличивается и количество подобных задач. Это, в свою очередь, приводит к развитию самих численных методов.

Реальное проведение любых вычислений проводится над числами, которые задаются не только точно, но и приближенно. Например, запись 7/3 обозначает число, но записать его в виде десятичной дроби можно только приближенно: 2,333… При проведении вычислений на компьютере приближенно приходится записывать большое количество чисел. В результате возникают ошибки, которые постепенно накапливаются и значительно искажают результат.

К настоящему времени все программные средства, благодаря которым на компьютерах проводятся вычисления, имеют возможность регулировать точность проводимых расчетов ***программно***. Можно, например, «поручить» компьютеру вести вычисления с точностью до трех знаков после десятичной запятой, но определение точности результата в этом случае может оказаться сложнейшей математической задачей. Несомненным достоинством численного решения задачи на компьютере является возможность получения решения с требуемой точностью.

Требования к вычислительным (численным) методам можно разделить на две группы. Первая связана с адекватностью дискретной модели

исходной математической задаче. Вторая группа связана с реализуемостью численного метода на ЭВМ.

К первой группе относятся такие требования, как устойчивость, сходимость, корректность численного метода.

Метод и алгоритм, по которому ведутся вычисления, может быть ***устойчивым*** к приближенным числам и может не быть таковым. Слова «устойчивый алгоритм» означают, что чем точнее задаются числа для обработки, тем точнее получается результат, или более строго – устойчивость алгоритма означает, что малым отклонениям в исходных данных соответствуют малые отклонения в результате (решении). Примерами методов, которые не являются устойчивыми к приближенным числам, являются метод последовательного исключения неизвестных для решения систем линейных алгебраических уравнений, методы Рунге – Кутта для решения дифференциальных уравнений и ряд других.

Отсутствие устойчивости (или неустойчивость) означает, что даже незначительные погрешности в исходных данных приводят к большим погрешностям в решении, а зачатую к неверному результату. О таких задачах также говорят, что они ***чувствительны*** к погрешностям исходных данных.

При анализе точности вычислительного процесса одним из важнейших критериев является сходимость численного метода. Она означает близость получаемого численного решения задачи к истинному решению. Под ***сходимостью*** численного метода (алгоритма) понимают способность метода приводить к решению исходной (точному решению) за конечное число шагов, с любой заданной точностью, при любых начальных приближениях.

Сходимость численного метода тесно связана с корректностью. Задача называется поставленной корректно, если для любых значений исходных данных из некоторого класса ее решение существует, единственно и устойчиво по исходным данным.

Иногда при решении корректно поставленной задачи может оказаться неустойчивым метод ее решения. Численный алгоритм (метод) называется ***корректным*** в случае существования и единственности численного решения при любых значениях исходных данных, а также в случае устойчивости этого решения относительно погрешностей исходных данных.

Вторая группа требований, предъявляемых к численным методам, связана с возможностью реализации данной дискретной модели на данной ЭВМ. Алгоритм, реализующий те или иные вычисления, может требовать различное время для своей работы. Чем меньшее время требует алгоритм, тем он имеет более высокое быстродействие. Точно так

же, чем больше компьютерной памяти требуется для реализации алгоритма, тем более высокую сложность по памяти он имеет.

1.2. Элементы теории погрешностей

При численном решении математических и прикладных задач неизбежно появление погрешностей на том или ином этапе решения.

Отклонение истинного решения от приближенного называется ***погрешностью***.

Полная погрешность численного решения складывается из следующих составляющих и приведена на рис. 1.1:

а) неустранимая погрешность – обусловлена погрешностью постановки задачи и неточностью исходных данных;

б) устранимая погрешность, включающая погрешность метода решения задачи и погрешность вычислений.

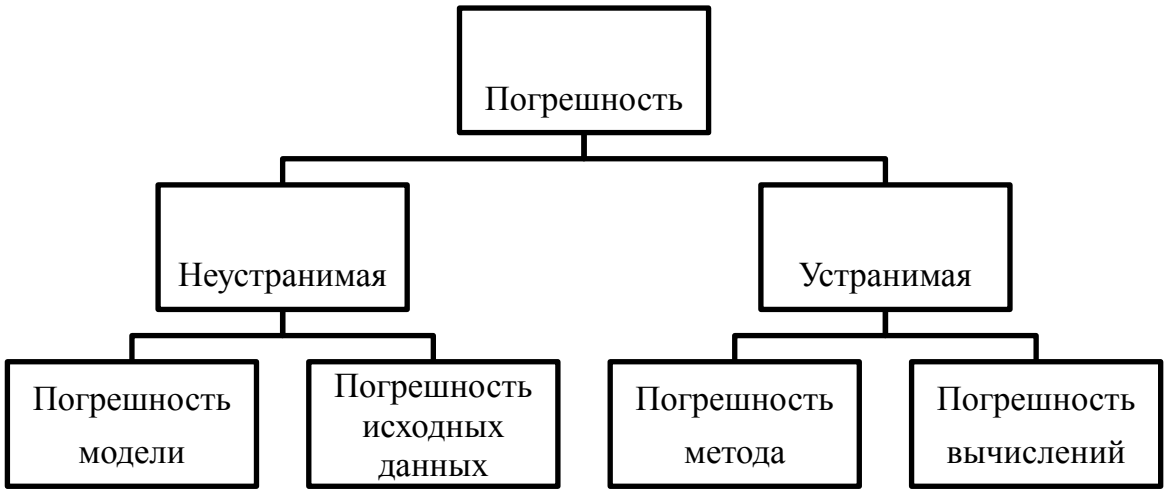


Рис. 1.1. Полная погрешность численного решения

Устранимая погрешность может быть уменьшена выбором более совершенного (точного) метода и увеличением разрядности вычислений.

Приближенным числом X^* называется число, незначительно отличающееся от точного X и заменяющее его в вычислениях.

Характеристиками точности решения задачи являются абсолютная и относительная погрешности.

Пусть X – точное решение задачи, X^* – приближенное решение.

Определение. Абсолютной погрешностью приближенного числа X^* называют величину Δ, которая является ограничением разности

$$\left|X - X^*\right| \le \Delta, \qquad (1.1а)$$

т.е.

$$X^* - \Delta \le X \le X^* + \Delta. \qquad (1.1б)$$

Другая форма записи абсолютной погрешности -

$$X = X^* \pm \Delta. \qquad (1.1в)$$

Форма записи абсолютной погрешности вида (1.1а) и (1.1б) чаще используется в математических оценках погрешности. Вид (1.1в) служит для обозначения погрешности физических систем или измерительных приборов. Например, на шкале вольтметра запись (5±0,01) В означает, что данный измерительный прибор допускает измерения до 5 В с абсолютной погрешностью 0,01 В.

Пример 1.1. В записи числа, подученного в результате измерения, обычно указывают его абсолютную погрешность. Пусть длина некого отрезка L = 10 см при этом точность измерения равна 0,05 см. В этом случае можно записать L = 10 см ± 0,05 см. Тогда абсолютная погрешность Δ = 0,05 см, а точная величина длины L отрезка заключена в гпределах 9,95 см < L < 10,05 см.

Пример 1.2. Найти абсолютную погрешность приближенного числа X^* = 3,141, заменяющего число π.

Решение. Пусть имеется неравенство

$$3{,}141 < \pi < 3{,}142.$$

Тогда $|X^* - X| < 0{,}001$ и, следовательно, можно принять абсолютную погрешность Δ = 0,001.

Рассмотрим другое представление приближенного числа π –

$$3{,}1415 < \pi < 3{,}1416.$$

Тогда абсолютная погрешность будет иметь более лучшую оценку: Δ = 0,0001.

Таким образом, возможно существование нескольких значений абсолютной погрешности, каждое из которых определяется границами приближенного числа.

Абсолютная погрешность не является достаточной характеристикой точности измерения (вычисления). Пусть измеряются длины двух отрезков: $L1$ = 200,35 см ± 0,5 см и $L2$ = 2,03 см ± 0,5 см. Здесь абсолютные погрешности совпадают, но, при этом, качество первого измерения выше, чем второго. Для точности данных измерений существенна абсолютная погрешность, приходящаяся на единицу длины, которая носит название *относительная погрешность*.

Определение. Относительной погрешностью приближенного числа X^* называют величину δ, определяемую выражением

$$\delta = \frac{\Delta}{X^*}, \qquad (1.2а)$$

или другая форма записи:

$$X^*(1 - \delta) \le X \le X^*(1 + \delta). \qquad (1.2б)$$

Более популярно в инженерных и технических приложениях выражение относительной погрешности в процентах.

$$\delta = \frac{\Delta}{X^*} \cdot 100\ \%. \qquad (1.2в)$$

Для большинства инженерных задач относительная погрешность вычисления или измерения 3–5 % считается допустимой и характеризует как вполне удовлетворительный результат. В отдельных задачах допустима более высокая относительная погрешность, порядка 10 %.

Пример 1.3. Пусть число e задано выражением

$$e = 2{,}718 \pm 0{,}001.$$

Требуется найти относительную погрешность вычисления.

Решение. В соответствии с формулой (1.1в)

$$X = e;\ X^* = 2{,}718;\ \Delta = 0{,}001.$$

Используя формулу (1.1в), вычислим относительную погрешность:

$$\delta = \frac{0{,}001}{2{,}718} \cdot 100\ \% \approx 0{,}036\ \%.$$

Пример 1.4. Найти относительные погрешности длин измеренных отрезков:

а) $L1 = 50{,}8$ см ± 0,5 см;

б) $L2 = 3{,}6$ см ± 0,5 см.

Решение

	а)	**б)**
X^*	50,8	3,6
Δ	0,5	0,5
δ	$\delta = \frac{0{,}5}{50{,}8} \cdot 100\ \% \approx 0{,}984\ \%$	$\delta = \frac{0{,}5}{3{,}6} \cdot 100\ \% \approx 13{,}888\ \%$

Таким образом, при одинаковой абсолютной погрешности $\Delta = 0{,}5$, относительные погрешности измерения отрезков $L1$ и $L2$ существенно отличаются. Относительная погрешность $\delta(L1) = 0{,}984$ % невелика. Погрешность измерения второго отрезка $L2$ весьма существенна $\delta(L2) = 13{,}888$ %.

Точность вычислений определяется количеством цифр результата, заслуживающих доверия. Характеристиками доверия к цифрам результата являются значащие, верные и сомнительные цифры.

Определение. Значащими цифрами числа X^* называют все цифры в его записи, начиная с первой ненулевой слева.

Любое число можно представить в виде

$$X^* = \alpha_1 \cdot \beta^n + \alpha_2 \cdot \beta^{n-1} + \ldots + \alpha_m \cdot \beta^{n-m+1}, \qquad (1.3)$$

где α_1 – первая значащая цифра;

β – основание системы счисления (2, 8, 10, 16), или основание позиционной системы;

$0 \le \alpha_i \le \beta$ – числа из базисного набора.

Запись вещественного числа в виде (1.3) называют его представлением в форме с *фиксированной* запятой.

В ЭВМ чаще всего используют представление чисел в форме с *плавающей* запятой, т.е. в виде

$$X^* = M \cdot \beta^p, \qquad (1.4)$$

где β – основание системы счисления;

p – порядок числа (целое число, положительное, отрицательное или нуль);

M – мантисса числа, $\beta^{-1} < M < 1$.

Пример 1.5. Провести разложение числа 2,718 в форме с фиксированной и плавающей запятой:

а) с фиксированной запятой:

$2{,}718 = 2 \cdot 10^0 + 7 \cdot 10^{-1} + 1 \cdot 10^{-2} + 8 \cdot 10^{-3}$;

$\beta = 10$; $\alpha_1 = 2$; $\alpha_2 = 7$; $\alpha_3 = 1$; $\alpha_4 = 8$; $n = 0$;

б) с плавающей запятой:

$2{,}718 = 0{,}2718 \cdot 10^1$;

$M = 0{,}2718$; $p = 1$.

В ЭВМ для записи каждого числа отводится определенное число разрядов, так называемая *разрядная сетка* ЭВМ. Например, 32-разрядная или 48-разрядная сетка ЭВМ. Чем больше разрядов в ЭВМ, тем больше диапазон допустимых чисел и, следовательно, меньше погрешность вычисления. Диапазон допустимых чисел при использовании представления с плавающей запятой значительно больше (на несколько порядков), чем для чисел с фиксированной запятой. Это объясняется тем, что для хранения чисел с фиксированной запятой требуются разряды для записи и целой, и дробной части. А для хранения чисел с плавающей запятой – только разряды под мантиссу и порядок основания.

Например, для 48-разрядной ЭВМ разряды распределены так:

- 1–40 – мантисса;
- знак мантиссы;
- порядок;
- знак порядка.

Шесть разрядов порядка (42–47) позволяют хранить максимальный порядок. В двоичной системе максимальный порядок 111 111 = 63. Мантисса $M < 1$. Отсюда диапазон представления чисел $2^{-63} – 2^{63}$, или в десятичной системе счисления $10^{-19} – 10^{19}$.

Для представления чисел с фиксированной запятой разряды между целой и дробной частью распределены поровну, т.е.

1–24 – дробная часть;

25–47 – целая часть.

Тогда максимальное число, которое можно представить с помощью данной разрядной сетки, будет равно $2^{23} \approx 10^7$.

Следовательно, диапазон допустимых чисел с фиксированной запятой значительно меньше, чем при представлении с плавающей запятой. Этим и объясняется преимущественное использование в ЭВМ чисел в форме с плавающей запятой.

В разложении с фиксированной запятой интерес в дальнейшем будет представлять число n, т.е. показатель степени основания при первой значащей цифре. В примере 1.5 $n = 0$.

Определение. Значащая цифра α_k считается *верной*, если выполняется неравенство

$$\Delta \le \omega \cdot \beta^{n-k+1}, \qquad (1.5)$$

в противном случае α_k – *сомнительная* цифра.

Число ω может принимать любое значение из набора $0{,}5 \le \omega \le 1$. Чаще всего в расчетах его принимают равным 0,75.

Решение. Вычислим значение приближенного числа Z^*, подставив значения входящих в него аргументов *a*, *b*, *c*:

$$Z^* = \frac{0,643 \cdot 2,17^3}{5,843} = 27,1814.$$

Абсолютные погрешности из условия задачи равны

$\Delta a = 0,0005;$

$\Delta b = 0,02;$

$\Delta c = 0,001.$

Относительные погрешности найдем по формуле (1.2):

$\delta a = 7,796 \cdot 10^{-3};$

$\delta b = 0,929;$

$\delta c = 0,171.$

Для вычисления погрешностей сложной функции *Z* воспользуемся формулой из табл. 1.1 для вычисления относительной погрешности произведения и деления. Получим

$$\delta Z = \delta a + \delta(b^3) + \delta(\sqrt{c}).$$

По формулам из таблицы относительных погрешностей приводим к виду, зависящему только от погрешностей входящих аргументов, и вычисляем:

$$\delta Z = \delta a + 3 \cdot \delta(b) + \frac{1}{2} \cdot \delta(c) = 2,8385\ \%.$$

Для вычисления абсолютной погрешности функции *Z* воспользуемся формулой (1.2в)

$$\Delta Z = \frac{\delta \cdot X^*}{100\ \%} = \frac{2,8385 \cdot 27,1814}{100} = 0,7769.$$

Пример 1.8. Вычислить значение аналитического выражения и оценить абсолютную и относительную погрешности сложной функции

$$Z = \frac{a^2 - b^2}{(a+b)^2} + h^2$$

при заданных значениях:

$a = 0,643 \pm 0,0005;$

$b = 2,17 \pm 0,002;$

$h = 5,843 \pm 0,001.$

Решение. Вычислим значение приближенного числа Z^*, подставив

Пример 1.6. Определить число верных знаков в записи числа e = 2,718±0,001.

Решение:

$$X^* = 2{,}718;\ \ n = 0.$$

Абсолютная погрешность $\Delta = 0{,}001$.

Для разрешения неравенства (1.5) удобно представить погрешность в форме числа с плавающей запятой, т.е. $\Delta = 0{,}1 \cdot 10^{-2}$.

Выберем $\omega = 0{,}75$;

$0{,}1 \cdot 10^{-2} \le 0{,}75 \cdot 10^{0-k+1}$.

Неизвестным в этом неравенстве является число k. Равенство оснований (10) и числа мантиссы (0,1< 0,75) позволяют перейти к неравенству относительно показателей:

$-2 < 1 - k$.

Отсюда $k \le 3$.

Таким образом, верными цифрами числа являются три первые значащие цифры, т.е. 2,71. Цифра 8 – сомнительная.

1.3. Общая формула погрешности

Нередко в процессе вычислений известны погрешности отдельных величин, входящих в результат. Общая погрешность результата определяется как величинами отдельных погрешностей, так и видом математического выражения.

Основные правила трансформации погрешностей в процессе вычисления следующие:

- Абсолютная погрешность суммы конечного числа приближенных чисел не превышает суммы абсолютных погрешностей этих чисел.
- Относительная погрешность произведения конечного числа приближенных чисел не превышает суммы относительных погрешностей этих чисел.

Приведенные выше правила лежат в основе оценки погрешности сложной функции по известным погрешностям аргументов этой функции – основной задачи теории погрешностей.

Пусть задана функция $Z = f(x_1, x_2, \ldots, x_n)$ и известны абсолютные (либо относительные) погрешности аргументов $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$ ($\delta x_1, \delta x_2, \ldots, \delta x_n$).

Абсолютная и относительная погрешности самой функции ΔZ, δZ определяются по формулам

$$\Delta Z = \sum_{i=1}^{n} \left| \frac{\partial Z}{\partial x_i} \right| \cdot \Delta x_i ; \tag{1.6}$$

$$\delta z = \sum_{i=1}^{n} \left| x_i \cdot \frac{\partial}{\partial x_i} \ln Z \right| \cdot \delta x_i . \tag{1.7}$$

Если сложная функция Z зависит только от двух аргументов, т.е. $Z = f(x, y)$, то формулы погрешностей имеют на основании (1.6), (1.7) достаточно простой вид. В табл. 1.1 приведены погрешности для наиболее часто встречающихся видов функциональных зависимостей.

Таблица 1.1

Погрешности основных математических операций

Абсолютная погрешность	Относительная погрешность
$\Delta(x \pm y) = \Delta x + \Delta y$	$\delta(x \pm y) = \frac{x \cdot \delta x + y \cdot \delta y}{x \pm y}$
$\Delta(x \cdot y) = x \cdot \Delta y + y \cdot \Delta x$	$\delta(x \cdot y) = \delta x + \delta y$
$\Delta\left(\frac{x}{y}\right) = \frac{y \cdot \Delta x + x \cdot \Delta y}{y^2}$	$\delta\left(\frac{x}{y}\right) = \delta x + \delta y$
$\Delta\left(x^m\right) = m \cdot x^{m-1} \cdot \Delta x$	$\delta\left(x^m\right) = m \cdot \delta x$

Приведенные в таблице формулы позволяют вычислить погрешности практически для любых сложных функций. При этом для одной из погрешностей (абсолютной или относительной), в зависимости от вида функции, существует более простая формула. Другая погрешность может быть вычислена на основе формул (1.1) или (1.2).

Пример 1.7. Вычислить значение аналитического выражения и оценить абсолютную и относительную погрешности сложной функции

$$Z = \frac{a \cdot b^3}{\sqrt{c}}$$

при заданных значениях:

a = 0,643±0,0005;

b = 2,17±0,002;

c = 5,843±0,001.

значения входящих в него аргументов a, b, h:

$$Z^* = 0{,}8307\,.$$

Абсолютные погрешности из условия задачи равны

$$\Delta a = 0{,}04;$$

$$\Delta b = 0{,}02;$$

$$\Delta h = 0{,}01.$$

Относительные погрешности найдем по формуле (1.2)

$$\delta a = 3{,}5057\ \%;$$

$$\delta b = 0{,}6337\ \%;$$

$$\delta h = 0{,}8772\ \%.$$

Так как в данной сложной функции Z^* основная математическая операция сложение, удобнее воспользоваться формулой для вычисления абсолютной погрешности. Предварительно упростим первое слагаемое функции

$$Z = \frac{a-b}{(a+b)} + h^2;$$

$$\Delta Z = \frac{2\cdot a\cdot(\Delta a + \Delta b)}{(a+b)^2} + 2\cdot h\cdot \Delta h\,.$$

Подставив численные значения абсолютных погрешностей, получим

$$\Delta Z = 0{,}043\,.$$

Относительную погрешность функции пересчитаем по формуле (1.2в):

$$\delta Z = 5{,}214\ \%\,.$$

Вопросы для самопроверки

1. Дайте определение погрешности
2. Дайте определение точного и приближенного числа.
3. Приведите определение абсолютной и относительной погрешности.
4. Укажите формы записи абсолютной (относительной) погрешности.
5. Что такое значащие цифры числа?
6. Укажите форму записи числа с фиксированной запятой.
7. Укажите форму записи числа с плавающей запятой.
8. Что такое верные и сомнительные цифры числа?
9. Приведите примеры устранимой и неустранимой погрешности.
10. Укажите основные источники погрешностей.

11. Как можно вычислить верные и сомнительные цифры числа?

12. Что такое значащие цифры числа?

13. Сформулируйте основную задачу теории погрешностей.

14. Дать понятие разрядной сетки ЭВМ.

15. Укажите источники погрешности численного результата.

16. Как влияет способ представления чисел в ЭВМ на точность расчетов?

17. Как найти погрешность сложной функции?

2. ФОРМУЛЫ ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ

В задачах, связанных с анализом, идентификацией, оценкой качества, моделированием различных процессов и устройств возникает необходимость вычисления определенных интегралов.

Часто на практике не удается вычислить интеграл аналитическим путем. В этих случаях применяют приближенные методы численного интегрирования.

Пусть функция $f(x)$ определена и непрерывна на интервале $[a, b]$. Требуется вычислить определенный интеграл

$$J = \int_a^b f(x)\mathrm{d}x .$$

Согласно формуле Ньютона – Лейбница

$$J = \int_a^b f(x)\mathrm{d}x = F(a) - F(b), \qquad (2.1)$$

где $F(x)$ первообразная, т.е. $F/(x) = f(x)$.

Однако нередко первообразная слишком сложна или ее нельзя выразить аналитически, либо функция $f(x)$ задана численно в виде отдельных табличных значений (отсчетов или измерений).

В этом случае, может быть поставлена задача численного (приближенного) интегрирования. Все методы численного интегрирования основаны на геометрическом смысле интеграла Ньютона-Лейбница. Согласно геометрической интерпретации определенный интеграл численно равен площади криволинейной трапеции, ограниченной графиком подынтегральной функции $f(x)$ и осью абсцисс.

В задаче численного интегрирования подынтегральная функция $f(x)$ может быть задана одним из трех способов:

1. В виде табличных значений функции $f_i = f(x_i)$ на сетке значений x_i отрезка$[a,b]$. Часто в таком виде задаются данные, полученные в ходе эксперимента либо измерительная информация, либо показания измерительных приборов.

2. В виде аналитического выражения (формулы), что часто встречается в исследованиях и экспериментах, в том числе и в лабораторных студенческих работах.

3. Функция $f(x)$ не задана явно, но ее значения могут быть переда-

ны из другой программы (подпрограммы) или файла.

При решении задачи численного интегрирования подынтегральная функция $f(x)$, заданная в любом виде, заменяется некоторой более простой функцией, для которой легко найти первообразную. Наиболее популярная функция замены – полиномы различной степени.

Тогда искомое значение интеграла J заменяется приближенным значением интеграла I, численно равного взвешенной сумме значений подынтегральной функции $f(x_i)$, $i = 1, 2, \ldots, n$:

$$I = \sum_{i=0}^{n} A_i \cdot f(x_i), \qquad (2.2)$$

где A_i – числовые коэффициенты;

x_i – узлы сетки, в которых задана или вычисляется функция $f(x)$;

n – число узлов на интервале $[a, b]$.

Формула (2.2) носит название *квадратурной формулы*. Термин «квадратура» означает вычисление площади (или квадрирование) и относится к древнейшей задаче о квадратуре круга. Основными параметрами квадратурных формул являются узлы интегрирования. Их выбором (числом и расположением) на отрезке $[a, b]$ можно регулировать точность вычисления.

Методы этого класса отличаются друг от друга степенью используемого полинома и, соответственно, количеством узлов, в которых необходимо вычислить функцию $f(x)$.

Наиболее популярными на практике являются формулы численного интегрирования для равноотстоящих узлов и при замене подынтегральной функции полиномом. Для этого случая квадратурные формулы носят название Ньютона – Котеса.

А этом случае для узлов x_i: $\{a = x_0 < x_1 < \ldots < x_n = b\}$ выполняются соотношения

$$x_i = x_{i-1} + h;$$

$$x_i = a + i \cdot h;$$

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

Использование неравноотстоящих узлов обеспечивает меньшую погрешность интегрирования. Но по сравнению с методами Ньютона – Котеса такие методы более сложны и требуют большего объема памяти.

К стохастическим методам численного интегрирования относятся методы Монте-Карло, узлы в которых выбираются с помощью датчика случайных чисел. Эффективность методов повышается при увеличении

числа статистических испытаний.

Ниже рассмотрены квадратурные формулы Ньютона – Котеса. На каждом частичном интервале $[x_i, x_{i-1}]$ подынтегральная функция $f(x)$ заменена полиномом определенной степени.

При замене подынтегральной функции на каждом шаге отрезками линий нулевого, первого и второго порядков, могут быть получены следующие приближенные формулы (методы) вычисления интеграла:

- метод прямоугольников;
- метод трапеций;
- метод Симпсона.

Полученные формулы просты для вычисления и программирования, этим и объясняется популярность их использования.

2.1. Метод прямоугольников

Разобьем интервал интегрирования $[a, b]$ на n равных частей. Внутри каждого i-го участка $[x_i, x_{i-1}]$ (частичный интервал) подынтегральная функция $f(x)$ заменяется многочленом 0-й степени $P_0(x)$, т. е. заменяется прямой линией, параллельной оси абсцисс. Подобная замена является неоднозначной, т.к. константу можно выбрать равной значению подынтегральной функции в любой точке частичного интервала.

Прямая линия, заменяющая на интервале подынтегральную функцию, может быть проведена через левую границу интервала $f(x_{i-1})$, правую – $f(x_i)$ либо через среднюю точку $f\left(\frac{x_{i-1}+x_i}{2}\right)$. Отсюда и название методов: левых, правых и средних прямоугольников.

Геометрически значение интеграла соответствует площади криволинейной трапеции, ограниченной осью абсцисс и функцией $f(x)$. В случае метода прямоугольников площадь криволинейной трапеции заменена на каждом частичном интервале площадью прямоугольника. Площадь всей криволинейной трапеции складывается из площадей на отдельных интервалах.

На рис. 2.1, 2.2, 2.3 приведены геометрические иллюстрации метода левых, правых и средних прямоугольников.

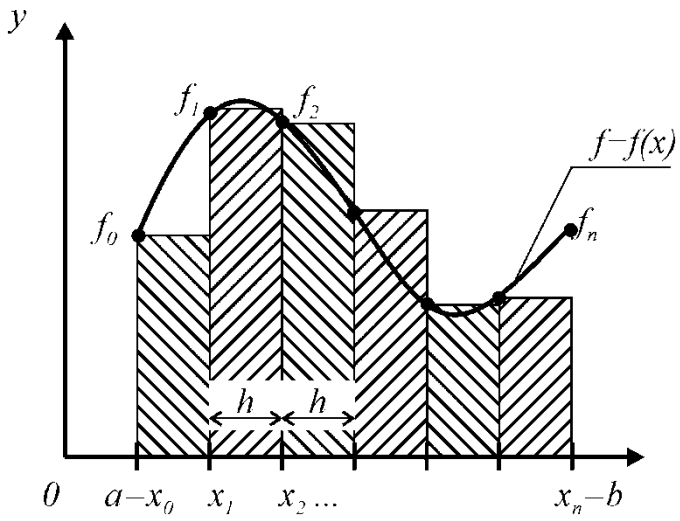


Рис. 2.1. Метод левых прямоугольников

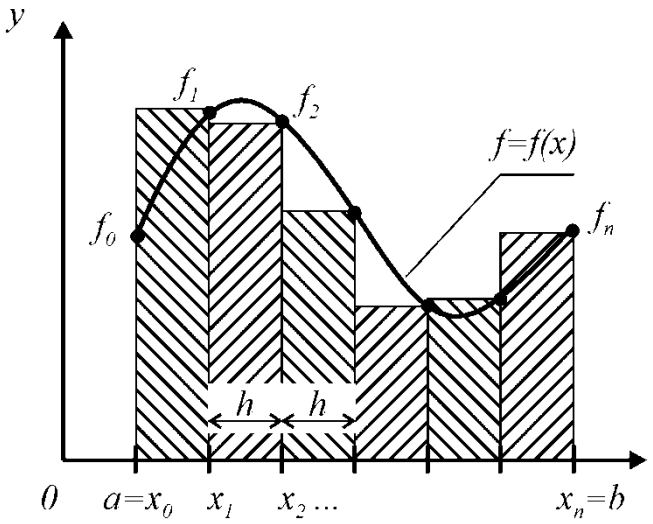


Рис. 2.2. Метод правых прямоугольников

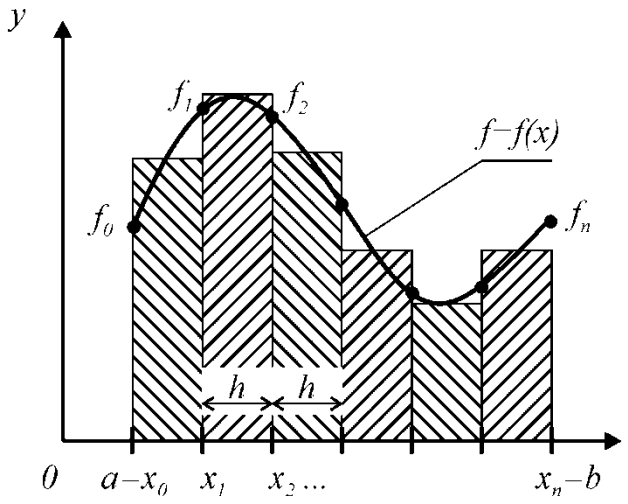


Рис. 2.3. Метод средних прямоугольников

Рассмотрим подробнее формулу средних прямоугольников.

Запишем площади отдельных прямоугольников S1, S2, …, Sn:

$$S1 = h \cdot f\left(\frac{x_0 + x_1}{2}\right);$$

$$S2 = h \cdot f\left(\frac{x_1 + x_2}{2}\right);$$

.....

$$Sn = h \cdot f\left(\frac{x_{n-1} + x_n}{2}\right).$$

Просуммировав площади отдельных прямоугольников, получим значение I, приближенно равное определенному интегралу J:

$$J \approx I = \frac{b-a}{n} \cdot \sum_{i=1}^{n} f\left(\frac{x_{i-1} + x_i}{2}\right). \qquad (2.3)$$

В формуле (2.3) произведена замена значения шага интегрирования $h = (b - a)/n$.

По аналогии запишем формулу левых (2.4) и правых (2.5) прямоугольников:

$$I = \frac{b-a}{n} \cdot \sum_{i=1}^{n} f(x_{i-1}); \qquad (2.4)$$

$$I = \frac{b-a}{n} \cdot \sum_{i=1}^{n} f(x_i). \qquad (2.5)$$

Замечание. Из геометрической иллюстрации видно, что метод средних прямоугольников дает меньшую погрешность. Ниже это будет показано аналитически. Однако метод может быть использован только в случае, если подынтегральная функция $f(x)$ задана аналитически или есть возможность вычисления функции в средней точке.

2.2. Метод трапеций

Если значение подынтегральной функции заменить на каждом частичном интервале $[x_{i-1}, x_i]$ полиномом первой степени $P_1(x)$, то квадратурная формула (2.2) имеет вид

$$I = \frac{b-a}{2 \cdot n} \cdot [f(x_0) + 2 \cdot \sum_{i=1}^{n-1} f(x_i) + f(x_n)]. \qquad (2.6)$$

Площадь криволинейной трапеции заменена в данном методе суммой площадей, прямолинейных трапеций. Как видно из геометрической интерпретации погрешность значительно уменьшается на участках, где подынтегральная функция близка к линейной. Способом уменьшения погрешности является уменьшение шага интегрирования h.

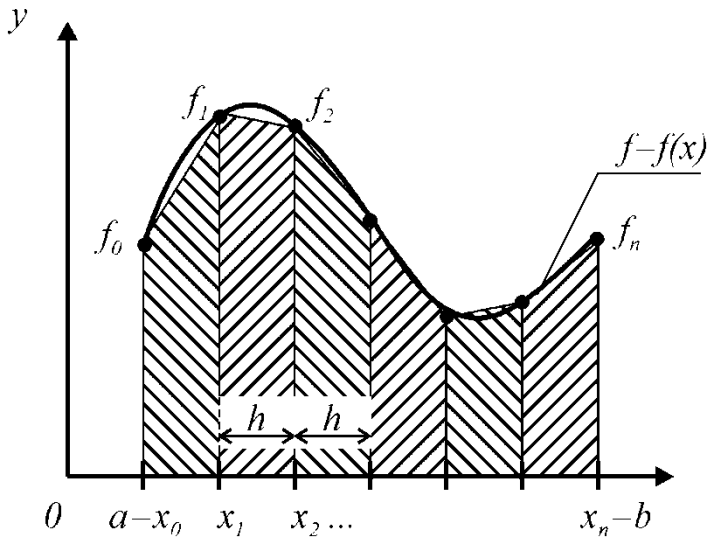


Рис. 2.4. Метод трапеций

2.3. Метод Симпсона

Формула Симпсона (формула парабол) более точная по сравнению с предыдущими методами. Подынтегральная функция заменяется на каждом частичном интервале $[x_{i-1}, x_i]$ полиномом второй степени $P_2(x)$ по трем равноотстоящим узлам.

На рис. 2.5 приведена иллюстрация для единственного интервала $[x_0, x_2]$.

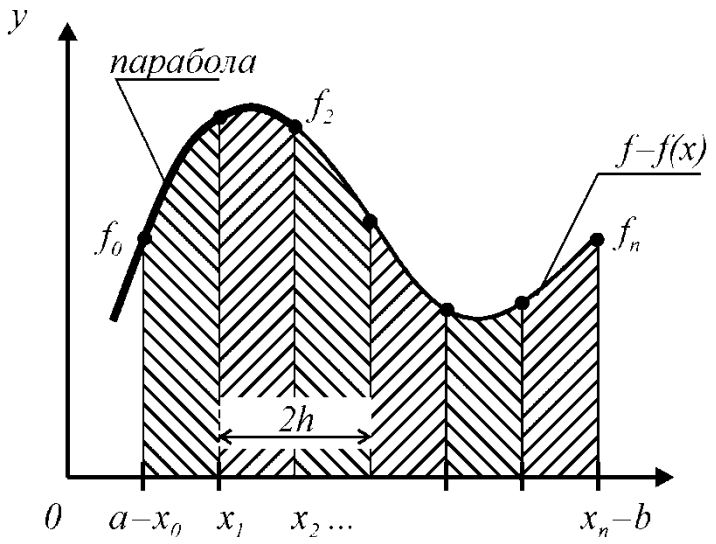


Рис. 2.5. Метод Симпсона

Доказано [2, 14, 24], что на этом интервале площадь криволинейной трапеции равна

$$S1 = \frac{h}{3} \cdot [f(x_0) + 4 \cdot f(x_1) + f(x_2)].$$

Для остальных интервалов формулы площадей аналогичны. Просуммировав все частичные площади, получим формулу Симпсона

$$I = \frac{b-a}{3 \cdot n} \cdot [f(x_0) + 4 \cdot \sum_{i=1,3}^{n-1} f(x_i) + 2 \cdot \sum_{i=2,4}^{n-1} f(x_i) + f(x_n)]. \qquad (2.7)$$

Замечание. Формула Симпсона (2.7) определена только для четного количества узлов n, что вытекает из замены подынтегральной функции параболой.

При нечетном n можно рекомендовать добавить дополнительный узел по какому-либо правилу или произвести суммирование в третьем слагаемом формулы (2.6) до узла (n – 2). Оба способа приводят к потере точности вычисления.

2.4. Оценка погрешности квадратурных формул

Оценка погрешности квадратурных формул может быть осуществлена двумя способами:

- по формуле остаточного члена;
- по принципу Рунге.

Остаточный член возникает вследствие разности между исходной подынтегральной функцией и полиномом $P_n(x)$, заменяющим её. По-

этому его называют остаточным членом аппроксимации. Интуитивно ясно, что чем выше степень полинома, тем точнее замена и меньше остаточный член. Для квадратурной формулы любого порядка k остаточный член определен следующей общей формулой:

$$\left|R_k(h)\right| \approx J - I\,.$$

Запишем оценки точности различных методов интегрирования на основе остаточного члена:

- метод средних прямоугольников:

$$\left|R_0(h)\right| \le \frac{b-a}{24} h^2 M_2; \tag{2.8}$$

- метод трапеций:

$$\left|R_1(h)\right| \le \frac{b-a}{12} h^2 M_2; \tag{2.9}$$

- метод Симпсона:

$$\left|R_2(h)\right| \le \frac{b-a}{180} h^4 M_4\,. \tag{2.10}$$

В формулах (2.8), (2.9), (2.10) величины M_2, M_4 характеризуют оценку производных II, IV степени на интервале:

$$M_2 = \max\left|f''(x)\right|;$$

$$M_4 = \max\left|f^{\mathrm{IV}}(x)\right|;$$

$$x \in [a,b].$$

Формулы (2.8) – (2.10) представляют собой теоретическую оценку погрешности вычисления интеграла. Такую оценку называют доопытной или априорной и она не требует знания вычисляемого интеграла. Для практических оценок погрешности эти формулы неудобны, но полезны для определения:

- порядка метода интегрирования;
- количества точек разбиения интервала [a, b] при вычислении интеграла с заданной точностью ε.

Порядок метода интегрирования однозначно определяется величиной шага h, а точнее, степенью этого шага.

Перепишем формулы (2.8) – (2.10), заменяя величину ($b - a$) значением $h \cdot n$.

Например, для *формулы средних прямоугольников* порядок метода может быть оценен так:

$$\left|R_0(x)\right| \le h^3 \frac{n}{24} M_2 \approx C \cdot h^3 = \mathrm{O}(h^3).$$

В этой формуле величина $n \cdot M2/24$ обозначена константой C и не зависит от шага интегрирования. Поэтому погрешность пропорциональна величине h^3, или, говорят, «порядка h^3».

По аналогии:

- погрешность метода трапеций $\left|R_1(x\right| = \mathrm{O}(h^3)$ имеет также второй порядок;
- погрешность метода Симпсона $\left|R_1(x\right| = \mathrm{O}(h^5)$ – пятый порядок.

***Замечания*:**

1. Чем выше порядок метода, тем меньше погрешность и выше точность вычисления интеграла.

2. Метод средних прямоугольников имеет меньшую погрешность, чем методы левых/правых прямоугольников и даже метод трапеций. Но для его реализации необходимо иметь возможность вычислять функцию в средней точке.

Формула остаточного члена позволяет найти количество точек разбиения интервала $[a, b]$ для вычисления интеграла с заданной точностью ε. Это важнейшая практическая задача. Её решение может быть найдено на основании формул (2.8), (2.9). Рассмотрим решение на конкретном примере.

Пример 2.1. Найти количество точек разбиения n интервала [0, 3] для вычисления интеграла функции $f(x) = \sin(x)$ методом трапеций с точностью $\varepsilon = 10^{-2}$.

Решение. Из условия задачи погрешность не должна превышать заданного значения ε, т.е.

$$\left|R_1(x)\right| \le \varepsilon.$$

Так как выбран метод трапеций, то на основании формулы (2.9)

$$\left|R_1(x)\right| \le \frac{b-a}{12} h^2 M_2 < \varepsilon.$$

Заменив в формуле значение шага

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

получим

$$R_1(x) \le \frac{b-a}{12} \cdot \frac{(b-a)^2}{n^2} \cdot M_2 = \frac{(b-a)^3}{12 \cdot n^2} \cdot M_2.$$

Из этого выражения найдем число разбиений n:

$$n \geq \sqrt{\frac{(b-a)^3 \cdot M_2}{12 \cdot \varepsilon}}.$$

Для вычисления n найдем значение M_2 (максимум модуля второй производной) на интервале [0, 3]:

$$f''(x) = -\sin(x);$$

$$\max\left|f''(x)\right| = 1 -$$

на любом интервале. Отсюда

$$n \geq \sqrt{\frac{3^3 \cdot 1}{12 \cdot 10^{-2}}} = 15,$$

т.е.разбив интервал [0, 3] не менее чем на 15 частей, можно вычислить интеграл от функции $f(x) = \sin(x)$ методом трапеции с точностью 10^{-2}.

На рисунке 2.6 приведен фрагмент программы MathCad для вычисления интеграла по формуле трапеций для заданной точности ε. Число узлов интегрирования выбрано исходя из формулы (2.9). Погрешность интегрирования оценена по формулам абсолютной и относительной погрешности. В качестве точного значения взято значение аналитического интеграла, полученное символьным вычислителем MathCad.

2.5. Принцип Рунге для оценки погрешности интегрирования

Принцип Рунге иногда называют практическим критерием оценки погрешности численного интегрирования. Он позволяет оценить погрешность интегрирования по вычисленным значениям интеграла для шага h и $2h$.

Пусть J – точное значение интеграла;

I – приближенное значение.

Причем I_h – значение интеграла, вычисленное с шагом h.

I_{2h} – с шагом $2h$.

Приближенные значения интегралов содержат погрешности. Оценим погрешность на основе формулы трапеций (2.9)

$$\left|R_1(x)\right| \leq \frac{b-a}{12} h^2 M_2.$$

Численное интегрирование

1 Исходные данные

$f(x) := \frac{1}{x}$ $a := 1$ $b := 3$ $\varepsilon := 10^{-3}$

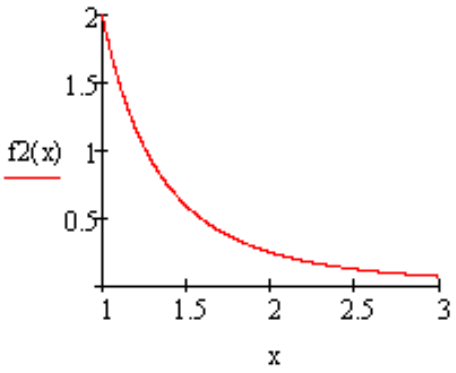
2 Точное значение интеграла

$J := \int_a^b f(x)\,dx$ $J = 1.098612$

3 Расчет числа разбиений интервала [a, b]

$f2(x) := \frac{d^2}{dx^2} f(x) \rightarrow \frac{2}{x^3}$ $M2 := \max(f2(a), f2(b))$ $M2 = 2$

$n := \text{ceil}\left[\sqrt{(b-a)^3 \cdot \frac{M2}{12 \cdot \varepsilon}}\right]$ $n = 37$



4 Сетка узлов

$k := 1..n$ $x_k := a + k \cdot \frac{b-a}{n}$

5 Формула трапеций

$I := \frac{b-a}{2 \cdot n} \cdot \left[f(a) + 2 \cdot \left(\sum_k f(x_k)\right) + f(b)\right]$

$I = 1.116847$

6 Погрешность

$\Delta := |I - J|$ $\Delta = 0.018234$

$\delta := \frac{\Delta}{|I|} \cdot 100$ $\delta = 1.632666$ (%)

Рис. 2.6.

Рассмотрим два случая.

а)	б)
Шаг равен $2h$; число разбиений интервала – n. Умножим формулу на сомножитель n/n: $\lvert R_1(x)\rvert \le \frac{b-a}{12}(2h)^2 M_2 \cdot \frac{n}{n} =$ $= \frac{n \cdot M_2}{12} \cdot \frac{b-a}{n} \cdot (2h)^2 = C \cdot (2h)^3.$ Независимый от шага первый сомножитель обозначен константой C	Шаг равен h; число разбиений интервала – $2n$. Умножим формулу на сомножитель $2n/2n$: $\lvert R_1(x)\rvert \le \frac{b-a}{12}(h)^2 M_2 \cdot \frac{2n}{2n} =$ $= \frac{2n \cdot M_2}{12} \cdot \frac{b-a}{2n} \cdot (h)^2 = 2 \cdot C \cdot h^3.$ Тот же самый сомножитель обозначен также константой C

Тогда

$$J = I_h + 2Ch^3;$$

$$J = I_{2h} + C(2h)^3.$$

Вычтем из первого уравнения второе:

$$J - J = I_h - I_{2h} + 2Ch^3 + C(2h)^3;$$

$$I_h - I_{2h} = C(2h)^3 - 2Ch^3 = (2Ch^3)(2^2 - 1).$$

Величина $(2Ch^3)$ определяет погрешность интеграла, вычисленного с шагом h. Обозначим её r_1 и выразим из последней формулы:

$$r_1 = \frac{I_h - I_{2h}}{3}.$$

Общая формула оценки погрешности, оцененная по принципу Рунге, для метода интегрирования порядка k

$$r_k = \frac{I_h - I_{2h}}{2^{k-1} - 1}, \qquad (2.11)$$

где $k = 3$ – метод трапеций;
$k = 5$ – метод Симпсона.

2.6. Вычисление кратных интегралов

Требуется вычислить двойной (кратный) интеграл

$$J = \iint\limits_D f(x, y)\mathrm{d}x\mathrm{d}y .$$

Область D может иметь произвольную форму. Рассмотрим простейший случай D – прямоугольная область (рис. 2.7).

Перепишем интеграл, разделив его вычисление по аргументам x, y:

$$J = \int\limits_a^b \mathrm{d}x \int\limits_c^d f(x, y)\mathrm{d}y = \int\limits_a^b F(x)\mathrm{d}x .$$

В этой формуле $F(x)$ – первообразная функции $f(x, y)$ по аргументу y:

$$F(x) = \int\limits_c^d f(x, y)\mathrm{d}y .$$

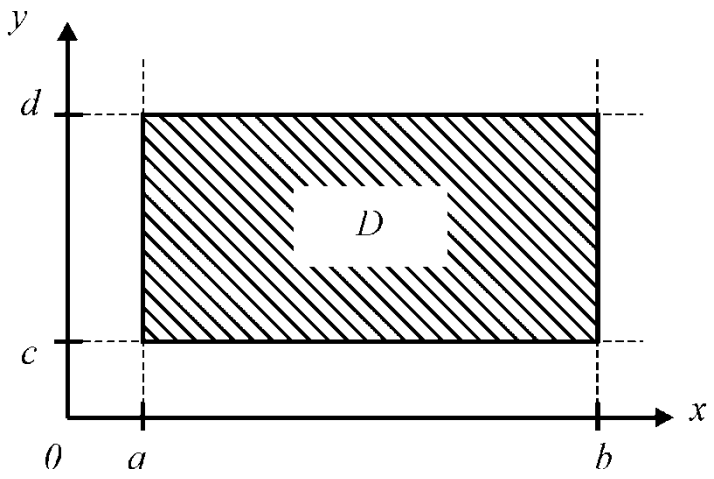


Рис. 2.7

Учитывая введенное обозначение, можно вычислить кратный интеграл по одной из известных квадратурных формул (прямоугольников, трапеций, Симпсона). Запишем для примера вычисление по формуле Симпсона:

$$J = \frac{b-a}{3n} \cdot [F(x_0) + 4F(x_1) + 2F(x_2) + \ldots + F(x_n)],$$

где $F(x_i)$ – $(n + 1)$ – значение интеграла, $i = 0 \ldots n$.

***Замечания*:**

1. Удобство предложенного способа состоит в том, что можно вос-

пользоваться подпрограммами вычисления простых интегралов и применить их (n + 1) раз. Объем вычислений при этом резко возрастает.

2. В случае произвольной (непрямоугольной) области D возможны следующие рекомендации:

а) область интегрирования помещают в прямоугольник и доопределяют функцию $f(x, y)$ так, чтобы вне области D она равнялась нулю. Тогда интеграл по прямоугольнику равен интегралу по области D;

б) использовать метод Монте-Карло, позволяющий заменить вычисление нужной величины вычислением вероятности некоторой эквивалентной приведенной задачи.

2.7. Методы Монте-Карло для вычисления интегралов

Термин Монте-Карло был введен в вычислительной математике во время Второй мировой войны фон Нейманом в связи с моделированием процессов в расщепляемых ядерных материалах. Хотя аналогичная идея была известна в начале XX века, и успешно (вручную) применялась в 30-х гг. Э. Ферма. Другое название метода Монте-Карло – метод статистических испытаний. Основан он на использовании случайных величин, датчики которых реализованы во всех математических пакетах программ.

Существует несколько модификаций этого метода. Рассмотрим две из них.

1. Случайной величиной в методе являются узлы интегрирования. До этого мы рассматривали методы, в которых интервал [a, b] разделялся на n частей и узлы равно отстояли друг от друга. Для примера формула прямоугольников в этом случае имела вид

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{n} \cdot \sum_{i=1}^{n} f(x_i).$$

Если узлы x_i выбрать случайным образом, например с помощью датчика равномерных случайных чисел, то формула прямоугольников будет иметь тот же вид. Погрешность вычисления интеграла в этом случае равна $\frac{1}{\sqrt{n}}$, что выше погрешности рассмотренных ранее методов.

2. В этом случае интеграл приводится к единичному диапазону

$$\int_a^b f(x)\mathrm{d}x = \int_0^1 f(x)\mathrm{d}x$$

и значение самого интеграла $0 \le J \le 1$.

Тогда две случайные величины (x_i, y_i) рассматриваются как координаты точки в единичном квадрате (рис. 2.8).

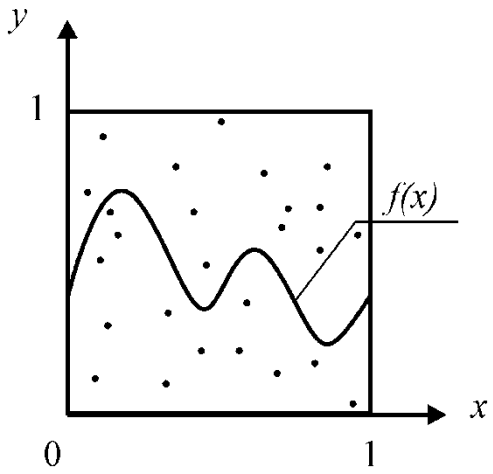


Рис. 2.8. Метод Монте-Карло

Если выбран равномерный закон распределения случайных чисел x_i, y_i, то за приближенное значение интеграла принимается отношение количества точек m, попавших под кривую $y = f(x)$, к общему числу испытаний (случайных чисел) n, т.е. приближенное значение интеграла

$$I = \frac{m}{n}.$$

Ясно, что точность вычисления интеграла повышается с увеличением числа испытаний.

Пример 2.1. Вычислить интеграл от функции $f(x) = x^2$ на интервале [0, 2] с шагом интегрирования $h = 1$ методами интегрирования:

а) средних прямоугольников;

б) трапеций;

в) Симпсона.

Оценить погрешность.

Решение. Вычислим интеграл аналитически:

$$J = \int_0^2 x^2 \mathrm{d}x = \frac{x^3}{3} = \frac{8}{3} \approx 2,666...$$

а) Формула средних прямоугольников –

$$I = \frac{b-a}{n} \cdot \sum_{i=1}^{n} f\left(\frac{x_{i-1} + x_i}{2}\right).$$

Подставим исходные данные в формулу. Получим

$$I == \frac{2-0}{2}\sum_{i=1}^{2}[f(0,5)+f(1,5)] = 1\cdot[0,5^2+1,5^2] = 2,5.$$

Оценим погрешность

$$|R_0(x)| \le \frac{b-a}{24}h^2M_2 :$$

$$M_2 = \max|f''(x)|, \quad f'(x) = 2x, \quad f''(x) = 2, \quad M_2 = 2;$$

$$|R_0(x)| \le \frac{2-0}{24}\cdot 1^2\cdot 2 = \frac{1}{6} = 0,166...$$

Полученная погрешность полностью согласуется с аналитическим результатом.

б) Формула трапеций –

$$I = \frac{b-a}{2\cdot n}\cdot[f(x_0)+2\cdot\sum_{i=1}^{n-1}f(x_i)+f(x_n)].$$

Подставим исходные данные в формулу. Получим

$$I = \frac{2-0}{2\cdot 2}\cdot[f(0)+2\cdot f(1)+f(2)] = \frac{1}{2}\cdot[0+2+4] = 3.$$

Погрешность

$$|R_1(x)| \le \frac{b-a}{12}h^2M_2 = \frac{2-0}{12}1^2\cdot 2 = \frac{1}{3} = 0,333...$$

в) Формула Симпсона –

$$I = \frac{b-a}{3\cdot n}\cdot[f(x_0)+4\cdot\sum_{i=1,3}^{n-1}f(x_i)+2\cdot\sum_{i=2,4}^{n-1}f(x_i)+f(x_n)].$$

Так как в примере всего три узла, то, исключая крайние узлы, в формуле будет присутствовать только слагаемое по нечетному узлу (с коэффициентом 4):

$$I = \frac{2-0}{3\cdot 2}\cdot[f(0)+4\cdot f(1)+f(2)] = \frac{1}{3}\cdot[0+4+2^2] = \frac{8}{3} = 2,666...,$$

То есть для данного примера формула Симпсона дает полное совпадение с аналитическим результатом. Проверим это с помощью фор-

мулы остаточного члена

$$\left|R_2(x)\right| \le \frac{b-a}{180} h^4 M_4,$$

где $M_4 = \max\left|f^{\mathrm{IV}}(x)\right|$, $f'(x) = 2x$, $f''(x) = 2$, $f''' = 0$;

$M_4 = 0$;

$\left|R_2(x)\right| = 0$.

Вопросы для самопроверки

1. В каком случае используется численное интегрирование?
2. Дайте понятие первообразной функции.
3. Приведите постановка задачи численного интегрирования.
4. Приведите понятие сетки узлов, шага сетки, частичного интервала.
5. Какие существуют методы интегрирования функций?
6. В чем состоит основная идея численного интегрирования?
7. Приведите графическую интерпретацию методов прямоугольников, трапеций и Симпсона.
8. Как оценить погрешность методов прямоугольников, трапеций и Симпсона?
9. Укажите оценку погрешности интегрирования по формуле остаточного члена.
10. Чем отличаются формулы метода трапеций и метода парабол?
11. Каковы преимущества имеют формулы парабол по сравнению с другими формулами численного интегрирования?
12. В чем состоит принцип Рунге для оценки погрешности численного интегрирования?
13. Как влияет на точность численного интегрирования величина шага?
14. Верны ли формулы Симпсона и парабол для неравноотстоящих узлов?
15. В каких случаях приближенные формулы трапеций и Симпсона оказываются точными?
16. Можно ли добиться неограниченного уменьшения погрешности интегрирования путем последовательного уменьшения шага?

3. МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Решение нелинейных уравнений является не только самостоятельной задачей, но и частью других задач вычислительной математики (нелинейных дифференциальных уравнений, нахождения собственных значений матриц и т.д.). Также с решением нелинейных уравнений связано построение различных моделей устройств и систем автоматики.

Любое уравнение в общем случае можно представить в виде

$$f(x) = 0. \qquad (3.1)$$

На рис. 3.1 представлена общая классификация уравнений в зависимости от их числа, предполагаемого характера и числа решений.

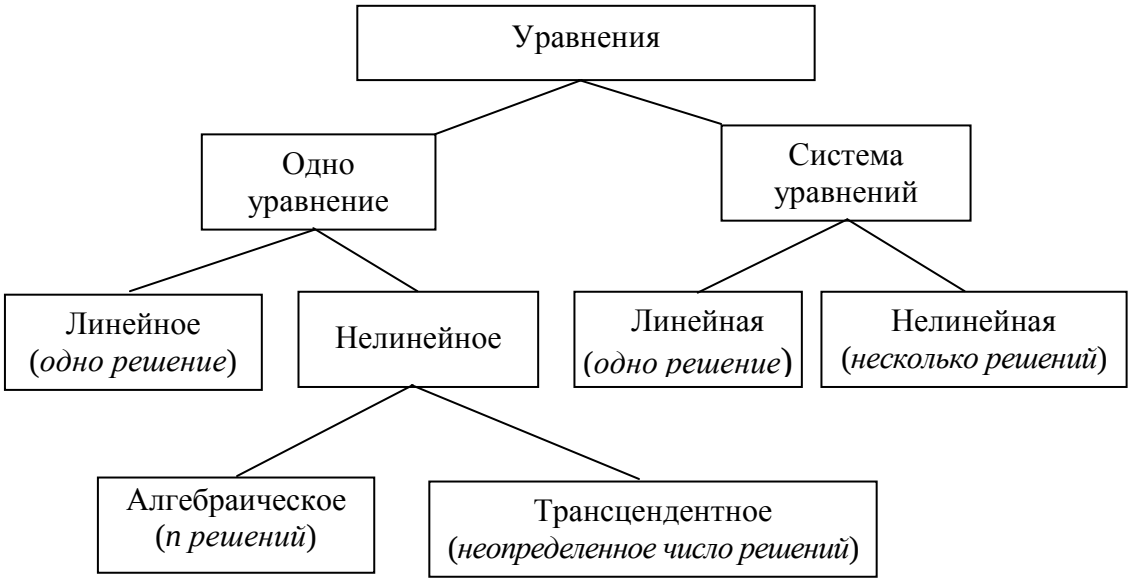


Рис. 3.1. Классификация уравнений и систем уравнений

Нелинейные уравнения включают два типа уравнений – алгебраические и трансцендентные.

Алгебраическими уравнениями называются уравнения, содержащие только алгебраические функции (целые, рациональные, иррациональные).

Алгебраическое уравнение можно представить многочленом n-й степени с действительными коэффициентами:

$$f(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_n.$$

Трансцендентными называются уравнения, содержащие специальные функции (логарифмические, показательные, тригонометрические и т.д.). Например:

$$f(x) = e^{-x} - x;$$
$$y(x) = \sin x - x + \pi.$$

Пусть задана непрерывная на интервале $[a,b]$ функция $f(x)$. Требуется найти корни уравнения $f(x) = 0$.

Корнем уравнения (или его нулем) называют всякое значение $x = \xi$, обращающее функцию $f(x)$ в тождество, т.е. $f(\xi) = 0$.

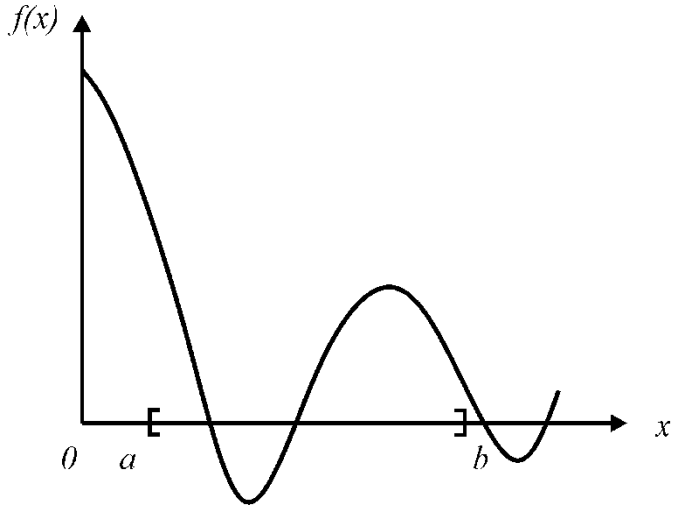


Рис.3.2

Методы решения нелинейных уравнений делятся на прямые и итерационные. **Прямые** методы позволяют найти корни уравнения в виде аналитического выражения (формулы).

Известны точные (прямые) методы решения полиномов 2–3 степени, ряда тригонометрических и логарифмических уравнений. Однако встречающиеся на практике уравнения не всегда удаётся решить простыми методами. Для их решения используются **итерационные** методы, т.е. методы последовательных приближений.

Приближенное нахождение корней нелинейных уравнений разделяется на ряд этапов:

1. Отделение корней, т.е. нахождение интервала $[\alpha, \beta]$, внутри которого находится только один требуемый корень.

2. Уточнение корней, т.е. вычисление корней с заданной точностью.

3.1. Отделение корней

Приближенное значение корня может быть найдено различными способами.

Графический метод отделения корней

Пусть требуется отделить корни нелинейного уравнения (3.1). Построим график функции $f(x) = 0$. Действительные корни этого уравнения

приближенно можно определить как абсциссы точек пересечения графика функции $f(x)$ с осью OX.

Часто на практике удобно уравнение (3.1) преобразовывать к более простому виду. Допустим, что

$$f(x) = \varphi_1(x) - \varphi_2(x) = 0,$$

или

$$\varphi_1(x) = \varphi_2(x),$$

где функции $\varphi_1(x)$ и $\varphi_2(x)$ более просты для построения, чем функция $f(x)$.

Строим графики функций $\varphi_1(x)$ и $\varphi_2(x)$. Корнями уравнения будут абсциссы точек пересечения этих графиков.

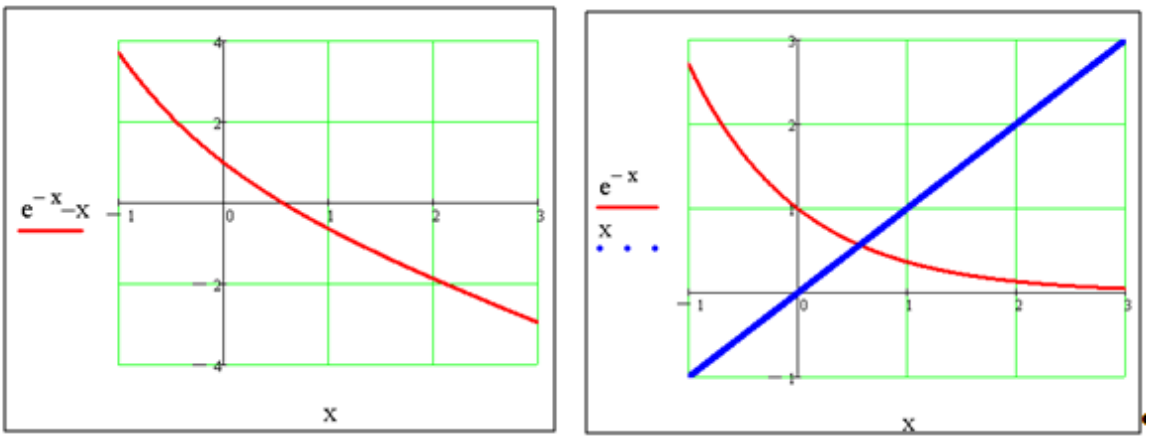
Пример 3.1. Отделить корни уравнения $f(x) = e^{-x} - x$.

Решение. Преобразуем $f(x)$ к виду

$$e^{-x} = x.$$

Построим графики функций $f_1(x) = e^{-x}$ и $f_2(x) = x$ (рис. 3.3, *а*). Абсцисса точки пересечения графиков функций $f_1(x)$ и $f_2(x)$ – решение исходного нелинейного уравнения.

На рис. 3.3, *б* приведен график исходной функции $f(x) = e^{-x} - x$. Отметим, что для рис. 3.3, *а*) и *б)* корни одинаковы и равны ≈ 0,567.



а) б)

Рис. 3.3. Графический метод отделения корней

Аналитический метод отделения корней

Этот метод отделения корней основан на теореме интервалов.

Теорема. Если непрерывная функция $f(x)$ на концах отрезка $[a,b]$

принимает значения разных знаков, то внутри интервала $[a,b]$ существует хотя бы один действительный корень уравнения $f(x) = 0$, т.е. существует такое число $x^* \in [a,b]$, что $f(x^*) = 0$.

При этом если на заданном отрезке $[a,b]$ существует первая производная $f'(x)$, сохраняющая знак внутри $[a,b]$ ($f(x) > 0$ или $f(x) < 0$), то корень x^* будет единственным.

На рис. 3.4 приведены иллюстрации теоремы интервалов. В обоих случаях на концах интервала $[a, b]$ функция $f(x)$ имеет разные знаки.

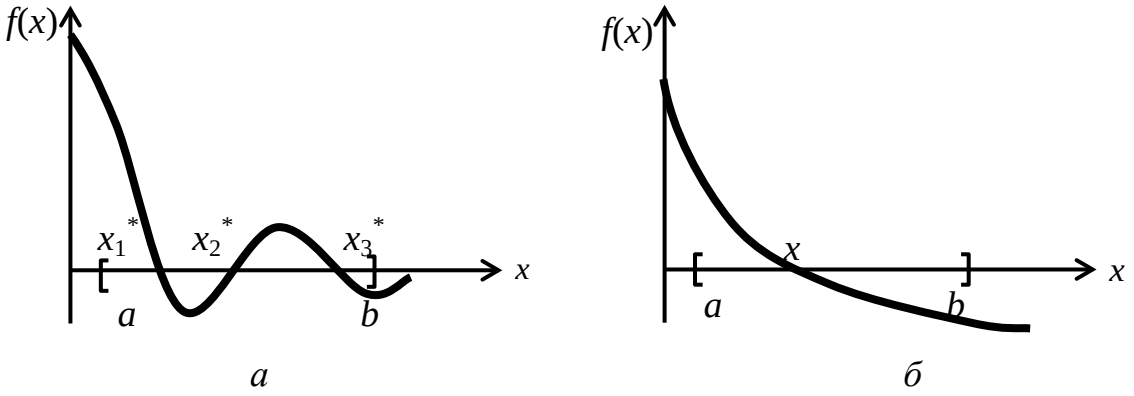


Рис. 3.4. Аналитический метод отделения корней

Однако в случае *а* – производная $f'(x)$ трижды меняет знак на интервале, соответственно на этом интервале нелинейное уравнение имеет три действительных корня. В случае *б* – производная не меняет знака на интервале, корень на интервале $[a,b]$ – единственный.

Процесс отделения корней начинается с установления знаков функции в граничных точках a и b. Затем определяются знаки в ряде промежуточных точек, после чего выделяются отрезки, на границе которых функция меняет знак на противоположный. Выделенные отрезки и содержат корень данного уравнения. Определение промежуточных точек носит либо случайный характер на числовой оси, либо определяется свойствами конкретной нелинейной функции. Например, тригонометрические функции изменяют свои свойства при переходе через абсциссы, кратные числу π. Для алгебраических функций, которые, как известно, легко дифференцируемы, можно находить промежуточные точки на основе первой производной.

Пример 3.2. Отделить корни уравнения $f(x) = x^4 - 4x - 1$.
Решение. Найдем производную $f'(x) = 4x^3 - 4 = 4(x^3 - 1)$;
$f'(x) = 0$ при $x = 1$.

Таким образом, промежуточная точка единственная и равна 1. Так как границы интервала $[a,b]$ не заданы, то выбираем их бесконечными. При заданных значениях аргумента x определяем знаки функции $f(x)$. Результаты поиска приведены в табл. 3.1.

Таблица 3.1

x	$-\infty$	-10	1	10	∞
$f(x)$	+	+	–	+	+

В результате поиска выделены два интервала [–10, 1] и [1, 10], на которых функция $f(x)$ имеет действительные корни: $x1 = -0{,}249$; $x2 = 1{,}663$.

3.2. Итерационные методы уточнения корней

Уточнение корней – второй этап решения нелинейного уравнения, позволяющий вычислить требуемые корни с заданной точностью. Все методы уточнения корней итерационные. Итерационный процесс состоит в последовательном (пошаговом) уточнении начального приближения x_0. Каждый такой шаг называется итерацией. В результате итерационных вычислений находится последовательность приближенных значений корня $x_0, x_1, x_2, \ldots, x_n$. Если с ростом n вычисленные значения корней приближаются к истинному значению корня, то итерационный процесс сходится.

Рассмотрим некоторые итерационные методы решения нелинейных уравнений.

Все методы решения нелинейных уравнений допускают хорошую геометрическую интерпретацию. Поэтому ниже, при изложении методов, будет приведена геометрическая иллюстрация метода и его пошаговое описание (алгоритмизация).

3.2.1. Метод половинного деления (дихотомии)

Пусть дано уравнение $f(x) = 0$.

Допустим, что на этапе отделения корней удалось найти отрезок $[a, b]$, на котором расположено значение корня $x^{\text{т}}$, т.е. $a < x^{\text{т}} < b$, $f(a) \cdot f(b) < 0$.

За начальное приближение корня x_0 принимаем середину отрезка $x_0 = (a + b)/2$. Далее исследуем значения функции: если $f(x_0) = 0$, то x_0 является корнем уравнения, т.е. $x^{\text{т}} = x_0$. Если $f(x_0) \neq 0$, следовательно корень не найден. И в качестве нового интервала выбираем одну из половин отрезка $[a, x_0]$ или $[x_0, b]$, на концах которой функция $f(x)$ имеет

противоположные знаки, т.е. содержит искомый корень. Для рис. 3.5 это интервал $[a, x_0]$. Интервал $[x_0, b]$ исключаем, на концах его знак $f(x)$ не меняется.

Отрезок $[a, x_0]$ вновь делим пополам. Находим новое приближение к корню – $x_1 = (a + x_0)/2$. Исследуем знаки функции $f(x)$ на концах отрезка. Теперь отбрасываем отрезок $[a, x_1]$, т.к. $f(x_1) > 0$ и $f(a) > 0$. Выбираем отрезок $[x_1, x_0]$, на концах которого функция имеет противоположные знаки $f(x_1) > 0$, $f(x_0) < 0$. Вновь делим пополам и получаем новое приближение корня $x_2 = (x_1 + x_0)/2$ и т.д. Итерационный процесс продолжаем до тех пор, пока длина отрезка после n-й итерации не станет меньше некоторого заданного малого числа (точности вычисления корня) ε, т.е. $|b - a| < \varepsilon$.

В качестве значения корня принимается последнее полученное приближение x_n. В этом случае говорят, что решение уравнения найдено с точностью ε. Одновременно подсчитываем количество шагов (итераций), за которое был найден корень.

Математические формулы метода дихотомии просты и определяют приближение к корню на очередной итерации и правило останова

$$\begin{aligned} c &= \frac{a+b}{2}; \\ |b-a| &\le \varepsilon, \end{aligned} \tag{3.2}$$

или

$$\begin{aligned} c &= \frac{x_{n-1}+x_n}{2}; \\ |x_n - x_{n-1}| &\le \varepsilon. \end{aligned} \tag{3.3}$$

Следующим этапом численного решения уравнения является разработка алгоритма метода дихотомии. Описание алгоритма возможно в виде блок-схемы или в виде пошагового описания.

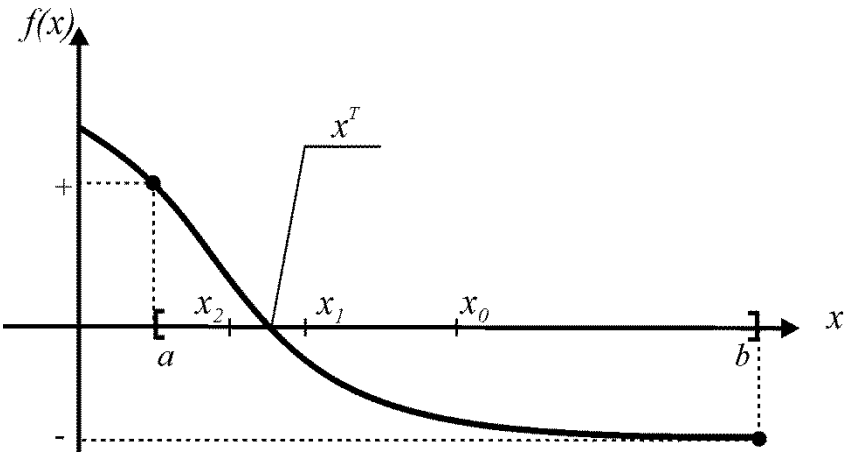


Рис. 3.5. Геометрическая иллюстрация метода половинного деления

Алгоритм метода дихотомии

Шаг 1. Выбор интервала $[a,b]$, такого, что $f(a) \cdot f(b) < 0$.
Задание точности вычисления корня ε.
Счетчик числа итераций $n = 0$.
Шаг 2. Вычисление $c = (a + b)/2$, $n = n + 1$.
Шаг 3. Если $f(c) = 0$, то – переход на Шаг 5,
иначе
если $f(a) \cdot f(c) < 0$, то $b = c$,
иначе $a = c$.
Шаг 4. Если $|b - a| > \varepsilon$, то – переход на Шаг 2,
иначе – переход на Шаг 5.
Шаг 5. Печать результатов: корень уравнения c,
число итераций n.
Выход из итерационной процедуры.

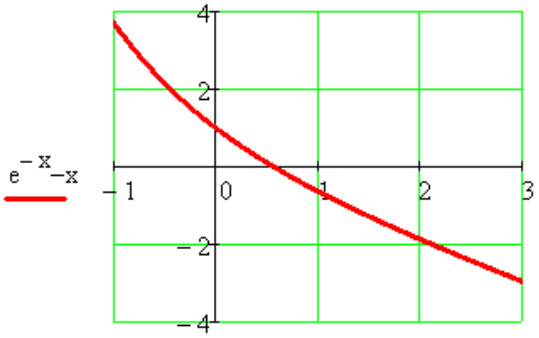
На рисунке 3.6 приведен листинг программы MathCAD, реализующей отделение корней и метод дихотомии для решения уже исследованного выше уравнения $f(x) = e^{-x} - x$. Метод дихотомии реализован в виде программы-функции от следующих аргументов: a, b (концы интервала) и точности вычисления корня ε. В последнем операторе программы-функции осуществлен вывод результатов: корня c и количества итераций n. Вызов программы- функции при различных значениях $\varepsilon 1 = 10^{-3}$ и $\varepsilon 2 = 10^{-6}$ дает близкие значение корня , вычисленные за различное число итераций.

1. Исходные данные: функция, точность

$y(x) := e^{-x} - x \qquad \varepsilon 1 := 10^{-3} \qquad \varepsilon 2 := 10^{-6}$

2. Графическое отделение корней

$a := 0 \qquad b := 1.5$



3. Программа для метода дихотомии
Аргументы: концы интервала и точность метода

$$\begin{aligned}
&\text{poldel}(a,b,\varepsilon) := \\
&\quad n \leftarrow 0 \\
&\quad \text{while } |b-a| > \varepsilon \\
&\qquad n \leftarrow n+1 \\
&\qquad c \leftarrow \frac{(a+b)}{2} \\
&\qquad b \leftarrow c \text{ if } y(a)\cdot y(c) < 0 \\
&\qquad a \leftarrow c \text{ otherwise} \\
&\quad \begin{pmatrix} c \\ n \end{pmatrix}
\end{aligned}$$

4. Результаты программы с различными значениями аргументов

$$\text{poldel}(a,b,\varepsilon 2) = \begin{pmatrix} 0.5671427 \\ 21 \end{pmatrix} \qquad \text{poldel}(a,b,\varepsilon 1) = \begin{pmatrix} 0.56763 \\ 11 \end{pmatrix}$$

Рис. 3.6. Листинг программы, реализующий метод дихотомии

Замечания.

1. Метод имеет малую сходимость. После n итераций

$$x_n - x_{n-1} = 0{,}5^{n/2}(x_1 - x_0),$$

где $(x_1 - x_0)$ – впервые найденный интервал;
$(x_n - x_{n-1})$ – интервал после n итераций.

Или

$$L_n = 0{,}5^{n/2} L_1,$$

где L_1, L_n – длины соответствующих интервалов.

2. Погрешность найденного решения заключена в пределах

$$0 \le x^{\text{т}} - x_n \le 0{,}5^{n/2} (x_1 - x_0).$$

Если на каждой итерации выбирается средняя точка c в качестве приближения к корню, то ошибка $|c - x^{\text{т}}|$ на каждой итерации уменьшается в среднем в два раза, т.е.

$$\frac{\left|x^m - x_n\right|}{\left|x^m - x_{n-1}\right|} \approx 0{,}5.$$

3.2.2. Особенности итерационных процедур

Большинство методов численного решения инженерных задач носят итерационный характер. Итерационные методы имеют ряд особенностей. Выделим основные из них:

1. Вычисления осуществляются по одной и той же итерационной (повторяющейся) формуле $x_k = \varphi(x_{k-1})$, т.е. каждое последующее приближение определяется через предыдущее.

2. Для начала итерационного процесса необходимо задание начального приближения x_0.

3. Итерационный процесс бесконечен. Для его останова необходима точность вычисления корня ε и правило (формула) останова, рисунок 3.7. Наиболее часто используются следующие правила останова:

$$|x_n - x_{n-1}| \le \varepsilon; \qquad (3.4)$$

$$|f(x_n) - f(x_{n-1})| \le \varepsilon. \qquad (3.5)$$

4. Важнейшая характеристика итерационного процесса – количество итераций (шагов) n получения решения с заданной точностью. Чем шагов меньше, тем быстрее может быть получен результат и тем привлекательнее метод.

5. Другая характеристика получения корня – скорость сходимости. Эта величина определяет, как быстро на каждой итерации уменьшается расстояние до точного значения корня $x^{\text{т}} - x_n$:

$$\frac{\left|x^m - x_n\right|}{\left|x^m - x_{n-1}\right|^r} = C, \qquad (3.6)$$

где C – некоторая конечная ненулевая константа;

показатель степени r называют скоростью сходимости.

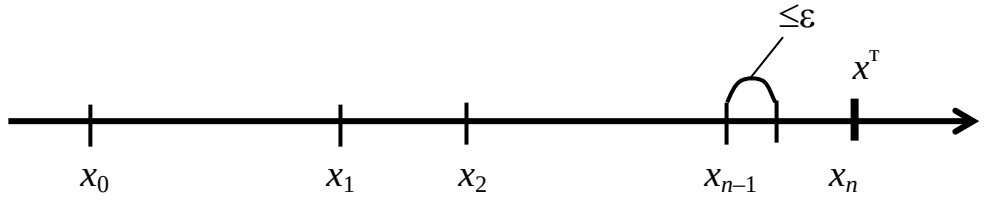


Рис.3.7 Иллюстрация сходимости итерационных процедур

Метод дихотомии имеет линейную скорость сходимости (r = 1).

Кроме линейной сходимости ряд методов обладают квадратичной сходимостью. Методы более высокой скорости сходимости встречаются реже.

Отметим, что, как видно из рисунка, на n-й итерации произошел останов итерационного процесса вычисления корня по правилу (3.4). Однако точного решения x^T достигнуть не удалось. Итерационная процедура позволила найти приближенное значение корня, но при любом численном решении существует погрешность вычисления $\left|x^T - x_n\right|$. В этом состоит особенность и сущность итерационных процедур.

Решение нелинейных уравнений процедурами пакета MathCad

Система MathCad имеет ряд встроенных процедур для решения нелинейных уравнений. основные из них следующие:

- **root**;
- вычислительный блок **Given - Find** (Дано - Найти);
- вычислительный блок **Given - Minerr** (Дано - Минимальная погрешность).

На рисунке 3.8 приведен листинг программы MathCad для решения исследуемого выше уравнения тремя процедурами. В данном случае все решения одинаковые.

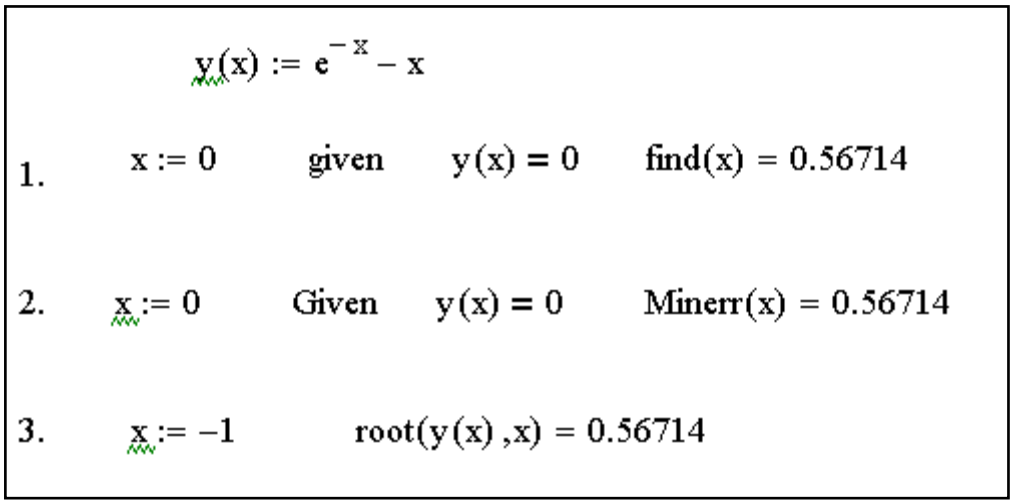


Рис.3.8 Решение нелинейных уравнений встроенными процедурами пакета MathCad

Процедура **root** позволяет найти только один корень уравнения. Причем, перед использованием процедуры требуется задать начальное значение (приближение) корня. Для процедуры **root** большое значение имеет начальное приближение к корню, если корней несколько.

При использовании вычислительных блоков необходимо учитывать следующие особенности:

1. Должны быть определены начальные приближения к корням (в данном листинге **x:=0**).

2. В равенствах должны быть использованы знаки тождественного равенства (*жирное равенство*, взятое с панели Булевой алгебры). Кроме равенств можно включать и неравенства, образуемые знаками < , > , ≤ , ≥.

3. Служебные слова **Given, Find** могут быть взяты из служебных слов, либо просто напечатаны.

Вычислительный блок **Given - Find** создает итерационную последовательность приближений к корню, начиная с заданного начального приближения. Полученное решение таково, что при подстановке его в уравнения правая и левая часть его отличается на величину **TOL** (**TOL**erance - точность, погрешность). По умолчанию величина **TOL**=10^{-3}. Если требуется более точный результат, то величину **TOL** можно сменить либо через меню, либо задав величину в рабочем листе программы, например TOL :=10^{-8}. (Имя переменной TOL записывается только в верхнем регистре).

Правила записи вычислительного блока **Given - Minerr** такие же. Однако итерационная процедура поиска корней ориентирована на поиск решения, минимизирующего разность правой и левой части уравнений.

Поэтому возможны ситуации, когда при одних и тех же начальных приближениях вы числительные блоки **Given - Minerr** и **Given - Find** приводят к разным решениям.

На рисунке 3.6 приведен листинг программы MathCad, релизующий метод половинного деления, использующий единственную переменную *c* для корня на каждой их *n* итераций в соответствии с формулами (3.2). Более общим случаем реализации метода в соответствии с формулами (3.3) является использование переменной массива для корня $x = \{x_0, x_1, ..., x_n\}$. То есть использование переменной с индексом. На рисунке 3.9 приведен фрагмент программы MathCad для такого случая.

Данная программа демонстрирует ряд важных особенностей реализации итерационных процедур в пакете MathCad.

1. В данной программе переменная использована для организации вывода результатов на экран. Если flag=1, осуществляется печать последовательности корней $x = \{x_0, x_1, ..., x_n\}$. Если flag=2 – число итераций, flag=3 – двумерный массив границ сокращаемых интервалов.

2. Интервал на последнем шаге [0.5625; 0.65625] . Длина этого интервала меньше заданной точности ε и по формальному правилу останова (3.3) программа закончила работу с последним значением корня 0.65625. Однако точное значение корня $x^T = 0.56714$.

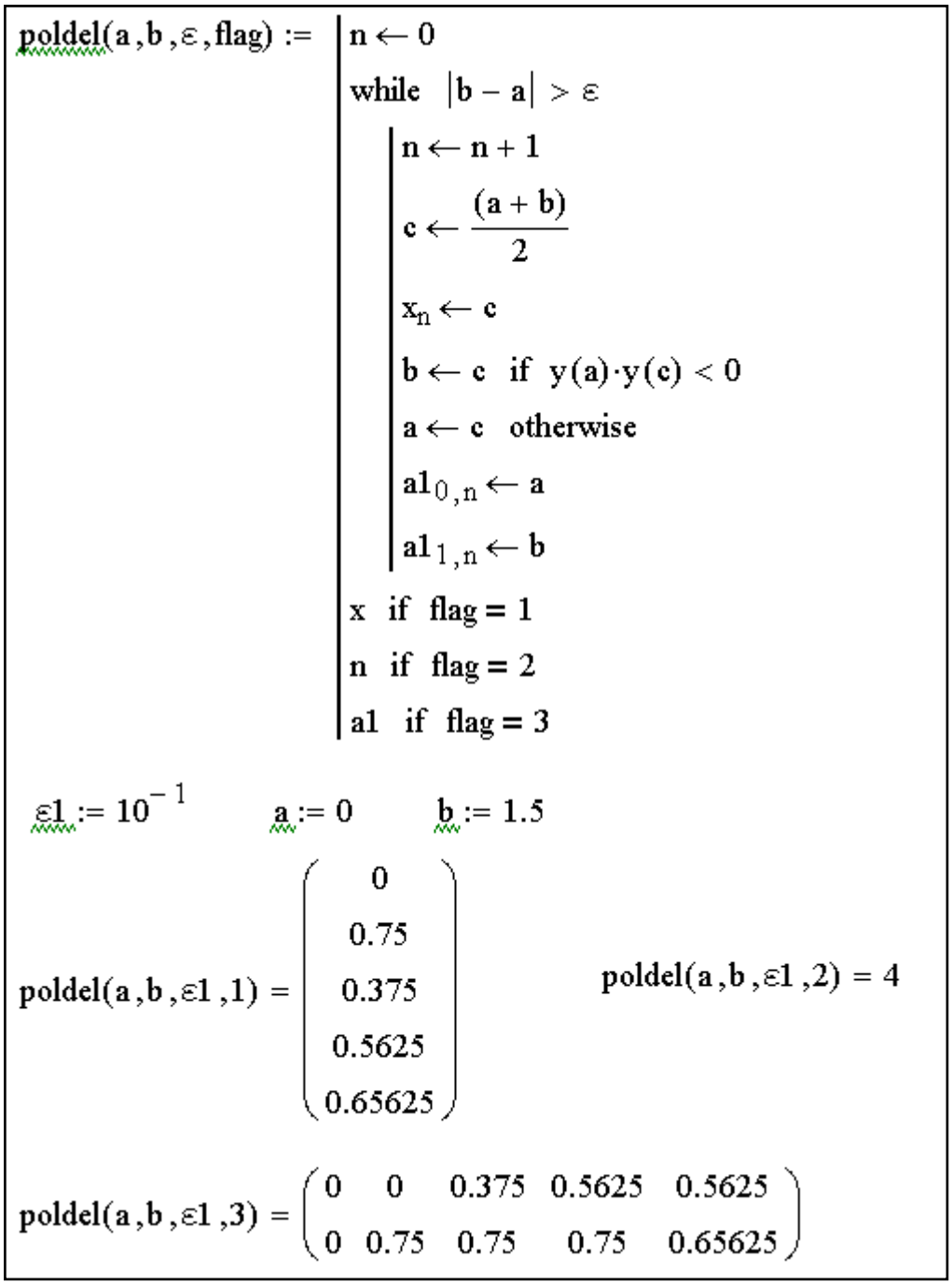


Рис.3.9 Фрагмент программы для метода половинного деления

3.2.3. Метод ложного положения (хорд)

Метод хорд также относится к интервальным, т.е. требует первоначального выбора интервала $[a,b]$. Алгоритмизация метода практически совпадает с методом половинного деления, за исключением шага 2.

Вместо деления интервала $[a,b]$ пополам метод предполагает по-

делить его в отношении $f(a)$ / $f(b)$. Графически (рис. 3.10) это означает построение хорды между точками $f(a)$ и $f(b)$. Уравнение прямой, проходящей через точки $f(a)$ и $f(b)$, имеет вид

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

Эта прямая пересекает ось OX при значении $x = c$, $f(c) = 0$. Отсюда найдем искомое приближение к корню $x^{\text{т}}$:

$$c = a - f(a)\cdot\frac{b-a}{f(b)-f(a)}. \qquad (3.7)$$

В остальном алгоритмизация метода хорд полностью повторяет алгоритмизацию метода половинного деления.

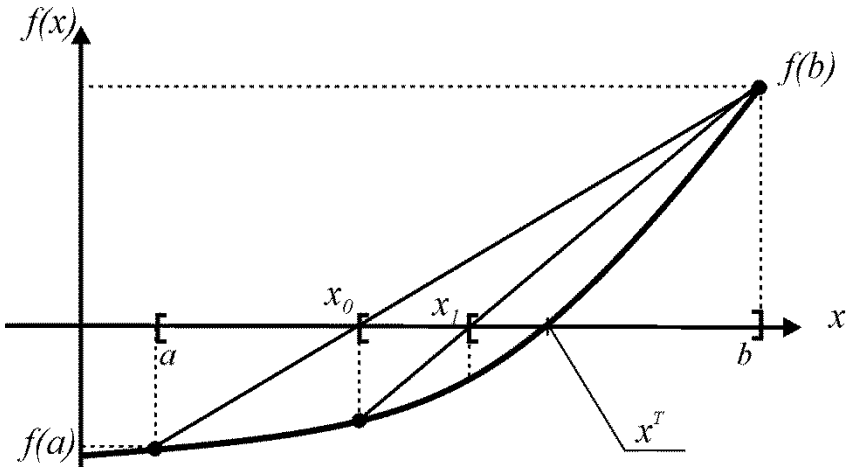


Рис. 3.10. Метод хорд

Алгоритм метода хорд

Шаг 1. Выбор интервала $[a, b]$, такого, что $f(a) \cdot f(b) < 0$.
Задание точности вычисления корня ε.
Счетчик числа итераций $n = 0$;
$x_n = a$ (либо b).

Шаг 2. Вычисление корня $c = a - f(a)(b - a)/f(b) - f(a))$;
$n = n + 1$;
$x_n = c$.

Шаг 3. Если $f(c) = 0$, то – переход на Шаг 5,
иначе
если $f(a) \cdot f(c) < 0$, то $b = c$, иначе $a = c$.

Шаг 4. Если $|x_n - x_{n-1}| > \varepsilon$, то – переход на Шаг 2,
иначе – переход на Шаг 5.

Шаг 5. Печать результатов: корень уравнения c, число итераций n.
Выход из итерационной процедуры.

Замечания:

1. В некоторых случаях (как, например, на рис. 3.10) стандартное правило останова итерационной процедуры для интервального метода $|b - a| < \varepsilon$ может не привести к решению, т.к. одна из границ остается неподвижной. В этом случае целесообразно использовать следующее правило останова (Шаг 4 алгоритма):

$$\left|x_n - x_{n-1}\right| \le \varepsilon,$$

где x_n, x_{n-1} – значения корней на двух соседних итерациях, найденные по формуле (3.7).

2. Погрешность найденного решения оценивается формулой

$$\left|x^m - x_{n-1}\right| \le \frac{M_1 - m_1}{m_1}\left|x_n - x_{n-1}\right|, \qquad (3.7а)$$

где M_1 и m_1 – соответственно наибольшее и наименьшее значения модуля производной $f'(x)$ на впервые найденном интервале $[a, b]$. Таким образом, скорость сходимости метода хорд также линейная, но с другой константой. В методе половинного деления константа одинакова для всех функций и равна 0,5. В методе хорд константа определяется свойствами производной, величинами M_1 и m_1.

3.2.4. Метод Ньютона (касательных)

Метод Ньютона очень популярен. Это объясняется тем, что, в отличие от интервальных методов дихотомии и хорд, метод Ньютона не требует первоначального задания интервала, на концах которого функция имеет разные знаки. Вместо приближения (интерполяции) по двум точкам в методе Ньютона осуществляется экстраполяция (прогнозирование) с помощью касательной к функции, рисунок 3.11.

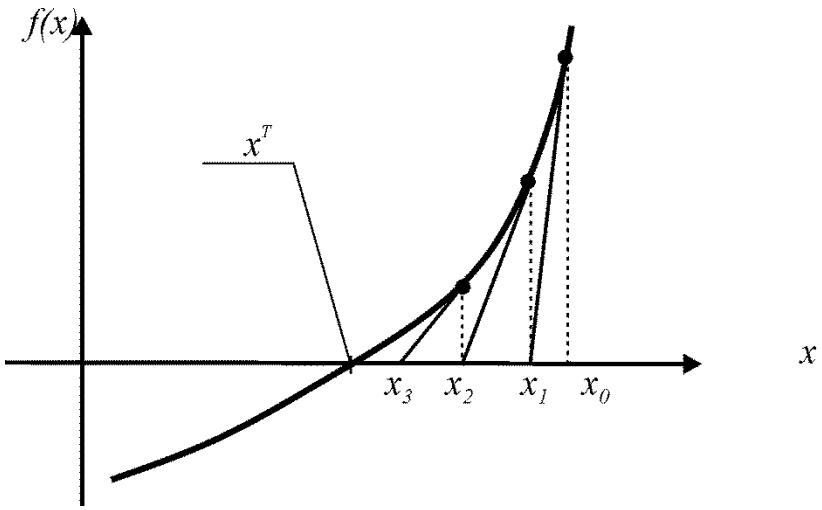


Рис. 3.11. Геометрическая иллюстрация метода Ньютона

Проведем касательную в точке $f(x_0)$. Первым приближением корня будет точка пересечения этой касательной с осью абсцисс – x_1. Через точку $f(x_1)$ снова проводим касательную, точка пересечения которой с осью абсцисс даст нам второе приближение корня – x_2 и т.д.

Процесс итераций состоит в том, что в качестве приближений к корню принимаются значения x_0, x_1, x_2, ... точек пересечения касательной к кривой $f(x)$ с осью абсцисс. Геометрически метод это означает замену небольшой дуги кривой $f(x)$ касательной. При этом не требуется задавать отрезок $[a, b]$, содержащий корень уравнения, а достаточно лишь найти некоторое начальное приближение корня x_0. К выбору этого начального приближения к корню метод Ньютона чрезвычайно критичен. Именно начальное приближение определяет сходимость (или расходимость) метода.

Достаточные условия сходимости метода Ньютона

Пусть корень функции отделен на интервале $[a, b]$, а производные $f'(x)$, $f''(x)$ сохраняют знак на интервале.

Тогда, исходя из начального приближения $x_0 \in [a, b]$, удовлетворяющего неравенству

$$f(x_0) \cdot f''(x_0) > 0, \tag{3.8}$$

можно построить итерационную последовательность

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}, \tag{3.9}$$

сходящуюся к единственному на $[a, b]$ решению x^T уравнения $f(x) = 0$.

Условия (3.8) позволяют найти начальное приближение, гаранти-

рующее сходимость метода Ньютона, а формула (3.9) – построить итерационную процедуру приближения к корню уравнения. В основе формулы (3.9) лежит разложение в ряд Тейлора в окрестности точки $f(x_n)$ и отбрасывание нелинейных членов ряда.

Правила останова стандартные – (3.4) или (3.5).

Алгоритм метода Ньютона

Шаг 1. Задание точности вычисления корня ε.
Счетчик числа итераций $n = 0$.
Выбор начального приближения x_0 (a или b) по условию (3.8).

Шаг 2. Вычисление корня $x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1})$;
$n = n + 1$.

Шаг 3. Если $|x_n - x_{n-1}| > \varepsilon$, то – переход на Шаг 2,
иначе – переход на Шаг 4.

Шаг 5. Выход из итерационной процедуры.
Печать результатов: корень уравнения x_n, число итераций n.

Замечания:

1. При выборе начального приближения x_0 предполагается, что требуемый корень отделен, на концах интервала $[a, b]$ функция $f(x)$ имеет разные знаки, т.е. $f(a)f(b) < 0$. Часто в качестве начальной точки проверяются сами концы интервала a или b. Условие выбора (3.8).

На рис. 3.12 приведены возможные варианты выбора правого или левого конца интервала в качестве начального приближения.

2. Метод Ньютона хорошо работает в окрестности корня. Он является более быстрым по сравнению с интервальными методами, и для него существует квадратичная сходимость

$$\left|x^m - x_n\right| \le \frac{M_2}{2m_1}(x_n - x_{n-1})^2. \qquad (3.10)$$

3. Иногда в формуле (3.9) вместо $f(x_n)$ используют $f(x_0)$. В этом случае сходимость линейная.

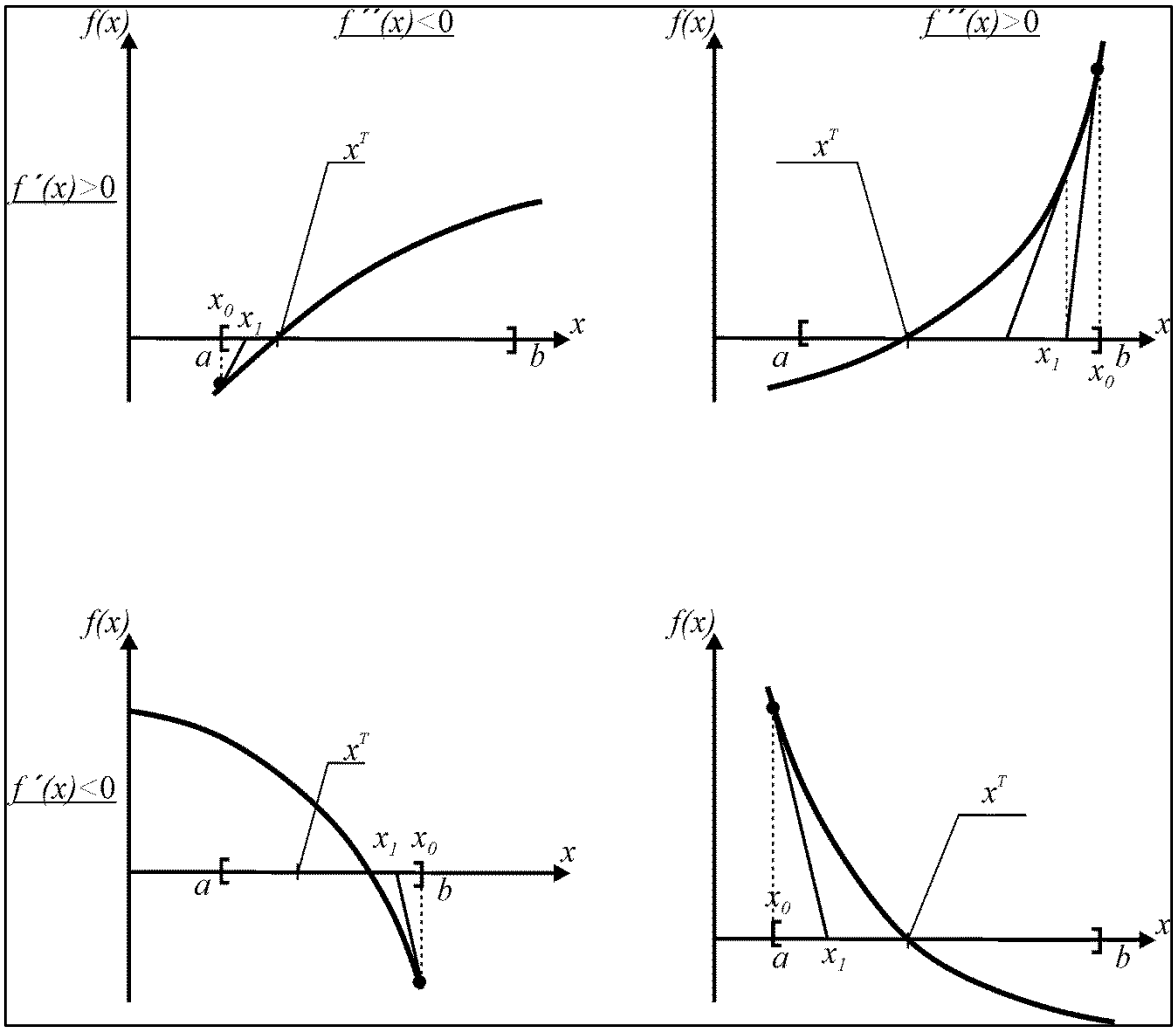


Рис. 3.12. Выбор начального приближения в методе Ньютона

3.2.5. Метод простых итераций

Метод простых итераций (или метод последовательных приближений) является важным численным методом решения нелинейных уравнений. Для применения этого метода нелинейное уравнение

$$f(x) = 0$$

разделяем на линейную (x) и нелинейную часть [$\varphi(x)$] и заменяем исходное уравнение эквивалентным:

$$x = \varphi(x). \qquad (3.11)$$

Выберем начальное приближение корня x_0. Подставляя это значение в правую часть уравнения (3.11), получаем новое приближение:

$$x_1 = \varphi(x_0).$$

Аналогично

$$x_2 = \varphi(x_1);$$
$$\ldots\ldots\ldots;$$
$$x_k = \varphi(x_{k-1}). \qquad (3.12)$$

Итерационные вычисления по формуле (3.12) продолжаются до тех пор, пока не будет выполнено правило останова итерационного процесса (3.4):

$$\left| x_n - x_{n-1} \right| \le \varepsilon.$$

Если значение $x^{\text{т}}$ – предел сходящейся последовательности $x_0, x_1, \ldots, x_n$, то $x^{\text{т}}$ является корнем нелинейного уравнения, который может быть вычислен по формуле (3.12) с любой степенью точности.

Очевидно, что гарантией сходимости последовательности $x_0, x_1, \ldots x_n$ является удачное разделение исходного уравнения на линейную (x) и нелинейную часть [$\varphi(x)$]. Достаточные условия сходимости метода простых итераций определяет следующая теорема.

Теорема. Пусть функция $\varphi(x)$ определена и непрерывна на интервале $[a, b]$. Тогда если

$$\left|\varphi(x)\right| < 1, \qquad x \in [a,b], \qquad (3.13)$$

то процесс итераций (3.12) сходится с любой степенью точности при любом начальном значении $x_0 \in [a, b]$.

Таким образом, успех метода простых итераций полностью определен «правильным» выделением нелинейной части уравнения. Ниже в примере будут показаны два способа выделения нелинейной части, приводящие к различным результатам.

Пример 3.3. Привести уравнение $f(x) = x^3 - x - 1$, где $x \in [1, 2]$ к виду, удобному для итераций (3.12).

Решение 1

$x = x^3 - 1;$

$\varphi(x) = x^3 - 1;$

$\varphi'(x) = 3x^2;$

$\varphi'(1) = 3, \varphi'(2) = 12;$

$\varphi'(x) > 1, x \in [1, 2].$

Метод итераций расходится.

Решение 2

$x = \sqrt[3]{x+1};$

$\varphi(x) = (x^3 - 1)^{1/3};$

$\varphi'(x) = \dfrac{1}{3\sqrt[3]{(x+1)^2}};$

$\varphi'(1) = \dfrac{1}{3\sqrt[3]{4}} < 1, \ \varphi'(2) = \dfrac{1}{3\sqrt[3]{9}} < 1;$

Метод итераций сходится.

Рассмотрим геометрическую иллюстрацию метода. Построим графики функций линейной части x и нелинейной (рис. 3.9).

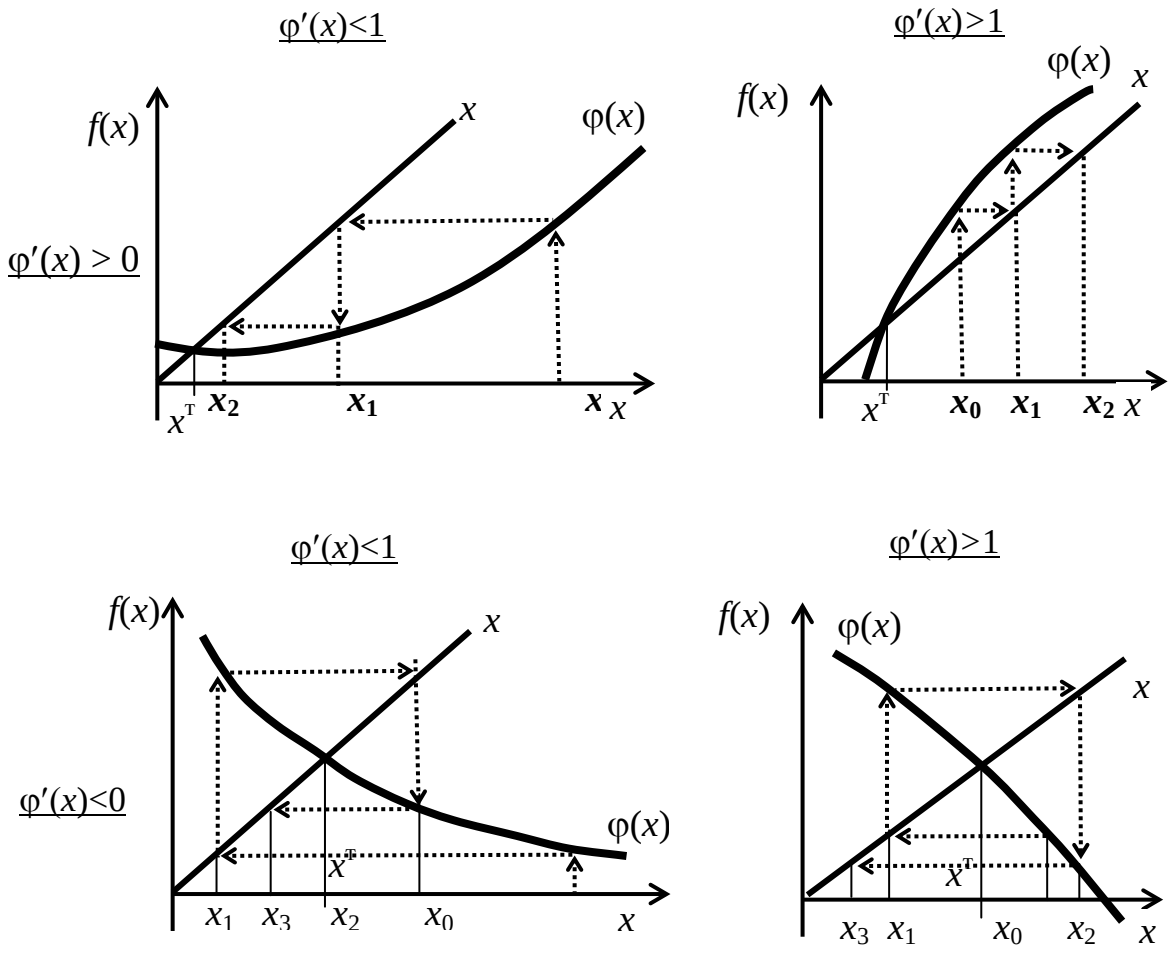


Рис. 3.13. Иллюстрация сходимости-расходимости метода простых итераций

Корнем исходного нелинейного уравнения является абсцисса точки пересечения линии $\varphi(x)$ с прямой x.

Из графиков видно, что в методе простых итераций возможны как сходящиеся, так и расходящиеся итерационные процессы. Скорость сходимости зависит от абсолютной величины производной $\varphi'(x)$. Чем меньше $\varphi'(x)$ вблизи корня, тем быстрее сходится процесс. Приближения к корню могут быть односторонними (типа лестницы) при $\varphi'(x) > 0$ и двусторонними (типа спирали) при $\varphi'(x) < 0$.

***Замечания*:**

1. Сходимость метода простых итераций определяется параметром $q = \max\left|\varphi'(x)\right| < 1,\ x \in [a, b]$ в соответствии со следующим выражением:

$$\left|x^m - x\right| \le \frac{q}{1-q}(x_1 - x_0),$$

т.е. погрешности приближения к корню убывают от шага к шагу, как члены геометрической погрешности со знаменателем q.

2. Метод простых итераций, как и все итерационные методы, обладает важным достоинством: он не накапливает вычислительную погрешность.

3.3. Приближенные методы решения систем нелинейных уравнений

Для решения систем нелинейных уравнений используются только итерационные методы, аналогичные методам решения нелинейных уравнений. Будем рассматривать системы нелинейных уравнений на примере системы двух уравнений вида

$$\begin{cases} f_1(x_1, x_2) = 0; \\ f_2(x_1, x_2) = 0. \end{cases} \qquad (3.14)$$

Функции f_1, f_2 – нелинейные функции двух аргументов x_1, x_2 (рис. 3.14).

Предполагается, что существует корень системы (3.14) – в данном случае двумерный вектор с координатами ($x_1^{\text{т}}$, $x_2^{\text{т}}$) и можно указать начальное приближение к корню, также вектор (x_1^{0}, x_2^{0}). В данном случае корень существует и он единственный.

В классификации уравнений на рис. 3.1 указано, что системы нелинейных уравнений могут иметь несколько решений. Проиллюстрируем это на следующем примере:

$$\begin{cases} f_1 = x_1^{\,2} - x_2 + \alpha = 0; \\ f_2 = -x_1 + x_2^{\,2} + \alpha = 0. \end{cases}$$

Здесь α – вещественный параметр из диапазона [–1, +1].

При $\alpha = 1$ – решений нет, $\alpha = 1/4$ – решение единственное, $\alpha = 0$ – два решения, $\alpha = -1$ – четыре решения.

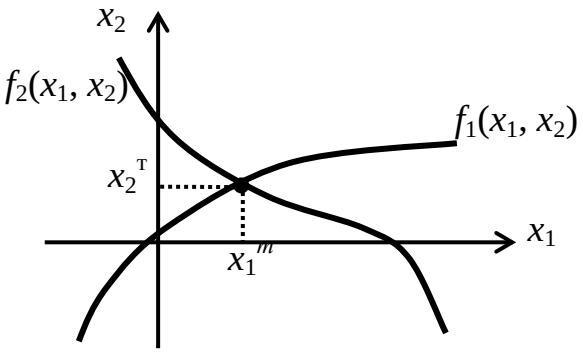


Рис. 3.14. Система двух
нелинейных уравнений

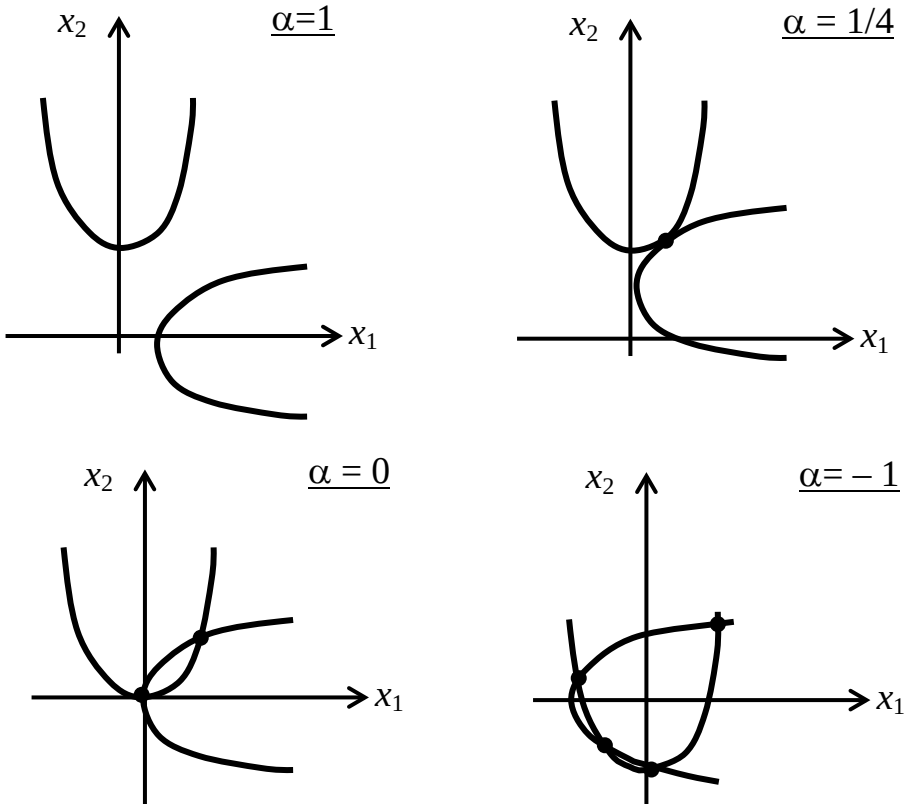


Рис. 3.15. Множество решений систем нелинейных уравнений

Метод Якоби, как и в одномерном случае, основан на предположении, что каждое нелинейное уравнение $f_1(x_1, x_2)$ и $f_2(x_1, x_2)$ системы (3.14) можно разделить на линейную (x) и нелинейную [$\varphi(x)$] часть, причем из первого уравнения линейно извлекаем x_1, а из второго – соответственно x_2. Получим

$$\begin{cases} x_1 = \varphi_1(x_1, x_2); \\ x_2 = \varphi_2(x_1, x_2). \end{cases} \quad (3.15)$$

Выбрав начальное приближение (x_1^0, x_2^0), подставим его в правую часть системы (3.15):

$$\begin{cases} x_1^1 = \varphi_1(x_1^1, x_2^1); \\ x_2^1 = \varphi_2(x_1^1, x_2^1). \end{cases}$$

Проделав аналогичные операции для (x_1^1, x_2^1), (x_1^2, x_2^2),…, (x_1^{k-1}, x_2^{k-1}), получим общую формулу метода простых итераций

$$\begin{cases} x_1^k = \varphi_1(x_1^{k-1}, x_2^{k-1}); \\ x_2^k = \varphi_2(x_1^{k-1}, x_2^{k-1}). \end{cases} \quad (3.16)$$

По формуле (3.16) может быть организована итерационная процедура приближения к корню системы нелинейных уравнений. Итерационный процесс продолжается до тех пор, пока обе координаты корня (x_1^k, x_2^k) будут меняться существенно. Обрыв итерационного процесса осуществляется на основе правила останова, аналогичного одномерному случаю:

$$\begin{cases} \left| x_1^k - x_1^{k-1} \right| \le \varepsilon; \\ \left| x_2^k - x_2^{k-1} \right| \le \varepsilon, \end{cases} \quad (3.17)$$

где ε – точность вычисления корня.

***Замечания*:**

1. Сходимость метода простых итераций линейная. Более быструю сходимость дает одна из модификаций метода простых итераций – метод Зейделя. Метод Зейделя отличается от метода Якоби тем, что при вычислении *k*-го приближения к корню используются уже вычисленные на *k*-м шаге корни. Применительно к системе двух уравнений формула (3.16) примет вид

$$\begin{cases} x_1^k = \varphi_1(x_1^{k-1}, x_2^{k-1}); \\ x_2^k = \varphi_2(x_1^k, x_2^{k-1}), \end{cases}$$

т.е. для вычисления второго корня используются значения первого корня с текущего шага, остальные значения берутся с предыдущего шага.

2. Основная сложность метода простых итераций – переход от вида системы (3.14) к (3.15), т.е. разделение на линейную и нелинейную

часть каждого уравнения.

3. Успех метода во многом определяется выбором начальных значений и выделением нелинейных частей, которые обеспечивают сходимость метода. Сходимость метода определяется следующей теоремой.

Теорема

Если норма Якобиана

$$\|J\| < 1, \tag{3.18}$$

то итерационный процесс (3.15) сходится.

Матрица Якоби (якобиан) для системы двух переменных имеет вид

$$J = \begin{pmatrix} \dfrac{\partial \varphi_1}{\partial x_1} & \dfrac{\partial \varphi_1}{\partial x_2} \\ \dfrac{\partial \varphi_2}{\partial x_1} & \dfrac{\partial \varphi_2}{\partial x_2} \end{pmatrix}.$$

В зависимости от принятой нормы 1 или 2 условия сходимости (3.18) имеют вид

$$\begin{aligned} \|J\|_1 &= \max_i \sum_{j=1}^{2} \left| \frac{\partial \varphi_i}{\partial x_j} \right| < 1; \\ \|J\|_2 &= \max_j \sum_{i=1}^{2} \left| \frac{\partial \varphi_j}{\partial x_i} \right| < 1. \end{aligned} \tag{3.19}$$

3.3.2. Метод Ньютона для решения систем нелинейных уравнений

Метод Ньютона обладает более быстрой сходимостью, чем метод простой итерации. Рассмотрим реализацию метода Ньютона для системы из двух нелинейных уравнений вида (3.14):

$$\begin{cases} f_1(x_1, x_2) = 0; \\ f_2(x_1, x_2) = 0. \end{cases}$$

При итерационном нахождении корней каждое последующее значение отличается от предыдущего на некоторую величину Δ. Запишем формально итерационные формулы для двух корней

$$\begin{cases} x_1^k = x_1^k + \Delta x_1^k; \\ x_2^k = x_2^k + \Delta x_2^k. \end{cases} \tag{3.20}$$

Разложим левые части нелинейных уравнений (3.14) в ряд Тейлора

в окрестности точки (x_1^k, x_2^k) и ограничимся производными первыми порядка. Получим

$$\begin{cases} f_1(x_1, x_2) \approx f_1\left(x_1^k, x_2^k\right) + \Delta x_1^k \left.\dfrac{\partial f_1}{x_1}\right|_k + \Delta x_1^k \left.\dfrac{\partial f_1}{x_2}\right|_k; \\ f_2(x_1, x_2) \approx f_2\left(x_1^k, x_2^k\right) + \Delta x_1^k \left.\dfrac{\partial f_2}{x_1}\right|_k + \Delta x_1^k \left.\dfrac{\partial f_2}{x_2}\right|_k. \end{cases}$$

Пусть приращения Δx_i^k (i = 1, 2) таковы, что функции f_1, f_2 принимают значения, близкие к корню, т.е. левые части предыдущей системы равны нулю. Перепишем систему с учетом этого:

$$\Delta x_1^k \left.\frac{\partial f_1}{\partial x_1}\right|_k + \Delta x_2^k \left.\frac{\partial f_1}{\partial x_2}\right|_k = -f_1(x_1^k, x_2^k);$$
$$\Delta x_1^k \left.\frac{\partial f_2}{\partial x_1}\right|_k + \Delta x_2^k \left.\frac{\partial f_2}{\partial x_2}\right|_k = -f_2(x_1^k, x_2^k),$$

или в векторно-матричной форме

$$\begin{pmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{pmatrix} \cdot \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = \begin{pmatrix} -f_1 \\ -f_2 \end{pmatrix}. \tag{3.21}$$

В сокращенной форме

$$W(\overline{x}^k) \cdot \Delta\overline{x}^k = F(\overline{x}^k), \tag{3.22}$$

где $W(\overline{x}^k)$ – матрица частных производных Якоби, вычисленная для приближенных корней x_i^k, i = 1, 2.

Неизвестными в этой системе являются приращения Δx_i^k, i = 1,2. Решим систему (3.22) относительно Δx_i^k

$$\Delta\overline{x}^k = (W(\overline{x}^k))^{-1} \cdot F(\overline{x}^k).$$

Найденные приращения используются как поправка к корню, полученному на предыдущем шаге x_i^k, i = 1, 2. Подставим поправки в формулу (3.20), получим окончательно формулу Ньютона

$$\Delta\overline{x}^k = \Delta\overline{x}^k + [W(\overline{x}^k)]^{-1} \cdot F(\overline{x}^k). \tag{3.23}$$

Если систему двух уравнений (3.21) разрешить относительно приращений не в матричной форме, а, например, по правилу Крамера, то получим выражения для приращений в виде

$$\Delta x_1^k = \frac{-f_1(x_1^k, x_2^k)\dfrac{\partial f_2}{\partial x_2} + f_2(x_1^k, x_2^k)\dfrac{\partial f_1}{\partial x_2}}{\dfrac{\partial f_1}{\partial x_1}\cdot\dfrac{\partial f_2}{\partial x_2} - \dfrac{\partial f_1}{\partial x_2}\cdot\dfrac{\partial f_2}{\partial x_1}};$$

$$\Delta x_2^k = \frac{-f_2(x_1^k, x_2^k)\dfrac{\partial f_1}{\partial x_1} + f_1(x_1^k, x_2^k)\dfrac{\partial f_2}{\partial x_1}}{\dfrac{\partial f_1}{\partial x_1}\cdot\dfrac{\partial f_2}{\partial x_2} - \dfrac{\partial f_1}{\partial x_2}\cdot\dfrac{\partial f_2}{\partial x_1}}.$$

Эти приращения затем используются в формуле (3.20) для нахождения приближения к корням системы на новом шаге.

***Замечания*:**

1. Окончание итерационного процесса возможно либо по стандартному условию останова (3.20), либо когда все приращения Δx_i^k (i = 1, 2) стремятся к нулю, т.е.

$$\left|\Delta x_i^k\right| \le \varepsilon, i = 1,2. \qquad (3.24)$$

Правило останова (3.24) означает, что найденные на данной итерации приращения практически не изменяются, т.е. корни уравнения найдены с заданной точностью.

2. Метод Ньютона обеспечивает более быструю сходимость, чем метод простых итераций. Особенно его эффективность заметна при удачном выборе начальных приближений. Сходимость ухудшается с увеличением числа уравнений.

Вопросы для самопроверки

1. Что называется корнем уравнения?
2. Приведите примеры алгебраических и трансцендентных уравнений.
3. Для чего производится процедура отделения корней нелинейного уравнения и предварительное исследование уравнений?
4. Правило останова итерационных методов решения нелинейных уравнений.
5. Выделите этапы решения нелинейного уравнения.
6. Способы отделения корней.
7. Перечислите интервальные методы нахождения корней.

8. Приведите геометрическую иллюстрацию метода дихотомии, хорд, касательных.

9. Сформулируйте теорему интервалов.

10. Геометрическая интерпретация сходимости метода простых итераций.

11. Опишите алгоритм метода дихотомии (хорд, касательных, простых итераций) для решения нелинейных уравнений.

12. Опишите основные свойства прямых и итерационных методов решения уравнений.

13. Что понимают под сходимостью итерационной процедуры?

14. Поясните, что такое скорость сходимости и как она связана с эффективностью метода.

4. ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧ ЛИНЕЙНОЙ АЛГЕБРЫ

Раздел математики, занимающийся изучением линейных уравнений и систем линейных уравнений, называется линейной алгеброй.

Рассмотрим неоднородную систему линейных алгебраических уравнений (СЛАУ)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1; \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2; \\ .....; \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n, \end{cases} \tag{4.1}$$

или в векторно-матричной форме –

$$\mathrm{A} \cdot \bar{x} = \bar{b}, \tag{4.2}$$

где A – квадратная матрица системы порядка n;

$\bar{x}$ – вектор неизвестных (решение СЛАУ);

$\bar{b}$ – вектор свободных членов.

Иногда матрично-векторную форму записывают в развернутом виде:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}. \tag{4.3}$$

Важнейшими вопросами теории СЛАУ являются:

- существование и единственность СЛАУ;
- мера обусловленности СЛАУ.

Существование решения зависит от определителя (детерминанта) матрицы A **detA**. Существуют три возможных варианта решения, представленные на рис. 4.1 на примере двух линейных уравнений с переменными x_1 и x_2:

а) **detA ≠ 0**, следовательно матрица A неособенная, а СЛАУ невырожденная и имеет единственное решение;

б) **detA = 0**, следовательно СЛАУ вырождена и не имеет решений (графики уравнений параллельны) или СЛАУ имеет бесконечное множество решений (графики уравнений накладываются друг на друга);

в) **detA ≈ 0**, система почти вырождена или плохо обусловлена. Это означает, что получить численное решение системы трудно, т.к. при непосредственном (векторно-матричном или с помощью обратной матрицы) решении системы (4.2) –

$$\mathbf{A^{-1}A}\cdot\boldsymbol{x} = \mathbf{A^{-1}}\cdot\boldsymbol{b}; \qquad \boldsymbol{x} = \mathbf{A^{-1}}\cdot\boldsymbol{b} \tag{4.4}$$

– необходимо нахождение обратной матрицы $\mathbf{A^{-1}}$. Методы обращения матрицы используют деление на величину **detA**. Если **detA ≈ 0**, то при делении на него получаются бесконечно большие числа, это делает невозможным численное решение СЛАУ или решение на ЭВМ. Количественно плохая обусловленность СЛАУ определяется следующими характеристиками:

- **detA** $< 10^{-2}$;
- $\mu = \max\lambda / \min > 10^3$ (λ – собственные числа системы).

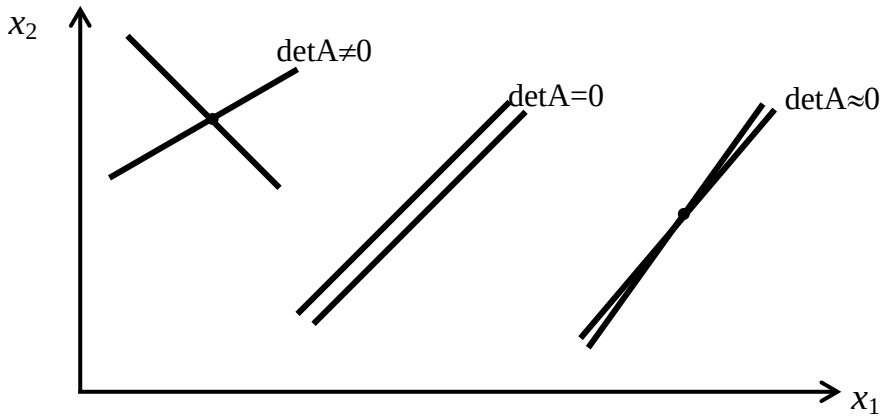


Рис. 4.1. Геометрическая иллюстрация решений СЛАУ

Выделяют четыре основные задачи линейной алгебры:

1. Решение СЛАУ, т.е. нахождение корней СЛАУ.
2. Вычисление определителя квадратной матрицы **detA.**
3. Нахождение обратной матрицы $\mathbf{A^{-1}}$.
4. Нахождение собственных чисел и собственных значений.

4.1. Методы решения СЛАУ

Методы решения СЛАУ разбиваются на две группы:

1. ***Прямые*** – приводят к решению за конечное число действий. Эти методы достаточно просты, пригодны для широкого класса линейных систем, нет погрешности метода. Вместе с тем прямые методы имеют ряд недостатков. Они применимы при ограниченной размерности системы, т.е. требуют хранения в оперативной памяти ЭВМ сразу всей

матрицы. Нерациональное использование оперативной памяти ЭВМ происходит также при неплотно заполненных матрицах, т.е. матрицах с большим количеством нулевых элементов. Недостатком прямых методов является также накопление погрешностей в процессе решения. К прямым методам решения СЛАУ относятся методы Гаусса, метод Крамера, схема Жордана, главных компонентов и ряд других.

2. ***Итерационные*** – число действий заранее определить нельзя. К достоинствам итерационных методов следует отнести то, что они требуют хранения в памяти машины не всей матрицы системы, а лишь отдельных векторов. Погрешности результатов не накапливаются, поскольку точность вычислений в каждой итерации определяется лишь результатами предыдущей итерации и практически не зависит от ранее выполненных вычислений. Итерационные методы могут использоваться для уточнения решений, полученных с помощью прямых методов. Целесообразно применять для слабо заполненных матриц большой размерности. К этой группе относятся: методы простых итераций, Зейделя, релаксации.

4.2. Метод Гаусса (метод последовательного исключения неизвестных)

Метод Гаусса основан на приведении матрицы коэффициентов к верхнетреугольному виду. Это достигается последовательным исключением неизвестных из уравнений системы. Вначале исключается x_1 из всех уравнений, кроме первого. Затем исключается x_2 из всех уравнений, кроме первого и второго. И так далее. В последнем уравнении исключены все неизвестные, кроме x_n. Это позволяет вычислять искомые неизвестные, начиная с последнего уравнения, из которого находим неизвестное x_n. Далее, используя это полученное значение, из предыдущего уравнения находим x_{n-1} и т.д. Последним находим x_1 из первого уравнения.

Существует множество модификаций метода Гаусса, но все они имеют целью получение верхнетреугольной матрицы. Оставляя пока в стороне формулы приведения, преобразуем матрицу системы к верхнетреугольному виду с помощью линейных преобразований над строками (т.е. сложение строк и умножение их на число).

Пример 4.1:

$$\begin{cases} 3x_1 - x_2 = 5; \\ -2x_1 + x_2 + x_3 = 0; \\ 2x_1 - x_2 + 4x_3 = 15, \end{cases}$$

или в векторно-матричной форме:

$$\begin{pmatrix} 3 & -1 & 0 \\ -2 & 1 & 1 \\ 2 & -1 & 4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 15 \end{pmatrix}.$$

Цель первого преобразования – исключить в последней строке x_1. Для этого сложим второе и третье уравнения:

$$\begin{pmatrix} 3 & -1 & 0 \\ -2 & 1 & 1 \\ 0 & 0 & 5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 15 \end{pmatrix}$$

Исключая x_1, одновременно удалось исключить и x_2 – обычно это достигается на втором шаге преобразования. Следующим шагом является исключение x_1 из второго уравнения. Для этого складываем первое и второе уравнения, при этом второе уравнение умножаем на 1,5:

$$\begin{pmatrix} 3 & -1 & 0 \\ 0 & 0.5 & 1.5 \\ 0 & 0 & 5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \\ 15 \end{pmatrix}$$

Получили матрицу верхнетреугольного вида. Из последнего уравнения находим $x_3 = 3$, из второго – $x_2 = 1$ и из первого – $x_1 = 2$.

Схем Гауссовых исключений множество. Рассмотрим одну из них – схему *единственного деления.*

Основная идея метода состоит в замене матрицы системы (4.1) эквивалентной системой с треугольной матрицей вида

$$\begin{cases} x_1 + \alpha_{12}x_2 + ... + \alpha_{1n}x_n = \beta_1; \\ x_2 + ... + \alpha_{2n}x_n = \beta_2; \\ .....; \\ x_n = \beta_n. \end{cases} \qquad (4.5)$$

Решение системы (4.5) находится, начиная с последнего уравнения в обратном порядке, по рекуррентным формулам.

Метод Гаусса состоит из двух этапов:

I. Прямой ход. Приведение матрицы **A** к верхнетреугольному виду.

Для этого выполняется n преобразований матрицы **A**. На каждом шаге преобразования выбирается главная k-я главная или ведущая строка. Диагональные элементы главной строки называются главными (ведущими) элементами.

На прямом ходе выполняются следующие действия:

- для всех строк, кроме главной, находим множитель l_{ik};
- к каждой неглавной добавляем главную, умноженную на множитель l_{ik};
- главную строку делим на главный элемент a_{kk};
- главную строку вычеркиваем, размерность системы становится $(n-1)$.

Преобразования с главной строкой, указанные выше, могут производиться, либо главная строка остается без изменения. В дальнейшем изложении рассмотрен второй вариант – без изменения главной строки.

На следующем шаге исключения главной строкой становится опять верхняя строка (под вычеркнутой), и все указанные выше преобразования повторяются. Преобразования повторяются столько раз, пока главная строка не становится единственной в системе.

В результате матрица **A** – верхнетреугольного вида.

II. Обратный ход. Вектор неизвестных СЛАУ $\bar{x}$ находится в обратном порядке. Для этого составляется матрица из вычеркнутых строк верхнетреугольного вида. Из последнего уравнения находится x_n, затем неизвестные находятся в порядке $x_{n-1}, x_{n-2},\ldots, x_1$.

Формулы метода Гаусса

I. Прямой ход:

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij}^{(k-1)} + l_{ik}\cdot a_{kj}^{(k-1)};\\ b_i^{(k)} &= b_i^{(k-1)} + l_{ik}\cdot b_k^{(k-1)};\\ l_{ik} &= -\frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}. \end{aligned} \qquad (4.6)$$

II. Обратный ход:

$$x_i = \frac{b_i^{(i)} - \sum\limits_{j=i+1}^{n} a_{ij}^{(i)}\cdot x_j}{a_{ii}^{(i)}}. \qquad (4.7)$$

В формулах (4.6), (4.7) k – номер шага преобразования. На обрат-

ном ходе номер шага совпадает с номером строки i. Последнее неизвестное x_n находят по формуле

$$x_n = \frac{b_n^{(n)}}{a_{ii}^{(n)}}. \qquad (4.7а)$$

Алгоритмизация метода Гаусса

Алгоритм метода Гаусса весьма прост логически, и поэтому нет смысла приводить его блок-схему. Как правило, блок-схемы хорошо иллюстрируют сложную разветвленную логику. Основная сложность метода Гаусса – в многочисленных матричных операциях, которые выполняются на основе циклических структур и индексов. Поэтому здесь использовано пошаговое описание алгоритма:

1. Цикл k =1, …, n – 1.
2. Цикл $i = k + 1$, …, n.
3. $l_{ik} = - a_{ik}/a_{kk}$.
4. Цикл $j = k + 1$, …, n.
5. $a_{ij} = a_{ij} + l_{ik} \cdot b_k$.
6. Конец цикла j.
7. $b_i = b_i + l_{ik} \cdot b_k$.
8. Конец циклов i, k.
9. $x_n = b_n/a_{nn}$.
10. Цикл $i = n - 1, n - 2$, …, 1.
11. $S = 0$.
12. Цикл $j = i + 1$, …, n.
13. $S = \sum_j a_{ij} \cdot x_j$.
14. Конец цикла j.
15. $x_i = (b_i - s)/a_{ii}$.
16. Конец цикла i.

Приведенный алгоритм полностью основан на формулах (4.6), (4.7). При алгоритмизации верхний индекс k, отвечающий за шаг преобразования, заменен отдельным циклом по той же переменной k.

Шаги алгоритма 1–8 соответствуют прямому ходу.

Шаги 9–15 – обратному ходу.

Пример 4.2. Используя схему единственного деления Гаусса, решить СЛАУ из примера 4.1:

$$\begin{cases} 3x_1 - x_2 = 5; \\ -2x_1 + x_2 + x_3 = 0; \\ 2x_1 - x_2 + 4x_3 = 15. \end{cases}$$

Пусть $k = 0$:

$$\begin{pmatrix} 3 & -1 & 0 \\ -2 & 1 & 1 \\ 2 & -1 & 4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 15 \end{pmatrix};$$

$k = 1$. Главная (или ведущая) строка – первая, её оставляем без изменения.

Для второй и третьей строк по формуле (4.6) найдем сомножители:

$$l_{21} = -(-2/3) = 2/3;$$

$$l_{31} = -(2/3) = -2/3.$$

Перепишем вторую строку, производя следующие преобразования: к каждому элементу второй строки (включая свободный член) добавляем стоящий над ним в матрице коэффициентов элемент первой строки, умноженный на l_{21}. Например:

$$a_{12} = 1+(-1) \cdot (2/3) = 1/3.$$

Сомножитель l_{ik} – единый для всей строки. k – номер шага.

Аналогичны преобразования для третьей строки. Например:

$$b_3 = 15 + 5 \cdot (-2/3) = 35/3.$$

Схему единственного деления Гаусса удобно приводить в виде таблицы. В таблице находятся коэффициенты матрицы, свободные члены и их изменения от шага к шагу. В таблице 4.1 приведена таблица метода Гаусса для примера 4.2. В ней на каждом шаге исключения выделены главные строки. Диагональные элементы главных чтрок являются ведущими.

Таблица 4.1

Метод Гаусса, схема единственного деления

№	k	l_{ik}	Коэффициенты при неизвестных			b	c
			x_1	x_2	x_3		

	0		3	–1	0	5	7
		2/3	–2	1	1	0	0
		–2/3	2	–1	4	15	20
	1		3	–1	0	5	7
			0	1/3	1	10/3	14/3
		1	0	–1/3	4	35/3	46/3
	2	0	3	–1	0	5	7
		0	0	1/3	1	10/3	14/3
			0	0	5	45/3	60/3
Обратный ход					$x_3 = 3$		$\bar{x}_3 = 4$
				$x_2 = 1$			$\bar{x}_2 = 2$
			$x_1 = 1$				$\bar{x}_1 = 3$

На этапе обратного хода составляется система из главных (выделенных) строк. И, начиная с последнего уравнения, находим неизвестные:

$5 \cdot x_3 = 45/3$, т.е. $x_3 = 3$;

$1/3 \cdot x_2 + 1 \cdot 3 = 10/3$, следовательно, $x_2 = 1$ и т.д.

Замечание. Метод Гаусса, особенно при ручном счете, требует большого внимания и точности. Для уменьшения возможности ошибок счета вводятся так называемые контрольные суммы (столбец c в таблице), с которым проделывают следующие преобразования:

а) *на прямом ходе* те же преобразования, что со столбцом свободных членов b:

$$c_i^{(k)} = c_i^{(k-1)} + l_{ik} \cdot c_k^{(k-1)}. \qquad (4.8)$$

Контроль правильности преобразования строки на очередном шаге проводится суммированием всех коэффициентов в строке и свободного члена. Эта сумма должна быть равна значению c из столбца контрольных сумм, вычисленного по формуле (4.8), т.е.

$$c_i^{(k)} = \sum_{j=1}^{n} a_{ij}^k + b_i^k;$$

б) *при обратном ходе* одновременно с вычислением корней x_i вычисляются корни $\bar{x}_i$, которые получены, если в выделенном в системе уравнении вместо свободного члена b_i использовать значение контрольной суммы c_i. Например:

$5 \cdot \bar{x}_3 = 6\ 0/3$, т.е. $\bar{x}_3 = 4$.

$1/3 \cdot \bar{x}_2 + 1 \cdot 4 = 14/3$, т.е. $\bar{x}_2 = 1$.

Корни $\bar{x}_i$, полученные по контрольным суммам, больше обычных

корней x_i на единицу: $\overline{x}_i = x_i + 1$.

В этом и состоит контроль обратного хода.

4.3. Задачи теории систем, сопутствующие реализации метода Гаусса

Метод Гаусса и его модификации популярны в вычислительной математике и её инженерных приложениях. Многие математические программные пакеты (MathCAD, Matematica) при реализации процедур решения СЛАУ используют метод Гаусса. Популярность метода Гаусса отчасти объясняется возможностью решения ряда задач теории систем, которые могут быть одновременно решены при реализации схемы Гаусса. Ниже рассмотрены некоторые часто встречающиеся задачи.

1. ***Треугольная факторизация матриц***

В результате реализации метода Гаусса матрица системы **A** может быть представлена в виде произведения двух треугольных матриц (**L-V** разложение, **L** – нижнетреугольная и **V** – верхнетреугольная):

$$\mathbf{A = L \cdot V.}$$

Матрица L имеет по главной диагонали единичные элементы, а ниже её – сомножители l_{ik}, вычисленные по формуле (4.6).

Матрица V составлена из элементов главных (вычеркнутых) строк в таблице:

$$\mathrm{A} = \begin{pmatrix} 1 & 0 & 0 & \ldots 0 \\ l_{21} & 1 & 0 & \ldots 0 \\ . & . & . & . \\ l_{n1} & l_{n2} & l_{n3} & \ldots 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & a_{22}^{(1)} & \ldots & a_{2n}^{(1)} \\ . & . & . & . \\ 0 & 0 & \ldots & a_{nn}^{(n-1)} \end{pmatrix}.$$

Пример 4.3. На основании табл. 4.1 провести треугольную матрицу системы из примера 4.1:

$$\mathrm{A} = \begin{pmatrix} 3 & -1 & 0 \\ -2 & 1 & 1 \\ 2 & -1 & 4 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2/3 & 1 & 0 \\ 2/3 & -1 & 1 \end{pmatrix}}_{\mathrm{L}} \cdot \underbrace{\begin{pmatrix} 3 & -1 & 0 \\ 0 & 1/3 & 1 \\ 0 & 0 & 5 \end{pmatrix}}_{\mathrm{V}}.$$

2. ***Вычисление определителей***:

$$\mathbf{\det A = \det L \cdot \det V = 1 \cdot a_{11} \cdot a_{22}^{(1)} \cdot \ldots \cdot a_{nn}^{(n-1)}.}$$

Определитель треугольной матрицы равен произведению диагональных элементов. По диагонали матрицы L стоят только единичные элементы, т.е. её определитель равен единице. И таким образом, определитель матрицы системы находится как *произведение главных элементов.*

Пример 4.4. Для примера 4.1 вычислить определитель на основании метода Гаусса:

$$\det A = 3 \cdot 1/3 \cdot 5 = 5.$$

На рисунке 4.2. приведена программа-функция MathCAD, реализующая метод Гаусса и задачи, ему сопутствующие: вычисление определителя и треугольную факторизацию матриц. Выделенные в правой части фрагменты области – вывод результатов прграммы.

В последнем операторе самой программы напечатан вектор- строка результатов (*x l a det*), где *x* корни, *l- a* соответственно L-V матрицы, *det* - определитель.

В качестве обучающего в MathCad примера вызов результатов осуществлен двумя способами. С одной стороны - как элемент матрицы размера 1x4, с другой - как элементы транспонированного вектора-столбца. Поэтому при вызове (печати) результатов обращение ведется к элементу вектора-строки с двойным индексом $MG(A, b)_{1,1}$, $MG(A, b)_{1,2}$ или к элементу—транспонированного вектора-столбца $MG(A, b)^T_3$, $MG(A, b)^T_3$.

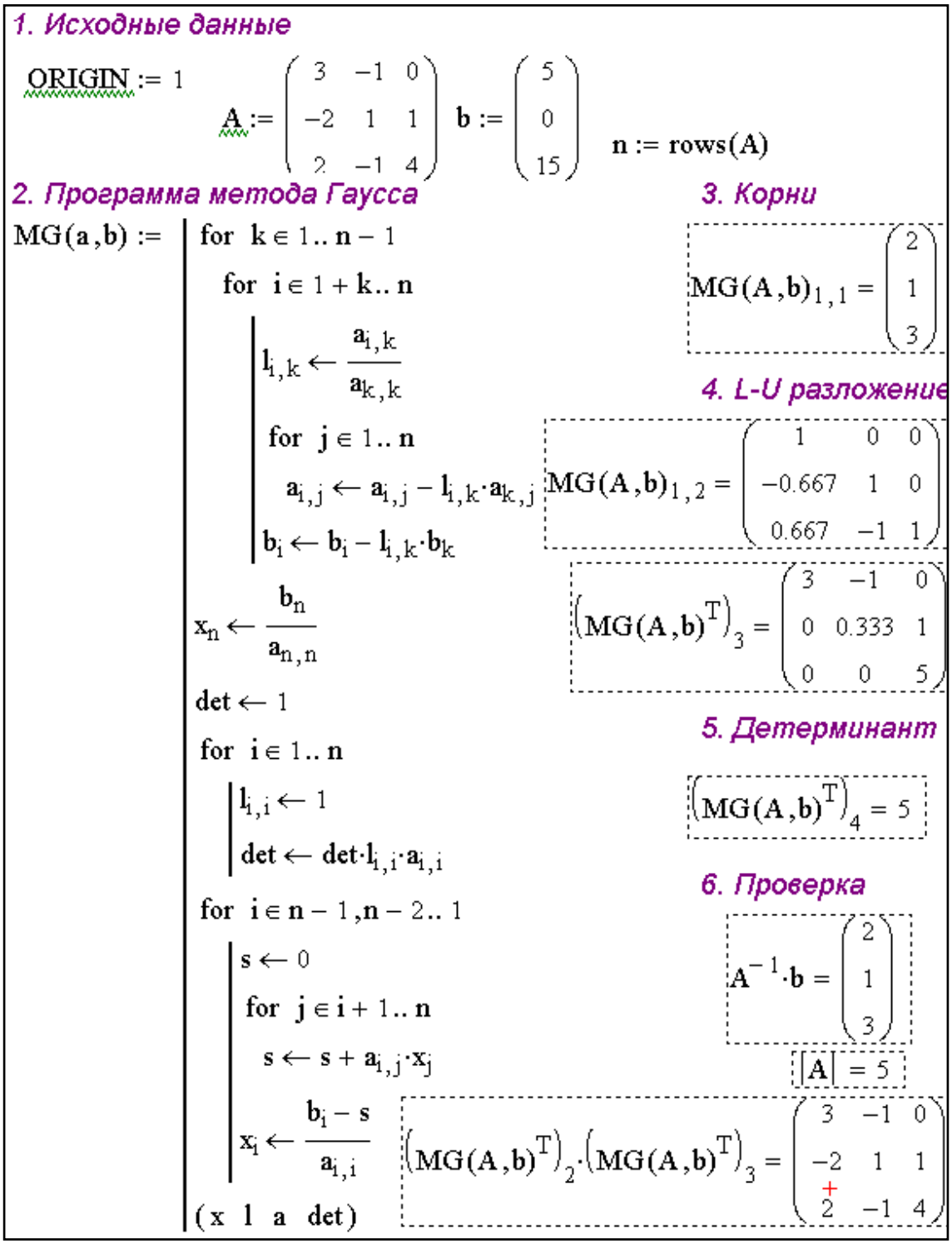

1. Исходные данные
ORIGIN := 1
A := (3 −1 0 ; −2 1 1 ; 2 −1 4)
b := (5 ; 0 ; 15)
n := rows(A)
2. Программа метода Гаусса
MG(a,b) :=
for k ∈ 1.. n − 1
for i ∈ 1 + k.. n
l_{i,k} ← a_{i,k} / a_{k,k}
for j ∈ 1.. n
a_{i,j} ← a_{i,j} − l_{i,k}·a_{k,j}
b_i ← b_i − l_{i,k}·b_k
x_n ← b_n / a_{n,n}
det ← 1
for i ∈ 1.. n
l_{i,i} ← 1
det ← det·l_{i,i}·a_{i,i}
for i ∈ n − 1, n − 2.. 1
s ← 0
for j ∈ i + 1.. n
s ← s + a_{i,j}·x_j
x_i ← (b_i − s) / a_{i,i}
(x l a det)
3. Корни
MG(A,b)_{1,1} = (2 ; 1 ; 3)
4. L-U разложение
MG(A,b)_{1,2} = (1 0 0 ; −0.667 1 0 ; 0.667 −1 1)
(MG(A,b)^T)_3 = (3 −1 0 ; 0 0.333 1 ; 0 0 5)
5. Детерминант
(MG(A,b)^T)_4 = 5
6. Проверка
A^{−1}·b = (2 ; 1 ; 3)
|A| = 5
(MG(A,b)^T)_2 · (MG(A,b)^T)_3 = (3 −1 0 ; −2 1 1 ; 2 −1 4)

Рис.4.2.

3. ***Вычисление обратной матрицы*** основано на основном свойстве обратной матрицы:

$$\mathbf{A}\cdot \mathbf{A}^{-1} = \mathbf{E},$$

где E – единичная матрица.

Запишем это уравнение в развернутом виде:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & . & . \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} y_{11} & y_{12} & \dots & y_{1n} \\ y_{21} & y_{22} & \dots & y_{2n} \\ . & . & . & . \\ y_{n1} & y_{n2} & \dots & y_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ . & . & . & . \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

$$\mathrm{A} \qquad\qquad \mathrm{A}^{-1} \qquad\qquad \mathrm{E}$$

Нахождение обратной матрицы сводится к решению n систем СЛАУ, у которых одна общая матрица А, а столбцами свободных членов являются столбцы единичной матрицы Е. Подробнее: решаем СЛАУ n раз. Первый раз в качестве столбца свободных членов выбираем первый столбец единичной матрицы. В результате решения получаем неизвестные, которые являются первым столбцом обратной матрицы. При втором решении получаем второй столбец обратной матрицы, вместо свободных членов – второй столбец единичной матрицы. И так поступаем n раз.

На рисунке 4.3. приведена программа-функция MathCAD, реализующая вычисление обратной матрицы на основе обращения к программе-функции метода Гаусса. Программа-функция весьма лаконична благодаря использованию в цикле верхней угловой скобки, определяющей весь вектор-столбец матрицы. Также осуществлена проверка обратной матрицы символьным вычислителем MathCad.

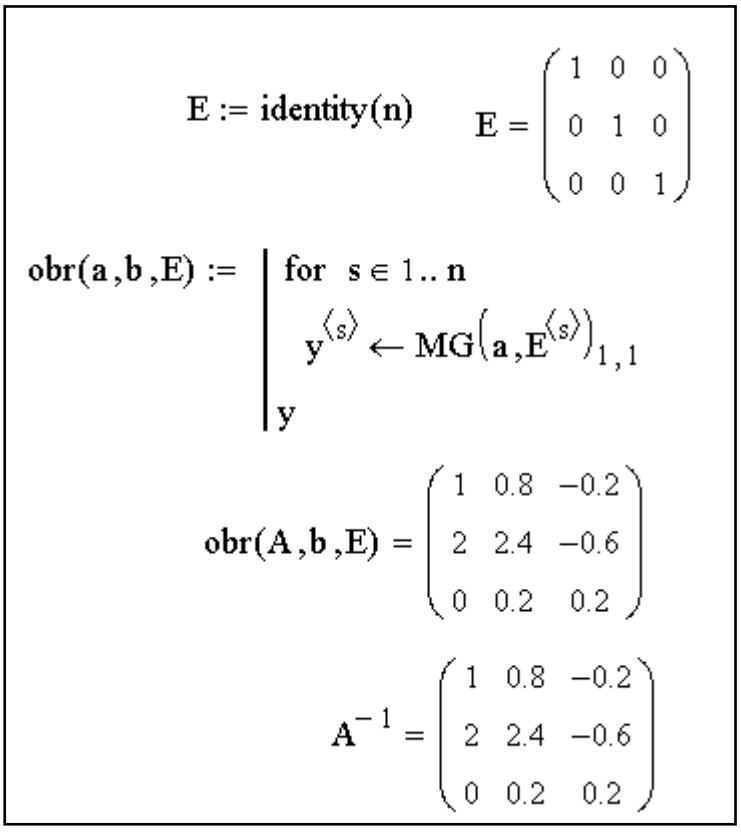


Рис.4.3.

4.4. Метод Гаусса с выбором главного элемента

При реализации метода Гаусса по схеме единственного деления предполагалось, что диагональные элементы не равны нулю. Это можно гарантировать только на нулевом шаге (k = 0) в исходной системе. Однако в ходе преобразований вполне возможны ситуации, что главные диагональные элементы $a_{kk}^{(k-1)}$ станут нулевыми. В этих условиях схема единственного деления становится неработоспособной, т.к. на главные элементы в ходе преобразований производится деление. Выходом в такой ситуации является использование модификации метода Гаусса – метод главных элементов.

Основная идея метода

1. На каждом шаге выбирают главный элемент матрицы A – максимальный по модулю коэффициент в матрице a_{pq}. Строка p – главная строка.

2. Для всех строк, кроме главной, вычисляются множители

$$m_i = -\frac{a_{iq}}{a_{pq}}. \qquad (4.9)$$

3. К каждой неглавной строке добавляют главную, умноженную на сомножитель m_i. В результате q-й столбец – нулевой.

4. Вычеркиваем p-ю строку и q-й столбец. В результате получаем матрицу A^1, размерность которой на единицу меньше предыдущей матрицы A.

5. Процедуру повторяем с первого шага $(n - 1)$ раз.

6. В результате составляем новую систему уравнений из вычеркнутых строк. Полученная матрица не треугольного вида, как в схеме единственного деления Гаусса. Но каждое уравнение содержит разное количество неизвестных: в последнем уравнении – 1 неизвестное, в $(n–1)$-м – 2 неизвестных и т.д., в первом уравнении – все n неизвестных. Такой вид преобразованной системы позволяет последовательно находить неизвестные, начиная с последнего уравнения.

Пример 4.5. Используя метод Гаусса с выбором главного элемента решить СЛАУ из примера 4.1.

Решение будет представлено в табличной форме, главные элементы обведены на каждом шаге кружком.

Таблица 4.2

Метод Гаусса с выбором главного элемента

	k	m_i	Коэффициенты при неизвестных			b	c
			x_1	x_2	x_3		
Прямой ход	0	0	3	–1	0	5	7
		–1/4	–2	1	1	0	0
		–1	2	–1	**4**	15	20
	1	–1	**3**	–1	–	5	7
		5/6	–5/2	5/4	–	–15/4	–5
			–	–	–	–	–
	2		–	5/12	–	5/12	5/6
			–	–	–	–	–
			–	–	–	–	–
Обратный ход				$x_2 = 1$			$\overline{x}_2 = 2$
			$x_1 = 2$				$\overline{x}_1 = 3$
					$x_3 = 3$		$\overline{x}_3 = 4$

Пояснения к таблице:

Пусть $k = 0$.

Максимальный (главный) элемент $a_{33} = 4$. Следовательно, главная строка – третья, $p = 3$, $q = 3$.

Главная (или ведущая) строка – первая, её оставляем без изменения.

Для первой и второй строк по формуле (4.9) найдем сомножители:

$$m_1 = -(0/4) = 0;$$
$$m_2 = -(1/4) = -1/4.$$

Перепишем первую строку матрицы, производя следующие преобразования: к каждому элементу первой строки (включая свободный член) добавляем стоящий под ним в матрице коэффициентов элемент главной (третьей) строки, умноженный на m_1. Например

$$b_1 = 5 + (15) \cdot (0) = 5.$$

Аналогичны преобразования для второй строки. Например:

$$a_{22} = 1 + (-1) \cdot (-1/4) = 5/4.$$

При выполнении обратного хода составляем систему из главных (выделенных) строк и, начиная с последнего уравнения, находим неизвестные:

$5/15 \cdot x_2 = 5/12$, т.е. $x_2 = 1$.

$3 \cdot x_1 + (-1) \cdot 1 = 5$, следовательно, $x_1 = 2$ и т.д.

Столбец контрольных сумм позволяет осуществлять такие же проверки, как и в схеме единственного деления и на прямом, и на обратном ходе. Действия со столбцом контрольных сумм приведены в табл. 4.2.

Замечания:

1. Метод Гаусса эффективен для СЛАУ общего вида с плотно заполненной матрицей. Для матриц особого вида (диагональных, симметричных) разработаны специальные методы.

2. При реализации на ЭВМ метод Гаусса используется для систем размерностью до 10^3. Для систем более высоких порядков используются итерационные методы.

4.5. Итерационные методы решения СЛАУ

При большом числе неизвестных линейной системы метод Гаусса и его модификации, дающие точное решение, становятся сложными. В этих условиях для получения решения можно использовать итерационные методы. Решение в этих методах получается как предел последовательных приближений к решению $x^{(0)}$, $x^{(1)}$, …, $x^{(k)} \to x^{т}$. Здесь $x^{т}$ – точное решение системы; $x^{(0)}$, $x^{(1)}$, … – вектора решений, полученные на 0, 1, …, k-й итерациях.

Для реализации итерационных методов необходимо:

1. Начальное приближение $x^{(0)}$.

2. Итерационная формула, определяющая последовательное приближение через предыдущее, $x^{(k)} = \varphi\,(x^{(k-1)})$.

3. Точность и условие проверки на точность $\left| x^{(k)} - x^{(k-1)} \right| \le \varepsilon$.

Считается, что решение найдено, если выполняется условие 3 и доказана сходимость последовательности векторов $x^{(0)}, x^{(1)}, \ldots, x^{(k)}$.

Для получения итерационной формулы СЛАУ в векторно-матричной форме (4.2)

$$\mathrm{A} \cdot \overline{x} = \overline{b}$$

преобразуем к виду, разрешенному относительно $\overline{x}$,

$$\mathrm{A} \cdot \overline{x} - \overline{x} = \overline{b} - \overline{x},$$

вынесем $\overline{x}$ в левой части

$$\overline{x} \cdot (\mathrm{A} - \mathrm{E}) = \overline{b} - \overline{x},$$

отсюда

$$\overline{x} = \underbrace{(\mathrm{E} - \mathrm{A})}_{\alpha} \cdot \overline{x} + \underbrace{\overline{b}}_{\beta}.$$

С учетом введенных обозначений

$$\overline{x} = \alpha \cdot \overline{x} + \overline{\beta}.$$

Справа подставляем предыдущее приближение, слева получаем последующее:

$$\overline{x}^{k} = \alpha \cdot \overline{x}^{k-1} + \overline{\beta}. \qquad (4.10)$$

Такова структура любой итерационной формулы решения СЛАУ. В зависимости от конкретного вида переменных α и $\overline{\beta}$ существует несколько методов решения СЛАУ. Наиболее известные из них – метод Якоби (простых итераций) и метод Зейделя.

4.5.1. Метод простых итераций (Якоби)

Для получения итерационных формул метода простых итераций вновь перепишем СЛАУ в развернутом виде, т.е. формулу (4.1):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots; \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n. \end{cases}$$

Проведем с системой следующие преобразования: из первого уравнений СЛАУ неизвестное выразим x_1, из второго – x_2, из последнего n-го уравнения – x_n. В результате получим систему

$$x_x = (b_1 - a_{12} \cdot x_2 - a_{13} \cdot x_3 - ... - -a_{1n} \cdot x_n) / a_{11};$$

$$x_2 = (b2 - a_{21} \cdot x_1 - a_{23} \cdot x_3 - ... - -a_{2n} \cdot x_n) / a_{22};$$

.....;

$$x_n = (b_n - a_{11} \cdot x_1 - a_{12} \cdot x_2 - ... - -a_{1n-1} \cdot x_{n-1}) / a_{nn}.$$

Предполагаем, что вектор начальных приближений $x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}$ задан. Далее, как обычно в итерационных методах, в левую часть уравнений подставляем предыдущее решение, в правой части получаем последующее. Запишем итерационные формулы для *k*-й итерации:

$$x_1^{(k)} = (b_1 - a_{12} \cdot x_2^{(k-1)} - a_{13} \cdot x_3^{(k-1)} - ... - -a_{1n} \cdot x_n^{(k-1)}) / a_{11};$$

$$x_2^{(k)} = (b2 - a_{21} \cdot x_1^{(k-1)} - a_{23} \cdot x_3^{(k-1)} - ... - -a_{2n} \cdot x_n^{(k-1)}) / a_{22};$$

.....;

$$x_n^{(k)} = (b_n - a_{11} \cdot x_1^{(k-1)} - a_{12} \cdot x_2^{(k-1)} - ... - -a_{1n-1} \cdot x_{n-1}^{(k-1)}) / a_{nn},$$

или в общем виде –

$$x_i^{(k)} = \frac{b_i}{a_{ii}} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{a_{ij}}{a_{ii}} \cdot x_j^{(k-1)}, \; i = 1,n; \qquad (4.11)$$

$$k = 1, 2, \ldots \text{ до тех пор, пока } \left| x^{(k)} - x^{(k-1)} \right| \leq \varepsilon. \qquad (4.12)$$

Введенные в формулу (4.10) матрица α и вектор $\bar{\beta}$ для метода Якоби, с учетом (4.11), имеют вид

$$\alpha = \begin{pmatrix} 0 & \frac{-a_{12}}{a_{11}} & \frac{-a_{13}}{a_{11}} ... & \frac{-a_{13}}{a_{11}} \\ \frac{-a_{21}}{a_{22}} & 0 & ... & \frac{-a_{n2}}{a_{22}} \\ . & . & . & . \\ \frac{-a_{n1}}{a_{nn}} & \frac{-a_{n2}}{a_{nn}} & ... & 0 \end{pmatrix}; \qquad \beta = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ ... \\ \frac{bn}{a_{nn}} \end{pmatrix}.$$

Сходимость метода простых итераций определяется теоремой.

Теорема. Для того чтобы метод простых итераций сходился при любом начальном значении вектора $\bar{x}^0$, достаточно, чтобы какая-нибудь норма матрицы α была меньше единицы

$$\|\alpha\| < 1. \qquad (4.13)$$

Напомним понятие нормы матрицы.

Определение. Нормой матрицы α называют такое вещественное число $\|\alpha\|$, которое удовлетворяет следующим аксиомам:

1. $\|\alpha\| > 0$, если $\alpha \neq 0$.
2. $\|\lambda \cdot \alpha\| = |\lambda| \cdot \|\alpha\|$, где λ – скаляр (число).
3. $\|\alpha + \beta\| < \|\alpha\| + \|\beta\|$.
4. $\|\alpha \cdot \beta\| < \|\alpha\| \cdot \|\beta\|$.

Наиболее употребимы в математическом анализе следующие нормы:

1. $\|\alpha\|_1 = \max\limits_i \sum\limits_{j=1}^{n} |\alpha_{ij}|$.
2. $\|\alpha\|_2 = \max\limits_j \sum\limits_{i=1}^{n} |\alpha_{ij}|$.
3. $\|\alpha\|_E = \sqrt{\sum\limits_{j=1}^{n}\sum\limits_{i=1}^{n} \alpha_{ij}^2}$.

Нормы 1 и 2 отличаются лишь суммированием по строкам или столбцам. Третья норма называется евклидовой.

В методе простой итерации удобно воспользоваться нормой 1 или 2. Достаточные условия сходимости метода (4.13) для этих норм и с учетом вида матрицы α будут выглядеть так:

$$\max_i \sum_{j=1}^{n} \left|\frac{\alpha_{ij}}{\alpha_{ii}}\right| < 1;$$

$$\max_j \sum_{i=1}^{n} \left|\frac{\alpha_{ij}}{\alpha_{jj}}\right| < 1,$$

или в более удобном виде:

$$\begin{aligned} |a_{ii}| &> \sum_{j=1}^{n} |a_{ij}|; \\ |a_{jj}| &> \sum_{i=1}^{n} |a_{ij}|. \end{aligned} \qquad (4.14)$$

Формулы (4.14) определяют условия сходимости метода Якоби. Означают они диагональное преобладание элементов матрицы системы, т.е. диагональный элемент каждой строки системы по модулю больше,

чем сумма остальных элементов в строке, также взятых по модулю.

Далеко не каждая СЛАУ удовлетворяет условиям сходимости (4.14). Например, для СЛАУ из примера 4.1

$$\begin{pmatrix} 3 & -1 & 0 \\ -2 & 1 & 1 \\ 2 & -1 & 4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 15 \end{pmatrix};$$

проверим условия сходимости;

$$|3| > |-1| + |0|, \qquad 3 > 1;$$
$$|1| < |-2| + |1|, \qquad 1 < 3;$$
$$|4| > |2| + |-1|, \qquad 4 > 3,$$

т.е. второе уравнение не удовлетворяет условиям сходимости и данной СЛАУ нельзя найти корни методом Якоби. Таких систем, для которых условия сходимости не выполняются, большое количество. Поэтому существует методика приведения СЛАУ к сходящемуся виду. Для приведения матрицы А к сходящемуся виду практически поступают следующим образом:

1. Из заданной системы выделяют уравнения, модули коэффициентов которых больше суммы модулей остальных коэффициентов в этих уравнениях.

2. Каждое из выделенных уравнений записывается в такую строку новой системы, чтобы этот максимальный элемент стал диагональным.

3. Из оставшихся и выделенных уравнений системы составляются линейно независимые между собой линейные комбинации уравнений так, чтобы выполнялся принцип 2, были заполнены все свободные строки новой системы и были использованы все уравнения исходной системы.

Пример 4.6. Привести систему

$$\begin{cases} x_1 + 5 \cdot x_2 - x_3 = 8; \\ x_1 - x_2 + 3 \cdot x_3 = 8; \\ 6 \cdot x_1 + 3 \cdot x_2 - 17 \cdot 4x_3 = -39, \end{cases}$$

к сходящемуся виду и вычислить первые четыре итерации методом Якоби при нулевых начальных условиях.

Каждое уравнение в исходной системе отметим соответственно буквами а)–б)–в). В соответствии с приведенной выше методикой, проанализируем каждое уравнение, начиная с первого.

Первое уравнение а) может быть поставлено в новой сходящейся

системе на вторую строку (римские цифры): 5 > 1 + 1.

Второе уравнение б) в новой системе может стать третьим уравнением (III): 3 > 1 + 1.

Третье уравнение 3) могло бы быть в новой системе третьим уравнением, однако его место уже занято. А нам требуется в новой системе первое уравнение. Оно может быть получено на основании третьей рекомендацией методики, при этом обязательно нужно использовать незадействованное пока уравнение в) и любую линейную комбинацию (сложение и умножение на число) других уравнений системы. Таких комбинаций может быть множество – это своего рода искусство. По логике приведения первый коэффициент системы максимальный в уравнении в) и равен 6, но коэффициент 17 должен быть компенсирован и по знаку, и значением. Это может быть достигнуто добавлением уравнения б) и умножением его на скаляр. В результате имеем формулу преобразования I = в) + 6 · б). Полученная система имеет вид

$$\begin{cases} 12 \cdot x_1 - 3 \cdot x_2 + x_3 = 9; \\ x_1 + 5 \cdot x_2 - x_3 = 8; \\ x_1 - x_2 + 3 \cdot x_3 = 8. \end{cases}$$

У системы диагональное преобладание – для всех уравнений, следовательно она может быть решена методом простых итераций.

Приведем систему к виду, удобному для итераций (4.11), по схеме Якоби:

$$x_1^{(k+1)} = \frac{9}{12} + \frac{1}{3} \cdot x_2^{(k)} - \frac{1}{12} x_3^{(k)};$$

$$x_2^{(k+1)} = \frac{8}{5} - \frac{1}{5} \cdot x_1^{(k)} + \frac{1}{5} x_3^{(k)};$$

$$x_3^{(k+1)} = \frac{8}{3} - \frac{1}{3} \cdot x_1^{(k)} + \frac{1}{3} x_2^{(k)}.$$

Вычислим первую итерацию, подставляя нулевые начальные условия в правые части уравнений. Затем полученные в левой части корни вновь подставим в правые части и таким образом вычислим первые 4 итерации:

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	0	0	0
1	9/12	8/5	8/3
2	0,9277	1,9833	2,9499
3	1	2,0044	3,0164
4	0,999	2,0003	3,0013

Отметим, что точное решение системы – $x_1 = 1$, $x_2 = 2$, $x_3 = 3$.

Замечание. Число итераций метода Якоби, дающих решение с точностью ε, определяется следующей формулой:

$$\left\| \overline{x}^{\mathrm{T}} - \overline{x}^{(k)} \right\| \le \frac{\|\alpha\|^{k+1}}{1-\|\alpha\|} \cdot \|\beta\| < \varepsilon. \qquad (4.15)$$

Пример 4.7. Вычислить число итераций метода Якоби, необходимых для получения решения системы из примера 4.6 с точностью $\varepsilon = 10^{-3}, 10^{-4}, 10^{-6}$.

Найдем нормы матрицы α столбца свободных членов β:

$$\|\beta\| = \max\{9/12,\ 8/5,\ 8/3\} = 8/3;$$

$$\|\alpha\| = \max\begin{Bmatrix} 1/4+1/12 = 1/3 \\ 1/5+1/5 = 2/5 \\ 1/3+1/3 = 2/3 \end{Bmatrix} = 2/3.$$

Подставим вычисленные нормы в формулу (4.15):

$$\frac{(2/3)^{k+1}}{1-2/3} \cdot 8/3 \le \varepsilon;$$

$$(2/3)^{k+1} \cdot 8 \le \varepsilon;$$

$$\frac{2}{3}^{k+1} \le \frac{1}{8} \cdot \varepsilon;$$

$$(k+1) \cdot \lg\frac{2}{3} \le \lg\frac{1}{8}\varepsilon;$$

$$\varepsilon = 10^{-3}, \quad \varepsilon = 10^{-4}, \quad \varepsilon = 10^{-6};$$

$$k = 21, \qquad k = 26, \qquad k = 38.$$

Алгоритмизация метода Якоби

Шаг 1. Задание точности вычисления корней ε,
начального приближения $\overline{x}^0$.
Счетчик числа итераций $k = 0$.
Вспомогательная переменная $m = 1$.

Шаг 2. Цикл $m > \varepsilon$.

Шаг 3. Цикл $i = 1, \ldots, n$.

Шаг 4. $m = 0$.

$S = 0$ (переменная для накопления суммы).

Шаг 5. Цикл $j = 1, \ldots, n$.

$S = S + A_{i,j} \cdot x_{j,k}$ для $i \neq j$.

Конец цикла j.

Шаг 6. $x_{j,k+1} = (b_i - S)/A_{i,i}$ для $i \neq j$.

Конец цикла j.

Шаг 7. Если $|b - a| > m$, то $m = |b - a|$.

Конец цикла i.

Шаг 8. $k = k + 1$.

Конец цикла m.

Шаг 9. Печать результатов: корни системы $\bar{x}$, число итераций k.

4.5.2. Метод Зейделя

Метод Зейделя – это модификация метода простых итераций. Основное отличие заключается в том, что при вычислении k-го приближения корня $x_i^{(k)}$ учитывается уже вычисленные на k-й итерации значения $x_1^{(k)}$, $x_2^{(k)}$, …, $x_{i-1}^{(k)}$. Значения $x_{i+1}^{(k-1)}$, $x_{i+2}^{(k-1)}$, …, $x_n^{(k-1)}$ берутся, вычисленные на предыдущем шаге:

$$
\begin{aligned}
x_1^{(k)} &= \frac{b_1}{a_{1i}} - \sum_{j=2}^{n} \frac{a_{1j}}{a_{11}} \cdot x_j^{(k-1)}; \\
x_2^{(k)} &= \frac{b2}{a_{2i}} - \frac{a_{21}}{a_{22}} \cdot x_1^{(k)} - \sum_{j=3}^{n} \frac{a_{2j}}{a_{22}} \cdot x_j^{(k-1)}; \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots; \\
x_n^{(k)} &= \frac{b_n}{a_{ni}} - \sum_{j=n}^{n-1} \frac{a_{nj}}{ann} \cdot x_j^{(k)}.
\end{aligned}
\qquad (4.16)
$$

Условия сходимости метода Зейделя те же, что и в методе Якоби (4.12).

Замечание. Сходимость метода Зейделя обычно выше, чем у метода Якоби. Однако бывают системы, сходящиеся при решении методом Якоби и расходящиеся по методу Зейделя. Бывают ситуации прямо противоположные, хотя достаточные условия сходимости в обоих методах одинаковые. Некоторые дополнительные гарантии сходимости дает теорема.

Теорема. Пусть **A** – вещественная, симметричная положительно определенная матрица, тогда метод Зейделя сходится при любых начальных приближениях.

Напомним, что в положительно определенной матрице положительны диагональные элементы и определители главных миноров. Способ приведения к симметричной, положительно определенной матрице системы можно выбрать такой: $A^T \cdot A \cdot \bar{x} = A^T \cdot \bar{b}$.

4.6. Решение задач линейной алгебры в ППП MathCad

Для решения СЛАУ (нахождения корней) возможно использование вычислительных блоков:

Given…Find;

Given…Minner.

Использование вычислительных блоков подробно описано в разд. 3, а на рис. 4.4 приведен фрагмент программы для решения СЛАУ из примера 4.1.

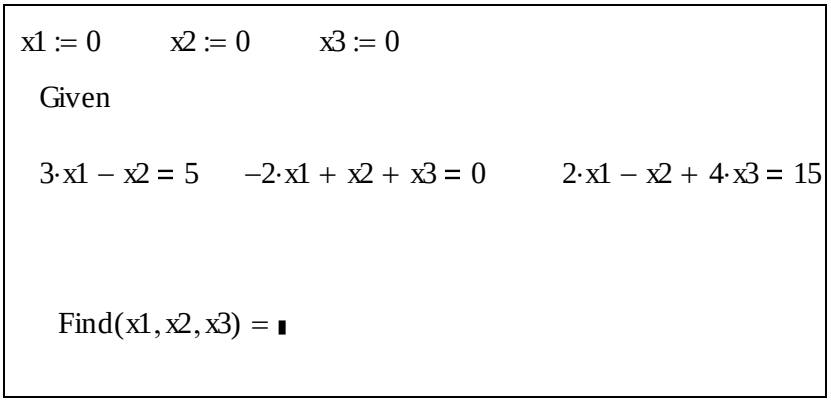


Рис. 4.4. Решение СЛАУ с помощью вычислительных блоков

Отметим, что вычислительные блоки не допускают использования переменных с индексом, поэтому элементы вектора с понижающим индексом x_1, x_2, x_3 заменены простыми переменными $x1$, $x2$, $x3$.

Пакет программ MathCad имеет встроенную функцию решения СЛАУ:

Описание функции	Аргументы функции
lsolve(*A*,*b*) Возвращается вектор решения ***x*** такой, что ***Ax* = *b*.**	***A*** – квадратная матрица. ***b*** – вектор, имеющий столько же строк, сколько строк в матрице ***A***.

На рис. 4.5 приведен фрагмент решения той же СЛАУ с помощью функции **lsolve.** Здесь показано нахождение определителя и обратной матрицы.

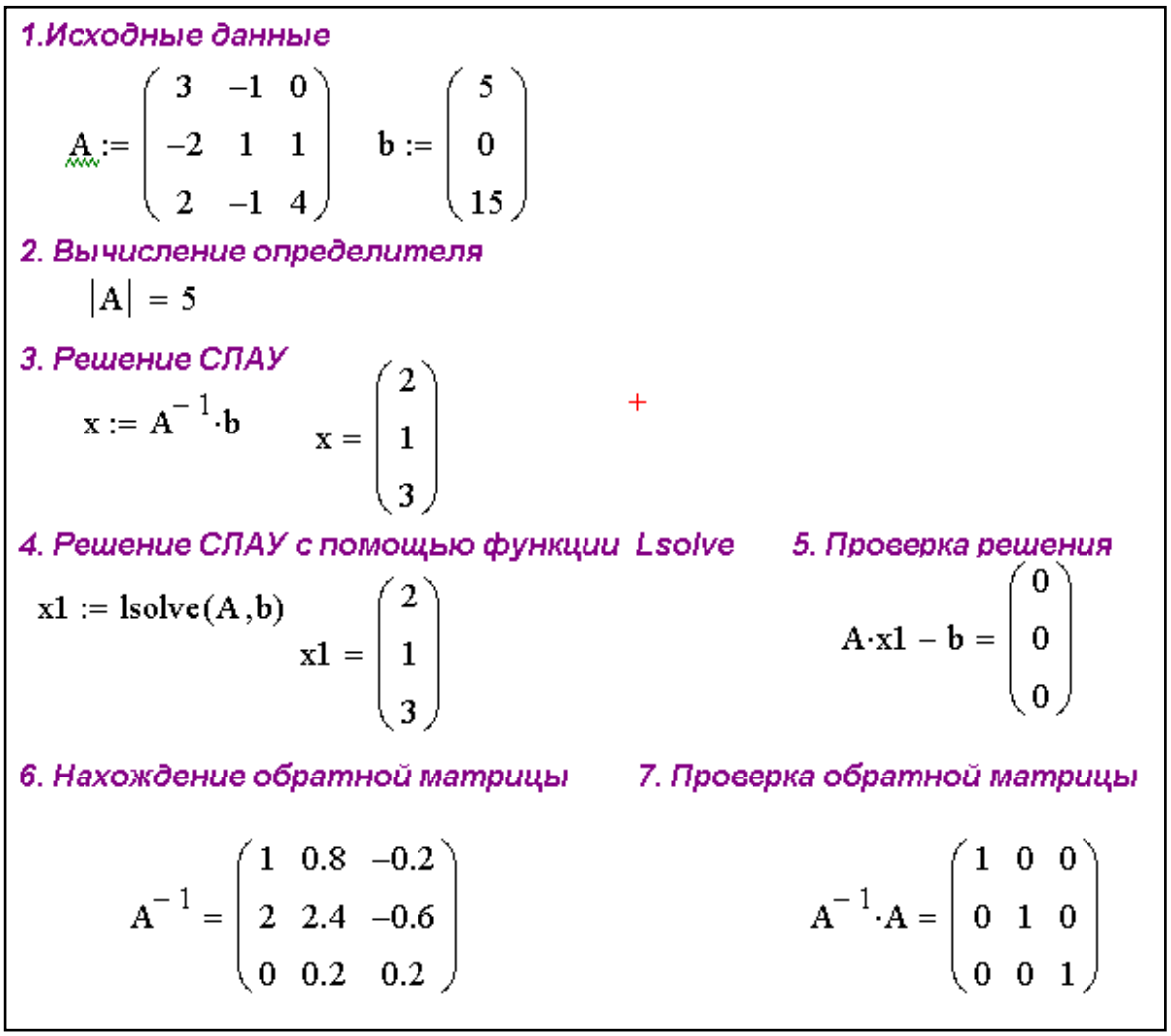


Рис. 4.5

Заслуживает внимания встроенная функция ППП MathCad, приводящая матрицу к ступенчатому виду **rref**. Это приведение реализует метод Гаусса, его прямой и обратный ход. Функция работает с расширенной матрицей, состоящей из матрицы системы A и вектора свободных членов *b*. Её результатом является также расширенная матрица, последним столбцом которой являются значения полученных корней. На рис. 4.6 приведен рабочий документ ППП MathCAD решения примера 4.1 методом Гаусса. В программе использован ряд дополнительных функций MathCAD. Они описаны ниже.

Описание функции	Аргументы функции
rref(*A*) Возвращается ступенчатая форма матрицы ***A***	***A*** – расширенная матрица системы
augment (*A*, *b*) Соединение двух матриц ***A*** и ***b***, имеющих одинаковое количество строк	***A*** – матрица системы ***b*** – матрица (вектор)
submatrix (*A*, *r*1, *r*2, *c*1, *c*2) Возвращается матрица, состоящая из всех элементов со строки ***r*1** по ***r*2** и столбцов с ***c*1** по ***c*2**.	***A*** – матрица системы ***r*1 < *r*2** – номера возвращаемых строк ***c*1< *c*2** – номера возвращаемых столбцов

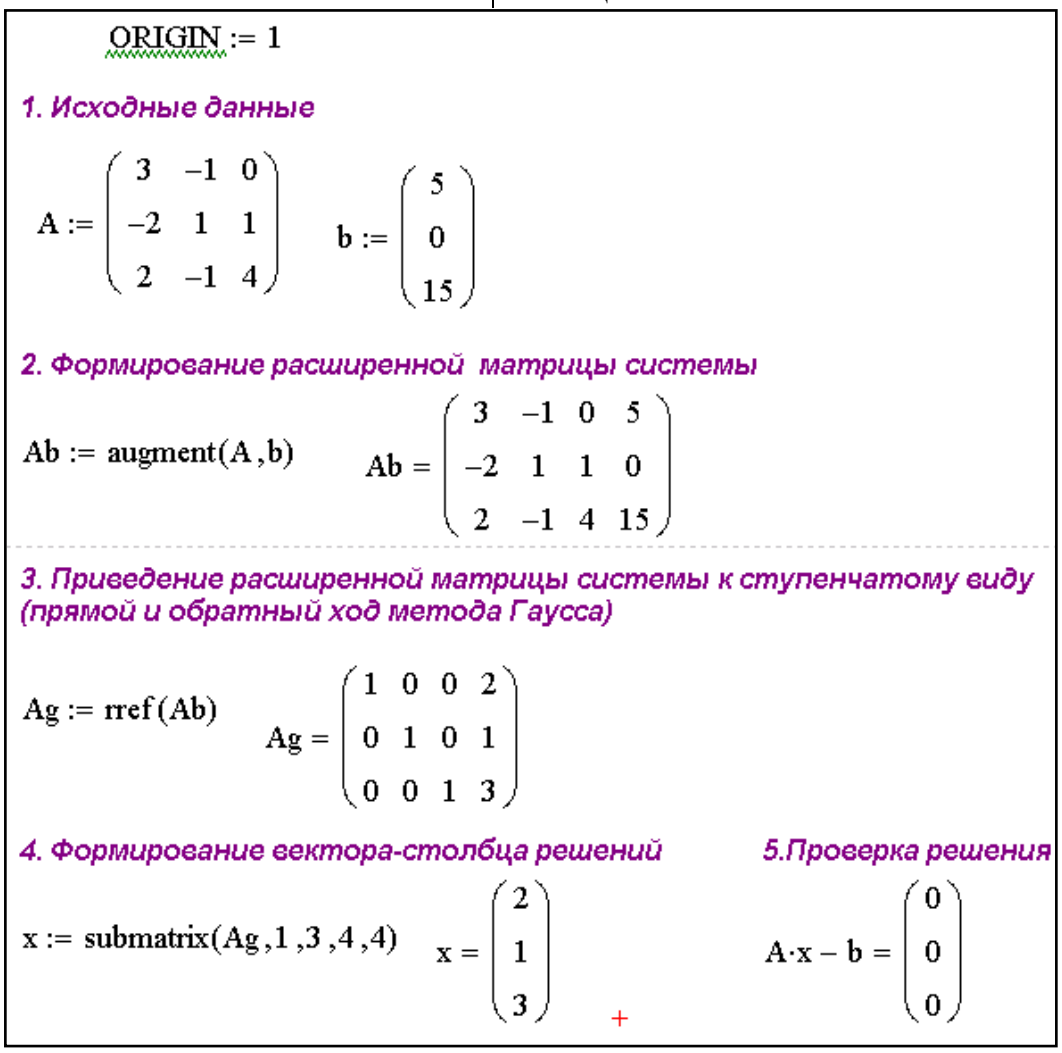


Рис. 4.6

ППП MathCAD позволяет реализовать итерационные методы решения СЛАУ. На рис. 4.7 приведен рабочий фрагмент программы решения

СЛАУ из примера 4.6. Программа ориентирована на сходящуюся систему (матрицы ***A*** и ***b***) и в виде, удобном для итераций (матрицы α и β). В программе условия сходимости проверяются функциями ППП MathCAD **norm1(*A*)**, **norm2(*A*)**, **norme(*A*)**, которые возвращают норму 1, 2 или евклидову матрицы ***A***.

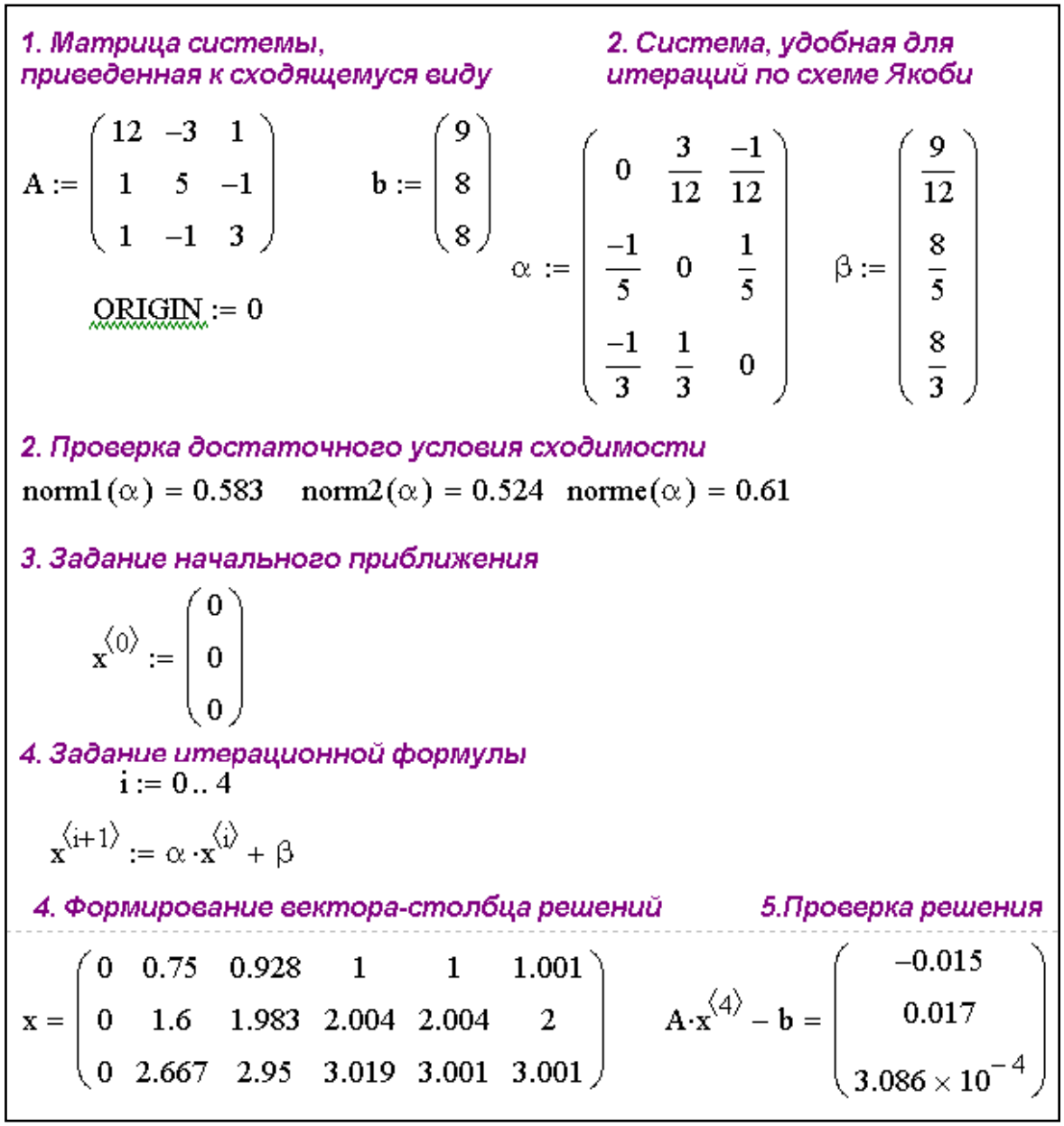


Рис. 4.7

Результаты, полученные при ручном счете в примере 4.6, и в программе полностью совпадают на всех итерациях.

В заключении следует отметить, что все приведенные в разделе алго-

ритмы могут быть реализованы в ППП MathCAD и с помощью программ пользователя, написанных на встроенном в пакете языке программирования. Эти программы просты и значительно дополняют возможности ППП MathCAD решения математических и инженерных задач.

Вопросы для самопроверки

1. Что значит решить систему уравнений?
2. Какие методы относятся к прямым и приближенным методам решения систем линейных алгебраических уравнений?
3. Какие вы знаете группы методов решения систем линейных алгебраических уравнений?
4. В чем заключается основная идея метода исключения Гаусса для решения систем линейных алгебраических уравнений? В чем состоят отличия метода Гаусса со схемой единственного деления и с выбором главного элемента?
5. Как организовать контроль прямого и обратного хода метода Гаусса?
6. Назовите задачи, сопутствующие реализации метода Гаусса.
7. В чем заключается суть метода простой итерации и Зейделя для решения систем линейных алгебраических уравнений? В чем состоит отличие этих методов?
8. Назовите достаточные условия сходимость итерационных методов решения систем линейных алгебраических уравнений.
9. Как привести систему к виду с диагональным преобладанием?
10. Дайте сравнительную оценку методам решения систем линейных алгебраических уравнений.
11. В каких случаях целесообразно использовать итерационные методы решения систем линейных алгебраических уравнений?
12. Что влияет на скорость сходимости итерационного процесса?
13. Сформулируйте достаточные условия сходимости метода Зейделя для решения систем линейных алгебраических уравнений.
14. Дайте определение и приведите примеры нормы матрицы.

5. РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Часто инженерные и научные задачи связаны с решением дифференциальных уравнений, так именно дифференциальные уравнения описывают динамику физических явлений и процессов в технических устройствах.

К сожалению, для многих практически задач получить точное решение дифференциальных уравнений оказывается затруднительным или невозможным. Однако решить подобные сложные задачи можно численно с использованием ЭВМ.

Дифференциальные уравнения устанавливают связь между независимыми переменными, функциями и их производными.

Если функция зависит от одной переменной, то дифференциальное уравнение называется обыкновенным.

В том случае, если искомая функция зависит от нескольких переменных, дифференциальное уравнение будет уравнением в частных производных.

Будем рассматривать обыкновенное дифференциальное уравнение (ОДУ), содержащее одну независимую переменную и производные по ней. ОДУ имеют множество решений. Для отыскания единственного решения необходимы дополнительные условия. Если дополнительные условия заданы при одном значении независимой переменной x, то условия называются начальными и имеет место задача Коши. Количество начальных условий соответствует порядку дифференциального уравнения.

Если дополнительные условия заданы в двух или более значениях независимой переменной, то задача называется краевой, а условия – граничными.

При решении этих задач используются разные методы и алгоритмы.

5.1. Задача Коши

Пусть задано обыкновенное дифференциальное уравнение 1-го порядка с начальным условием:

$$\begin{aligned} y'(x) &= f(x, y); \\ y(a) &= y_0. \end{aligned} \tag{5.1}$$

Требуется найти функцию $y(x)$ при $x \in [a, b]$, удовлетворяющую как

указанному уравнению, так и начальному условию.

Данная задача носит название задачи Коши.

Все методы решения задачи Коши делятся на аналитические и приближенные. Аналитические методы основаны на интегрировании исходных дифференциальных уравнений, поэтому искомую функцию $y(x)$ часто называют, при решении дифференциальных уравнений, интегральной кривой. С помощью аналитических методов удается решить относительно узкий круг задач.

Приближенные методы решения задачи Коши дают только численное решение. При этом значения интегральной кривой $y(x)$ представлены в виде таблицы значений $y_k = y(x_k)$, на сетке узлов x_k, т.е. интервал $[a, b]$ разбивается на n частей и образуется сетка узлов x_k:

$$a = x_0 < x_1 < x_2 < \ldots < x_i < \ldots < x_n = b,$$

где $x_k = x_{k-1} + h$, $k = 1, 2, \ldots, n$, или $x_k = a + k \cdot h$;

$h = \dfrac{b-a}{n}$ – шаг сетки, чаще всего постоянный.

Приближенное значение функции $y(x_k)$ возможно найти лишь в этих внутренних точках интервала, т.е. на сетке узлов

x_0	x_1	x_2	…	x_k	…	x_n
y_0	y_1	y_2	…	y_k	…	y_n

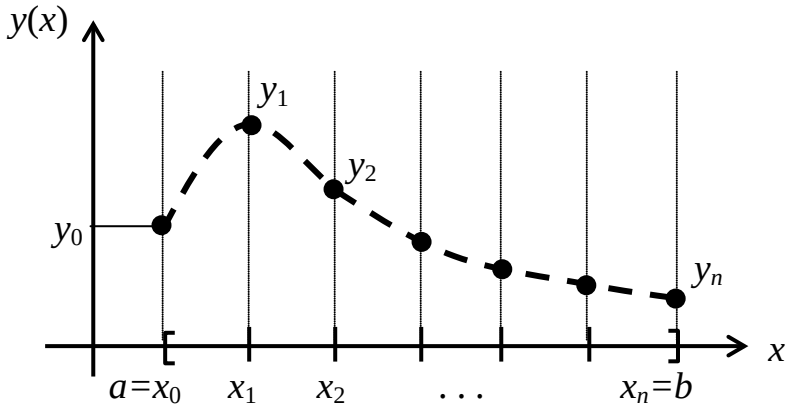


Рис. 5.1. Численное решение задачи Коши

Основным недостатком всех численных методов, в т.ч. и задачи Коши, является то, что методы дают лишь частное решение и не позволяют построить общее. Кроме того, численные методы можно применять лишь к устойчивым задачам, а задача Коши – неустойчива,

т.е. малым изменениям исходных данных не всегда соответствуют малые изменения в решении. Например, пусть дано дифференциальное уравнение и два варианта начальных условий – а) и б), отличающихся друг от друга на небольшую величину:

$$y'(x) = 2 \cdot x, \qquad x \in [0,100];$$

а) $y_1(0) = 0;$

б) $y_2(0) = 10^{-6}.$

Проинтегрируем дифференциальное уравнение, учитывая постоянную интегрирования

$$y(x) = x^2 + Ce^x.$$

Для нахождения константы C используем начальные условия:

а) $0^2 + C \cdot e^0 = 0, \qquad C = 0;$

б) $0^2 + C \cdot e^0 = 10^{-6}, \quad C = 10^{-6}.$

С учетом полученных констант интегрирования найдем решение дифференциального уравнения в правом конце интервала при x = 100:

а) $y(100) = 100^2 + 0, \qquad y(100) = 10^4;$

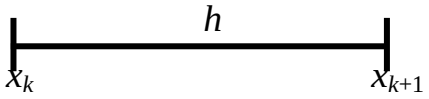
б) $y(100) = 100^2 + 10^{-6}e^{100}, \qquad y(100) = 2,7 \cdot 10^{37}.$

Как видно, решение значительно отличается при небольшом отличии начальных условий.

5.2. Методы Рунге – Кутта

Эти методы позволяют строить вычислительные схемы различного порядка точности. Методы весьма удобны для реализации на ЭВМ. Для нахождения значения искомой функции в последующей точке y_{k+1} нужна информация только о предыдущей точке (y_k, x_k). Платой за столь малую информативность является большая трудоемкость методов. Для получения последующего значения искомой функции требуется несколько раз вычислить значение производной $y'(x)$, т.е. обращаться к правой части дифференциального уравнения. При громоздкой правой части решение дифференциального уравнения становится сложным.

Основная идея метода



Для получения формулы Рунге–Кутта проинтегрируем дифференциаль-

ное уравнение (5.1) в интервале $[x_k, x_{k+1}]$. Получим

$$y(x_{k+1}) = y(x_k) + \int_{x_k}^{x_{k+1}} f\left[t, y(t)\right]\mathrm{d}t,$$

или

$$y_{k+1} = y_k + \Delta y_k . \tag{5.2}$$

Формула (5.2) носит название формулы Рунге – Кутта. Член Δy_k в формуле определяет численное значение интеграла на участке $[x_k, x_{k+1}]$, где $x_{k+1} = x_k + h$. Для практического применения Δy_k раскладывают в ряд Тейлора в окрестности точки x_k и заменяют полиномом некоторой степени:

$$\Delta y_k = \sum_{i=1}^{r} P_i k_i , \tag{5.3}$$

где P_i – коэффициенты разложения;

k_i – полиномы вида

$k_1 = h \cdot f(x_k, y_k)$;

$k_2 = h \cdot f(x_k + \alpha_2 h, y_k + \beta_{21} k_1)$;

$k_3 = h \cdot f(x_k + \alpha_3 h, y_k + \beta_{31} k_1 + \beta_{32} k_2)$;

. . . . ;

$k_r = h \cdot f(x_k + \alpha_r h, y_k + \beta_{r1} k_1 + \beta_{r2} k_2 + \ldots + \beta_{r\,r-1} k_{r-1})$.

Таким образом, Δy_k раскладывается по выбранной системе параметров P_i, $i = \overline{1,\ldots,r}$:

P_1	P_2	…	P_r	
α_2	α_3		α_r	k_1
β_{21}				k_2
β_{31}	β_{32}			k_3
.	.	.	.	.
β_{r1}	β_{r2}		$\beta_{r\,r-1}$	k_r

При удачном выборе системы параметров можно сколь угодно близко подойти к точному решению. В результате такого приближения появляется погрешность

$$R(h) = \Delta y_k - \sum_{i=1}^{r} P_i k_i .$$

Считается, что в методах Рунге – Кутта порядок ошибки равен *r*, т.е.

$$R(h) = \mathrm{O}(h^{r+1}). \tag{5.4}$$

В зависимости от степени полинома различают методы Рунге – Кутта 1, 2, 4 степени.

5.2.1. Метод Эйлера (Рунге – Кутта 1-го порядка)

Запишем формулы (5.2) для $r = 1$:

$$\begin{aligned} &y_{k+1} = y_k + \Delta y_k; \\ &\Delta y_k = P_1 k_1; \\ &P_1 = 1, \end{aligned}$$

или

$$\begin{aligned} &y_{k+1} = y_k + h \cdot f(x_k, y_k); \\ &x_k = x_{k-1} + h. \end{aligned} \tag{5.5}$$

Формула (5.5) – формула Эйлера. Она позволяет, начиная с начальных условий $\{x_0, y_0\}$, получить последовательность приближений к решению дифференциального уравнения $\{y_1, y_2, \ldots, y_n\}$ с шагом h и, таким образом, решить задачу Коши.

Формула Эйлера (5.5) может быть получена также другим способом. В основе ее лежит разложение функции $y(x)$ в ряд Тейлора в окрестности точки x_0:

$$y(x) = y(x_0) + (x - x_0) \cdot y'(x_0) + \frac{(x - x_0)^2}{2} \cdot y''(x_0) + \ldots$$

В точке $x_0 + h$, при малых значениях h, можно ограничиться двумя членами ряда Тейлора. Тогда

$$y(x_0 + h) = y(x_0) + h \cdot y'(x_0) + O(h^2),$$

где $O(h^2)$ – бесконечно малая величина порядка h^2, определяющая погрешность метода. Далее мы ее опускаем. Заменим производную $y'(x_0)$, на правую часть уравнения (5.1):

$$y_1 = y_0 + h \cdot f(x_0, y_0),$$

где $y_0 = y(x_0)$, $y_1 = y(x_1) = y(x_0 + h)$.

Далее этот процесс можно распространить на x_2, x_3, …И, таким образом, общая формула будет иметь вид (5.5).

Иногда метод Эйлера называют методом ломаных. Это связано

с его геометрической интерпретацией – рис. 5.2 Искомая функция $y(x)$ заменяется ломаной линией, представляющей собой отрезки касательных к этой функции в узлах x_0, x_1, ...

На рисунке изображены первые два шага решения дифференциального уравнения в точках x_1, x_2. Интегральные кривые 0, 1, 2 описывают точные решения уравнения. Кривая 0 соответствует точному решению задачи Коши, т.к. проходит через начальную точку x_0, y_0. Отрезок AB – это касательная к кривой 0 в точке x_0, y_0, ее наклон характеризуется значением производной $y_0' = f(x_0, y_0)$. Касательная BC уже проводится к другой интегральной кривой *1*, которая получена перенесением интегральной кривой в точку *B*. Таким образом, на каждом шаге решение переходит на другую интегральную кривую. При этом каждый последующий шаг включает, собственно, погрешность самого шага и погрешности всех предыдущих шагов. Определяется погрешность порядком отброшенных членов в разложении Тейлора

$$R(h) = O(h^2). \qquad (5.6)$$

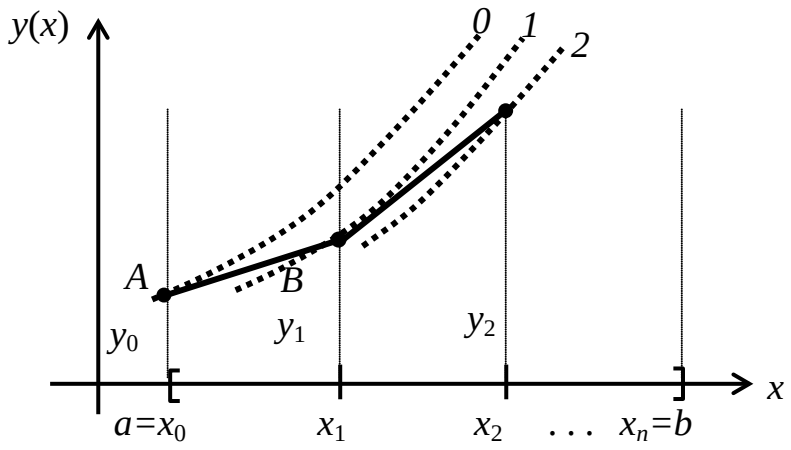


Рис. 5.2. Иллюстрация метода Эйлера

Метод Эйлера имеет довольно большую погрешность вычисления. Кроме того, именно метод Эйлера оказывается неустойчивым к ошибкам в исходных данных. Малые локальные ошибки приводят к значительной глобальной погрешности.

5.2.2. Метод Рунге – Кутта 2-го порядка

Существуют различные модификации метода Эйлера, позволяющие увеличить его точность. Все они основаны на том, что производ-

ную, вычисленную в начале интервала, заменяют на среднее значение производной на данном интервале. Такая замена основана на увеличении количества членов разложения в ряд Тейлора или степени полинома (5.3), заменяющего приращение Δy_k. Если использовать полиномы второго порядка, то получим методы Рунге – Кутта 2-го порядка. Общие расчетные формулы имеют следующий вид:

$$y_{k+1} = y_k + \Delta y_k;$$
$$\Delta y_k = P_1 \cdot k_1 + P_2 \cdot k_2;$$

$$k1 = h \cdot f(x_k, y_k);$$
$$k2 = h \cdot f(x_k + \alpha_2 h, y_k + \beta_{21} k1).$$

В зависимости от соотношения параметров α_2, β_{21} возможны два варианта метода Рунге – Кутта 2-го порядка:

а) метод Эйлера – Коши. В этом методе среднее значение производной оценено в конце интервала $[x_k, x_{k+1}]$. Значения параметров равны соответственно $P1 = 0{,}5$, $P2 = 0{,}5$, $\alpha_2 = 1$, $\beta_{21} = 1$.

При таком наборе параметров

$$\begin{cases} y_{k+1} = y_k + \Delta y_k; \\ \Delta y_k = \dfrac{(k1 + k2)}{2}; \\ k1 = h \cdot f(x_k, y_k); \\ k2 = h \cdot f(x_k + h, y_k + k1). \end{cases} \qquad (5.7)$$

Формулы (5.7) – это аналог формулы трапеций, если сопоставлять с формулами численного интегрирования. Формулы согласуются с разложением в ряд Тейлора функции $y(x)$, включая члены степени h^2. И, таким образом, погрешность метода имеет порядок

$$R(h) = \mathrm{O}(h^3). \qquad (5.8)$$

б) модифицированный метод Эйлера. В этом методе среднее значение производной оценено в середине интервала $[x_k, x_{k+1}]$ и использовано для аппроксимации решения на всем интервале. Значения параметров равны соответственно $P1 = 0$, $P2 = 1$, $\alpha_2 = 0{,}5$, $\beta_{21} = 0{,}5$.

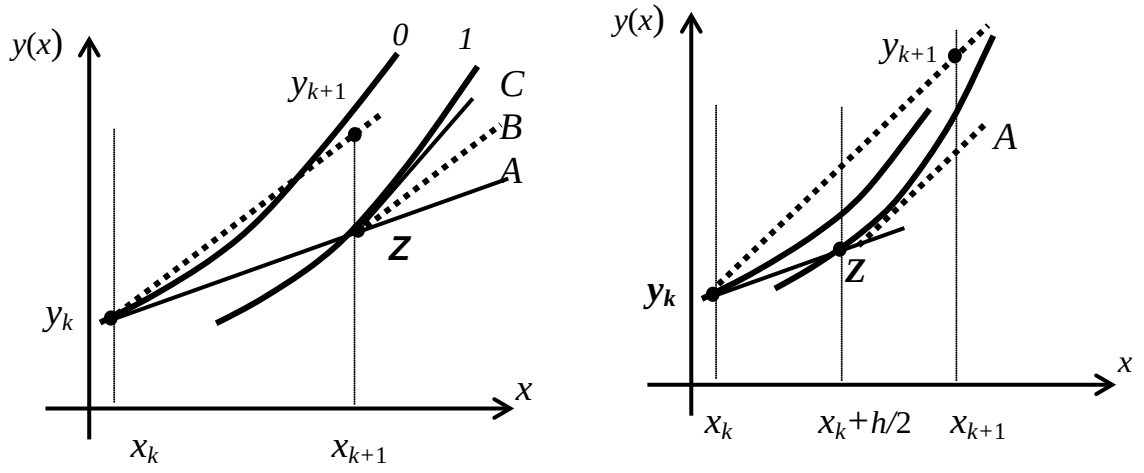
При таком наборе параметров

$$\begin{cases} y_{k+1} = y_k + \Delta y_k; \\ \Delta y_k = k2; \\ \\ k1 = h \cdot f(x_k, y_k); \\ k2 = h \cdot f(x_k + \dfrac{h}{2}, y_k + \dfrac{k1}{2}). \end{cases} \qquad (5.9)$$

Погрешность метода та же и определяется формулой (5.8).

Вариант а) метода Рунге – Кутта 2-го порядка является аналогом формулы трапеции (численное интегрирование), вариант б) – средних прямоугольников.

Геометрическая интерпретация методов представлена на рис. 5.3.



а *б*

Рис. 5.3. Иллюстрация методов Рунге – Кутта 2-го порядка

Для случая а) вначале вычисляется приближенное решение дифференциального уравнения в точке $x_{k+1} = x_k + h$ по формуле Эйлера $y_k + h \cdot f(x_k, y_k)$. Это точка *Z*. В эту точку переносится интегральная кривая *1*, и в точке *Z* строится касательная *ZC*. Прямая *ZB* – это средний наклон (*ZA* + *ZC*)/2. Далее осуществляется параллельный перенос прямой *ZC* в точку (x_k, y_k). Эта линия продолжается до пересечения с линией сетки x_{k+1} – искомое значение y_{k+1}.

Для варианта б) вначале делается половинный шаг $x_k + h/2$, и в этой точке вычисляется грубое решение по формуле Эйлера – точка *Z*. В эту точку также переносится интегральная кривая, к которой строится касательная *ZA*. Прямая параллельно переносится в точку (x_k, y_k), продол-

жение которой и приводит к очередной точке решения y_{k+1}.

Пример 5.1. Найти решение задачи Коши

$$\begin{cases} y' = x^2 + y^2; \\ y(0) = 0 \end{cases}$$

методами Рунге – Кутта 2-го порядка на интервале $x \in [0, 1]$ с шагом $h = 0{,}5$.

Для данного примера $f(x, y) = x^2 + y^2$.

Запишем формулы Рунге – Кутта 2-го порядка для данной функции:

а)

$$y_{k+1} = y_k + \Delta y_k;$$

$$\Delta y_k = \frac{(k1 + k2)}{2};$$

$$k1 = h \cdot (x_k + y_k);$$

$$k2 = h \cdot [(x_k + h)^2 + (y_k + k1)^2].$$

б)

$$y_{k+1} = y_k + \Delta y_k;$$

$$\Delta y_k = k2;$$

$$k1 = h \cdot \left(x_k^2 + y_k^2\right);$$

$$k2 = h \cdot [\left(x_k + \frac{h}{2}\right)^2 + \left(y_k + \frac{k1}{2}\right)^2].$$

а) б)

x_k	$k1$	$k2$	Δy_k	y_k	$k1$	$k2$	Δy_k	y_k	y^T
0,5	0	0,125	0,0625	0,0625	0	0,0313	0,031	0,0313	0,042
1	0,1269	0,5179	0,3222	0,3849	0,125	0,2856	0,2856	0,316	0,35

Так как $h = 0{,}5$, то на интервале [0, 1] решение будет включать три значения интегральной кривой $y(x)$: $y_0 = y(0)$, $y_1 = y(0{,}5)$, $y_1 = y(1)$. Решения для вариантов а) и б), а также промежуточные вычисления приведены в табличном виде.

5.2.3. Метод Рунге – Кутта 4-го порядка

Метод Рунге – Кутта 4-го порядка является наиболее употребимым методом решения задачи Коши. Формула (5.3) в этом случае должна содержать четыре коэффициента k_i, $i = 1,\ldots,r$, $r = 4$:

$$\begin{cases} y_{k+1} = y_k + \Delta y_k; \\ \Delta y_k = \dfrac{1}{6} \cdot (k1 + 2 \cdot k2 + 2 \cdot k3 + k4); \\ k1 = h \cdot f(x_k, y_k); \\ k2 = h \cdot f(x_k + \dfrac{h}{2}, y_k + \dfrac{k1}{2}); \\ k3 = h \cdot f(x_k + \dfrac{h}{2}, y_k + \dfrac{k2}{2}); \\ k4 = h \cdot f(x_k + h, y_k + k3). \end{cases} \quad (5.10)$$

Аналог – формулы Симпсона.

Погрешность метода – порядка (h^5).

Замечания:

1. Все методы Рунге – Кутта (кроме метода 1 порядка Эйлера) имеют хорошую сходимость.
2. Методы Рунге – Кутта являются явными, т.е. значение в последующей точке y_{k+1} находится по явным формулам, зависящим только от значения x_k, y_k.
3. Методы Рунге – Кутта допускают неравномерную дискретизацию.
4. Методы Рунге – Кутта не требуют вычисления производных.
5. Методы Рунге – Кутта удобны для реализации на ЭВМ.
6. К недостаткам методов можно отнести отсутствие простых методов оценки погрешности.

5.2.4. Погрешность решения задачи Коши

Погрешность методов Рунге – Кутта включает две составляющие: методологическую и вычислительную.

Погрешность метода возникает при отбрасывании членов разложения в ряд Тейлора. На каждом шаге в предположении, что предыдущий шаг был точным, эта погрешность определяется порядком r (5.4) и называется локальной:

$$l_k = \mathrm{O}(h^{r+1}) = C \cdot h^{r+1},$$

где C – константа.

Глобальная погрешность метода складывается из локальных погрешностей отдельных шагов:

$$g_k = n \cdot C \cdot h^{r+1} = \frac{b-a}{h} \cdot h^{r+1} = C_1 \cdot h^r .$$

В этой формуле введена новая константа $C_1 = (b-a) \cdot C$.

Вычислительная погрешность определяется ошибкой округления ЭВМ и на каждом шаге равна ε_k. Глобальная вычислительная погреш-

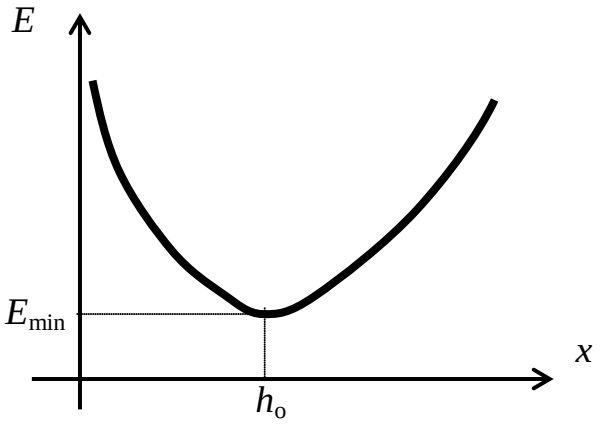


Рис. 5.4

ность на всем интервале соответственно равна

$$\varepsilon = n \cdot \varepsilon_k = \frac{b-a}{h} \cdot \varepsilon_k = C_2 \cdot \frac{\varepsilon_k}{h} .$$

Общая глобальная погрешность включает методологическую и вычислительную погрешность

$$E = C_1 \cdot h^k + C_2 \cdot \frac{\varepsilon_k}{h} . \qquad (5.11)$$

Формула (5.11) дает качественную картину поведения общей погрешности E, представленную на рис. 5.4. Погрешность полностью определяется шагом вычисления функции h. Вид математической зависимости $E(h)$ предполагает наличие оптимального значения шага h_o, приводящего к решению задачи Коши с минимальной погрешностью E_{min}.

Для практической оценки погрешности решения задачи Коши используют принцип Рунге, позволяющий оценить погрешность на каждом шаге по формуле

$$r(x_k) = \frac{y_k^h - y_k^{2h}}{2^{r-1} - 1}, \qquad (5.12)$$

где r – порядок метода Рунге – Кутта;

y_h, y_{2h} – значения интегральной кривой, вычисленные с шагом h и $2h$.

Оценка по формуле (5.12) должна быть проведена для каждой точки $k = 1, 2, \ldots, n$ и быть меньше заданной точности вычисления ε.

На основе формулы (5.12) имеется возможность построения формул Рунге – Кутта с автоматическим выбором шага, обеспечивающим заданную точность вычисления ε. Для этого (после вычисления значения y_{k+1} с шагом h) все вычисления повторяются с шагом $h/2$. Затем по формуле (5.12) вычисляется погрешность и сравнивается с заданной погрешностью ε. Если $r(x_k) < \varepsilon$, то расчеты продолжаются для последующих точек x_{k+1}. В противном случае шаг уменьшается до $h/2$ и все расчеты повторяются. Приведенная процедура лежит в основе алгоритма метода Рунге – Кутта с адаптивным выбором шага.

5.3. Решение систем дифференциальных уравнений

Методы Рунге – Кутта могут быть применены и к решению систем дифференциальных уравнений первого порядка. Каждое уравнение должно быть разрешено относительно производной:

$$\begin{cases} y_0' = f_0(x, y_0, y_1, ..., y_n); \\ y_1' = f_1(x, y_0, y_1, ..., y_n); \\ .....; \\ y_n' = f_n(x, y_0, y_1, ..., y_n). \end{cases} \qquad (5.13)$$

где $y_0(x)$, $y_1(x)$, …, $y_n(x)$ – неизвестные функции одного аргумента x, подлежащие определению при заданных начальных условиях:

$$\begin{cases} y_0(x_0) = y_0^0; \\ y_1(x_0) = y_1^0; \\ .....; \\ y_n(x_0) = y_n^0. \end{cases} \qquad (5.14)$$

Значения y_0^0, y_1^0, …, y_n^0 – некоторые константы (числа). Верхний индекс 0 означает номер точки вычисления интегральной кривой. Нижний индекс – номер функции.

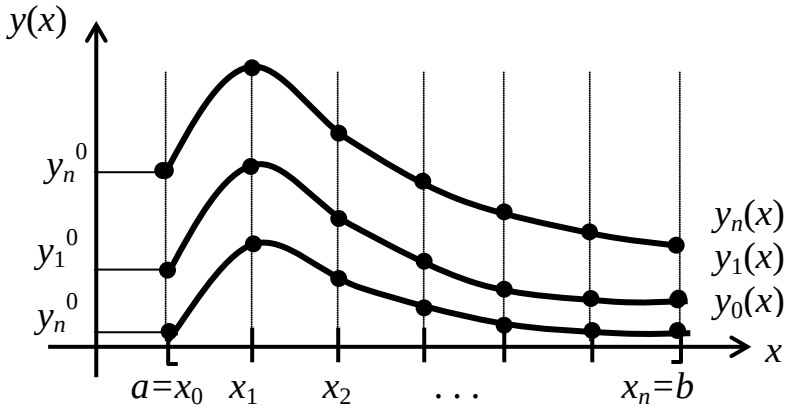


Рис. 5.5. Семейство интегральных кривых

Требуется найти функции $y_0(x)$, $y_1(x)$, …, $y_n(x)$ при $x \in [a, b]$, удовлетворяющие системе уравнений (5.13) и начальным условиям (5.14).

Методы решения системы обыкновенных дифференциальных уравнений с начальными условиями те же, что и для решения одного уравнения. Например, формулы Эйлера для системы дифференциальных уравнений имеют вид

$$\begin{cases} y_0^{k+1} = y_0^k + h \cdot f_0(x_k, y_0^k, y_1^k, ..., y_n^k); \\ y_1^{k+1} = y_1^k + h \cdot f_1(x_k, y_0^k, y_1^k, ..., y_n^k); \\ .....; \\ y_n^{k+1} = y_n^k + h \cdot f_n(x_k, y_0^k, y_1^k, ..., y_n^k). \end{cases} \qquad (5.15)$$

Формулы (5.15) позволяют по явным формулам получить неизвестные – семейство n интегральных кривых на сетке узлов x_k, $x_k = a + k \cdot h$.

Аналогично одномерному случаю могут быть записаны методы Рунге – Кутта более высоких порядков – второго или четвертого. Четвертый порядок наиболее употребим в численном анализе. Однако формулы его для многомерного случая слишком громоздки и здесь не приводятся.

5.4. Решение дифференциального уравнения *n*-го порядка

Дифференциальное уравнение n-го порядка должно быть разрешено относительно старшей производной, т.е. иметь вид

$$y_0^{(n)} = f(x, y, y', ..., y^{(n-1)}) \qquad (5.16)$$

с начальными условиями

$$y(a) = y_0\ , y'(a) = y_0{}',\ldots, y^{(n-1)}(a) = y_0{}^{(n-1)}. \qquad (5.17)$$

Требуется найти функцию $y(x)$ при $x \in [a, b]$.

Для нахождения решения введем обозначения: $y(x)$ обозначим новой функцией $y_0(x)$, $y'(x)$ обозначим $y_1(x)$ и т.д. В результате получим систему обозначений:

$$\begin{cases} y(x) = y_0(x); \\ y'(x) = y_1(x); \\ \ldots\ldots\ldots\ldots; \\ y^{(n-1)}(x) = y_{n-1}(x). \end{cases} \qquad (5.18)$$

Продифференцируем обе части последней системы уравнений:

$$\begin{cases} y'(x) = y_0'(x); \\ y''(x) = y_1'(x); \\ \ldots\ldots\ldots\ldots; \\ y^{(n)}(x) = y_{n-1}'(x). \end{cases}$$

Производные из правых частей предыдущей системы приравняем функциям (5.18), введенным при замене переменных:

$$\begin{cases} y_0'(x) = y_1(x); \\ y_1'(x) = y_2(x); \\ \ldots\ldots\ldots\ldots; \\ y_{n-1}'(x) = f(x, y_0, y_1, \ldots, y_{n-1}). \end{cases} \qquad (5.19)$$

В результате получили систему дифференциальных уравнений первого порядка относительно неизвестных функций $y_0(x)$, $y_1(x)$, …, $y_n(x)$. Решив её, получим интегральные кривые, одна из которых $y_0(x)$ и является решением исходного уравнения n-го порядка (5.16). Начальные условия (5.17), с учетом замены переменных, будут иметь вид

$$y_0(a) = y_0{}^0, y_1(a) = y_0{}', \ldots, y_{n-1}(a) = y_0{}^{(n-1)}. \qquad (5.20)$$

Таким образом, дифференциальное уравнение n-го порядка преобразовано к системе n дифференциальных уравнений 1-го порядка. Решение системы дифференциальных уравнений методами Рунге – Кутта

рассмотрено в разд. 5.3.

Пример 5.2. Решить методом Эйлера дифференциальное уравнение 2-го порядка $y'' + 2 \cdot y' + 2 \cdot y = 2 \cdot e^x \cdot \cos(\pi \cdot x)$ на интервале $x \in [0,\ 2]$ с шагом $h = 1$ при начальных условиях $y(0) = 1$, $y'(0) = 0$.

Разрешим дифференциальное уравнение относительно старшей (второй) производной

$$y'' = -2 \cdot y' - 2 \cdot y + 2 \cdot e^x \cdot \cos(\pi \cdot x).$$

Так как задано уравнение второго порядка, то введем две переменных

$$\begin{cases} y(x) = y_0(x); \\ y'(x) = y_1(x). \end{cases}$$

Продифференцируем:

$$\begin{cases} y_0'(x) = y_1(x); \\ y_1'(x) = -2y_1 - 2y_0 + 2e^x \cos(\pi x). \end{cases}$$

Начальные условия –

$$y_0(0) = 1, \quad y_1(0) = 0.$$

Формула Эйлера для одномерного случая имеет вид

$$y^{k+1} = y^k + h \cdot f(x_k, y^k).$$

Для системы двух заданных уравнений

$$\begin{cases} y_0{}^{k+1} = y_0^k + h \cdot y_1^k; \\ y_1{}^{k+1} = y_1^k + h \cdot (-2y_1^k - 2y_0^k + 2e^{x_k} \cos(\pi \cdot x_k). \end{cases}$$

Решение приведем в виде таблицы

x_k	0	1	2
y_0	1	1	1
y_1	0	0	$-2 - 2 \cdot e$

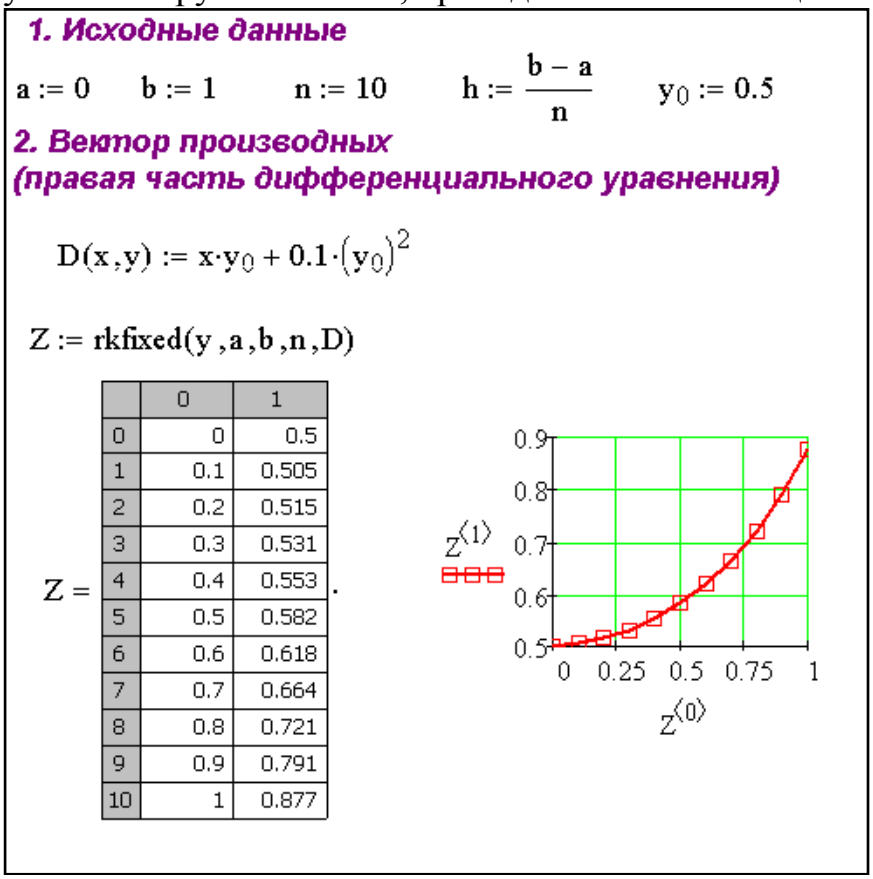
Решение дифференциального уравнения – функция $y_0(x_k)$.

5.5. Решение задачи Коши в ППП MathCad

Для решения дифференциальных уравнений ППП MathCad имеет ряд встроенных функций, в частности функцию **rkfixed**, реализующую метод Рунге – Кутта 4-го порядка с фиксированным шагом.

Описание функции	Аргументы функции
Rkfixed(*y, a, b, n, D*) Возвращает матрицу. Первый столбец – узлы сетки. Второй столбец – решение уравнения в узлах сетки. Остальные столбцы – производные, если порядок уравнения выше первого.	***y*** **–** вектор начальных значений; ***a, b*** **–** границы интервала; ***n*** **–** число точек внутри интервала [*a*,*b*], в которых получено решение; ***D*** **–** вектор первых производных функции.

На рис. 5.6 приведен фрагмент программы решения дифференциального уравнения из примера 5.1 с использованием функции **rkfixed**. Решение этой же задачи методами Рунге – Кутта 1-го и 2-го порядка приведено на рис. 5.7. Отметим, что решения полностью согласуются с результатами ручного счета, приведенными в таблице.



	0	1
0	0	0.5
1	0.1	0.505
2	0.2	0.515
3	0.3	0.531
4	0.4	0.553
5	0.5	0.582
6	0.6	0.618
7	0.7	0.664
8	0.8	0.721
9	0.9	0.791
10	1	0.877

Рис. 5.6. Решение задачи Коши в ППП MathCAD

1. Исходные данные

1.1. Дифференциальное уравнение

$f(x,y) := x^2 + y^2$

1.2. Начальные условия

$a := 0 \qquad b := 1 \qquad h := 0.5 \qquad y_0 := 0$

$n := \frac{b-a}{h} \qquad n = 2$

2. Сетка узлов

$k := 0..n-1 \qquad x_k := a + k \cdot h$

3. Метод Эйлера

$y_{k+1} := y_k + h \cdot f(x_k, y_k) \qquad y = \begin{bmatrix} 0 \\ 0 \\ 0.125 \end{bmatrix}$

4. Метод Рунге-Кутта 2 порядка -a)

$k1(x,y) := h \cdot f(x,y)$

$k2\_a(x,y) := h \cdot f\left(x + \frac{h}{2}, y + \frac{k1(x,y)}{2}\right)$

$y\_a_0 := y_0$

$y\_a_{k+1} := y\_a_k + k2\_a(x_k, y\_a_k) \qquad y\_a = \begin{bmatrix} 0 \\ 0.03125 \\ 0.31692 \end{bmatrix}$

5. Метод Рунге-Кутта 2 порядка -b)

$k2\_b(x,y) := h \cdot f(x + h, y + k1(x,y))$

$y\_b_0 := y_0$

$y\_b_{k+1} := y\_b_k + \frac{\left(k1(x_k, y\_b_k) + k2\_b(x_k, y\_b_k)\right)}{2} \qquad y\_b = \begin{bmatrix} 0 \\ 0.0625 \\ 0.38495 \end{bmatrix}$

Рис. 5.7. Решение дифференциального уравнения методами Рунге – Кутта

Более общим случаем является обыкновенное дифференциальное уравнение *n*-го порядка. Задача Коши в этом случае содержит само дифференциальное уравнение с производными до *n*-го порядка включительно и *n* начальных условий. Решение дифференциального уравнения *n*-го порядка сводится к решению системы *n* дифференциальных уравнений 1-го порядка. Рассмотрим методику этого метода на примере дифференциального уравнения 2-го порядка

$$y''(x) = 3 \cdot y'(x) + e^{5x}.$$

Начальные условия

$$y(0) = 5,5;$$

$$y'(0) = 0,8.$$

Требуется найти функцию $y(x)$ при $x \in [0, 1]$.

Введем следующие обозначения:

$$\begin{cases} y(x) = y_0(x); \\ y'(x) = y_1(x). \end{cases}$$

Продифференцируем

$$\begin{cases} y_0'(x) = y_1(x); \\ y_1'(x) = 3 \cdot y_1 + e^{5 \cdot x}. \end{cases}$$

Начальные условия – $y_0(0) = 5,5; \quad y_1(0) = 0,8.$

На рис. 5.8 приведен фрагмент программы MathCAD для решения приведенного уравнения.

Результатом является матрица Z, содержащая три столбца. Первый столбец – аргумент функции в виде узлов сетки; второй столбец – решение дифференциального уравнения – интегральная кривая в узлах сетки; последний столбец – значения первой производной. Решение дифференциального уравнения также приведено графически.

Процедура **Rkfixed** использует метод Рунге – Кутта с фиксированным (постоянным) шагом. Это не самый быстрый метод, но достаточно универсальный к виду анализируемой функции. Если известно, что искомая функции быстро меняется, то целесообразно использовать процедуру **Rkadapt**, реализующую метод Рунге – Кутта с переменным (адаптивным) шагом. Это позволяет повысить точность решения. При этом переменный шаг используется только при нахождении функции $y(x)$. При выводе результатов – шаг постоянный.

1. Начальные условия

$$y := \begin{bmatrix} 5.5 \\ 0.8 \end{bmatrix} \quad a := 0 \qquad b := 1$$

2. Матрица первых производных

$$D(x, y) := \begin{bmatrix} y_1 \\ 3 \cdot y_1 + e^{5 \cdot x} \end{bmatrix}$$

$$Z := rkfixed(y, a, b, 5, D)$$

$$Z = \begin{bmatrix} 0 & 5.5 & 0.8 \\ 0.2 & 5.754 & 1.905 \\ 0.4 & 6.369 & 4.685 \\ 0.6 & 7.907 & 11.839 \\ 0.8 & 11.843 & 30.551 \\ 1 & 22.094 & 80.073 \end{bmatrix}$$

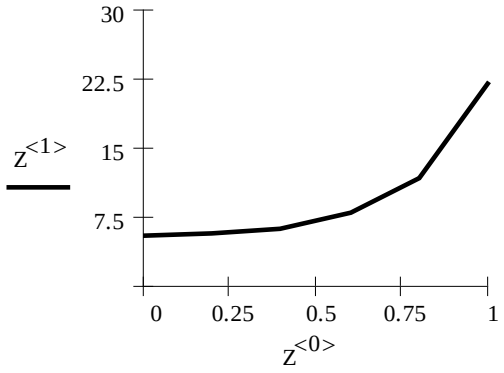


Рис. 5.8. Решение дифференциального уравнения 2-го порядка

Процедура **Bulstoer** использует иной численный метод – метод Булирша – Штера. Она рекомендуется для применения, если известно, что решение является гладкой функцией.

И в заключение раздела на рис. 5.9 приведен листинг пользовательской процедуры для решения дифференциального уравнения методом Рунге – Кутта второго порядка в ППП MathCAD.

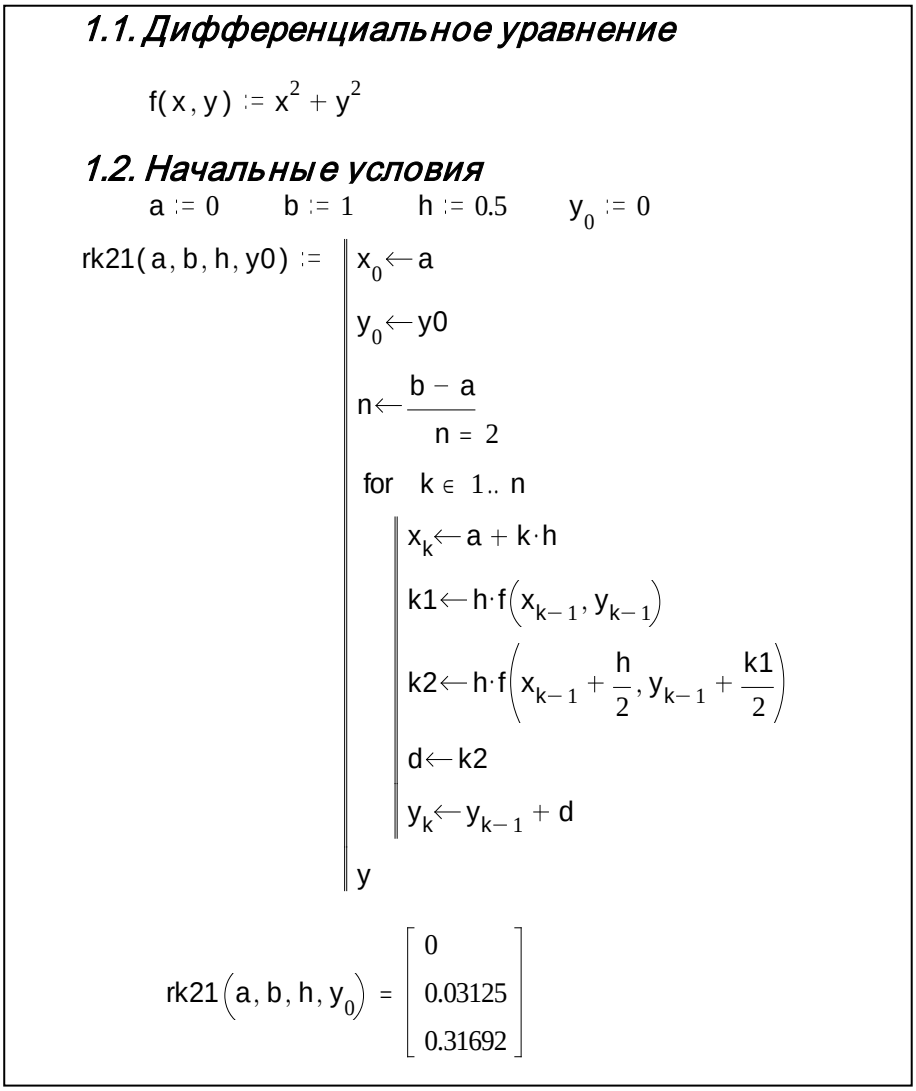


Рис. 5.9. Программа решения дифференциального уравнения методом Рунге – Кутта 2-го порядка

Вопросы для самопроверки

1. Сформулируйте постановку задачи Коши.
2. В чем разница между аналитическим и численным решением дифференциального уравнения?
3. В чем состоит основная идея методов Рунге- Кутта?
4. Как оценить порядок метода Рунге- Кутта?
5. Приведите формулы методов Рунге- Кутта 1, 2 и 4-го порядка.
6. В чем состоит идея метода Рунге- Кутта с адаптивным шагом?
7. Приведите классификацию дифференциальных уравнений.
8. Приведите геометрическую интерпретацию решения задачи Коши методами Рунге- Кутта.

9. Какие существуют источники погрешности решения задачи Коши.

10. Определите локальную и глобальную погрешность задачи Коши.

11. В чем состоит идея решения дифференциального уравнения высокого порядка.

12. Как можно использовать методы Рунге- Кутта для решения системы дифференциальных уравнений?

13. В какой форме можно получить решение дифференциального уравнения методами Рунге-Кутта?

14. Объясните, в чесм состоит принцип Рунге для оценки шага при решении задачи Коши.

6. АППРОКСИМАЦИЯ ФУНКЦИЙ И ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

Под экспериментальными данными понимается числовая информация, полученная при измерении некоторых параметров наблюдаемого процесса (явления).

При обработке экспериментальных данных часто возникает потребность построения некоторой эмпирической (опытной) формулы по полученным в ходе эксперимента данным. Полученная формула называется ***аппроксимирующей***, а сам способ приближения функции – ***аппроксимацией***.

Различают три задачи приближения экспериментальной информации: интерполяция, сглаживание и экстраполяция (прогноз) данных.

Интерполяция данных: требуется построить функцию, которая как можно ближе проходит через все значения экспериментальных данных. Иногда под интерполяцией понимают получение данных в промежутках между узлами $[x_i, x_{i+1}]$.

Сглаживание данных: получение функции, устраняющей случайные погрешности, попавшие в экспериментальные данные.

Экстраполяция данных (прогноз): предсказание значений функции вне интервала наблюдения функции. Как правило, требуется предсказать поведение функции в будущие моменты времени.

6.1. Интерполяция данных

Пусть на отрезке $[a, b]$ заданы $(n + 1)$ точек $a = x_0 < x_1 < \ldots < x_n = b$ и $(n + 1)$ значений функции $f(x)$: $\{f_0, f_1, \ldots, f_n\}$. $f_i = f(x_i)$, $i = 0,\ldots,n$.

Значения x_i называются сеткой узлов, f_i – решетчатой функцией, или таблично заданной:

x	x_0	x_1	…	x_n
f	f_0	f_1	…	f_n

В задаче интерполяции требуется по табличным значениям $\{x_i, f_i\}$ построить функцию $\varphi(x)$, такую, что значения $\varphi(x)$ легко вычисляются при $x \in [a, b]$ и при этом

$$\varphi(x_i) = f(x_i), \quad i \in 0,\ldots,n. \tag{6.1}$$

Последнее равенство носит название критерия интерполяции:
$f(x)$ – интерполируемая (исходная) функция;
$\varphi(x)$ – интерполирующая функция.

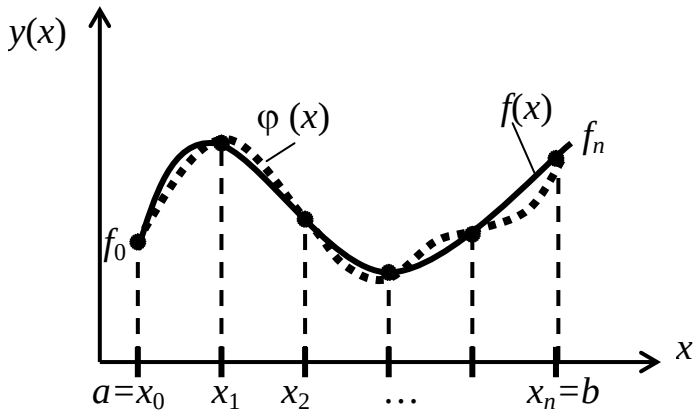


Рис. 6.1. Интерполяция функции

Интерполирующая функция $\varphi(x)$ может быть получена в двух видах:

- в виде аналитического выражения (формула);
- численно, т.е. в виде отдельных числовых значений для $x \in [a, b]$.

При получении аналитической зависимости $\varphi(x)$ строится в виде линейной комбинации элементарных (базисных) функций:

$$\varphi(x) = \sum_{k=0}^{m} a_k \cdot \varphi_k(x), \qquad (6.2)$$

где $\varphi_k(x)$ – базисные функции;
a_k – коэффициенты разложения по базисным функциям.

Функции $\varphi_k(x)$ могут быть любые (тригонометрические, логарифмические), но чаще других используются полиномиальные функции:

$$\varphi(x) = \sum_{k=0}^{m} a_k \cdot x^k .$$

Основная задача интерполяции по известным значениям (x_i, f_i), где $i = 0,\ldots,n$ найти коэффициенты разложения $a_0, a_1, \ldots, a_m$.

В теории интерполяции существуют классические полиномы Ньютона и Лагранжа, позволяющие найти коэффициенты a_k по известным формулам. Однако эти выражения очень громоздки и в последние годы редко используются в прикладных задачах. Наиболее современным аппаратом интерполяции являются сплайн-функции. Эти функции составлены (склеены) из отдельных кусков многочленов определенной степе-

ни. Между каждыми двумя узлами $[x_i, x_{i+1}]$ строится отдельный многочлен со своими коэффициентами a_k^i. На всем интервале $[a, b]$ сплайн-функция непрерывна вместе с несколькими производными. Это существенно повышает точность интерполяции, сохраняя однотипность математической формулы.

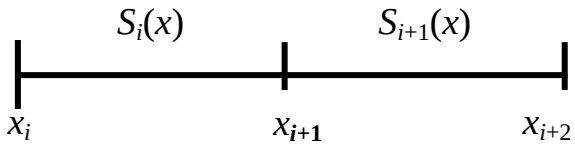
6.1.1. Сплайн-интерполяция

Сплайны представляют собой некоторую математическую модель гибкого тонкого стержня из упругого материала. Исторически сплайном (spline) называли гибкую линейку или рейку чертежника. Ее закрепляли между некоторыми установленными опорами (колышками) и, таким образом, получали гладкую кривую с заданными свойствами. А точки закрепления становились узлами интерполяции.

Наиболее популярны в вычислительной математике и в практических приложениях полиномиальные сплайны 1-, 2- и 3-го порядков.

Линейный сплайн представляет собой кусочно-линейную функцию. Для каждого частичного интервала $[x_i, x_{i+1}]$ исходная функция $f(x)$ заменяется линейной функцией

$$S_i(x) = a_i + b_i(x - x_i);$$
$$x_i \le x \le x_{i+1}, \quad i = 1,...,n-1. \tag{6.3}$$



Неизвестными в формуле (6.3) являются коэффициенты a_i, b_i. Для их нахождения используются следующие условия:

1) $S_i(x_i) = f_i$, $i = 0, n - 1$ (критерий интерполирования);

2) $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$, $i = 0, \ldots, n - 2$ (непрерывность сплайна в узлах).

Из условия 1 находим

$$a_i = f_i,\ i = 0,\ldots,n - 1. \tag{6.4}$$

Из условия 2 –

$$b_i = \frac{f_{i+1} - f_i}{h}, \quad i = 1,...,n-1. \tag{6.5}$$

Определив коэффициенты a_i, b_i, полностью описываем линейную

сплайн-функцию.

Параболический сплайн – это функция, склеенная в узлах интерполяции из полиномов 2-го порядка. В узлах интерполяции сплайн имеет непрерывную производную. На каждом частичном интервале $[x_i, x_{i+1}]$ исходная функция $f(x)$ заменяется квадратичной функцией

$$\begin{aligned} &S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2; \\ &x_i \le x \le x_{i+1}, \quad i = 1,...,n-1. \end{aligned} \tag{6.6}$$

Для параболического сплайна неизвестными являются уже три группы параметров a_i, b_i, c_i, $i = 0, \ldots, n - 1$. Они находятся из условий:

1) $S_i(x_i) = f_i$, $i = 0, \ldots, n - 1$ (критерий интерполирования);

2) $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$, $i = 0, \ldots, n - 2$ (непрерывность сплайна в узлах);

3) $S'_i(x_{i+1}) = S'_{i+1}(x_{i+1})$, $i = 0, \ldots, n - 2$ (непрерывность производной сплайна в узлах).

Из первого условия

$$a_i = f_i,\ i = 0,\ldots,n-1. \tag{6.7}$$

Продифференцируем уравнение (6.6)

$$\begin{aligned} &S_i'(x) = b_i + 2 \cdot c_i(x - x_i); \\ &S_i'(x_i) = b_i; \\ &S_i'(x_{i+1}) = b_i + 2 \cdot c_i h_i = b_{i+1}; \\ &h_i = x_{i+1} - x_i. \end{aligned}$$

Отсюда можно найти коэффициент

$$c_i = \frac{b_{i+1} - b_i}{2h_i}, \quad i = 0,...,n-1. \tag{6.8}$$

Из второго условия следует

$$\begin{aligned} &S_i'(x_{i+1}) = b_i + 2 \cdot c_i h_i + c_i h_i^2 = S_{i+1}(x_{i+1}) = f_{i+1}; \\ &\frac{f_{i+1} - f_i}{h_i} = b_i + c_i h_i. \end{aligned}$$

Подставим в последнее уравнение значение коэффициента c_i:

$$\frac{2}{h_i}(f_{i+1} - f_i) = b_i + b_{i+1}, \quad i = 0,...,n-1.$$

Имеем n уравнений для определения $(n + 1)$-го неизвестного. Для корректного определения всех коэффициентов требуется дополнительное (граничное) условие. Чаще всего оно задается в виде

$$b_0 = m_0,$$

где значение m_0 – это значение первой производной в левом конце интервала.

Тогда, обозначив

$$m_i = \frac{2}{h_i}(f_{i+1} - f_i), \quad i = 1,...,n-1,$$

получим формулы для явного нахождения коэффициентов

$$b_{i+1} = m_i - b_i, \quad i = 1,...,n-1. \tag{6.9}$$

Таким образом, параболический сплайн определяется уравнением (6.6), коэффициенты которого находятся по формулам (6.7), (6.8), (6.9).

Кубический сплайн – это функция, склеенная в узлах из полиномов 3-го порядка. Именно эта функция является математической моделью механического сплайна чертежника, для которого из теории сопротивления материалов уравнение свободного равновесия будет иметь вид $S^{IV}(x) = 0$. Тогда между каждой парой узлов сплайна функция $S(x)$ будет являться кубическим полиномом. Интерполяционный полином 3-го порядка проходит через все заданные узлы и имеет непрерывные первую и вторую производные. На каждом частичном интервале $[x_i, x_{i+1}]$ интерполирующая функция $S(x)$ имеет вид

$$\begin{aligned} &S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3; \\ &x_i \le x \le x_{i+1}, \quad i = 1,...,n-1. \end{aligned} \tag{6.10}$$

Для кубического сплайна неизвестными являются группы параметров a_i, b_i, c_i, d_i, $i = 0, \ldots, n - 1$, т.е. необходимо найти $4(n - 1)$ неизвестных коэффициентов полиномов. Они находятся из условий:

1) $S_i(x_i) = f_i$, $i = 0, \ldots, n - 1$ (критерий интерполирования);

2) $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$, $i = 0, \ldots, n - 2$ (непрерывность сплайна в узлах);

3) $S'_i(x_{i+1}) = S'_{i+1}(x_{i+1})$, $i = 0, \ldots, n - 2$ (непрерывность производной сплайна в узлах);

4) $S''_i(x_{i+1}) = S''_{i+1}(x_{i+1})$, $i = 0, \ldots, n - 2$ (непрерывность второй производной сплайна в узлах).

Всего имеется $4n - 2$ уравнения. Два недостающих уравнения можно получить, задавая граничные условия. Чаще всего задается нулевая кривизна функции на краях интервала $[a, b]$.

Для нахождения коэффициентов сплайна формируется система линейных алгебраических уравнений с трехдиагональной матрицей:

$$\frac{h_i}{6} M_i + \frac{h_i + h_{i+1}}{3} M_{i+1} + \frac{h_{i+1}}{6} M_{i+2} = y_{i+1} - y_i, \qquad (6.11)$$

где $y_i = f_{i+1} - f_i$.

Эффективным способом решения таких систем является метод прогонки. После определения M_i коэффициенты d_i, c_i, b_i находятся из обычных алгебраических выражений:

$$\begin{aligned} d_i &= \frac{M_{i+1} - M_i}{6h_i}; \\ c_i &= \frac{M_i}{2}; \\ b_i &= y_i - \frac{M_{i+1} + 2M_i}{6h_i}; \\ a_i &= f_i. \end{aligned} \qquad (6.12)$$

6.1.2. Интерполяция в ППП MathCad

Для решения задачи интерполяции ППП MathCad имеет ряд встроенных функций. В частности, функция **linterp** предназначена для линейной интерполяции. **Lspline**, **pspline**, **cspline –** интерполяция линейными, параболическими и кубическими сплайнами.

Описание функций	Аргументы функций
***linterp* (*x*,*y*, *z*)** Возвращает значение (значения) интерполяции для независимого аргумента **z**. ***Lspline*(*x*,*y*), *Pspline*(*x*,*y*), *Cspline*(*x*,*y*)** Возвращает вектор (производных) сплайна.	***x*, *y* –** векторные значения аргумента и функции. ***z*** – значения аргумента, в которых необходимо выполнить интерполяцию.

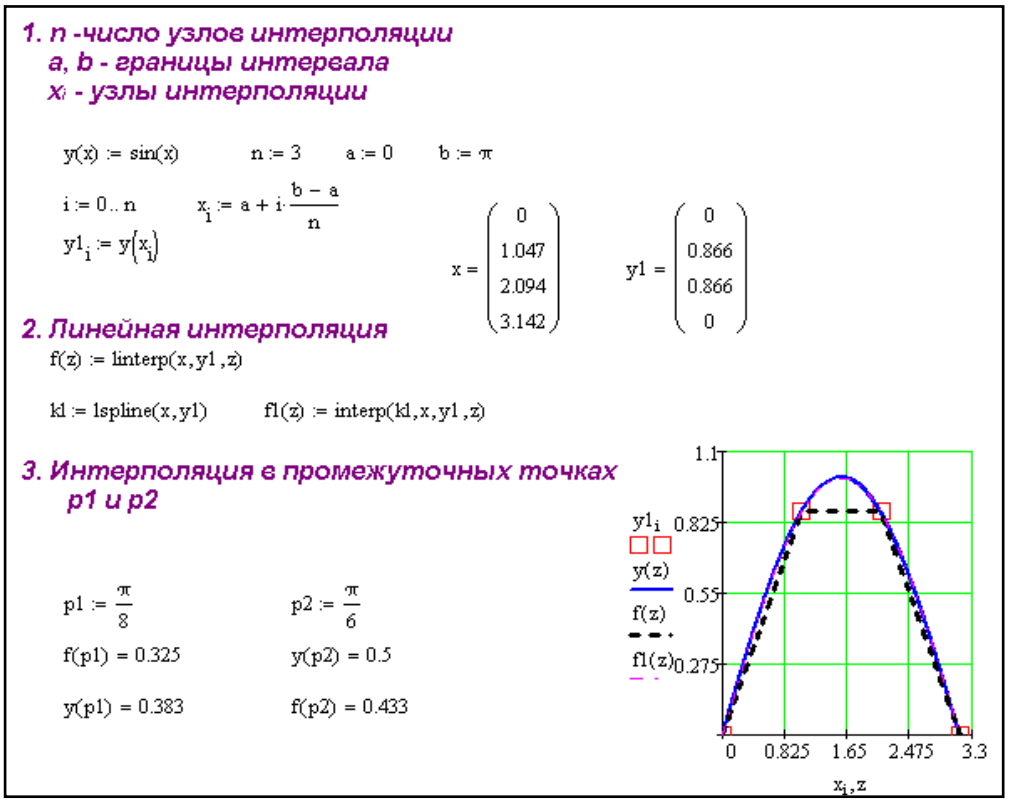


Рис. 6.2. Линейная интерполяция в ППП MathCAD

Как видно из графика, в узлах интерполирующая функция полностью совпадает с исходной. Вне узлов функции могут значительно различаться. Это подтверждает расхождение функций в промежуточных точках *p*1 и *p*2.

Векторы *x*, *y* должны быть одной размерности. Новый аргумент *z* может быть непрерывной переменной либо также представлять собой векторные значения.

Процедуры **lspline**, **pspline**, **cspline** позволяют получить коэффициенты сплайн-функции 1-го, 2-го и 3-го порядков соответственно. Эти коэффициенты являются затем входной информацией процедуры **interp**, производящей интерполяцию между узловыми значениями.

На рис. 6.3 представлен фрагмент программы MathCAD для интерполяции заданной функции *y*(*x*) = sin(*x*) сплайном 1-го, 2-го, 3-го порядка.

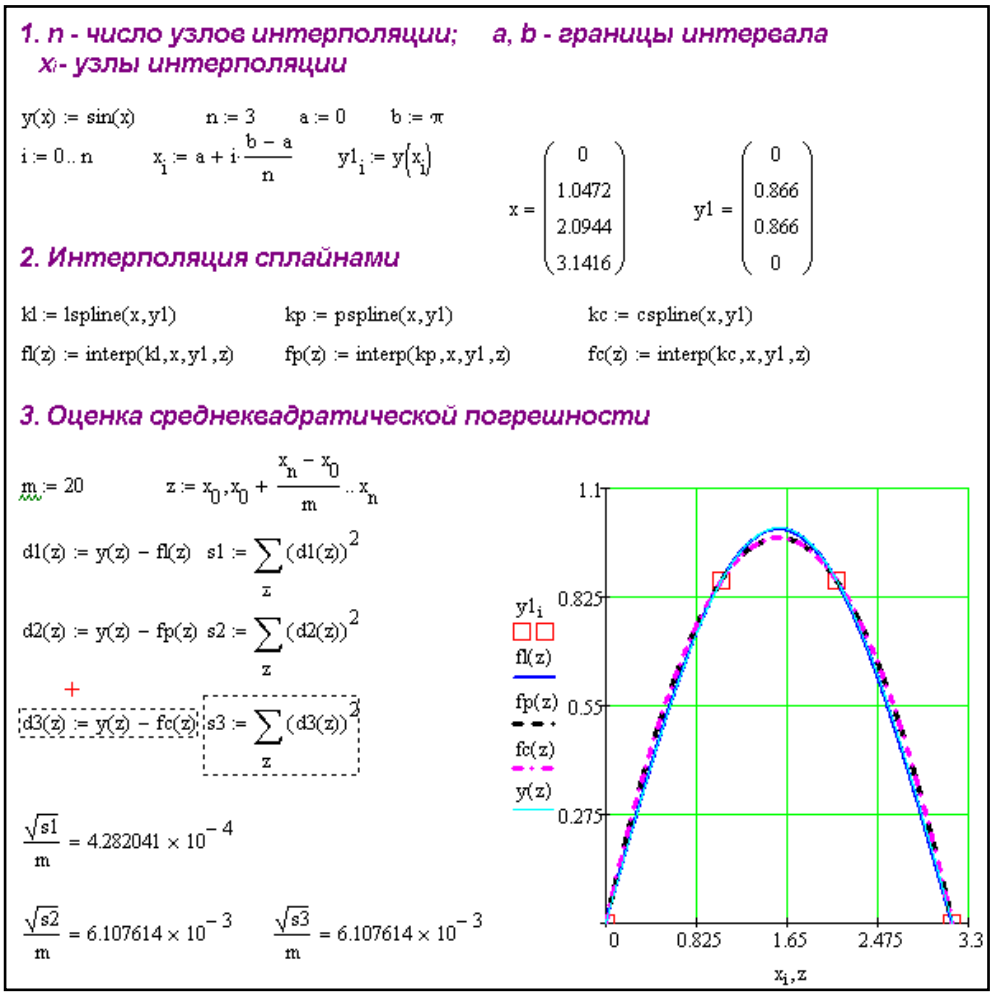


Рис. 6.3. Сплайн-интерполяция в ППП MathCAD

Здесь *x*, *y* – векторные значения аргумента и функции экспериментальных данных; *z* – новые значения аргумента, по которым проводится интерполяция. Размерность вектора *z*, как правило, намного больше, чем размерность исходного вектора *x*.

Из рисунка видно, что результаты интерполяции различными сплайнами практически не отличаются во внутренних точках интервала и совпадают с точными значениями функции. Отличие более заметно при отсутствии узлов интерполяции в местах изменения функции. Для количественной оценки интерполяции выбрана среднеквадратическая приведенная погрешность.

6.2. Сглаживание данных

Сглаживание данных – это получение функции, устраняющей случайные погрешности, попавшие в экспериментальные данные.

Сглаживание данных возможно в двух вариантах:

а) получение аналитической функции (регрессионного уравнения), позволяющей строить функцию при любых значениях аргумента;

б) получение числовых значений функции, более гладких, чем исходные данные, но только при исходных значениях аргумента.

Регрессионная функция является оптимальной в среднеквадратическом смысле. Графически это свойство соответствует построению линии, визуально усредняющей экспериментальные данные.

На рис. 6.4 точками (•) показаны экспериментальные данные; сплошной линией – функция, их соединяющая; пунктирной – линия регрессии.

Регрессионное уравнение может включать любые элементарные функции, однако чаще других используются полиномы.

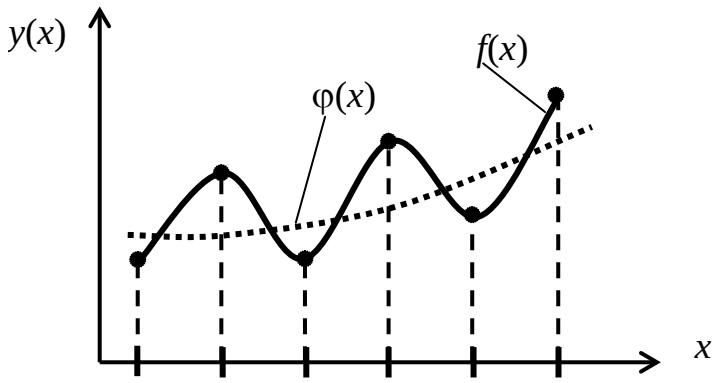


Рис. 6.4. Сглаживание данных

В пакете MathCAD для получения регрессионного полиномиального уравнения предназначена процедура **regress()**. На рис.6.5 приведен фрагмент использования процедуры медианной фильтрации **medsmooth** и **regress**.

1. n -число узлов; a, b - границы интервала; h- шаг сетки
d - диапазон погрешности

$f(x) := \ln(x+1)$ $a := 0$ $b := 2$ $h := 0.2$ $n := \frac{(b-a)}{h}$ $n = 10$

$i := 0..n$ $x_i := a + i\cdot\frac{b-a}{n}$ $fl_i := f(x_i)$ $d := 0.3\cdot\max(fl)$ $y_i := fl_i + rnd(d) - \frac{d}{2}$

2. Сглаживание

$p := medsmooth(y,3)$

$z2 := regress(x,y,2)$

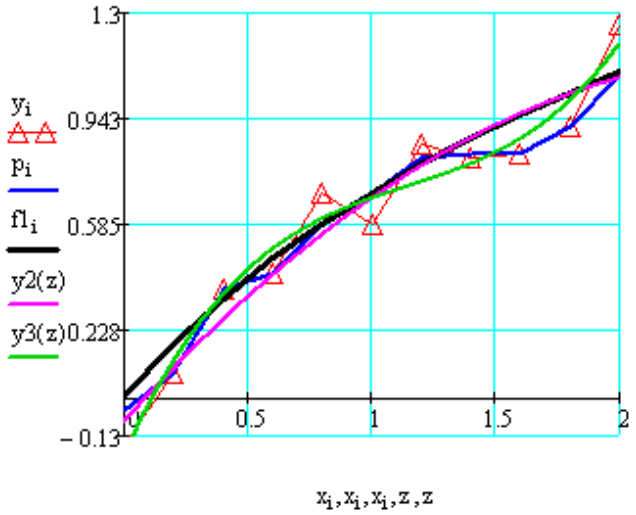
$z3 := regress(x,y,3)$

$y2(t) := z2_3 + z2_4\cdot t + z2_5\cdot(t)^2$

$y3(t) := z3_3 + z3_4\cdot t + z3_5\cdot(t)^2 + z3_6\cdot(t)^3$

$z2 = \begin{pmatrix} 3 \\ 3 \\ 2 \\ -0.081 \\ 0.93 \\ -0.175 \end{pmatrix}$ $z3 = \begin{pmatrix} 3 \\ 3 \\ 3 \\ -0.196 \\ 1.842 \\ -1.371 \\ 0.399 \end{pmatrix}$

$fl_i - y2(x_i) =$	$fl_i - y3(x_i) =$
0.081	0.196
0.084	0.061
0.073	-0.011
0.056	-0.032
0.037	-0.017
0.019	0.019
$5.42\cdot10^{-3}$	0.059
$-2.591\cdot10^{-3}$	0.085
$-3.576\cdot10^{-3}$	0.081
$3.5\cdot10^{-3}$	0.026
0.019	-0.095



3. Оценка среднеквадратической погрешности

$m := 20$ $z := x_0, x_0 + \frac{x_n - x_0}{m} .. x_n$

$d1(z) := f(z) - y2(z)$ $s1 := \sum_z (d1(z))^2$ $\frac{\sqrt{s1}}{m} = 0.01066$

$d2(z) := f(z) - y3(z)$ $s2 := \sum_z (d2(z))^2$ $\frac{\sqrt{s2}}{m} = 0.01632$

Рис. 6.5. Сглаживание полиномами и медианным фильтром в ППП MathCAD

Описание функции	Аргументы функции
regress (*x*,*y*, *k*) Возвращает значения коэффициентов полинома	***x*, *y* –** векторные значения аргумента и функции. ***k*** – степень полинома.
Medsmooth(y,m) Возвращает сглаженные значения функции	***y* –** векторные значения функции. ***m*** – число значений функции, участвующих в сглаживании (глубина сдвигаемого окна).

Векторы *x*, *y* должны быть одной размерности. Их значения должны быть упорядочены.

Построим регрессионное уравнение 2-го порядка, предполагая, что векторные значения x, y заданы в пяти дискретных точках.

Первые две переменные вектора *z* (z_0, z_1) содержат служебную информацию, z_2 – степень полинома, z_3, z_4, z_5, z_6 – коэффициенты полинома. Вектор *f*1 содержит дискретные значения исходной функции в пяти точках. Вектор *y* – исходная функция в условиях помех, на графике представлена отдельными значениями. Аппроксимирующие полиномы y_2, y_3 сглаживают зашумленную исходную функцию. Из рисунка видно, что полином 2-го порядка больше совпадает с исходной функцией без шума. Значение среднеквадратической погрешности подтверждает этот результат.

Для получения сглаживания в виде набора значений в пакете Mathcad предназначены процедуры:

- medsmooth;
- ksmooth;
- supsmooth.

Процедура **medsmooth(*y*,*k*)** позволяет получить значения функции**,** сглаженные с помощью скользящей медианы. Параметры процедуры:

y – вектор исходных данных (значения функции);

k **–** ширина окна сглаживания.

Результатом работы оператора MathCAD

***z*:= medsmooth(*y*,*k*)**

является вектор z, содержащий сглаженные значения. Размерность вектора z совпадает с размерностью вектора y.

6.2.1. Метод наименьших квадратов

Функция **regress(*x*, *y*, *k*)** позволяет построить регрессионное уравнение только в виде полинома. Если требуется построить сглаживающую функцию по произвольному базису, целесообразно использовать метод наименьших квадратов (МНК).

Идея МНК состоит в построении регрессионной функции $\varphi(x)$, оптимальной в среднеквадратическом смысле. Пусть на сетке узлов $\{a = x_0, x_1, \ldots, x_n = b\}$ заданы отсчеты функции $y_i = y(x_i)$, $i = 1,\ldots,n$. Аппроксимирующая функция $\varphi(x)$ представляет собой линейную комбинацию произвольных базисных функций $\{\varphi_j(x)\}$, $j = 1,\ldots,m$, $m < n$:

$$\varphi(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \ldots + a_m\varphi_m(x). \qquad (6.13)$$

Неизвестными в уравнении (6.13) являются коэффициенты a_j, $j = 1,\ldots,m$. Вид и количество базисных функций $\{\varphi_j(x)\}$, $j = 1,\ldots,m$, заданы.

Оценку параметров $a_0, a_1, \ldots, a_m$ определяем из условия минимума суммы квадратов отклонений измеренных значений $y_i = y(x_i)$ от расчетных $\varphi(x_i)$:

$$S(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} (\varphi(x_i) - y_i)^2 \to \min. \qquad (6.14)$$

Выражение (6.14) носит название среднеквадратического критерия, т.е. правила, которое определяет свойства $\varphi(x)$. Функция, удовлетворяющая критерию (6.14), является наилучшей в среднеквадратическом смысле.

Для выполнения условия (6.14) необходимо, чтобы при полученных параметрах $a_0, a_1, \ldots, a_m$ функция невязок $S(a_0, a_1, \ldots, a_m)$ имела стационарную точку, т.е.

$$\frac{\partial S(a_0, a_1, \ldots, a_m)}{\partial a_j} = 0,\ j = 0, m.$$

Перепишем критерий (6.14) с учетом конкретного вида $\varphi(x)$:

$$S(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} [a_0\varphi_0(x_i) + a_1\varphi_1(x_i) + \ldots a_m\varphi_m(x_i) - y_i]^2 \to \min.$$

Найдем частные производные:

$$\frac{\partial S(a_0,a_1,...,a_m)}{\partial a_0} = 2\cdot\sum_{i=1}^{n}[a_0\varphi_0(x_i)+...+a_m\varphi_m(x_i)-y_i]\cdot\varphi_0(x_i)=0;$$

$$\frac{\partial S(a_0,a_1,...,a_m)}{\partial a_1} = 2\cdot\sum_{i=1}^{n}[a_0\varphi_0(x_i)+...+a_m\varphi_m(x_i)-y_i]\cdot\varphi_1(x_i)=0;$$

. . . . . ;

$$\frac{\partial S(a_0,a_1,...,a_m)}{\partial a_m} = 2\cdot\sum_{i=1}^{n}[a_0\varphi_0(x_i)+...+a_m\varphi_m(x_i)-y_i]\cdot\varphi_m(x_i)=0.$$

Внеся знак суммы (Σ) в каждое слагаемое, получим систему линейных уравнений относительно неизвестных a_0, a_1, …, a_m. Запишем систему линейных алгебраических уравнений в векторно-матричном виде:

$$\begin{bmatrix} \sum_{i=1}^{n}\varphi_0^2(x_i) & \sum_{i=1}^{n}\varphi_0(x_i)\varphi_1(x_i) & . & \sum_{i=1}^{n}\varphi_0(x_i)\varphi_m(x_i) \\ \sum_{i=1}^{n}\varphi_1(x_i)\varphi_0(x_i) & \sum_{i=1}^{n}\varphi_1^2(x_i) & . & \sum_{i=1}^{n}\varphi_1(x_i)\varphi_m(x_i) \\ . & . & . & . \\ \sum_{i=1}^{n}\varphi_m(x_i)\varphi_0(x_i) & \sum_{i=1}^{n}\varphi_m(x_i)\varphi_1(x_i) & . & \sum_{i=1}^{n}\varphi_1^2(x_i) \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ . \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}\varphi_0(x_i)y_i \\ \sum_{i=1}^{n}\varphi_1(x_i)y_i \\ . \\ \sum_{i=1}^{n}\varphi_m(x_i)y_i \end{bmatrix},$$

или

$$\Phi\cdot\bar{a}=\bar{y}. \qquad (6.15)$$

Решение системы (6.15) единственно и существует, если определитель матрицы $\Phi \neq 0$. Если в качестве базиса будут выбраны линейно независимые функции $\varphi_j(x)$, $j = 0,\ldots,m$, то определитель матрицы $\Phi \neq 0$.

После получения системы линейных алгебраических уравнений она может быть решена любым методом. Чаще всего используют один из прямых методов, метод Гаусса. В результате ее решения найдены коэффициенты разложения по базисным функциям: a_0, a_1, …, a_m.

***Замечания*:**

1. Матрица Φ называется матрицей Грамма. Матрица симметрична относительно главной диагонали, поэтому нет необходимости вычислять всю матрицу. Кроме того, матрица Φ является положительно определенной.

2. МНК эффективен при относительно большом числе измерений n. Обязательным условием является $n > m$, в противном случае имеет место недоопределенная система.

3. Если система базисных функций ортогональна, т.е. $\varphi_j(x) \cdot \varphi_i(x)$, $i \neq j$, то в матрице Грамма ненулевые только диагональные элементы. Поэтому решение ее тривиально.

В качестве примера рассмотрим использование метода МНК для построения аппроксимирующего (регрессионного) уравнения, включающего 2 базисные функции $\{1, \sin x\}$:

$$\varphi(x) = a_0 + a_1 \cdot \sin(x).$$

Критерий аппроксимации определяет близость регрессионного уравнения и исходной функции (или измеренных значений):

$$S(a_0, a_1) = \sum_{i=0}^{n} [\varphi(x_i) - y(x_i)]^2 \to \min,$$

где n – число измерений функции $y(x)$ в узлах x_i.

Выражение $[\varphi(x_i) - y(x_i)]$ определяет разницы между исходной и аппроксимирующей функциями и называется *невязками*.

Параметры регрессионного уравнения a_0, a_1 найдем из условия минимума функции невязок $S(a_0, a_1)$:

$$\frac{\partial S(a_0, a_1)}{\partial a_0} = 0;$$

$$\frac{\partial S(a_0, a_1)}{\partial a_1} = 0.$$

Перепишем критерий оптимальности с учетом конкретного вида $\varphi(x)$:

$$S(a_0, a_1) = \sum_{i=0}^{n} [a_0 + a_1 \cdot \sin(x_i) - y(x_i)]^2 \to \min.$$

Найдем частные производные:

$$\frac{\partial S(a_0,a_1)}{\partial a_0} = 2\cdot\sum_{i=1}^{n}[a_0 + a_1\cdot\sin(x_i) - y(x_i)] = 0;$$

$$\frac{\partial S(a_0,a_1)}{\partial a_1} = 2\cdot\sum_{i=1}^{n}[a_0 + a_1\cdot\sin(x_i) - y(x_i)]\cdot\sin(x_i) = 0.$$

Перепишем правые части последних уравнений, вынося за знак суммы неизвестные коэффициенты регрессионного уравнения a_0, a_1:

$$\begin{cases} a_0\cdot\sum_{i=1}^{n}1^2 + a_1\cdot\sum_{i=1}^{n}\sin(x_i) = \sum_{i=1}^{n}y(x_i)\cdot 1; \\ a_0\cdot\sum_{i=1}^{n}\sin(x_i) + a_1\cdot\sum_{i=1}^{n}\sin^2(x_i) = \sum_{i=1}^{n}y(x_i)\cdot\sin(x_i). \end{cases}$$

Полученные уравнения представляют собой систему линейных алгебраических уравнений относительно неизвестных a_0, a_1. Запишем систему в матрично-векторном виде:

$$\begin{bmatrix} \sum_{i=1}^{n}1 & \sum_{i=1}^{n}\sin(x_i) \\ \sum_{i=1}^{n}\sin(x_i) & \sum_{i=1}^{n}\sin^2(x_i) \end{bmatrix}\cdot\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}y(x_i) \\ \sum_{i=1}^{n}y(x_i)\cdot\sin(x_i) \end{bmatrix},$$

или

$$\Phi\cdot\overline{a} = \overline{y}\,.$$

Отсюда вектор искомых коэффициентов регрессионного уравнения

$$\overline{a} = \Phi^{-1}\cdot\overline{y}\,.$$

На рис. 6.6 приведен фрагмент программы MathCAd получения регрессионного уравнения для рассмотренного примера. Исходная функция $y(x)$ содержит полезный сигнал $f(x)$ и случайную помеху с равномерным законом распределения.

В программе рассмотрены 2 варианта получения регрессионных уравнений по базису – {1, sin$x$}и {1, x}. Как видно из графика, оба регрессионные уравнения хорошо устраняют случайную помеху, полученные функции более гладкие по сравнению с исходной. Качество аппроксимации оценивается с помощью среднеквадратической погрешности (s и $s1$). Значение $s1$ меньше. Это означает, что регресси-

онное уравнение $\varphi(x)= a_0+a_1 \cdot x$ в данной задаче предпочтительнее.

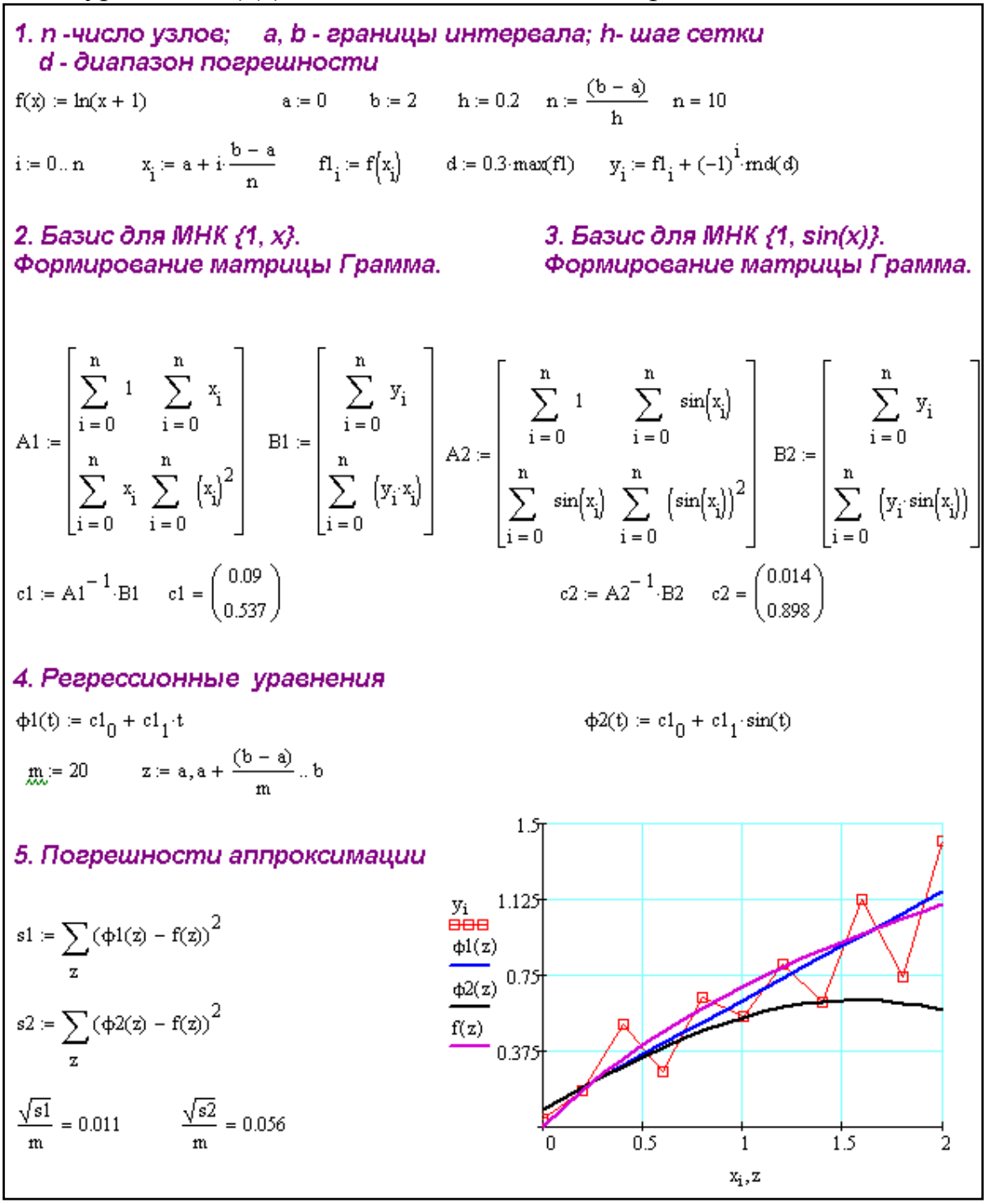


Рис. 6.6. Получение регрессионного уравнения методом наименьших квадратов

6.3. Экстраполяция данных

В ППП MathCAD существует процедура **predict(*y, n, m*)**, позволяющая прогнозировать (экстраполировать) данные на основании имеющихся табличных значений.

Описание функции	Аргументы функции
predict(*y, n, m*) Возвращает прогнозируемые значения функции.	***y*** **–** векторные значения функции; ***d*** – глубина предыстории; ***m*** – количество прогнозируемых значений.

Глубина предыстории – это количество последних значений вектора $\underline{y}\ \{y_0, y_1, \ldots, y_d\}$, которые учитываются в прогнозе, хотя число измеренных данных может быть много больше.

Для примера рассмотрим функцию $f(x) = \ln(x+1)$, заданную в 50 дискретных значениях. Вектор обозначен $\{f1_i\}$.

На рис. 6.7 приведен фрагмент программы MathCAD для прогноза значений данной функции.

Прогноз осуществляется на *m* значений вперед. На графике пунктирной линией обозначено продолжение исходной линии. Прогнозируемая кривая полностью совпадает с точными значениями функции в начале интервала (краткосрочный прогноз) и значительно отличается ри удалении от правого конца интервала *b*.

***Замечания*:**

1. Экспериментальных данных должно быть достаточное количество (>>10).

2. В прогнозе (в предыстории) должны участвовать не все экспериментальные данные, а лишь часть из них.

3. Эффективен только краткосрочный прогноз. Это видно и из графика в программе. Расхождение с истинными значениями заданной функции особенно заметно в последних точках прогнозируемых данных.

4. Шаг прогноза целесообразно выбирать такой же, как и в экспериментальных данных.

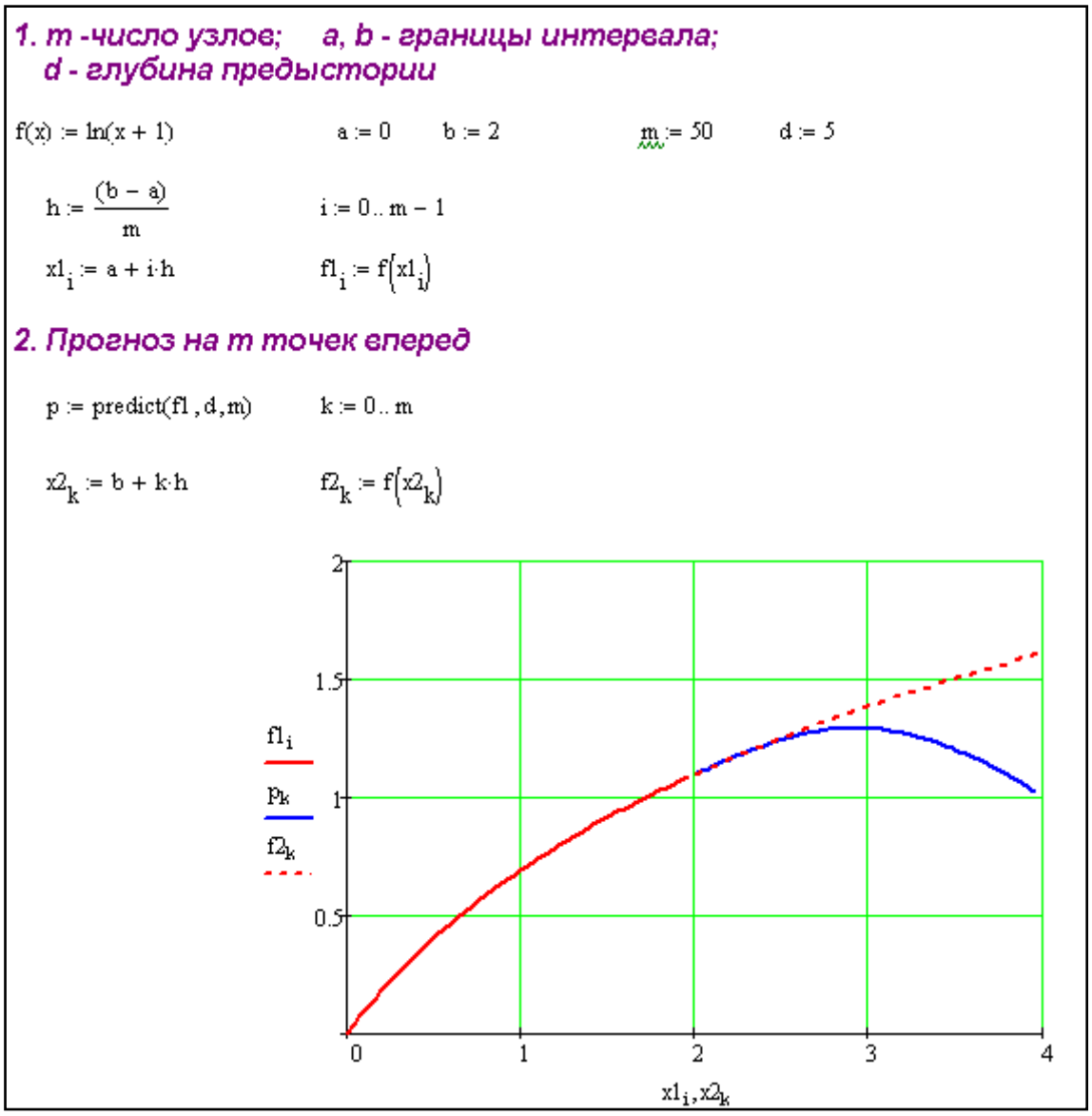


Рис. 6.7. прогнозирование данных в ППП MathCAD

6.4. Эффективность аппроксимации

Для оценки эффективности аппроксимации используются качественные и количественные показатели.

Качественная оценка проводится путем визуального оценивания графиков истинной и аппроксимирующей функций. Такое сравнение весьма наглядно, но субъективно.

Для количественной оценки вводят показатели точности аппроксимации.

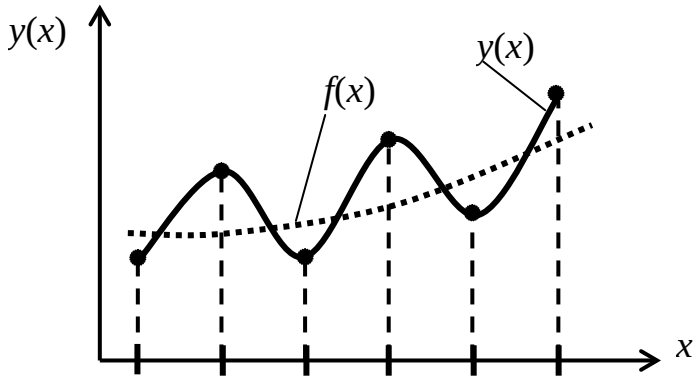


Рис. 6.8

Пусть $f(x)$ – истинная (заданная) функция, $f(x_i)$ – ее табличные значения;

$y(x_i) = f(x_i) + \xi_i$ – измеренные значения функции;

ξ_i – измерительный шум;

$\varphi(x)$ – аппроксимирующая функция, найденная по измеренным значениям $y(x_i)$.

Основными показателями эффективности аппроксимации являются:

- среднеквадратическая погрешность

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=0}^{n}[\varphi(x_i) - f(x_i)]^2}\,; \qquad (6.16)$$

- максимальная погрешность

$$\Delta = \max_i \left|\varphi(x_i) - f(x_i)\right|. \qquad (6.17)$$

Среднеквадратическая погрешность является усредненным показателем качества аппроксимации. При этом в разных частях интервала погрешность также различна. Как правило, на концах интервала погрешность выше. Часто в практических задачах бывает задана максимально допустимая погрешность.

Оценки погрешностей по формулам (6.16), (6.17) значительно зависят от числовых значений функции. Более показательной величиной является приведенная среднеквадратическая погрешность

$$\sigma 1 = \frac{\sigma}{\left|f_{\max} - f_{\min}\right|} \cdot 100\%, \qquad (6.18)$$

где $f_{\max}$, $f_{\min}$ – максимальное и минимальное значения исходной функции.

Приведенная погрешность выражена в процентах, является показательной и удобной. Например, погрешность 3–5 % допустима в технических приложениях, а погрешность 20–30 % говорит о плохом качестве аппроксимации.

Иногда приведение погрешности осуществляют не к диапазону функции $f_{\max} - f_{\min}$, а к ее среднему значению.

Вопросы для самопроверки

1. Основные задачи аппроксимации функций
2. Что такое узлы интерполяции?
3. Приведите понятие интерполирования данных.
4. В чем разница между глобальной и локальной интерполяцией?
5. Как происходит процесс интерполирования кубическими сплайнами?
6. Чем кусочно-линейная функция отличается от сплайна первого порядка?
7. Приведите понятие сплайн-функции.
8. Что такое базисные функции? Примеры базисных функций.
9. Что такое аппроксимация экспериментальных или табличных данных?
10. Что такое сглаживание данных?
11. Что такое регрессионное уравнение?
12. Что такое экстраполяция и прогноз данных?
13. В чем состоит основная идея сплайн аппроксимации?
14. Приведите критерий метода наименьших квадратов.
15. Укажите основные показатели качества аппроксимации.
16. Как получить регрессионное уравнение методом наименьших квадратов?
17. В чем заключается идея метода наименьших квадратов?

УЧЕБНО-МЕТОДИЧЕСКАЯ ЛИТЕРАТУРА

Литература основная

1. Амосов А.А., Дубинский Ю.А., Копченова Н.В. Вычислительные методы для инженеров: учеб. пособие. – М.: Высш. шк., 1994. – 544 с.

2. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы: учеб. пособие. – 6-е изд. – М.: БИНОМ. Лаборатория знаний, 2008. – 636 с.

3. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – М.: Лаборатория базовых знаний, 2001.

4. Боглаев Ю.П. Вычислительная математика и программирование: учеб. пособие для студентов втузов. – М.: Высш. шк., 1990. – 544 с.

5. Вержбицкий В. М. Основы численных методов / В. М. Вержбицкий. – М. : Высшая школа, 2002. – 840 с.

6. Волков Е.А. Численные методы: учеб. пособие для вузов. – 4-е изд., стер. – СПб.: Лань, 2007. – 256 с

7. Волков Е.А. Численные методы: учеб. пособие для ВУЗов. – 2-е изд., испр. – М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 248 с.

8. Воробьева Г.Н., Данилова А.Н. Практикум по вычислительной математике: учеб. пособие. – 2-е изд., перераб. и доп. – М.: Высш. шк., 1990. – 203 с.

9. Демидович Б.Р., Марон И.А. Основы вычислительной математики. – М.: Наука, 1970. – 664 с.

10. Демидович Б.П., Марон И.А. Основы вычислительной математики. – М.: Наука. Гл. ред. физ.-мат. лит., 1966. – 664 с.

11. Дробышевич В.И., Дымников В.П., Ривин Г.С. Задачи по вычислительной математике. – М.: Наука, 1980. – 144 с.

12. Дьяконов В.П., Абраменкова И.В. MathCAD 7 в математике, в физике и в Internet. – М.: Нолидж.,1998. – 304 с.

13. Заварыкин В.М., Житомирский В.Г., Лапчик М.П. Численные методы: учеб. пособие для студентов физ.-мат. спец. пед. ин-тов. – М.; Просвещение, 1990. – 176 с.

14. Завъялов Ю.С., Квасов Б.И., Мирошниченко В.Л. Методы сплайн-функций. – М.: Наука, 1980. – 353 с.

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

16. Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение: пер. с англ. – М.: Мир, 1998. – 575 с.

17. Киселевская С.В., Ушаков А.А. вычислительная математика. Численные методы : учебное пособие. – Владивосток : Изд-во ВГУЭС,

2009. – 96 с.

18. Крылов В.И., Бабков В.В., Монастырский П.И. Вычислительные методы: В 2т. – М.: Наука, 1976. – Т. 1. – 304 с.; 1977. – Т.2. – 400 с.

19. Маликов В.Т., Кветный Р.Н. Вычислительные методы и применение ЭВМ: учеб. пособие. – Киев: Выща школа. Головное изд-во, 1989. – 212 с.

20. Марчук Г.И. Методы вычислительной математики. – М.: Наука, 1989. – 456 с.

21. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. – Томск: МП «Раско», 1991. – 272 с.

22. Ортега Дж., Пул У. Введение в численные методы решения дифференциальных уравнений / пер. с англ.; под ред. А.А. Абрамова. – М.: Наука. Гл. ред. физ.-мат. лит., 1986. – 288 с.

23. Очков В.Ф. MathCAD 8 Pro для студентов и инженеров. – М.: КомпьютерПресс, 1999.

24. Очков В.Ф. MathCAD 8 Pro для студентов и инженеров. – М.: КомпьютерПресс, 1998. – 521 с.

25. Рыжиков Ю.И. Вычислительные методы : учебное пособие / Ю.И. Рыжиков. – Санкт-Петербург : БХВ-Петербург, 2007. – 400 с.

26. Самарский А.А., Гулин А.В. Численные методы. – М.: Наука, 1989. – 432 с.

27. Соболь И.М. Численные методы Монте-Карло. – М.: Наука, 1985. – 80 с.

28. Тейлор Дж. Введение в теорию ошибок / пер. с англ. – М.: Мир, 1985. – 282 с.

29. Турчак Л.И. Основы численных методов: учеб. пособие. – М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 320 с.

30. Формалев В.Ф., Ревизников Д.Л. Численные методы: учебник / под ред. А.И. Кибзуна. – М.: ФИЗМАТЛИТ, 2004. – 400 с.

31. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений: пер. с англ. – М.: Мир, 1980. – 279 с.

32. Шикин Е.В., Плис А.И. Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователей. – М.: ДИАЛОГ–МИФИ, 1996. – 240 с.

Литература дополнительная

33. Алексеев Е.Р., Чеснокова О.В., Рудченко Е.А. Scilab: Решение инженерных и математических задач. – М.: ALT Linux; Бином. Лаборатория знаний, 2008. – 260 с.

34. Бакланова Л.В., Огородников А.С., Офицеров В.В. Лабораторный практикум по численным методам: учеб. пособие. – Томск: изд.

ТПИ, 1990.

35. Воробьев Г. Н., Данилова А. Н. Практикум по численным методам. - М.:Высш. шк., 2007 г. -184 с.

36. Воскобойников Ю. Е. Программирование и решение задач в пакете MathCAD : учеб. пособие / Ю. Е. Воскобойников, В. Ф. Очков. – Новосибирск : НГАСУ, 2003. – 132 с.

37. Кацман Ю.Я. Прикладная математика. Численные методы: учеб. пособие. – Томск: Изд. ТПУ, 2000. – 68 с.

38. Левицкий А.А. Информатика. Основы численных методов: Лабораторный практикум / А. А. Левицкий. Красноярск: ИПЦ КГТУ, 2005. 111 с.

39. Плис А.И., Сливина Н.А. MathCAD: математический практикум. – М.: Финансы и Статистика, 1999. – 656 с.

40. Плис, А. И. MathCAD 2000. Математический практикум для экономистов и инженеров / А. И. Плис, Н. А. Сливина. М.: Финансы и статистика, 2000. 656с.

Учебное пособие

ТЕОРИЯ И РЕАЛИЗАЦИЯ ЗАДАЧ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ В ПАКЕТЕ MathCad

Учебное пособие

Составители
КОЧЕГУРОВ АЛЕКСАНДР ИВАНОВИЧ
КОЧЕГУРОВА ЕЛЕНА АЛЕКСЕЕВНА

Научный редактор
кандидат технических наук, доцент кафедры АиКС ИК
И.В. Цапко

Редактор Н.Т.Синельникова

Верстка *Л.А. Егорова*

Отпечатано в Издательстве ТПУ в полном соответствии с качеством предоставленного оригинал-макета

Подписано к печати ******** Формат 60×84/16.
Бумага «Снегурочка». Печать Xerox.
Усл. печ. л. **** Уч.-изд. л****.
Заказ . Тираж 50 экз.

Национальный исследовательский
Томский политехнический университет
Система менеджмента качества
Издательства Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту BS EN ISO 9001:2008

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30.
Тел./факс: 8(3822)56-35-35, www.tpu.ru