Саратовский государственный университет
имени Н. Г. Чернышевского

Введение в вычислительную физику: Функции, уравнения, интегралы

Учебно-методическое пособие
для студентов физического факультета

Саратов
2017

УДК 518

Автор: *С. В. Овчинников*

Введение в вычислительную физику: Функции, уравнения, интегралы : учеб.-метод. пособие по дисциплине «Вычислительная физика» для студентов физического факультета направления подготовки 03.03.02 «Физика» (бакалавриат) [Электронное издание] /

С. В. Овчинников. – Саратов : СГУ имени Н.Г. Чернышевского, 2017.

Рекомендуют к опубликованию :

кафедра общей физики;
доктор физико-математических наук, профессор *Г.В. Симоненко*
(Саратовский государственный университет)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Физику принято называть «точной» наукой. Это означает, что физика стремится дать ответы на поставленные перед ней задачи в количественной форме.

В задаче, связанной с натурным экспериментом, это будет совокупность приближенных значений измеряемых величин с учетом погрешности эксперимента. В задаче теоретической ответ дается в виде математических выражений того или иного уровня сложности. Причем очевидно, что для анализа полученного результата в соответствии с этими математическими выражениями необходимо провести числовые вычисления. Необходимо особо отметить, что вычислительный этап теоретической задачи зачастую является наиболее сложным и проблематичным.

Современные компьютеры кардинально облегчают проведение вычислений. Однако ими нужно пользоваться умело, и подходить к делу творчески. Компьютер освобождает человека от большого объема рутинной вычислительной работы. Но компьютер – инструмент, и ему надо подробно, без всякого намека на двусмысленность «втолковать», как производить вычисления. Поэтому творческая часть работы остается прерогативой человека. Роль компьютера в реализации теоретической задачи легче понять, если процесс ее реализации условно разбить на ряд этапов [1, 2]:

1. **Постановка задачи**. На этом этапе происходит осмысление исследуемого объекта или процесса, формулируются исходные положения, и определяется конечная цель исследования.

2. **Построение математической модели**. На этом этапе дается математическая формулировка поставленной задачи. Данный этап начинается с выделения тех факторов, которые следует принять во внимание. Здесь необходимо помнить, что построение модели является неизбежным компромиссом между учетом всех вероятных факторов, играющих роль в данной задаче, и сохранением математической модели достаточно простой, чтобы ее можно было решать имеющимися в нашем распоряжении средствами.

Далее записываются основные соотношения, соответствующие процессам в исследуемом объекте. В подавляющем большинстве случаев они имеют вид совокупности дифференциальных уравнений, дополненных условиями однозначности и необходимыми дополнительными соотношениями, например, материальными уравнениями в системе уравнений Максвелла.

Предполагается, что в окончательной формулировке математическая модель со всеми соответствующими граничными, начальными и дополнительными условиями имеет *единственное* решение.

3. **Реализация математической модели**. На этом этапе должен быть найден подходящий метод решения тех уравнений, которые составляют математическую модель. Для проблем, представляющих интерес сегодня, редко удается найти искомое решение в замкнутой аналитической форме. Поэтому на первый план сегодня вышли численные методы реализации математических моделей. Задача исследователя - правильно выбрать и грамотно применить тот или иной метод.

4. **Разработка алгоритма решения**. Выбранный метод решения должен быть выражен в виде конечной последовательности достаточно простых шагов, каждый из которых может быть однозначно сформулирован на каком-либо алгоритмическом языке. Это – алгоритм решения задачи. Часто алгоритм изображают графически в виде блок-схемы.

5. **Программирование:** перевод разработанного алгоритма на язык, понятный компьютеру. Совокупность соответствующих команд-операторов и является итоговой вычислительной программой[1].

6. **Отладка программы**. Составленная программа содержит, как правило, синтаксические и логические ошибки, которые на данном этапе должны быть устранены. После этого производится тестирование программы на специально подобранных тестовых задачах, решение которых известно.

7. **Проведение расчетов**. После отладки и тестирования программа может быть использована для проведения расчетов в соответствии с исходной задачей. Однако необходимо убедиться в ***обоснованности*** математической модели. То есть подтверждение того, что полученное решение и составленная на его основе программа являются удовлетворительными для тех целей, которые были поставлены. Имеется два главных источника возможных ошибок: 1) погрешности самой модели и 2) ошибки числовых вычислений. Первый источник ошибок обычно связан с факторами, учитываемыми или отбрасываемыми при построении математической модели, а второй будет описан в следующей главе данного пособия.

Проверка обоснованности модели является самостоятельной и весьма тонкой задачей. Эта проблема лежит вне рамок нашего методического пособия.

[1] Здесь мы не будем рассматривать вопросы компьютерного оформления и представления результатов вычислений.

Частично она должна рассматриваться в рамках дисциплины «Численные методы и математическое моделирование». Отметим только, что проверка обоснованности нередко приводит к необходимости модификации модели.

8. Проверка обоснованности модели прямо связана с завершающим и самым важным этапом работы: **анализом результатов**. При компьютерном решении прикладных задач нужно помнить девиз Р. Хемминга: «Цель расчетов – не числа, а понимание» [3]. В случае успеха полученные данные расчетов приводят к получению новых знаний; в случае неудачи приходится возвращаться на один из перечисленных выше этапов, иногда и на самый первый – уточнять саму постановку задачи.

Предлагаемое Вашему вниманию учебно-методическое пособие предназначено в помощь студентам, изучающим дисциплину «Вычислительная физика» и желающим начать приобретать навыки грамотной разработки вычислительного алгоритма и соответствующей компьютерной программы. В нем мы, естественно, не сможем сколько-нибудь детально изучить все эти восемь этапов. Его цель гораздо скромнее: *первоначальное ознакомление* с некоторыми вычислительными методами, применяемыми при решении физических задач. При этом предлагаемый теоретический материал желательно сопровождать решением серии практических заданий на занятиях в компьютерных классах.

В качестве языка программирования автором избран «старый» ФОРТРАН. Листинги вычислительных ФОРТРАН-программ намного понятнее, чем листинги, написанные на любом другом алгоритмическом языке. И по времени вычислений ФОРТРАНу нет конкурентов. Большинство серьезных моделей (модели физики атмосферы, геомагнитного поля, гравитационного поля Земли и др.) написаны именно на Фортране. Как отмечено во многих Internet- публикациях:

- ФОРТРАН прост в обучении, синтаксис понятен неподготовленному человеку. Познав основы, легко переучиться на любой другой язык.
- Бесплатный набор средств позволяет не получать лишних вопросов от правообладателей.
- Широко распространен по всему миру и содержит обширные математические библиотеки.
- Стандартизирован, может быть установлен на любые платформы и совместим с ранними версиями.
- Полезен для студентов технических, а особенно физико-математических специальностей, ввиду ориентации на научные и инженерные вычисления.

1. ОШИБКИ, ВОЗНИКАЮЩИЕ ПРИ КОМПЬЮТЕРНЫХ ВЫЧИСЛЕНИЯХ

Анализ ошибок полученного числового результата является необходимой составляющей любого серьезного вычисления. Исходные данные вычислительной задачи очень редко бывают точными, т. к. они обычно определяются экспериментально или основаны на приблизительных оценках. Сам процесс вычислений также вносит в результат различного рода ошибки.

Возможными источниками погрешностей при компьютерных вычислениях являются [2]:

1) ошибки в исходной информации;
2) ошибки ограничения;
3) ошибки округления.

Кроме того, для уверенности в конечном результате вычислений необходимо провести анализ выбранного алгоритма вычислений на его устойчивость и провести анализ самой решаемой задачи на устойчивость к малым изменениям исходных данных.

Ошибки в исходной информации возникают: 1) из-за естественной неточности экспериментальных данных, 2) из-за невозможности представить нужную величину конечной дробью и 3) из-за грубых просмотров. Поясним данное утверждение.

Всякая экспериментально измеренная физическая величина представляется интервалом значений, причем величина этого интервала соответствует погрешности эксперимента. Так заряд элементарный $e = (1{,}60210 \pm 0{,}00007)\cdot 10^{-19}$ Кл, число Авогадро $N_A = (6?02252 \pm 0{,}00028)\cdot 10^{23}$ моль$^{-1}$. Эти данные получены с помощью прецизионных экспериментов, и мы считаем указанные величины практически точными, так как их доверительные интервалы малы и составляют тысячные доли процента самих значений. Большинство же физико-технических данных, представленных в справочниках, даются с куда большей погрешностью (до 10 – 15 %). Именно на основе таких данных очень часто формулируются исходные положения для теоретической задачи.

Многие числа нельзя представить точно ограниченным числом значащих цифр. Числа π и e явный тому пример. Даже обыкновенные дроби часто нельзя представить с помощью конечного числа десятичных знаков: 1/3 = 0,33333...

Бывает, что дроби, которые являются конечными в одной системе счисления, становятся бесконечными в другой. Например, число 0,1. В двоичной

системе счисления 0,1 = 0,0001100011000110011... Хранящееся в памяти компьютера число 0,1 в двоичной системе естественно содержит ограниченное число разрядов. Поэтому, вычисляя на компьютере сумму из десяти чисел 0,1, значение 1,0 точно получить нельзя.

Что же касается грубых ошибок в исходной информации, не связанных с сутью решаемой задачи, то их выявление – задача самого исследователя.

1.1. Ошибки ограничения и округления

Ошибки ограничения возникают в результате ограничения бесконечного математического процесса при вычислениях. Например, известно представление функции $Sin(x)$ в виде бесконечного ряда Тейлора:

$$Sin(x) = x - x^3/3! + x^5/5! - x^7/7! + ...$$

Этот ряд знакопеременный и сходится абсолютно, следовательно, абсолютная ошибка вычислений не превосходит первого из отбрасываемых слагаемых. Поскольку при вычислениях невозможно использовать все члены указанного бесконечного ряда, то полученная ошибка и является примером ошибки ограничения.

Ошибки округления возникают из-за того обстоятельства, что все реальные компьютеры работают с конечным количеством разрядов представления числа (16, 32 или 64).

Вопросы округления относятся только к операциям с *действительными* числами. При выполнении арифметических операций с целыми числами потребности в округлении не возникает, если результат сам должен быть представлен целым числом.

Любое действительное число в компьютере представляется в нормализованном виде:

$$x = f \cdot 10^r.$$

Здесь *мантисса* f – десятичная дробь, у которой первая значащая цифра не равна нулю, а число r называется порядком. Например, представление числа π в нормализованном виде: $\pi = 0{,}314159265... \cdot 10^1$.

При проведении вычислений с действительными числами компьютер автоматически выравнивает их порядки. Это означает, что порядки сначала сравниваются, а затем мантисса *меньшего* по абсолютной величине числа сдвигает-

ся *вправо* на столько разрядов, сколько необходимо, чтобы порядки стали одинаковыми (термин «плавающая точка» возник именно по этой причине).

Рассмотрим простой пример [2]. Пусть необходимо сложить два числа: 162,4 и 1,769 – в некотором «гипотетическом» компьютере, который работает с четырьмя значащими цифрами в мантиссе и с одной цифрой в порядке. Сначала компьютер представит эти числа в виде: $0{,}1624 \cdot 10^3$ и $0{,}1769 \cdot 10^1$. Затем при операции сложения произойдет выравнивание порядков к значению $r = 3$, т.е.

$$0{,}1624 \cdot 10^3 + 0{,}001769 \cdot 10^3 = 0{,}164169 \cdot 10^3.$$

Но «гипотетический» компьютер работает с четырьмя значащими цифрами! Поэтому результат сложения в нашем примере будет равен $0{,}1641 \cdot 10^3$ или $0.1642 \cdot 10^3$ в зависимости от того, каким образом в компьютере реализовано правило округления – симметричное округление или правило отбрасывания. Данный результат отличен от точного значения результата сложения, и это отличие представляет собой ошибку округления. Отметим, что большинство компьютеров, работающих с широкой разрядной сеткой чисел (32 или 64), при округлении для экономии машинного времени используют правило отбрасывания. Можно показать, что при представлении действительного числа в компьютере с использованием правила отбрасывания максимальная относительная ошибка округления зависит только от количества значащих цифр в числе и не зависит от величины этого числа: $|\varepsilon| \le 10^{-t+1}$, где t - число значащих цифр в мантиссе числа.

Отметим, что ошибка симметричного округления никогда не превосходит ошибки округления с отбрасыванием и в среднем вдвое меньше ее.

Важным является следующий вопрос: как ошибка, возникшая в каком-либо месте вычислительного процесса, распространяется дальше, т.е. усиливается или уменьшается ее влияние по мере того, как производятся последующие операции. Для ответа на этот вопрос найдем выражения для абсолютной ошибки результата каждого из 4-х арифметических действий как функцию величин, участвующих в этих действиях.

Пусть мы имеем два каких-либо точных значения x и y и два соответствующих приближенных значения $\bar{x}$ и $\bar{y}$. Соответствующие абсолютные ошибки обозначим как e_x и e_y.

Рассмотрим операцию сложения:

$$x + y = \bar{x} + e_x + \bar{y} + e_y = (\bar{x} + \bar{y}) + (e_x + e_y).$$

Ошибка суммы равна сумме абсолютных ошибок: $e_{x+y} = e_ч + e_y$.

Аналогичным образом для операции вычитания получим: $e_{x-y} = e_ч - e_y$.

Для операции умножения: $x \cdot y = (\bar{x} + e_x)(\bar{y} + e_y) = \overline{xy} + \bar{x}e_y + \bar{y}e_x + e_x e_y$. Пренебрегая последним слагаемым в силу его малости, получим:

$x \cdot y \approx \overline{xy} + \bar{x}e_y + \bar{y}e_x$. Таким образом, ошибка произведения равна:

$$e_{xy} \approx \bar{x} \cdot e_y + \bar{y} \cdot e_x.$$

Теперь рассмотрим операцию деления. Имеем:

$$\frac{x}{y} = \frac{\bar{x} + e_x}{\bar{y} + e_y} = \frac{\bar{x} + e_x}{\bar{y}} \cdot \frac{1}{\left(1 + \frac{e_y}{\bar{y}}\right)}.$$

Так как $\left|\frac{e_y}{\bar{y}}\right| << 1$, то $\frac{x}{y} \approx \frac{\bar{x}}{\bar{y}} + \frac{e_x}{\bar{y}} - \frac{\bar{x}}{\bar{y}^2} e_y$. Следовательно, $e_{x/y} \approx \frac{e_x}{\bar{y}} - \frac{\bar{x}}{\bar{y}^2} e_y$.

Что означают приведенные формулы? Арифметическая операция начинает выполняться с двумя приближенными значениями $\bar{x}$ и $\bar{y}$, имеющими соответствующие ошибки e_x и e_y. Ошибки эти могут быть любого происхождения: разброс экспериментальных результатов, или предварительные вычисления с ошибками ограничения, или результаты предыдущих арифметических действий с соответствующими ошибками, или комбинация всего перечисленного здесь. Вышеприведенные формулы дают возможность оценить ошибку результата каждого из арифметических действий как функцию от $\bar{x}$, $\bar{y}$, e_x и e_y. Ошибка же округления самого арифметического действия при этом *не учитывается*. Для оценки того, как распространяется в последующих арифметических операциях ошибка данного результата, необходимо к вычисленной по одной из четырех формул ошибке результата *прибавить отдельно ошибку округления*.

1.2. Переполнение и машинный ноль

Снова рассмотрим «гипотетический» компьютер, в котором любое действительное число представляется в виде мантиссы f с $t = 4$ значащими цифрами и порядком r из одной цифры ($-9 \le r \le 9$). Очевидно, что максимальное число, возможное для представления в таком компьютере, равно x_{max} =

$0{,}9999 \cdot 10^9$. Также очевидно, что результат любой арифметической операции, больший, чем x_{max}, выходит за границы возможного представления числа в нем. Попытка произвести арифметическую операцию с таким результатом обычно вызывает в компьютере сигнал **переполнения**, после чего вычисления, как правило, прекращаются, поскольку в пределах имеющейся разрядной сетки невозможно дать разумное толкование такому числу-результату. Переполнение наиболее характерно для операций умножения и возведения в степень.

При «плавающем» (по расположению десятичной точки) умножении возможно возникновение «**машинного нуля**», когда два ненулевых сомножителя имеют ненулевое произведение, но меньшее по абсолютной величине наименьшего положительного числа, возможного к представлению в разрядной сетке конкретного компьютера. В нашем компьютере таким числом будет являться величина $x_{min} = 0{,}1000 \cdot 10^{-9}$.

В конкретном компьютере точность «плавающей» операции сложения характеризуется так называемым *машинным эпсилон* (ε) – наименьшим числом с плавающей точкой, которое может распознать компьютер в операции сравнения $1 + \varepsilon > 1$ при заданной разрядной сетке представления чисел.

Существует несколько методов для приближенного вычисления ε. Вот один из вариантов:

Метки Позиции 1 – 5	6	Операторы. Позиции 7 - 72
		program epsilon
C		вычисление машинного эпсилон, i – номер итерации
		real*8 eps,eps1
		eps=1.
1		i=0
		eps=eps*.5
		i=i+1
		eps1=eps+1.
		if(eps1.gt.1.) goto 1
		print 10,eps*2,i
10		format(' eps=',e13.8,' i=',i4)
		read*
		end

Результат: eps=0.22204460e–15, i=53.

Теперь вычислим «машинный ноль». Для его определения применим следующую программу:

Метки Позиции 1 – 5	6	Операторы. Позиции 7 - 72
		program zero
C		вычисление «машинного нуля», i – номер итерации
		real*8 eps,eps1
		eps=1
1		i=0
		eps1=eps
		eps=eps*.5
		i=i+1
10		if(eps.gt.0.) goto 1
		print 10,eps1,i
		format(' eps1=zero=',e16.8,' i=',i4)
		read*
		end

Результат[2]: zero =0.49406565e–323, i=1075.

1.3. Пример влияния ошибок округления и ограничения

Посмотрим как на результаты вычислений влияют ошибки округления, сопровождающие арифметические операции с плавающей точкой.

Известно следующее разложение экспоненты e^x в ряд Тейлора:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots \quad .$$

С помощью приведенной ниже простой программы попробуем применить эту формулу для различных значений показателя экспоненты и различном числе слагаемых в ряду Тейлора. Будем рассматривать только отрицательные показатели, чтобы не связываться с очень большими числами, возникающими из-за быстрого роста экспоненциальной функции.

Метки Позиции 1 – 5	6	Операторы. Позиции 7 - 72
		program expon
C		x–аргумент, e1 – точное значение,
C		e2 – значение, вычисленное с помощью ряда Тейлора с числом слагаемых n+1
		real*8 x,e1,e2,a

[2] Вычисления проводились на 64-х разрядной машине с процессором «Athlon»

```
      read 5 x,n
5     format(f5.1,i6)
      e1=exp(x)
        a=1.
      e2=1.
      do 1 i=1,n
       a=a*x/i
      e2=e2+a
1     continue
       print 10,x,e1,e2,n
10     format('  x=',f5.1,'  e1=',e12.6,'  e2=',e12.6,'  n='i6)
       read*
      end
```

Сначала положим $x = -5{,}5$. Значение экспоненты, вычисленное с помощью стандартной функции, $exp(-5{,}5)=0{,}408677\cdot10^{-2}$. Ряд Тейлора с $n=50$ дает то же самое значение. Для $x = -10$ при $n=50$ оба значения при шести значащих цифрах также совпадают: $exp(-10)=0{,}453999\cdot10^{-4}$.

Однако для $x = -15$ значения экспоненты, вычисленные с помощью стандартной функции и с помощью ряда Тейлора, не совпадают ни при каких n. Это иллюстрировано таблицей 1.1.

Таблица 1.1

Сравнения значений экспоненты, вычисленной с помощью стандартной функции и с помощью ряда Тейлора

x	$exp(x)$	Значения, вычисленные с помощью ряда Тейлора
–15	$0{,}305902\cdot10^{-6}$	$0{,}305855\cdot10^{-6}$ для всех $n>100$
–20	$0{,}206115\cdot10^{-8}$	1,04692 для $n=50$ $0{,}589002\cdot10^{-9}$ для всех $n>100$
–30	$0{,}935762\cdot10^{-13}$	$0{,}227128\cdot10^{-4}$ для всех $n>100$

Представленные результаты говорят о том, что, начиная с некоторого значения x, вычисления экспоненты с помощью ряда Тейлора теряют точность. В рассмотренном примере потеря точности возникла из-за того, что в бесконечных сходящихся рядах типа $\Sigma\, x^k/k!$ для $|x|>(2\ldots8{,}5)$ слагаемые сначала возрастают по абсолютной величине, поскольку x^k сначала растет быстрее, чем $k!$. Ошибка, накапливающаяся при суммировании в таких рядах больших по абсолютной величине слагаемых, может полностью свести на нет сумму оставшихся малых слагаемых из-за ограниченного числа разрядов, используемых для представления чисел. При больших $|x|$ даже может возникнуть ситуация, когда численное значение целой части одного или нескольких слагаемых в таком

ряду не помещается в разрядной сетке компьютера. Естественно, произойдет катастрофическая потеря точности. В нашем примере положение можно исправить, по меньшей мере, двумя способами.

1). Провести вычисление ряда в обратном порядке, начиная суммирование с последнего слагаемого, наименьшего по абсолютной величине. Однако отметим, что чем больше $|x|$, тем больше вероятность ошибочного конечного результата и при таком вычислении.

2). Реальные алгоритмы вычисления экспоненты начинаются с разложения ее аргумента x на целую и дробную части: $x=m+f$, где m – целое, а $0 \le f < 1$. Далее используются известные свойства показательной функции:

$$e^x = e^{m+f} = e^m \cdot e^f.$$

Первый сомножитель e^m вычисляется простым перемножением числа e нужное число раз, а степенной ряд для второго сомножителя сходится очень быстро, поскольку $f < 1$.Ну а для $x < 0$ $exp(-x) = 1/exp(x)$.

1.4. Пример неустойчивого алгоритма

Часто причиной получения неверных результатов является нерациональный вычислительный алгоритм.

Рассмотрим пример. Предположим, что нам нужно вычислить серию интегралов [2]:

$$E_n = \int_0^1 x^n e^{x-1} dx, \qquad n = 1,2,\ldots.$$

Путем интегрирования по частям несложно получить следующие рекуррентные соотношения:

$$E_n = 1 - n \cdot E_{n-1}, \quad n=2,\ldots,$$

где $E_1 = 1/e$. При вычислениях в десятичной системе с шестиразрядной сеткой получаются следующие значения первых девяти значений последовательности E_n:

$E_1 \approx 0.367879$; $E_2 \approx 0.264242$; $E_3 \approx 0{,}207274$;
$E_4 \approx 0.170904$; $E_5 \approx 0.145480$; $E_6 \approx 0.127120$;
$E_7 \approx 0.110160$; $E_8 \approx 0.118720$; $E_9 \approx -0.06848$.

Интеграл E_9 получился отрицательным, чего быть не может, так как подынтегральное выражение $x^9 \cdot e^{x-1}$ положительно во всем отрезке интегрирования.

Причиной является единственная ошибка округления, сделанная при вычислении $E_1 = 1/e$, когда проводилось деление. Затем допущенная ошибка округления умножалась на (–2) при вычислении E_2, затем ошибка в E_2 умножается на (–3) при вычислении E_3 и т.д. В итоге, ошибка в E_9 есть ошибка, допущенная в E_1, умноженная на (–2)(–3)...(–9) = 9! Такое огромное увеличение исходной ошибки является результатом неустойчивости выбранного алгоритма. Подлинное значение $E_9 = 0.0916$ (с тремя значащими цифрами).

В этом примере избавиться от неустойчивости алгоритма несложно. Достаточно переписать исходное рекуррентное соотношение в виде

$$E_{n-1} = \frac{1 - E_n}{n}, \quad n = \ldots 3,2,1.$$

Теперь на каждом шаге вычислений ошибка в E_n умножается на множитель $1/n$. Следовательно, если мы начнем со значения для некоторого E_n при $n >> 1$ и будем вычислять в обратном направлении, то любая начальная ошибка и промежуточные ошибки округлений будут уменьшаться на каждом шаге. Это и есть *устойчивый алгоритм.* Как выбрать начальное значение? Для этого заметим, что имеет место неравенство: $E_n \le 1/(n+1)$, то есть E_n уменьшается с ростом n. Поэтому, если мы заменим нулем, например, число E_{20}, то сделаем начальную ошибку не большую 1/21. Эта ошибка умножается на 1/20 при вычислении E_{19}, поэтому ошибка при вычислении E_{19} не больше $(1/20)\cdot(1/21) \approx 0.0024$. Ко времени вычисления E_{15} начальная ошибка уменьшится до величины, меньшей $4 \cdot 10^{-8}$, что меньше единственной ошибки округления для используемой разрядной сетки. Поэтому вычисленные значения от E_{15} до E_9 будут точными в пределах шести значащих цифр.

В заключении отметим, что в физике встречаются и так называемые неустойчивые задачи, решение которых может очень сильно меняться при незначительном изменении исходных данных. В таких задачах неустойчивость обусловлена исключительно их природой и никак не связана с методами и алгоритмами их решения.

Проблемы исследования неустойчивых задач и изучения методов их решения выходят за рамки данного учебного курса.

***Задания к разделу* 1.**

1. Вычислить ошибку результата следующих операций: $u = (x + y)\cdot z$.
2. Найти корни квадратного уравнения $x^2 + 0{,}4002\,x + 0{,}00008 = 0$, проводя вычисления с точностью до 4-х и до 8-ми значащих цифр. Сравнить результаты.
3. Провести вычисления синуса угла $2\pi n + 30^o$, n=0,1...4, используя разложение синуса в ряд Тейлора:

$$Sin(x) = x - x^3/\,3! + x^5/\,5! - x^7/\,7! + x^9/\,9! - \ldots$$

При вычислениях число слагаемых проварьировать (в разумных пределах), а их суммирование проводить в двух направления - от первого к последнему и наоборот. Результаты сравнить. $Sin\ 30^o = 0{,}5$. Обратить внимание на материал параграфа 1.3.

4. Реализовать на компьютере вычисления величин E_n (параграф 1.4) в пределах одинарной точности разрядной сетки Вашего компьютера. Определить значение n, для которого вычисление E_n, начиная с больших n, даст точный результат в пределах 8-ми значащих цифр для величин $E_1 \ldots E_{30}$.

Запомните:

1. Если необходимо производить сложение - вычитание длинной последовательности чисел, работайте сначала с *наименьшими* числами.
2. По возможности, избегайте вычитания почти равных чисел.
3. Выражение вида $a(b\text{-}c)$ можно свести к выражению $ab - ac$, а выражение $(b - c)/a$ к выражению $b/a - c/a$. Если числа в разности близки между собой, не ленитесь доводить вычисления до умножения или деления. При этом задача не будет осложняться дополнительными ошибками округления.
4. Старайтесь свести к минимуму число необходимых арифметических операций, но при этом учитывайте сказанное выше в разделе 1 данного учебного пособия.

2. МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Огромное количество расчетных задач связано с решением систем линейных алгебраических уравнений. В частности, численные методы решения задач математической физики часто сводятся к решению таких систем.

Напомним, что система алгебраических уравнений

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &= b_1; \\ \bullet \quad \bullet \quad \bullet & \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n &= b_n \end{aligned} \qquad (2.1)$$

называется линейной, если все a_{ij} и b_i есть постоянные. Квадратная матрица $A = (a_{ij})$ размерности $n \cdot n$ называется матрицей коэффициентов, а вектор $\boldsymbol{B} = (b_i)$ называется столбцом свободных членов. Мы будем предполагать, что $\det(A) \neq 0$, так что решение системы (2.1) существует и единственно.

Сделаем замечание о так называемых *плохо обусловленных* системах уравнений. С точки зрения математики система линейных уравнений либо вырождена, либо нет. С точки же зрения практических вычислений могут существовать почти вырожденные системы, при решении которых получаются недостоверные значения неизвестных. Рассмотрим простой пример. Имеем систему из двух уравнений:

$$\begin{aligned} 5x + 7y &= 12; \\ 7x + 10y &= 17. \end{aligned} \qquad (2.2)$$

Эта система имеет единственное решение: $x = 1$ и $y = 1$. Теперь рассмотрим следующие значения: $x = 2{,}415$ и $y = 0$. При их подстановке в (2.2) мы получим:

$$\begin{aligned} 5x + 7y &= 12{,}075; \\ 7x + 10y &= 16{,}905. \end{aligned} \qquad (2.3)$$

С погрешностью до 0,1 или с точностью до двух значащих цифр (именно так задана система (2.2)) значения (2,415; 0) также могут считаться решением системы (2.2).

Дело в том, что две прямые, описываемые уравнениями (2.2), почти параллельны, см. рис. 2.1. И все точки пространства (x, y), лежащие между этими

прямыми вблизи точки их пересечения (x=1, y=1) будут с некоторой не слишком большой погрешностью удовлетворять системе (2.2). Такие системы и называются ***плохо обусловленными***. Найти с высокой точностью численное решение такой системы трудно, особенно с учетом конечноразрядного представления чисел в компьютере. Однако этот пример показывает, что проверка исходной системы линейных алгебраических уравнений на обусловленность перед ее решением (или в процессе решения) весьма желательна. Графическая интерпретация хорошо и плохо обусловленных систем, а также вырожденных систем уравнений на примере системы из двух уравнений с двумя неизвестными представлена на рис. 2.1.

Методы численного решения системы (2.1) делятся на две группы: прямые методы и итерационные методы. В *прямых методах* решение системы (2.1) находится за конечное число арифметических действий. Эти методы еще называют точными методами решения, хотя при их реализации на компьютере термин “точные” можно использовать лишь с поправкой на ошибки округления.

Итерационные методы (методы последовательных приближений) состоят в том, что решение (x_i) системы (2.1) находится как предел при $k\to\infty$ последовательных приближений $x_i^{(k)}$, где k – номер итерации. Как правило, за конечное число итераций этот предел не достигается. Обычно задается некоторое малое число ε (погрешность), и вычисления проводятся до тех пор, пока не будет выполнено условие $|(x_i)^{(k)} - (x_i)^{(k-1)}| \le \varepsilon$.

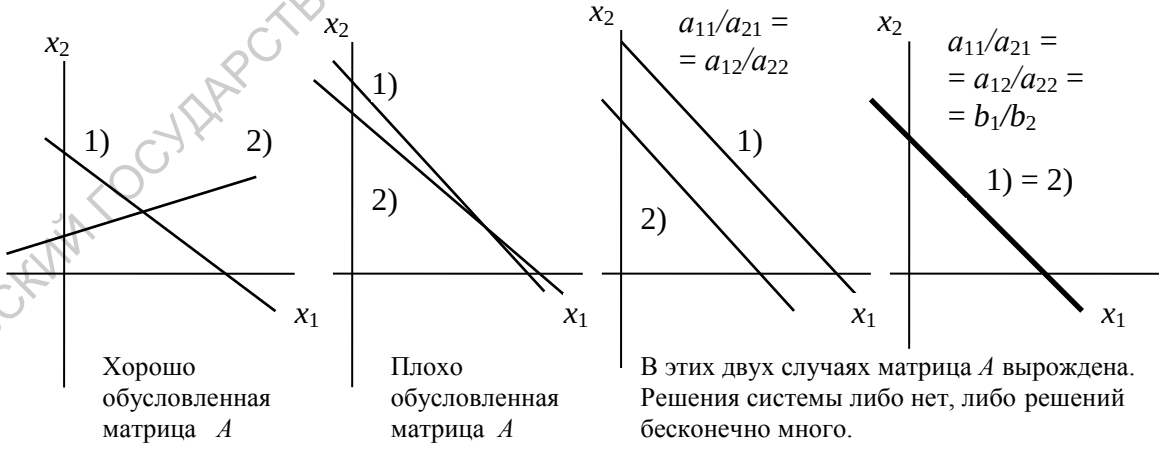


Рис. 2.1. Графическое представление обусловленности матрицы коэффициентов для системы:

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

Решение системы – точка пересечения прямых 1) и 2).

В настоящее время разработано очень большое число алгоритмов решения линейных систем, многие из которых ориентированы на матрицы A специального вида (трехдиагональные, симметричные, ленточные, разреженные). Здесь мы рассмотрим два “классических” метода решения систем (2.1): прямой метод Гаусса и итерационный метод Гаусса - Зейделя [2 – 6].

2.1. Метод последовательных исключений Гаусса

Метод Гаусса является одним из наиболее известных и широко применяемых на практике прямых методов решения систем линейных алгебраических уравнений.

Для иллюстрации этого метода рассмотрим систему из трех уравнений с тремя неизвестными:

$$a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 = b_1, \tag{2.4}$$

$$a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 = b_2, \tag{2.5}$$

$$a_{31}\,x_1 + a_{32}\,x_2 + a_{33}\,x_3 = b_3. \tag{2.6}$$

Без потери общности будем считать, что $a_{11} \neq 0$ – этого всегда можно добиться перестановкой строк и/или столбцов исходной системы.

Введем множитель $m_2 = a_{21} / a_{11}$. Умножим первое уравнение (2.4) на m_2 и вычтем его из второго уравнения (2.5). Получим

$$(a_{21} - m_2 a_{11})\cdot x_1 + (a_{22} - m_2 a_{12})\cdot x_2 + (a_{23} - m_2 a_{13})\cdot x_3 = b_2 - m_2 b_1. \tag{2.7}$$

Но так как $a_{21} - m_2 a_{11} = 0$, то значение x_1 из уравнения (2.7) исключается (именно для этого и выбирался вид множителя m_2). Уравнение (2.7) преобразуем к виду

$$a_{22}'\,x_2 + a_{23}'\,x_3 = b_2', \tag{2.8}$$

где $a_{22}' = a_{22} - m_2 a_{12}$; $a_{23}' = a_{23} - m_2 a_{13}$; $b_2' = b_2 - m_2 b_1$.

Заменим уравнение (2.5) исходной системы уравнением (2.7) и проделаем аналогичные действия с уравнением (2.6). Введем множитель $m_3 = a_{31} / a_{11}$, умножим первое уравнение (2.4) исходной системы на этот множитель и вычтем его из третьего уравнения (2.6) исходной системы. Коэффициент при x_1 снова станет нулевым, и третье уравнение приобретет вид

$$a_{32}'\,x_2 + a_{33}'\,x_3 = b_3', \tag{2.9}$$

где $a_{32}' = a_{32} - m_3 a_{12}$; $a_{33}' = a_{33} - m_3 a_{13}$; $b_3' = b_3 - m_3 b_1$.
Преобразованная система будет выглядеть следующим образом:

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1 , \quad (2.4)$$
$$a_{22}' x_2 + a_{23}' x_3 = b_2' , \quad (2.8)$$
$$a_{32}' x_2 + a_{33}' x_3 = b_3' . \quad (2.9)$$

Понятно, что новая система полностью эквивалентна исходной, но имеет то преимущество, что x_1 не входит ни во второе, ни в третье уравнение. То есть второе и третье уравнения представляют собой систему из двух уравнений с двумя неизвестными. Если ее решить, то есть найти x_2 и x_3, то результат можно подставить в первое уравнение и определить x_1.

Дальнейшие действия очевидны. Исключим x_2 из двух последних уравнений. Считаем, что $a_{22}' \neq 0$, в противном случае производим перестановку. Если же окажется, что $a_{22}' = 0$ и $a_{32}' = 0$, то исходная система уравнений вырождена и либо не имеет решения, либо имеет бесконечное множество решений – этот случай во внимание не принимаем. Вводя множитель $m_3' = a_{32}' / a_{22}'$, умножая на него уравнение (2.8) и вычитая результат из уравнения (2.9), получим преобразованное уравнение (2.9):

$$a_{33}'' x_3 = b_3'' ; \quad \text{где} \quad b_3'' = b_3' - m_3' b_2' .$$

В результате всех упомянутых операций исходная система будет преобразована к треугольному виду:

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1 , \quad (2.4)$$
$$a_{22}' x_2 + a_{23}' x_3 = b_2' , \quad (2.8)$$
$$a_{33}'' x_3 = b_3'' . \quad (2.10)$$

Получение системы (2.4), (2.8), (2.10) часто называют ***прямым ходом*** метода Гаусса. ***Обратный ход*** заключается в нахождении неизвестных x_1, x_2 и x_3 очевидным образом:

$$x_3 = b_3'' / a_{33}'' ; \quad x_2 = (b_2' - a_{23}' x_3)/a_{22}' ; \quad x_1 = (b_1 - a_{12} x_2 - a_{13} x_3)/a_{11} .$$

Еще раз отметим, что мы подразумевали возможную перестановку в уравнениях таким образом, чтобы a_{11} и a_{22}' *не были равными нулю.* Если окажется, что $a_{33}''= 0$, то исходная система *вырождена*.

Теперь обобщим рассмотренный пример на систему из n уравнений с n неизвестными. Введем следующую индексацию:

- k означает номер того уравнения, которое вычитается из остальных, а также номер того неизвестного, которое исключается из оставшихся $n–k$ уравнений;
- i означает номер уравнения, из которого в данный момент исключается неизвестное;
- j означает номер столбца.

Тогда для **прямого хода:**

$$m_i^{(k-1)} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)}, \; i = k+1, \ldots n; \quad \text{причем} \; a_{kk}^{(k-1)} \neq 0; \qquad (2.11)$$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_i^{(k-1)} a_{kj}^{(k-1)}, \; \text{где} \; i = k+1, \ldots n, \; j = k, \ldots n, \qquad (2.12)$$

$$b_i^{(k)} = b_i^{(k-1)} - m_i^{(k-1)} b_k^{(k-1)}. \qquad (2.13)$$

Индекс k последовательно принимает значения от 1 до n–1 включительно. При $k = n - 1$ происходит исключение x_{n-1} из последнего уравнения системы. Необходимо помнить, что в выражениях для $a_{ij}^{(k)}$ первый индекс определяет номер строки, а второй – номер столбца.

Полезно заметить, что процесс исключения неизвестных при прямом ходе **не изменяет** *абсолютной величины* определителя исходной матрицы коэффициентов $A = (a_{ij})$, хотя знак определителя меняется при каждой перестановке. Значение определителя будет равно произведению диагональных элементов получившейся треугольной матрицы, причем знак этого произведения надо изменить на *обратный*, если количество перестановок было *нечетным*.

Обратная подстановка определяется формулами:

$$\begin{aligned} x_n &= b_n^{n-1} / a_{nn}^{n-1}, \\ x_{n-1} &= \frac{b_{n-1}^{(n-2)} - a_{n-1,n}^{(n-2)} \cdot x_n}{a_{n-1,n-1}^{(n-2)}}, \\ &\ldots \\ x_j &= \left(b_j^{(j-1)} - a_{jn}^{(j-1)} \cdot x_n - \ldots - a_{j,j+1}^{(j-1)} \cdot x_{j+1}\right) / a_{jj}^{(j-1)}, \end{aligned} \qquad (2.14)$$

для $j = n–2, n–3, \ldots 1$.

В заключение отметим, что для минимизации ошибок округления при прямом ходе желательно, чтобы значения $|m_i^{(k-1)}|$ были возможно меньшими. Для этого перестановку уравнений нужно осуществлять так, чтобы на каждом этапе исключения *наибольший по абсолютной величине* коэффициент при x_k попадал на главную диагональ; в результате ни один из множителей $m_i^{(k-1)}$ не превзойдет единицу по абсолютной величине. Такая реализация перестановок получила название перестановок с *выбором ведущего элемента*.

В Приложении Б приведен пример программы нахождения решения системы линейных алгебраических уравнений методом исключений Гаусса с выбором ведущего элемента. Эта программа получена путем модифицирования программ, представленных в [4] .

2.2. Уточнение решения системы линейных алгебраических уравнений

Весьма вероятна ситуация, когда ошибки округления влияют на результат. Обычно это происходит, когда размерность системы уравнений достаточно большая или решение системы было проведено без перестановок с выбором ведущего элемента. Совершенно очевидно, как провести проверку найденного решения. Для этого достаточно подставить найденные значения в исходные уравнения.

Пусть найдено решение системы (2.1), и мы сомневаемся в его точности. Обозначим это решение через $x_1^{(0)} \ldots x_n^{(0)}$. Подставим его в левые части системы (2.1) и получим:

$$\begin{gathered} a_{11} x_1^{(0)} + a_{12} x_2^{(0)} + a_{13} x_3^{(0)} + \ldots + a_{1n} x_n^{(0)} = b_1^{(0)}; \\ \bullet \qquad \bullet \qquad \bullet \\ a_{n1} x_1^{(0)} + a_{n2} x_2^{(0)} + a_{n3} x_3^{(0)} + \ldots + a_{nn} x_n^{(0)} = b_n^{(0)} . \end{gathered} \tag{2.15}$$

Если $b_i^{(0)}$ сильно отличаются от b_i, то $x_i^{(0)}$ не является достаточно хорошим приближением к точному решению. С другой стороны, даже если все $b_i^{(0)}$ достаточно близки к b_i, то $x_i^{(0)}$ все равно может явиться плохим приближением к точному решению, см. пример (2.2) – (2.3).

Если вычесть каждое уравнение системы (2.15) из соответствующего уравнения исходной системы (2.1) и обозначить

$$e_i^{(0)} = x_i - x_i^{(0)}, \; t_i^{(0)} = b_i - b_i^{(0)}, \; i = 1, \ldots n, \tag{2.16}$$

то получим:

$$a_{11}\, e_1^{(0)} + a_{12}\, e_2^{(0)} + a_{13}\, e_3^{(0)} + \ldots + a_{1n}\, e_n^{(0)} = t_1^{(0)};$$
$$\bullet \qquad \bullet \qquad \bullet \tag{2.17}$$
$$a_{n1}\, e_1^{(0)} + a_{n2}\, e_2^{(0)} + a_{n3}\, e_3^{(0)} + \ldots + a_{nn}\, e_n^{(0)} = t_n^{(0)}\ .$$

Решая систему (2.17) методом исключений, найдем все $e_i^{(0}$. Тогда новое приближение к точному решению системы (2.1) определяется следующим образом:

$$x_i^{(1)} = x_i^{(0)} + e_i^{(0)}, \quad i = 1,\ldots n.$$

Подставляя это приближение в исходную систему (2.1), и вводя затем

$$e_i^{(1)} = x_i - x_i^{(1)},\ t_i^{(1)} = b_i - b_i^{(1)},\ i = 1, \ldots n,$$

нетрудно записать новую систему для определения $e_i^{(1)}$:

$$a_{11}\, e_1^{(1)} + a_{12}\, e_2^{(1)} + a_{13}\, e_3^{(1)} + \ldots + a_{1n}\, e_n^{(1)} = t_1^{(1)};$$
$$\bullet \qquad \bullet \qquad \bullet \tag{2.18}$$
$$a_{n1}\, e_1^{(1)} + a_{n2}\, e_2^{(1)} + a_{n3}\, e_3^{(1)} + \ldots + a_{nn}\, e_n^{(1)} = t_n^{(1)}\ .$$

Новое приближение после этого запишется в следующем виде:

$$x_i^{(2)} = x_i^{(1)} + e_i^{(1)}, \quad i = 1,\ldots n.$$

Этот процесс можно продолжать до тех пор, пока все e_i не станут меньше некоторой наперед заданной погрешности. Обращаем внимание на то обстоятельство, что *нельзя останавливать* вычисления только потому, что все t_i стали малыми – рассмотренный выше пример (2.2) – (2.3) показывает, что при малых t_i решение системы не обязательно будет точным.

Описанный здесь метод фактически представляет собой итерационный процесс последовательных приближений. Метод последовательных уточнений некоторого значения путем подстановки этого значения или линейной комбинации на его основе в определяющее выражение широко используется в практике вычислений.

2.3. Итерационный метод Гаусса - Зейделя

Рассмотрим систему из трех уравнений с тремя неизвестными, на примере которой мы рассматривали реализацию метода исключений Гаусса:

$$a_{11}\, x_1 + a_{12}\, x_2 + a_{13}\, x_3 = b_1, \tag{2.4}$$
$$a_{21}\, x_1 + a_{22}\, x_2 + a_{23}\, x_3 = b_2, \tag{2.5}$$
$$a_{31}\, x_1 + a_{32}\, x_2 + a_{33}\, x_3 = b_3. \tag{2.6}$$

Будем считать, что диагональные элементы матрицы коэффициентов данной системы отличны от нуля. В противном случае необходимо провести

соответствующие перестановки. В этих условиях систему (2.4) – (2.6) можно переписать в виде:

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}\ ; \tag{2.19}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}\ ; \tag{2.20}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}\ . \tag{2.21}$$

Выберем некоторое приближение x_1^0, x_2^0 и x_3^0 как решение системы (2.19) – (2.21); это приближение будет нулевым по номеру. Нулевое приближение подставим в систему (2.19) – (2.21), соблюдая следующие правила такой подстановки:

1) сначала вычисляется новое значение $x_1^{(1)} = (b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)})/a_{11}$;
2) используя только что вычисленное значение $x_1^{(1)}$ и начальное значение x_2^0 , вычисляется новое значение $x_2^{(1)} = (b_2 - a_{21}x_1^{(1)} - a_{21}x_3^{(0)})/a_{22}$;
3) Используя найденные значения $x_1^{(1)}$ и $x_2^{(1)}$, вычисляется новое значение

$$x_3^{(1)} = (b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)})/a_{33}\ .$$

На этом первая итерация заканчивается. На второй итерации исходные значения x_1^0, x_2^0 и x_3^0 заменяются на x_1^1, x_2^1 и x_3^1, и процесс вычислений по пунктам 1) … 3) повторяется снова.

В общем случае *k*-тое приближение определяется формулами:

$$\begin{aligned} x_1^{(k)} &= (b_1 - a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)})/a_{11};\\ x_2^{(k)} &= (b_2 - a_{21}x_1^{(k)} - a_{21}x_3^{(k-1)})/a_{22};\\ x_3^{(k)} &= (b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)})/a_{33}. \end{aligned} \tag{2.22}$$

Описанный метод решения системы линейных алгебраических уравнений известен как *итерационный метод Гаусса - Зейделя*. Этот метод очень удобен для реализации на компьютере. Итерационный процесс определения x_i^k продолжается до тех пор, пока не выполнится условие:

$$max\left|\frac{x_i^{(k)} - x_i^{(k-1)}}{x_i^{(k)}}\right| \le \varepsilon,$$

где ε - требуемая относительная погрешность вычислений.

Обобщим расчетные выражения (2.22) на случай системы из *n* уравнений. По прежнему предполагаем, что все диагональные коэффициента матрицы *A* отличны от нуля: для всех i $a_{ii} \neq 0$. Тогда *k*-тое приближение к решению задается следующей формулой:

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - a_{i1}x_1^{(k)} - \ldots - a_{i,i-1}x_{i-1}^{(k)} - a_{i,i+1}x_{i+1}^{(k-1)} - \ldots - a_{in}x_n^{(k-1)}\right), \tag{2.23}$$

$$i = 1,2..n\ .$$

Замечание. Сходимость итерационного процесса по методу Гаусса – Зейделя гарантируется при условии, что главная диагональ матрицы коэффициентов системы состоит из максимальных по абсолютной величине значений коэффициентов соответствующих строк, удовлетворяющих условию [5 – 6]:

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|, \quad (i = 1, 2 \ldots n). \qquad (2.24)$$

Этого можно добиться путем перестановок строк и столбцов исходной матрицы. Однако существуют такие системы уравнений, в которых указанное условие может не выполняться.

Как привести систему уравнений к виду, удобному для ее решения итерационным методом, то есть к виду типа (2.19) – (2.21)? На практике поступают следующим образом. Из заданной системы выделяют уравнения с коэффициентами, модули которых больше суммы модулей остальных коэффициентов уравнения. Каждое выделенное уравнение вписывают в такую строку новой системы, чтобы наибольший по модулю коэффициент оказался *диагональным*.

Из оставшихся неиспользованных и выделенных уравнений системы составляют другие линейно независимые строчки новой системы с соблюдением указанного выше условия (2.24). Поясним сказанное на примере.

Имеем систему уравнений:

$$2x_1 + 3x_2 - 4x_3 + x_4 = 3, \qquad (A)$$
$$x_1 - 2x_2 - 5x_3 + x_4 = 2, \qquad (B)$$
$$5x_1 - 3x_2 + x_3 - 4x_4 = 1, \qquad (C)$$
$$10x_1 + 2x_2 - x_3 + 2x_4 = -4. \qquad (D)$$

Видно, что в уравнениях (B) и (D) модули коэффициентов при x_3 и x_1 соответственно больше суммы модулей остальных коэффициентов. Поэтому уравнение (D) можно принять первым уравнением новой системы, а уравнение (B) - третьим ее уравнением:

$$10x_1 + 2x_2 - x_3 + 2x_4 = -4, \qquad (I)$$
$$\ldots\ldots\ldots\ldots\ldots \qquad (II)$$
$$x_1 - 2x_2 - 5x_3 + x_4 = 2, \qquad (III)$$
$$\ldots\ldots\ldots\ldots\ldots \qquad (IV)$$

Для получения уравнения (II) можно из уравнения (A) вычесть уравнение (B) , получим:

$$x_1 + 5x_2 + x_3 + 0 \cdot x_4 = 1. \qquad (II)$$

Последнее уравнение новой системы получаем, используя следующую линейную комбинацию: 2(A) – (B) + 2(C) – (D). Тогда

$$3x_1 + 0\cdot x_2 + 0\cdot x_3 - 9x_4 = 10. \qquad \text{(IV)}$$

Теперь полученную преобразованную систему (I) – (IV) нужно привести к виду (2.19) – (2.21), удобному для проведения итераций. Для этого систему (I) – (IV) достаточно разрешить относительно диагональных элементов:

$x_1 = \ 0\cdot x_1 - 0{,}2\, x_2 + 0{,}1\, x_3 - 0{,}2\, x_4 - 0{,}4;$

$x_2 = -\,0{,}2\, x_1 + \ 0\cdot x_2 - 0{,}2\, x_3 + \ 0\cdot x_4 + 0{,}2;$

$x_3 = \ 0{,}2\, x_1 - 0{,}4\, x_2 + \ 0\cdot x_3 + 0{,}2\, x_4 - 0{,}4\ ;$

$x_4 = 0{,}33(3)\, x_1 + 0\cdot x_2 + 0\cdot x_3 + \ 0\cdot x_4 - 1,\ 11(1)\ .$

Исходная система (A) – (D) приведена к виду, удобному для итераций.

2.4. Метод прогонки

Многие задачи вычислительной физики при их численной реализации приводят к системам линейных алгебраических уравнений определенного вида. Матрица $A = (a_{ij})$ коэффициентов в них имеет отличные от нуля элементы только на ее главной диагонали и на двух диагоналях, смежных с нею. Такая матрица коэффициентов называется трехдиагональной:

$$\begin{gathered}
a_{11}x_1 + a_{12}x_2 + \ 0 + \quad \ldots \quad + 0 = b_1; \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + 0 \ldots + 0 = b_2; \\
\cdots\cdots \\
0 + \ldots 0 + a_{i,i-1}x_{i-1} + a_{i,i}x_i + a_{i,i+1}x_{i+1} + 0 + \ldots + 0 = b_i; \\
\cdots\cdots \\
0 + \ldots + 0 + \ldots + a_{n,n-1}x_{n-1} + a_{n,n}x_n = b_n
\end{gathered} \qquad (2.25)$$

Обычно системы типа (2.25) решают *методом прогонки*, являющимся модификацией метода последовательных исключений Гаусса.

Будем искать решение системы (2.25) в виде

$$x_i = \alpha_i \cdot x_{i+1} + \beta_i, \quad (i = 1, 2 \ldots n-1) \qquad (2.26)$$

где α_i и β_i – пока неизвестные величины. Определим их:

1) Из первого уравнения системы (2.25) с учетом (2.26) получим:

$$x_1 = \alpha_1 \cdot x_2 + \beta_1 \ \rightarrow \ \alpha_1 = -a_{12}/a_{11}; \ \ \beta_1 = b_1/a_{11}.$$

2) Из второго уравнения:

$$x_2 = \alpha_2 \cdot x_3 + \beta_2 \quad \rightarrow$$

$$\rightarrow \quad a_{21}(\alpha_1 \cdot x_2 + \beta_1) + a_{22}(\alpha_2 \cdot x_3 + \beta_2) + a_{23} \cdot x_3 = b_2 \quad \rightarrow$$

$$\rightarrow \quad \alpha_2 = -\frac{a_{23}}{a_{21} \cdot \alpha_1 + a_{22}}; \quad \beta_2 = \frac{b_2 - a_{21} \cdot \alpha_1}{a_{21} \cdot \alpha_1 + a_{22}}.$$

3) На шаге с номером i:

$$x_i = \alpha_i \cdot x_{i+1} + \beta_i;$$

$$\alpha_i = -\frac{a_{i,i+1}}{a_{i,i-1} \cdot \alpha_{i-1} + a_{i,i}}; \qquad (2.27)$$

$$\beta_i = \frac{b_i - a_{i,i-1} \cdot \alpha_{i-1}}{a_{i,i-1} \cdot \alpha_{i-1} + a_{i,i}}, \quad i = 2,3,..n-1.$$

4) На последнем шаге с номером n определим x_n.

Поскольку $x_{n-1} = \alpha_{n-1} \cdot x_n + \beta_{n-1}$, то из последнего уравнения системы получим:

$$a_{n,n-1} \cdot (\alpha_{n-1} \cdot x_n + \beta_{n-1}) + a_{n,n} x_n = b_n.$$

Отсюда

$$x_n = \frac{b_n - a_{n,n-1} \cdot \beta_{n-1}}{a_{n,n-1} \cdot \alpha_{n-1} + a_{n,n}}. \qquad (2.28)$$

Нахождение коэффициентов α_i и β_i по формулам (2.27) называется *прямой прогонкой*, а сами коэффициенты – *прогоночными* коэффициентами [5]. После того, как все прогоночные коэффициенты найдены и определено значение x_n, по рекуррентной формуле (2.26)

$$x_i = \alpha_i \cdot x_{i+1} + \beta_i$$

определяются все x_i (обратная прогонка).

Очевидно, что метод прогонки можно применять, если знаменатели выражений (2.27) и (2.28) отличны от нуля.

Приведем один из возможных вариантов подпрограммы, реализующей метод прогонки.

Сначала заметим, что всю матрицу коэффициентов A с учетом всех нулевых элементов в память компьютера заносить не нужно. Достаточно запомнить три одномерных массива: коэффициенты левой диагонали $c(i)=a_{i,i-1}$, коэффициенты главной диагонали $a(i)=a_{i,i}$ и коэффициенты правой диагонали $d(i)=a_{i,i+1}$, $i=1,2,\ldots n-1$. При этом очевидно, что формально $c(1)=0$ и $d(n)=0$. Кроме того, необходимо запомнить массив $b(i)=b_i$ – столбец свободных членов исходной системы, ввести два вспомогательных массива прогоночных коэффициентов $alf(i)=\alpha_i$ и $bet(i)=\beta_i$ ($i=1,2,\ldots n-1$) и массив $x(i)$ ($i=1,2,\ldots n$) – столбец решения.

Метки Позиции 1 – 5	6	Операторы. Позиции 7 - 72
		Subroutine progon(n,c,a,d,b,x)
C		реализация метода прогонки
		real c(n),a(n),d(n),b(n),x(n),alf(n),bet(n)
C		n – формальная размерность матрицы коэффициентов
C		c(n) – коэффициенты левой диагонали
C		a(n) – коэффициенты главной диагонали
C		d(n) – коэффициенты правой диагонали
C		b(n) – столбец свободных членов
C		x(n) – столбец-решение
		c(1)=0.
		d(n)=0.
		n1=n–1
		alf(1)=–d(1)/a(1)
		bet(1)=b(1)/a(1)
C		прямая прогонка
		do 1 i=2,n1
		y=c(i)*alf(i–1)
		zn=y+a(i)
		alf(i)=–d(i)/zn
		bet(i)=(b(i)–y)/zn
1		continue
C		обратная прогонка
		x(n)=(b(n)–c(n)*bet(n1))/(c(n)*alf(n1)+a(n))
		do 2 j=1,n1
		i=n–j
		x(i)=alf(i)*x(i+1)+bet(i)
2		continue
		return
		end

2.5 . Краткие сравнительные характеристики

Естественный вопрос: какой из методов предпочтительнее?

Метод исключений конечен и с его помощью (теоретически!) можно решить любую невырожденную систему уравнений. Итерационный метод Гаусса – Зейделя может быть применен не всегда, гарантия его сходимости оговорена в Замечании к предыдущему параграфу. Однако когда итерационные методы сходятся, они обычно предпочтительнее [2 – 4]: 1) время вычислений итераци-

онным методом пропорционально n^2, время вычислений по методу исключений пропорционально n^3; 2) ошибки округления при итерационном методе обычно меньше, иногда это соображение может оказаться важным.

Многие системы уравнений, возникающие в практических задачах, имеют среди коэффициентов много нулей. В этом случае итерационные методы (при условии их сходимости) в высшей степени предпочтительны. При решении с помощью компьютера такой системы можно проверять коэффициенты и не производить умножения, если они равны нулю. Кроме того, бывают такие большие системы уравнений, что их точное решение методом исключений просто невозможно.

Задания к разделу 2.

1. Решить систему уравнений методом Гаусса, проводя вычисления вручную и с помощью компьютерной программы:

$2x_1 + 2x_2 - x_3 + x_4 = 4,$
$4x_1 + 3x_2 - x_3 + 2x_4 = 6,$
$8x_1 + 5x_2 - 3x_3 + 4x_4 = 12,$
$3x_1 + 3x_2 - 2x_3 + 2x_4 = 6.$

Ответ для контроля:
$x_1 = 1, x_2 = 1, x_3 = -1, x_4 = -1.$

2. На основе метода Гаусса - Зейделя решить систему уравнений, первоначально приведя ее к виду, удобному для итераций:

$4x_1 - x_2 + x_3 = 4,$
$x_1 + 6x_2 + 2x_3 = 9,$
$-x_1 - 2x_2 + 5x_3 = 2.$

Точное решение: $x_1 = x_2 = x_3 = 1.$

Указание: положить $x_1^0 = x_2^0 = x_3^0 = 0.$

3. Методом прогонки решить следующую систему уравнений:

$2x_1 + x_2 = -5,$
$x_1 + 10x_2 - 5x_3 = -18,$
$x_2 - 5x_3 + 2x_4 = -40,$
$x_3 + 4x_4 = -27.$

Ответ для контроля:
$x_1 = -3, x_2 = 1, x_3 = 5, x_4 = -8.$

3. ПРАКТИЧЕСКОЕ ВЫЧИСЛЕНИЕ ФУНКЦИЙ

Экспериментальные данные практически всегда доступны в виде некоторого набора дискретных значений (таблицы данных). С другой стороны, зависимость между различными параметрами какого-либо процесса удобно представлять в виде функциональной зависимости. Такая зависимость позволяет определить значения в промежуточных точках таблицы данных, вычислить интеграл или производную и, в конце концов, сделать графическое представление данных в виде гладкой функции.

Эта приближающая дискретную таблицу данных гладкая функция в большом количестве случаев основана на приближении полиномами, поскольку с полиномами легко обращаться – интеграл и производная от полинома тоже полином. Однако для многих целей используются приближения и функциями других классов.

Существуют три группы функций, широко применяемых в численном анализе [2, 5]. Первая группа включает в себя линейные комбинации функций типа

$$1, x, x^2, \ldots x^n,$$

что совпадает с классом всех полиномов степени n (или меньше).

Вторую группу образуют функции

$$sin(a \cdot x), \ cos(a \cdot x).$$

Эти функции используются для приближенного описания периодических процессов, в том числе с помощью рядов Фурье.

Третья группа образована функциями $exp(-a \cdot x)$.

Каждая из этих трех групп обладает одним важным свойством: конечное множество функций такой группы переходит само в себя, когда x заменяется на $x+k$. Например, если $P(x)$ – полином степени n, то $P(x+k)$ – также полином степени n, хотя, конечно, их коэффициенты различны. То же самое относится к двум другим группам функций.

Кроме того, для полиномиальной группы функции выполняется еще одно важное свойство: замена x на $k \cdot x$ также переводит множество функций этой группы само в себя. Экспоненциальные и тригонометрические функции данным свойством не обладают.

Ниже мы рассмотрим простейшие случаи полиномиального приближения дискретной таблицы данных.

3.1. Вычисление функций с помощью полиномиальных приближений

Назовем полиномом целой степени n следующую функцию аргумента x:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n , \qquad (3.1)$$

где $a_0 ... a_n$ - коэффициенты полинома.

Многие конкретные задачи и многие методы численного анализа приводят к необходимости выполнения расчетов в соответствии с (3.1). Главное неудобство прямых вычислений по этой формуле – необходимость вычислений высоких степеней x. Но формулу (3.1) нетрудно привести к виду:

$$P_n(x) = a_0 + x\,(a_1 + x\,(a_2 + ... + x\,(a_{n-1} + x\,a_n)\,...)). \qquad (3.2)$$

Подразумевается, что при вычислениях самые внутренние скобки должны быть раскрыты первыми. Этот метод вычисления полинома получил название *правила (схемы) Горнера* [4, 7]. Очевидно, вычисление конкретного значения полинома при $x=\xi$ сводится к циклическому применению следующих формул:

$$b_0 = a_n ; \quad b_k = b_{k-1} * \xi + a_{n-k} ; \; k = 1, 2, ... n; \quad P_n(\xi) = b_n .$$

Для вычислений по схеме Горнера требуется n умножений и n сложений. Число же умножений при использовании выражения (3.1) равно $n(n+1)/2$, если каждая степень x получается путем последовательного перемножения. Доказано, что для полиномов общего вида нельзя построить схему более экономичную по числу операций. Кроме того, вычисления по схеме Горнера приводят к меньшим ошибкам округления.

Наш интерес к вычислению полиномов по схеме Горнера связан с тем, что для многих специальных функций математической физики имеются достаточно точные *полиномиальные аппроксимации*, см., например, справочник [8].

3.1.1. Интерполяционные полиномы Лагранжа и Ньютона

Интерполяция – приближенное представление некоторой функции $f(x)$, заданной дискретно $y_i = f(x_i)$ в конечном наборе точек x_i, принадлежащих области ее определения. Значения $\{x_i , y_i\}$ называются узловыми значениями или просто ***узлами***. При этом значения приближающей функции должны *совпадать* со значениями приближаемой функции в узловых точках. Нередко добавляется требование и о совпадении в узловых точках значений производных (до некоторого порядка включительно) приближающей и приближаемой функций.

Пусть известны значения

$$y_0 = f(x_0), y_1 = f(x_1), \ldots y_k = f(x_i), \ldots y_n = f(x_n)$$
$$(i = 0,1..n)$$

некоторой физической величины $f(x)$ в точках x_i, а требуется определить ее значения в других точках $x \in [x_0, x_n]$, не совпадающих с обозначенными. При этом из каких-то соображений примерно ясен вид функции $f(x)$. Тогда задача интерполирования состоит в определении некоторой функции $F(x)$, точно совпадающей с $f(x)$ во всех точках x_i – ***узлах*** интерполяции.

Ясно, что такая задача допускает сколь угодно *много* решений. Предположим, что приближаемая функция на отрезке своего определения гладкая и не носит сильно выраженного колебательного характера. Тогда задача интерполирования становится *однозначной*, если в качестве интерполирующей функции $F(x)$ для функции $f(x)$, заданной n+1 своими значениями, выбрать полином $P_n(x)$ степени не выше n, такой, что

$$P_n(x_0) = y_0 , \quad P_n(x_1) = y_1, \ \ldots \ \ P_n(x_n) = y_n . \tag{3.3}$$

Полином $P_n(x)$, удовлетворяющий этим условиям, называется ***интерполяционным*** полиномом.

В этом параграфе кратко изложены методы полиномиальной интерполяции. Такая интерполяция просто реализуема и широко используется в практике вычислений [4 – 7].

Условие (3.3) позволяет однозначно определить коэффициенты интерполяционного полинома. Действительно, пусть $P_n(x)$ – полином степени n, определенный выражением (3.1). Его коэффициенты $a_0 \ldots a_n$ - постоянные действительные и пока неизвестные величины. Тогда из (3.1) нетрудно получить следующую систему n+1 линейных алгебраических уравнений для определения a_0, $\ldots a_n$:

$$\begin{aligned} &a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n = y_0 ; \\ &a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n = y_1 ; \\ &\qquad\qquad \bullet \quad \bullet \quad \bullet \\ &a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n = y_n . \end{aligned} \tag{3.4}$$

Неизвестные $a_0 \ldots a_n$ находятся по формулам Крамера:

$$a_0 = \Delta_1/\Delta; \quad a_1 = \Delta_2/\Delta; \quad \ldots \ a_n = \Delta_n/\Delta,$$

где Δ - определитель системы (3.4), а Δ_i – определитель, получающийся заменой столбца коэффициентов при a_i столбцом свободных членов $(i = 0,1..n)$. Таким образом, интерполяционный полином определен. Решение системы (3.4) можно проводить любыми удобными методами, например, методом Гаусса.

В частности, интерполяционный полином можно представить в следующем широко распространенном виде:

$$P_n(x) = L_n(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)\cdot\ldots\cdot(x-x_{i-1})\cdot(x-x_{i+1})\cdot\ldots\cdot(x-x_n)}{(x_i-x_0)\cdot\ldots\cdot(x_i-x_{i-1})\cdot(x_i-x_{i+1})\cdot\ldots\cdot(x_i-x_n)}, \qquad (3.5)$$

Полином вида (3.5) носит название ***интерполяционного полинома Лагранжа*** и обычно обозначается как $L_n(x)$. Данный полином может быть использован как для равно отстоящих, так и произвольно расположенных узлов интерполирования, что часто является важным. Кроме того, он содержит узловые значения интерполируемой функции в явном виде. Рассмотрим два частных случая полинома Лагранжа.

При $n = 1$ имеем две узловые точки, и полином Лагранжа в этом случае представляет собой уравнение прямой, проходящей через две заданные точки:

$$y(x) = y_0 (x - b) / (a - b) + y_1 (x - a)/(b - a) ,$$

где a, b - абсциссы этих точек.

При $n = 2$ имеем уравнение параболы, проходящей через три точки с абсциссами a, b и c:

$$y(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)} y_0 + \frac{(x-a)(x-c)}{(b-a)(b-c)} y_1 + \frac{(x-a)(x-b)}{(c-a)(c-b)} y_2 .$$

Другой широко используемый интерполирующий полином определяется формулой Ньютона:

$$P_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + a_3(x-x_0)(x-x_1)(x-x_2) + \\ + \ldots + a_n(x-x_0)(x-x_1)(x-x_2)\cdot\ldots\cdot(x-x_{n-1}). \qquad (3.6)$$

Одним из преимуществ формулы Ньютона является простота нахождения коэффициентов полинома. Действительно, полагая $x = x_0$, получим

$$a_0 = f(x_0),$$

так как все слагаемые, кроме первого, станут равными нулю. Далее, выбирая $x = x_1$, будем иметь

$$P_n(x_1) = f(x_1) = a_0 + a_1(x_1 - x_0) = f(x_0) + a_1(x_1 - x_0),$$

откуда $a_1 = \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$.

Теперь положим $x = x_2$. Тогда

$$P_n(x_2) = f(x_2) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1).$$

Поэтому

$$a_2 = \frac{f(x_2) - f(x_0) - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0)}{(x_2 - x_0)(x_1 - x_0)} =$$

$$= \frac{y_2 - y_0 - \dfrac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0)}{(x_2 - x_0)(x_1 - x_0)} = \frac{(y_2 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_0)}{(x_2 - x_0)(x_1 - x_0)^2}.$$

Следующие коэффициенты (a_3, a_4, … a_n) в интерполяционном полиноме Ньютона вычисляются по той же схеме. Так, с учетом того, что a_0, a_1 и a_2 формально вычислены, имеем:

$$a_3 = \frac{y_3 - a_0 - a_1(x_3 - x_0) - a_2(x_3 - x_0)(x_3 - x_1)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)},$$

$$a_4 = \frac{y_4 - a_0 - a_1(x_4 - x_0) - a_2(x_4 - x_0)(x_4 - x_1) - a_3(x_4 - x_0)(x_4 - x_1)(x_4 - x_2)}{(x_4 - x_0)(x_4 - x_1)(x_4 - x_2)(x_4 - x_3)},$$

. . . . . .

Отметим, что и в формуле (3.6) и при определении последнего коэффициента a_n не используется последнее узловое значение $\{x_n, y_n\}$. Поэтому эта формула не удобна для интерполирования дискретно заданной функции $f(x)$ вблизи конца таблицы данных. Однако формулу (3.6) можно переписать несколько в другом виде:

$$P_n^*(x) = a_0 + a_1(x - x_n) + a_2(x - x_n)(x - x_{n-1}) + a_3(x - x_n)(x - x_{n-1})(x - x_{n-2}) + \\ + \ldots + a_n(x - x_n)(x - x_{n-1})(x - x_{n-2}) \cdot \ldots \cdot (x - x_1). \qquad (3.6,а)$$

Как поступать дальше понятно. Полагаем $x = x_n$, тогда $a_0 = P_n^*(x_n) = y_n$. Затем полагаем $x = x_{n-1}$. Тогда $a_0 + a_1(x_{n-1} - x_n) = f(x_{n-1}) = y_{n-1}$, поэтому

$$a_1 = \frac{y_{n-1} - y_n}{x_{n-1} - x_n}$$

и так далее.

Формула (3.6) называется первой интерполяционной формулой Ньютона (интерполирование «вперед»), а формула (3.6,а) – второй (интерполирование «назад»).

Еще одним замечательным свойством интерполяционных формул Ньютона является очевидное соотношение:

$$P_n(x) = P_{n-1}(x) + a_n(x - x_0)(x - x_1)(x - x_2)\cdot\ldots\cdot(x - x_{n-1}).$$

Оно позволяет для повышения точности интерполяции добавлять в полиномиальную сумму новые слагаемые, что требует подключения дополнительных узлов. При этом для формул Ньютона безразлично, в каком порядке эти новые узлы подключаются.

Отметим, что для полинома Лагранжа при добавлении новых узлов все расчеты надо производить заново.

3.1.2. Погрешность интерполяции

Еще раз подчеркнем, что существует один и только один интерполяционный полином при заданном наборе узлов интерполяции. Формулы Лагранжа, Ньютона и другие (здесь не рассмотренные) описывают один и тот же полином (при условии, что вычисления проводятся точно). Повышение точности интерполяции целесообразно проводить за счет уменьшения шага и специального расположения точек x_i. Выбор способа интерполяции определяется различными соображениями: точностью, временем вычисления, погрешностями округления и др.

Естественный вопрос, возникающий при интерполяции, заключается в том, насколько близко построенный полином заданной степени n приближается к функции $f(x)$ в ***неузловых*** точках отрезка $[x_0, x_n]$ области определения переменной x. Очевидно, что такую оценку корректно проводить только в тех случаях, когда приближаемая функция $f(x)$ сама достаточно гладкая. Поэтому потребуем, что на отрезке $x_0 \le x \le x_n$, содержащем узлы интерполяции, функция $f(x)$ имела все производные $f'(x)$, $f''(x)$, ... $f^{(n+1)}(x)$ до $(n+1)$-го порядка включительно. Тогда для интерполяционного полинома в виде полинома Лагранжа или Ньютона справедлива следующая оценка [5, 6] для абсолютной погрешности:

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

где $M_{n+1} = \max\limits_{x_0 \le x \le x_n} |f^{(n+1)}(x)|$, $\Pi_{n+1}(x) = (x - x_0)(x - x_1)(x - x_2)\ldots(x - x_n)$.

Теперь предположим, что вид функции $f(x)$, которую необходимо заменить интерполяционным полиномом $L_n(x)$, известен. Возникает следующий вопрос: будет ли стремиться к нулю погрешность такой интерполяции $f(x) - L_n(x)$, если число узлов n интерполяции и, следовательно, степень полинома неограниченно увеличивать? В общем случае ответ на этот вопрос *отрицательный*

[4]. Случайные ошибки в значениях приближаемой функции сильно искажают интерполяционные многочлены высоких степеней. Причем необходимо учитывать сильное накопление погрешностей при вычислениях для систем с большим числом узлов. С другой стороны, интерполяция полиномами низкой степени функции, заданной большим числом узловых точек, может привести к потере существенной информации. Поэтому на практике избегают интерполяционных полиномов высокой степени, предпочитая применять для большого числа узлов либо кусочно-полиномиальную интерполяцию, разбивая совокупность узлов на относительно малые подобласти и применяя для них интерполяционные полиномы невысокой степени (в частности, сплайн-интерполяция), либо ***аппроксимацию*** полиномами или рациональными функциями с минимизацией среднеквадратичной или абсолютной ошибки приближения.

Примеры.

1. Построить интерполяционный полином второго порядка, совпадающей с функцией $f(x) = 3^x$ ($x \in [-1, 1]$) в точках $x_0 = -1$, $x_1 = 0$, $x_2 = 1$.

Решение. Степень полинома равна числу узловых точек без одной. Поэтому

$$P_2(x) = a_0 + a_1 x + a_2 x^2.$$

Тогда: $a_0 - a_1 + a_2 = 3^{-1}$;

$a_0 = 3^0 = 1$;

$a_0 + a_1 + a_2 = 3^1 = 3$.

Отсюда $a_0 = 1$, $a_1 = 4/3$, $a_2 = 2/3$ и $P_2(x) = 1 + (4/3)\,x + (2/3)\,x^2$.

Задание: по тем же данным построить полином Лагранжа $L_2(x)$ и сравнить с $P_2(x)$.

2. С какой точностью можно вычислить $\sqrt{115}$ с помощью интерполяционного полинома Лагранжа для функции $y = \sqrt{x}$, выбрав узлы интерполирования $x_0 = 100, x_1 = 121, x_2 = 144$?

Решение. Имеем $y''' = \frac{3}{8} x^{-5/2}$. Отсюда

$$M_3 = max\left|y'''\right| = \frac{3}{8\sqrt{100^5}} = \frac{3}{8}\cdot 10^{-5} \text{ при } 100 \le x \le 144.$$ Поэтому

$$\left|y - L_n(x)\right| \le \frac{3}{8}\cdot 10^{-5}\,\frac{1}{3!}(115-100)(115-121)(115-144) \approx 1{,}6\cdot 10^{-3}.$$

3.1.3. Полиномиальная аппроксимация функций

Аппроксимация – замена одних математических объектов другими, в том или ином смысле близкими к исходным. Аппроксимация позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов, таких, характеристики которых легко вычисляются, а свойства которых известны.

Аппроксимация используется в вычислительной математике не только для приближения функций, но и для приближения дифференциальных операторов и других математических объектов.

Здесь мы рассмотрим только задачу приближения функции $f(x)$, свойства которой оговорены в параграфе 3.1.1.

На практике часто бывает, что заданная степень m полинома $P_m(x)$, с помощью которого необходимо описать таблицу значений $y_i = f(x_i)$, $i = 0, 1, \ldots n$, должна быть меньше n . Возникает задача о том, как оптимальным образом построить такой полином. Очевидно, что условие (3.3) в данном случае применять нельзя: степень полинома меньше числа узловых значений. В данной ситуации отказываются от условия точного совпадения приближающего полинома со значениями описываемой функции в узлах. В качестве критерия оптимальности построения полинома выбирают условие ***минимальности квадратичного отклонения*** между полиномом и функцией во всех узлах таблицы.

Пусть функция $f(x)$ задана таблично $n + 1$ значениями $y_i = f(x_i)$, $i = 0, 1,\ldots n$. Необходимо найти такой полином $P_m(x)$, причем $m < n$, приближающий функцию $f(x)$ во всех точках отрезка $[x_0, x_n]$ так, что величина суммарного квадратичного отклонения

$$S_m = \sum_{i=0}^{n}\left[P_m(x_i) - f(x_i)\right]^2 = \sum_{i=0}^{n}\left[\left(a_0 + a_1x_i + a_2x_i^2 + \ldots + a_mx_i^m\right) - y_i\right]^2 \qquad (3.7)$$

принимала бы наименьшее значение.

Рассмотрим S_m как функцию коэффициентов $a_0 \ldots a_m$ полинома. Эти коэффициенты нам пока неизвестны. Однако требование минимальности величины S_m приводит к условию:

$$\text{для всех } i = 0, 1 \ldots n \quad \partial S_m / \partial a_i = 0. \qquad (3.8)$$

Вычислив n+1 выражений из условия (3.7), получим:

$$\sum_{i=0}^{n}\left[\left(a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m\right) - y_i\right]\cdot 1 = 0;$$

$$\sum_{i=0}^{n}\left[\left(a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m\right) - y_i\right]\cdot x_i = 0; \qquad (3.9)$$

$$\ldots\ldots$$

$$\sum_{i=0}^{n}\left[\left(a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m\right) - y_i\right]\cdot x_i^m = 0.$$

Данная система уравнений служит для определения неизвестных коэффициентов a_0, a_1, ... a_m искомого полинома. Введем следующие обозначения:

$$q_k = x_0^k + x_1^k + x_2^k + \ \ldots\ + x_n^k; \qquad k = 0, 1, 2, \ldots m, \ldots 2m;$$
$$t_k = x_0^k y_0 + x_1^k y_1 + x_2^k y_2 + \ \ldots\ + x_n^k y_n; \qquad k = 0, 1, 2, \ldots m.$$

Теперь систему (3.8) можно преобразовать к следующему виду:

$$\begin{aligned}
&a_0 q_0 + a_1 q_1 + a_2 q_2 + \ldots + a_m q_m = t_0;\\
&a_0 q_1 + a_1 q_2 + a_2 q_3 + \ldots + a_m q_{m+1} = t_1;\\
&a_0 q_2 + a_1 q_3 + a_2 q_4 + \ldots + a_m q_{m+2} = t_2; \qquad (3.10)\\
&\qquad\qquad \ldots\ldots\\
&a_0 q_m + a_1 q_{m+1} + a_2 q_{m+2} + \ldots + a_m q_{2m} = t_m.
\end{aligned}$$

Понятно, что $q_0 = 1 + n$, $t_0 = y_0 + y_1 + y_2 + \ldots + y_n$.

Следует обратить внимание на то, что в системе (3.10) мы имеем $2m + 1$ значения q_k и $m + 1$ значение t_k. Система (3.10) будет иметь единственное решение, если среди точек x_0, x_1, x_2, ... x_n не будет совпадающих. Это решение и определяет полином $P_m(x) = a_0 + a_1 x + \ldots + a_m x^m$, приближающий функцию $f(x)$, заданную таблично, с наименьшим квадратичным отклонением. Такой полином называется *аппроксимирующим*, а изложенный здесь метод его построения называется **методом наименьших квадратов**.

Если $n = m$, то аппроксимирующий полином будет совпадать с интерполяционным полиномом Лагранжа для той же системы $\{x_i, f(x_i)\}$, $i = 0, 1, 2, \ldots n$.

Пример. Построить аппроксимирующий полином второй степени для данных, представленных в табл. 3.1.

Таблица 3.1

Данные	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
x	0,78	1,56	2,34	3,12	3,81
y	2,50	1,20	1,12	2,25	4,28

Решение. В нашем примере степень искомого полинома $m = 2$, а $n = 4$. Для того, чтобы разобраться в построении системы (3.9), запишем еще одну таблицу (табл. 3.2), в которую внесем все необходимые для построения системы значения с учетом 3-х цифр после запятой. Эта таблица содержит значения q_k и t_k, необходимые для построения определяющей системы.

Таблица 3.2

x^0	x^1	x^2	x^3	x^4	y	xy	$x^2 y$
1	0,78	0,608	0,475	0,370	2,50	1,95	1,520
1	1,56	2,434	3,796	5,922	1,20	1,872	2,921
1	2,34	5,476	12,813	29,982	1,12	2,621	6,133
1	3,12	9,734	30,371	94,759	2,25	7,02	21,902
1	3,81	14,516	55,306	210,717	4,28	16,307	62,128
5	11,61	32,768	102,761	341,750	11,35	29,770	94,604
q_0	q_1	q_2	q_3	q_4	t_0	t_1	t_2

Как следует из данных табл. 3.2, система для определения a_0, a_1 и a_2 будет содержать следующие три уравнения:

$$\begin{aligned} 5\,a_0 + \ \ 11{,}61\,a_1 + \ \ 32{,}768\,a_2 &= 11{,}35\,; \\ 11{,}61\,a_0 + \ 32{,}768\,a_1 + 102{,}761\,a_2 &= 29{,}77\,; \\ 32{,}768\,a_0 + 102{,}761\,a_1 + 341{,}750\,a_2 &= 94{,}604\,. \end{aligned}$$

Решая эту систему, получим $a_0 = 5{,}045$; $a_1 = -4{,}043$; $a_2 = 1{,}009$. Следовательно, искомый аппроксимирующий полином имеет вид:

$$P_2(x) = 5{,}045 - 4{,}043\,x + 1{,}009\,x^2\,.$$

Замечание. Введем в рассмотрение величину $\Omega = S_m / (n - m)$, где значение S_m определено в соответствии с (3.7). Тогда наилучшим приближающим полиномом будет полином такой степени m, для которого Ω минимально [4 – 6] – критерий Гаусса.

Метод наименьших квадратов можно использовать и в случае, если необходимо некоторую непрерывную функцию $f(x)$ представить в виде полинома $P_m(x)$ заданной степени m на определенном отрезке $[a, b]$ изменения аргумента. В этом случае за меру отклонения принимают величину

$$J_m = \int_a^b \left[P_m(x) - f(x)\right]^2 dx.$$

Величина J_m есть функция коэффициентов a_0, a_1, ... a_m аппроксимирующего полинома $P_m(x)$. Для минимизации этой величины, как и ранее, вычислим m+1 производную вида

$$\partial J_m / \partial a_i = 0, \qquad i = 0, 1, \ldots m .$$

Тогда получим следующие m+1 уравнения:

$$\begin{aligned}
&\int_a^b \left[P_m(x) - f(x)\right]^2 \cdot 1 \cdot dx = 0 ; \\
&\int_a^b \left[P_m(x) - f(x)\right]^2 \cdot x \cdot dx = 0 ; \\
&\qquad \bullet \quad \bullet \quad \bullet \\
&\int_a^b \left[P_m(x) - f(x)\right]^2 \cdot x^m \cdot dx = 0 .
\end{aligned} \tag{3.11}$$

Раскроем скобки под интегралами и введем обозначение

$$q_k = \int_a^b x^k \, dx = (b^{k+1} - a^{k+1})/(k+1), \quad k = 0, 1, \ldots m, \ldots 2m .$$

Тогда систему (3.11) можно представить в виде:

$$\begin{aligned}
&q_0 a_0 + q_1 a_1 + \ldots + q_m a_m = \int_a^b f(x)\, dx ; \\
&q_1 a_0 + q_2 a_1 + \ldots + q_{m+1} a_m = \int_a^b x f(x)\, dx ; \\
&\qquad \bullet \quad \bullet \quad \bullet \\
&q_m a_0 + q_{m+1} a_1 + \ldots + q_{2m} a_m = \int_a^b x^m f(x)\, dx .
\end{aligned} \tag{3.12}$$

Решение этой системы относительно a_0, a_1, ... a_m позволяет построить полином $P_m(x)$, который имеет с исходной функцией $f(x)$ минимальное квадратичное расхождение. Интегралы, стоящие в правой части системы (3.12), могут быть рассчитаны численно.

Метод наименьших квадратов широко используется в практике вычислений для обработки экспериментальных данных, выявления вида кривых, упрощения сложных функций и т.п. Этот метод используется не только для полиномиальной аппроксимации, но и для аппроксимации *произвольными функциями*

требуемого вида. Однако только в случае аппроксимации полиномами система определяющих уравнений (3.10) или (3.12) линейна и легко решаема.

Пример. Найти наилучшую аппроксимацию в виде полинома второй степени для функции $f(x) = x^{0,5}$ на отрезке [0, 1].

Решение. Очевидно, что в нашем примере $q_k = (k + 1)^{-1}$. Вид искомого полинома

$$P_2(x) = a_0 + a_1 x + a_2 x^2.$$

Вычислим интегралы, стоящие в правой части системы (3.11) в нашем примере.

$$\int_0^1 x^{0,5}\,dx = 2/3\,; \qquad \int_0^1 x \cdot x^{0,5}\,dx = 2/5\,; \qquad \int_0^1 x^2 \cdot x^{0,5}\,dx = 2/7\,.$$

Поэтому система уравнений, определяющая коэффициенты полинома, записывается следующим образом:

$$a_0 + (1/2)\,a_1 + (1/3)\,a_2 = 2/3;$$
$$(1/2)\,a_0 + (1/3)\,a_1 + (1/4)\,a_2 = 2/5;$$
$$(1/3)\,a_0 + (1/4)\,a_1 + (1/5)\,a_2 = 2/7.$$

Ее решение: $a_0 = 6/35$, $a_1 = 48/35$, $a_2 = -4/7$. Следовательно, искомый полином имеет вид:

$$P_2(x) = 6/35 + x \cdot 48/35 - x^2 \cdot 4/7.$$

Замечания.

1. Система (3.10), определяющая коэффициенты аппроксимирующего полинома, содержит в качестве множителей при неизвестных величины q_k , представляющие собой сумму k-тых степеней всех аргументов. Так как максимальное значение $k = 2m$, где m – степень выбранного аппроксимирующего полинома, то величины q_k могут различаться на порядки. Это может привести к большим погрешностям за счет ошибок округления при решении системы (3.10) на компьютере методом Гаусса и, следовательно, к неправильному виду полинома. Выход из положения заключается в использовании метода Гаусса с обязательным выбором ведущих элементов и, возможно, в построчной нормировке исходной системы на соответствующий ведущий элемент. В последнем случае решение системы (3.10), найденное на основе итерационного метода Гаусса – Зейделя, будет более точным, чем решение, найденное методом исключений. Очевидно, что данное замечание в полной мере относится и к системе (3.12).

2. Выше кратко изложены методы приближения функций алгебраическими полиномами. Однако очевидно, что не всякую функцию целесообразно приближать алгебраическим полиномом. В задачах приближения функций используется тригонометрическая интерполяция, приближение рациональными функциями, дробно-линейная интерполяция и др.

Например, если приближаемая функция $f(x)$ периодическая с периодом t, то приближение строится в виде тригонометрического полинома степени n:

$$T_n(x) = a_0 + \sum_{k=1}^{n} [a_k \cos(\pi k x / t) + b_k \sin(\pi k x / t)].$$

Коэффициенты этого полинома в случае задачи интерполяции определяются из системы уравнений, проистекающей из условий совпадения значений полинома и приближаемой функции в узловых точках:

$$T_n(x_j) = f(x_j), \quad j = 1, 2, ..., 2n+1,$$

где $x_0 < x_1 < .... < x_j < ... < x_{2n+1}$, причем $x_{2n+1} - x_0 = t$.

Приближение рациональными функциями производится следующим образом. Пусть функция $f(x)$ задана $n+1$ своим значением в точках $x_0, x_1, ... x_n$. Приближающая функция строится в виде:

$$R_{k,m}(x) = \frac{a_0 + a_1 x + a_2 x^2 + ... + a_k x^k}{x^m + b_{m-1} x^{m-1} + .. b_1 x + b_0}.$$

Потребуем, чтобы

$$R_{k,m}(x_j) = f(x_j), \quad j = 0, 1, ... n.$$

Последнее равенство представляет собой систему из $n+1$ уравнений, решение которой будет однозначным, если число неизвестных $a_0, a_1, ... a_k, b_0, b_1, ... b_m$ будет равняться числу уравнений, т.е. $k + m = n$. Тех, кто хочет более подробно ознакомится с задачами приближения функций, отсылаем к соответствующей литературе, например, монографиям [4, 9].

3.2. Применение цепных дробей

Для вычисления многих функций эффективным оказывается применение *цепных дробей*.

Пусть $\{a_k\}_{k=0}^{\infty}$ и $\{b_k\}_{k=0}^{\infty}$ – две последовательности вещественных, комплексных чисел, или функций одной или нескольких переменных. Тогда выражение вида [7]

$$a_0+\cfrac{b_1}{a_1+\cfrac{b_2}{a_2+\cfrac{b_3}{a_3+}\ldots}}=\left[a_0,\frac{b_1}{a_1},\frac{b_2}{a_2},\frac{b_3}{a_3},\ldots\right]$$

называется *цепной* или *непрерывной* дробью, отвечающей заданным последовательностям $\{a_k\}_{k=0}^{\infty}$ и $\{b_k\}_{k=0}^{\infty}$. Выражения

$$a_0+\frac{b_1}{a_1},\ a_0+\cfrac{b_1}{a_1+\cfrac{b_2}{a_2}},\ a_0+\cfrac{b_1}{a_1+\cfrac{b_2}{a_2+\cfrac{b_3}{a_3}}},\ldots$$

называются соответственно первой, второй, третьей и т. д. *подходящей* дробью для данной бесконечной дроби и обычно обозначаются

$$\frac{P_1}{Q_1},\ \frac{P_2}{Q_2},\ \frac{P_3}{Q_3},\ldots,\ \frac{P_n}{Q_n}.$$

Цепная дробь называется *сходящейся*, если существует конечный предел подходящей дроби P_n/Q_n при $n\to\infty$, причем этот предел и принимается за значение дроби. Для сходящейся цепной дроби величина P_n/Q_n является ее приближенным значением.

Разложение функций в непрерывные дроби осуществляется не единственным способом (в отличие от разложения в степенные ряды).

При работе на компьютере подходящие цепные дроби можно находить с помощью следующей последовательности операций:

$$c_1=\frac{b_n}{a_n},\qquad d_1=a_{n-1}+c_1,$$

$$c_2=\frac{b_{n-1}}{d_1},\qquad d_2=a_{n-2}+c_2,$$

.....

$$c_k=\frac{b_{n-k+1}}{d_{k-1}},\qquad d_k=a_{n-k}+c_k,$$

.....

$$c_n=\frac{b_1}{d_{n-1}},\qquad d_n=a_0+c_n=\frac{P_n}{Q_n}.$$

Такая последовательность действий легко программируется.

Другой способ вычисления подходящих дробей состоит в использовании следующих рекуррентных соотношений:

$$P_n=a_n\cdot P_{n-1}+b_nP_{n-2},\ \ Q_n=a_n\cdot Q_{n-1}+b_nQ_{n-2}\ \ (n=2,3,\ldots),$$

где $P_{-1}=1,\ Q_{-1}=0,\ P_0=a_0,\ Q_0=1.$

Преимущество этого метода состоит в том, что на каждом этапе полученное значение функции можно сравнить с предыдущим приближением и прекратить вычисления, если нужная точность уже достигнута. С другой стороны, поскольку a_n и b_n обычно больше единицы, P_n и Q_n возрастают очень быстро. Это не способствует уменьшению погрешности вычислений.

Рассмотрим конкретный пример. Оказывается, что для *arctg x* справедливо следующее разложение:

$$arctg(x) = \left[0, \frac{x}{1}, \frac{x^2}{3}, \frac{(2x)^2}{5}, \frac{(3x)^2}{7}, \ldots, \frac{((n-1)x)^2}{2n-1}, \ldots\right],$$

которое сходится во всех точках непрерывности этой функции. Положив x=1, получим:

$$arctg(1) = \left[0, \frac{1}{1}, \frac{1}{3}, \frac{2^2}{5}, \ldots, \frac{(n-1)^2}{2n-1}, \ldots\right].$$

На основании приведенных выше рекуррентных соотношений для членов подходящих дробей составим следующую схему вычислений данной величины, см. табл. 3.5:

Таблица 3.5

k	–1	0	1	2	3	4	5	6	7
b_k	-	-	1	1	4	9	16	25	36
a_k	-	0	1	3	5	7	9	11	13
P_k	1	0	1	3	19	160	1744	23184	364176
Q_k	0	1	1	4	24	204	2220	29520	463680
P_k/Q_k	-	-	1	0,75	0,7843137	0,7843137	0,7855856	0,7853659	0,7854037

Заметим, что табличное значение величины *arctg*(1) = 0,785398163397. Следовательно, уже шестая подходящая дробь обеспечивает четыре верные цифры.

3.3. Вычисление функций методом последовательных приближений

Пусть нам нужно вычислить значение непрерывной функции $y = f(x)$ для заданного значения аргумента x=ξ. Представим эту функцию *неявно*, то есть уравнением вида:

$$F(x, y) = 0. \qquad (3.13)$$

Такое преобразование можно сделать огромным количеством способов.

Часто решение уравнения (3.13) удается свести к однообразным повторяющимся операциям, которые легко программируются. Один из возможных спо-

собов построения такого *итерационного* процесса заключается в следующем [5, 6].

Предположим, что $F(x, y)$ непрерывна и имеет непрерывную частную производную $F_y'(x, y) \neq 0$.

Пусть y_n – приближенное значение y. Применив формулу конечных приращений[3] Лагранжа, получим:

$$F(x, y_n) = (y_n - y)\, F_y'(x, \bar{y}_n),$$

где $\bar{y}_n$ – некоторое промежуточное значение между y_n и y. Отсюда находим:

$$y = y_n - \frac{F(x, y_n)}{F_y'(x, \bar{y}_n)},$$

причем значение $\bar{y}_n$ нам не известно. Полагая приближенно $\bar{y}_n \cong y_n$, имеем следующую итерационную формулу для вычисления $y \cong y_n$:

$$y_{n+1} = y_n - \frac{F(x, y_n)}{F_y'(x, y_n)}, \quad (n = 0,1,2,\ldots) \tag{3.14}$$

Если первая и вторая производные функции $F(x,y)$ по переменной y существуют и сохраняют постоянные знаки в рассматриваемом интервале, содержащем корень уравнения (3.13), то итерационный процесс сходится к значению этого корня $y(x=\xi)$.

Начальное значение y_0, вообще говоря, произвольно и выбирается, по возможности, близким к искомому значению y. Процесс итерации по формулу (3.14) продолжается до тех пор, пока в пределах заданной погрешности ε два последовательных значения y_n и y_{n-1} не совпадут между собой: $|y_n - y_{n-1}| < \varepsilon$. Заметим, что при этом *не гарантируется*, что

$$| y(x=\xi) - y_{n-1} | < \varepsilon.$$

Рассмотрим в качестве иллюстрации применения инерционного метода один из способов вычисления квадратного корня.

Пусть $y = \sqrt{x}$ $(x > 0)$. Преобразуем это уравнение к виду:

$$F(x, y) \equiv y^2 - x = 0.$$

Применив общую итерационную формулу (3.14), получим:

$$y_{n+1} = \frac{1}{2}\left(y_n + \frac{x}{y_n} \right) \quad (n = 0,1,2,\ldots).$$

[3] Если $f(x)$ непрерывна и дифференцируема на $[a, b]$, то $f(b) - f(a) = f'(\xi)(b - a)$, $a < \xi < b$.

Эта формула называется ***формулой Герона***. Вычислим с ее помощью $\sqrt{7}$ с погрешностью не более 10^{-5}. За начальное приближение возьмем y_0=2. Тогда получим следующую последовательность чисел:

$$y_1 = \frac{1}{2}\left(2 + \frac{7}{2}\right) = \frac{11}{4} = 2{,}75000; \qquad y_2 = \frac{1}{2}\left(\frac{11}{4} + \frac{28}{11}\right) = \frac{233}{88} = 2{,}64772;$$

$$y_3 = \frac{1}{2}\left(\frac{233}{88} + \frac{616}{233}\right) = 2{,}64575; \qquad y_4 = \frac{1}{2}\left(2{,}64575 + \frac{7}{2{,}64575}\right) = 2{,}64575.$$

Заметив, что в значениях y_3 и y_4 совпадают пять цифр после запятой, приближенно полагаем, что $\sqrt{7} \approx 2{,}64575..$

3.4. Кубическая сплайн-интерполяция

Термин «сплайн» (spline) буквально означает гибкую линейку, которую можно использовать для построения на бумаге графика функции по известным узловым значениям. Эту линейку ставят на ребро и, придерживая ее в некоторых местах, добиваются того, что ребро проходит сразу через много точек.

Уравнение свободного равновесия такой линейки можно описать некоторой функцией $S(x)$, четвертая производная которой равна нулю: $S'''(x)=0$. Значит, в промежутках между каждой парой соседних узлов интерполяционная функция является кубическим полиномом.

Пусть, как и ранее, задана таблица узловых значений некоторой функции, подлежащей кусочно-полиномиальной интерполяции:

$$y_0 = f(x_0), y_1 = f(x_1), \ldots y_k = f(x_i), \ldots y_n = f(x_n) \qquad (i = 0,1..n)$$

Кубическим сплайном, соответствующим заданной функции $f(x)$ и ее узловым значениям, называется функция $S(x)$, удовлетворяющая следующим условиям [5]:

1) На каждом сегменте $[x_{i-1}, x_i]$ области задания аргумента функция $S(x)$ является полиномом третьей степени.
2) Функция $S(x)$, а также ее первая и вторая производные непрерывны на $[x_0, x_n]$.
3) $S(x_i) = f(x_i)$, i=0,1, … n.

Таким образом, в промежутке между каждой парой соседних узлов интерполяционная функция записывается в виде:

$$S_i(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3,$$
$$x_{i-1} \le x \le x_i, \quad i = 1,2,..n. \tag{3.15}$$

Коэффициенты полинома на каждом интервале определяются из условий в узлах и из условий непрерывности функции $S(x)$ и ее производных.

В узлах полином должен принимать табличные значения приближаемой функции:

$$S_{i-1}(x) = y_{i-1} = a_i, \qquad i = 1,2,..n;$$
$$S_i(x) = y_i = a_i + b_i \cdot h_i + c_i \cdot h_i^2 + d_i \cdot h_i^3, \text{ где } h_i = x_i - x_{i-1}. \tag{3.16}$$

Видно, число уравнений (3.16) вдвое меньше числа неизвестных коэффициентов. Дополним эту систему условиями на производные функции $S(x)$. Вычислим их:

$$S_i'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2,$$
$$S_i''(x) = 2c_i + 6d_i(x - x_{i-1}),$$
$$x_{i-1} \le x \le x_i, \quad i = 1,2,..n.$$

Из условия непрерывности этих производных в узлах получим следующие соотношения:

$$b_{i+1} = b_i + 2c_i \cdot h_i + 3d_i \cdot h_i^2; \quad 1 \le i \le n-1;$$
$$c_{i+1} = c_i + 3d_i \cdot h_i; \quad 1 \le i \le n-1. \tag{3.17}$$

Осталось добавить всего 2 условия, чтобы система уравнений для определения коэффициентов была полностью определена. Предположим, что концы линейки свободные. Тогда (в соответствии с теорией сопротивления материалов)

$$\frac{1}{2}S_1''(x_0) = c_1 = 0;$$
$$\frac{1}{2}S_n''(x_n) = c_n + 3d_n h_n = 0. \tag{3.18}$$

Уравнения (3.16) – (3.18) образуют систему линейных алгебраических уравнений для определения $4n$ неизвестных коэффициентов, согласованных условиями непрерывности в узловых точках.

Обычно систему (3.16) – (3.18) переписывают в более удобном виде, так как ее часть (3.17) – (3.18) от (3.16) не зависит. Из (3.17) и (3.18) следует, что

$$c_1 = 0; \; d_i = \frac{c_{i+1} - c_i}{3h_i}; \; d_n = -\frac{c_n}{3h_n}. \tag{3.19}$$

Подставляя (3.19) в (3.16), получим с учетом $y_{i-1} = a_i$:

$$b_i = \frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i); \quad i \le i \ge n-1,$$
$$b_n = \frac{y_n - y_{n-1}}{h_n} - \frac{h_n c_n}{3} \ . \tag{3.20}$$

Теперь с помощью (3.20) исключим из (3.17) все b_i. Тогда остается система линейных уравнений для c_i:

$$\begin{gathered} c_1 = 0; \\ h_{i-1} \cdot c_{i-1} + 2 \cdot (h_{i-1} + h_i) \cdot c_i + h_i \cdot c_{i+1} = \\ = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}} \right); \ i = 2,3,..n; \\ c_n = 0. \end{gathered} \tag{3.21}$$

Система уравнений (3.21) относится к трехдиагональным системам. В них ненулевыми элементами являются элементы главной диагонали матрицы коэффициентов и элементы двух диагоналей, смежных с главной. Определив из нее все c_i, нетрудно определить все b_i (3.20), все d_i (3.19) и, наконец, все a_i (3.16). Обычно систему (3.21) решают методом прогонки, см. параграф 2.4 настоящего пособия.

Доказано (например, [5]), что интерполирование кубическими сплайнами является сходящимся процессом, то есть при неограниченном увеличении числа узлов n соответствующая последовательность сплайн-функций сходится к приближаемой функции $f(x)$.

Методам сплайн-функций и сплайн-интерполяций посвящено большое количество литературных источников, в частности, монография [10].

3.5 Замечания по вычислению значений специальных функций

Решение многих научных и технических проблем связано с использованием так называемых *специальных функций*.

Под специальными функциями математической физики обычно понимают класс функций, появляющихся при решении дифференциальных уравнений в частных производных и не представляющих собой конечную линейную комбинацию «обычных» элементарных функций.

Специальные функции определяются с помощью степенных рядов, бесконечных произведений, интегральных представлений, тригонометрических рядов, рядов по системам ортогональных функций и др.

Для специальных функций имеются справочники, содержащие таблицы значений, таблицы интегралов и рядов, представляющих их. Также для многих специальных функций существуют их асимптотические представления, значительно облегчающие получение числовых значений. Особенно это важно для функций, определяемых с помощью степенных рядов при больших по модулю значениях аргумента.

Наиболее полным справочником по специальным функциям является справочник [8]. В нем представлены как подробные таблицы значений многих специальных функций, так и их приближенные выражения.

Типичными примерами функций, определяемых степенными рядами, являются функции Бесселя 1-го рода нулевого и первого порядков $J_0(x)$ и $J_1(x)$:

$$J_0(x)=\sum_{k=0}^{\infty}(-1)^k\frac{\left(\frac{x}{2}\right)^{2k}}{(k!)^2},\ J_1(x)=\sum_{k=0}^{\infty}(-1)^k\frac{\left(\frac{x}{2}\right)^{2k+1}}{k!(k+1)!}. \tag{3.22}$$

Примерами функций, определяемыми интегральными представлениями, являются:

– интегральная показательная функция

$$E_1(x)=\int_x^{\infty}\frac{exp(-t)}{t}dt; \tag{3.23}$$

– интегральный синус

$$Si(x)=\int_0^{x}\frac{sin(t)}{t}dt; \tag{3.24}$$

– интегралы вероятностей

$$erf(x)=\frac{2}{\sqrt{\pi}}\int_0^{x}exp(-t^2)\cdot dt=\frac{2}{\sqrt{\pi}}\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{2n+1}}{n!(2n+1)};$$
$$erfc(x)=\frac{2}{\sqrt{\pi}}\int_x^{\infty}exp(-t^2)\cdot dt=1-erf(x), \tag{3.25}$$

и другие.

В частности, случайная величина Y имеет нормальное распределение со средним значением $m=\langle Y\rangle$ и дисперсией σ^2, если вероятность P события $\{Y\le y\}$ определяется формулой:

$$P(Y \le y) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{y} exp\left(-\frac{(t-m)^2}{2\sigma^2} \right) dt.$$

Отметим, что многие специальные функции могут быть представлены как интегрально, так и в виде сходящегося ряда, например, интеграл вероятностей (3.25):

$$erf(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n+1}}{n!(2n+1)} . \qquad (3.26)$$

При наличии подробной таблицы данных какой-либо специальной функции вычисление промежуточных значений на узких диапазонах изменения аргумента можно производить с помощью полиномиальной интерполяции (параболической, кубической, но обычно не выше пятой степени).

Если специальные функции определяются интегральными представлениями, в отсутствии таблиц данных их можно достаточно точно вычислить с помощью методов численного интегрирования (с учетом возможных особенностей внутри отрезка интегрирования)[4].

При вычислении функций, определяемых бесконечными степенными рядами, могут возникнуть проблемы, рассмотренные в разделе 1. Действительно, вычисления по формулам (3.22) при $|x| \ge 8$ не дадут правильного результата, поскольку функция x^{2k} сначала растет с увеличением k значительно быстрее функции $(k!)^2$, и набранные в разрядной сетке компьютера начальные значения суммы ряда не смогут уточняться дальнейшими малыми членами суммы, обусловленными слагаемыми с $(k!)^{-2}$, из-за их отбрасывания. Именно поэтому в справочниках приводятся формулы асимптотических приближений таких функций для больших значений аргумента. Кроме того, например, в [8], представлены аппроксимационные формулы, приближающие функции Бесселя многочленами, причем абсолютная погрешность таких приближений не превосходит $5 \cdot 10^{-8}$.

Подпрограммы-функции, вычисляющие функции Бесселя 1-го рода нулевого и первого порядков $J_0(x)$ и $J_1(x)$, приведены в Приложении В.

[4] Некоторые методы численного интегрирования будут рассмотрены в разделе 5 настоящего пособия.

***Задания к разделу* 3.**

1. Для функции *six*(*x*) при $x \in [0,\pi]$. С помощью полиномов Лагранжа и Ньютона исследовать абсолютную погрешность ее интерполирования в промежуточных точках в зависимости от числа $n \geq 2$ – степени полинома. Указание: узловые точки выбирать с условием постоянства шага *h*: x_0=0, x_n=π, *h*=π/*n*.

2. Показать, что если с помощью метода наименьших квадратов искать уравнение кривой в виде $y = ax + b$, причем исходная информация ограничена двумя точками (x_1, y_1) и (x_2, y_2), то аппроксимационная прямая совпадет с интерполяционной прямой.

3. Рассмотрим следующие исходные данные:

Таблица 3.6

i	**0**	**1**	**2**	**3**	**4**
x	0	1	2	3	4
y	0	20	40	50	70

Построить для этих данных аппроксимирующие линейную и квадратичную функции, нарисовать их графики и сравнить по критерию Гаусса. Также построить для этих данных полином Лагранжа и сравнить поведение интерполирующей и аппроксимирующих зависимостей в интервалах между узловыми точками.

4. Дана функция $f(x) = \dfrac{a + bx + cx^2}{1 + qx}$. Найти соответствующую цепную дробь вида:

$$f(x) = k_1 + \cfrac{x}{k_2 + \cfrac{x}{k_3 + x/k_4}}.$$

5. Составить компьютерную программу определения коэффициентов первого интерполяционного полинома Ньютона для *n* = 5, считая, что значения массивов *x*[0, 5] и *y*[0, 5] заданы – либо рассчитываются, либо вводятся посредством оператора ввода данных.

4. ЧИСЛЕННОЕ РЕШЕНИЕ УРАВНЕНИЙ

Нахождение корней уравнения – важная вычислительная задача. Многие физические задачи приводят к необходимости решения уравнений вида $f(x)=0$, в которых функция $f(x)$ не является линейной – она может быть алгебраической или трансцендентной, но мы предполагаем, что эта функция гладкая (дифференцируемая).

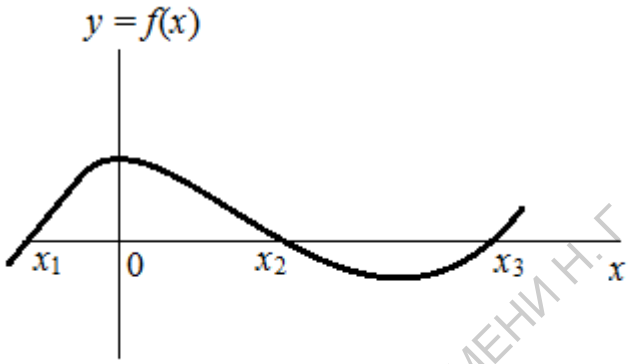


Рис 4.1. Геометрическая интерпретация корней уравнения $f(x) = 0$

Если $f(x)$ выражается через тригонометрические, логарифмические, показательные, специальные и прочие трансцендентные функции, то и уравнение принято называть *трансцендентным*. В общем же случае уравнение $f(x)=0$ называют *нелинейным,* при этом аналитические формулы, определяющие его корни, отсутствуют.

Рассмотрим конкретный пример задачи, опирающейся на трансцендентное уравнение. Оказывается, что длина водяных волн λ на мелководье глубиной h связана с длиной L тех же волн на очень глубокой воде следующим соотношением

$$\lambda - L \cdot th\left(\frac{2\pi h}{\lambda}\right) = 0. \qquad (4.1)$$

Пользуясь этим уравнением, можно по заданными значениям параметров h и L вычислить длину волны λ на мелководье.

Наиболее часто решение трансцендентных уравнений требуется в физических задачах, основанных на дифференциальных уравнениях в частных производных. Решения таких задач обычно выражаются через бесконечные рады типа рядов Фурье, Бесселя и подобных им. Собственные числа в них и определяются соответствующими трансцендентными уравнениями.

Задача решения нелинейного уравнения сводится к нахождению всех точек x_i пересечения графика функции $f(x)$ с осью x (см. рис. 4.1.). Из рисунка

видно, что точек пересечения графика функции с осью x может быть несколько. Поэтому в качестве первого шага при решении любого нелинейного уравнения проводят ***отделение*** его корней. Это означает, что ось x разбивают на такие отрезки, что в каждом из них содержится только один корень уравнения. После этого задача сводится лишь к уточнению положения каждого из корней в пределах допустимой погрешности.

Процесс отделения корней основан на следующей теореме: если непрерывная функция $f(x)$ принимает значения разных знаков на концах отрезка $[a, b]$, то есть $f(a)\cdot f(b) < 0$, то внутри этого отрезка содержится, по меньшей мере, один корень уравнения $f(x) = 0$. На основе этой теоремы реализуются самые простые и надежные методы численного определения корней уравнений: метод половинного деления и метод хорд [4, 6, 11].

Процесс отделения корней начинается с установления знаков $f(x)$ в граничных точках интересующего нас отрезка определения переменной x: $x = a$ и $x = b$. Затем определяются знаки $f(x)$ в ряде промежуточных точек $x = a_1, a_2 \ldots$, выбор которых должен учитывать особенности функции $f(x)$. Если окажется, что $f(a_i)\cdot f(a_{i+1}) < 0$, то в интервале (a_i, a_{i+1}) есть корень уравнения $f(x) = 0$. Необходимо выяснить, *является ли этот корень* ***единственным*** на данном интервале.

Для отделения корней практически достаточно провести процесс ***половинного деления***, последовательно деля исходный отрезок $[a, b]$ на 2, 4, 8 и т. д. равных частей и определяя знаки $f(x)$ в точках деления. Полезно помнить, что алгебраическое уравнение степени n: $a_0 + a_1x + a_2x^2 + \ldots + a_nx^n = 0$, – имеет не более n действительных корней. Поэтому если для такой системы мы получили n+1 перемену знака $f(x)$, то все его корни отделены.

Пример. Отделить корни уравнения $x^3 - 6x + 2 = 0$. Имеем $f(x) = x^3 - 6x + 2$. Составим приблизительную таблицу перемены знака $f(x)$:

Таблица 4.1

x	$-\infty$	-3	-1	0	1	3	$+\infty$
знак $f(x)$	–	–	+	+	–	+	+

Из данных, приведенных в таблице, следует, что в интервалах $(-3, -1)$; $(0, 1)$ и $(1, 3)$ происходит перемена знака $f(x)$. Так как $f(x)$ – полином 3-й степени, то все корни в заданном уравнении отделены.

В дальнейшем предполагается, что все корни в решаемом уравнении отделены.

4.1. Метод половинного деления

Пусть на отрезке $[a, b]$ находится ровно один корень уравнения $f(x)=0$, см. рис. 4.2. Знак $f(a)$ противоположен знаку $f(b)$. Необходимо уточнить положение корня с погрешностью ε.

Метод половинного деления заключается в следующем. Определим середину отрезка $[a, b]$: $c = (a+b)/2$, и вычислим функцию $f(c)$. Далее делаем выбор, какую из двух частей отрезка взять для дальнейшего уточнения корня. Очевидно, что корень будет находиться в той половине отрезка, на концах которой функция имеет разные знаки. На рис. 4.2 таким будет отрезок $[a, c]$. Для очередного шага уточнения корня отрезок $[c, b]$ из рассмотрения исключаем, а с отрезком $[a, c]$ продолжаем процесс деления так же, как и с первоначальным отрезком $[a, b]$. При этом сделаем переобозначения: $a_1 = a$, $b_1 = c$.

В итоге получается последовательность вложенных друг в друга отрезков все уменьшающейся длины: $[a_1, b_1]$, $[a_2, b_2]$ ... $[a_n, b_n]$. Этот повторяющийся (итерационный) процесс необходимо продолжать до тех пор, пока длина отрезка $[a_n, b_n]$ не станет меньше заданной погрешности ε вычислений. Тогда искомый корень $\xi \approx a_n \approx b_n \approx c = (a_n + b_n)/2$.

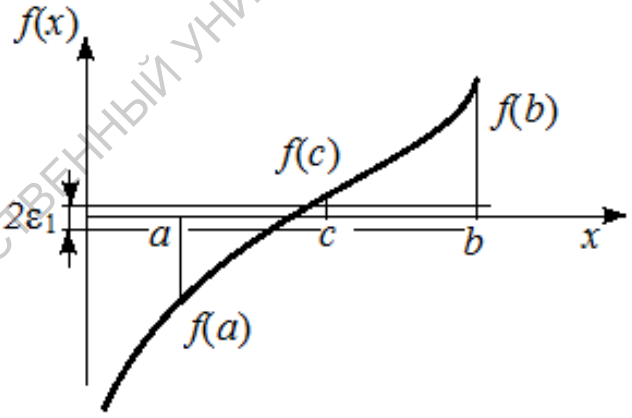


Рис. 4.2. Иллюстрация метода половинного деления

Следует учитывать, что функция $f(x)$ вычисляется с некоторой абсолютной погрешностью ε_1. Вблизи корня значения функции $f(x)$ малы по абсолютной величине и могут оказаться сравнимыми с погрешностью ее вычисления. Другими словами, при подходе к корню мы можем попасть в "полосу шумов" $2\varepsilon_1$ (рис. 4.2), и дальнейшее уточнение корня становится бессмысленным. Поэтому надо задать ширину полосы шумов и прекратить итерационный процесс при попадании в нее. Также необходимо иметь в виду, что при уменьшении длины

интервала $[a_n, b_n]$ увеличивается погрешность вычисления его длины $b_n - a_n$ за счет вычитания двух близких чисел.

Метод половинного деления (другие названия: метод бисекции, метод дихотомии) обладает довольно большой скоростью сходимости. Так как за каждую итерацию интервал, где расположен корень, уменьшается в два раза, то через n итераций интервал будет $(b - a)/2^n$. За 10 итераций интервал уменьшится в 2^{10}=1024 раза, за 20 итераций – в $2^{20} \approx 10^6$ раз.

Метод половинного деления легко программируется. Например, можно предложить следующую подпрограмму вычисления корня уравнения $f(x)=0$, при условии, что на отрезке $[a, b]$ находится только один корень:

Метки Позиции 1 – 5	6	Операторы. Позиции 7 - 72
		Subroutine bisection(a,b,c,eps,eps1)
C		a, b – исходные границы отрезка, a < b
C		c – текущая середина отрезка и корень
C		func(x) – функция, определяющая уравнение; задается головной программой
C		eps – погрешность вычисления корня
C		eps1 – погрешность вычисления функции
C		ВНИМАНИЕ! Стандартную функцию SIGN (знак) не используем
C		
		real*8 a,b,c,eps,eps1,y,fa,fy
		integer na,ny
		na=1
		fa=func(a)
1		if(fa.lt.0.) na= –1
		c=(a+b)*0.5
		y=func(c)
		if(abs(y).lt.eps1) return
		ny=1
		if (y.lt.0.) ny= –1
		if(ny.eq.na) then
		a=c
		else
		b=c
		end if
		if (abs(b–a).gt.eps) goto 1
		return
		end

Функция $f(x)$ здесь обозначена как func(x) и должна быть описана в головной программе.

4.2. Метод хорд

В предположениях предыдущего параграфа укажем более быстрый способ нахождения корня ξ уравнения $f(x) = 0$. В этом методе очередное приближение к корню берется не в середине отрезка $[a, b]$, а в точке x_1, где прямая, соединяющая точки $f(a)$ и $f(b)$, пересекает ось абсцисс, см. рис. 4.3. Эту прямую часто называют хордой – отсюда и название метода.

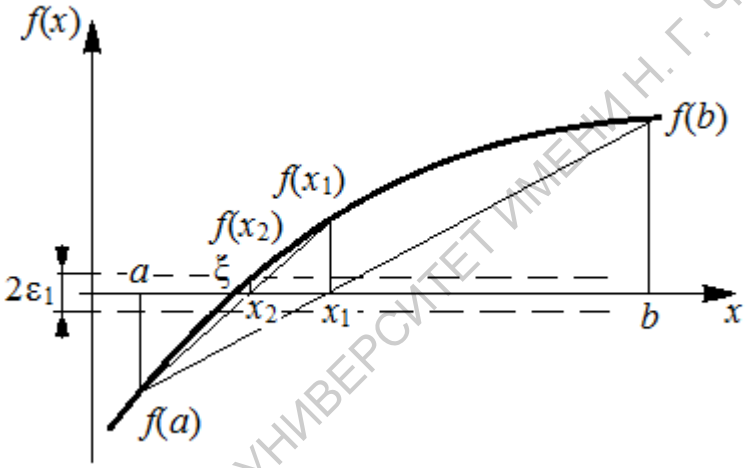


Рис. 4.3. Иллюстрация метода хорд

В качестве нового интервала для продолжения итерационного процесса выбираем ту из двух частей – $[a,x_1]$ или $[x_1,b]$ – отрезка $[a, b]$, на концах которого функция $f(x)$ меняет знак.

Процесс уточнения корня заканчивается, когда расстояние между очередными приближениями станет меньше требуемой погрешности ε вычислений:

$$|x_n - x_{n-1}| < \varepsilon,$$

или когда значения функции попадут в область шума, то есть

$$|f(x)| < \varepsilon_1.$$

Понятно, что уравнение хорды имеет вид (см. параграф 3.1.1):

$$f(x) = \frac{f(b) - f(a)}{b - a} \cdot (a - x) + f(a),$$

Тогда, поскольку x_1 – точка пересечения хордой оси $0x$ и, следовательно, $f(x_1) = 0$, для значения x_1 получим:

$$x_1 = a - \frac{b-a}{f(b)-f(a)} \cdot f(a). \qquad (4.2)$$

Схема вычислений при решении трансцендентного уравнения методом хорд почти полностью совпадает со схемой вычислений по методу бисекции. Различие заключается только в выборе точки деления отрезка, содержащего корень. Поэтому подпрограмма, реализующая решение трансцендентного уравнения методом хорд может быть, например, такой:

Метки Позиции 1 – 5	6	Операторы. Позиции 7 – 72
		Subroutine horda(a,b,c,eps,eps1)
C		a, b – исходные границы отрезка, a < b
C		c – точка деления отрезка и корень
C		func(x) – функция, определяющая уравнение; задается головной программой
C		eps – погрешность вычисления корня
C		eps1 – погрешность вычисления функции
C		ВНИМАНИЕ! Стандартную функцию SIGN (знак) не используем
C		
		real*8 a,b,c,eps,eps1,y,fa,fy
		integer na,ny
1		na=1
		fa=func(a)
		fb=func(b)
		if(fa.lt.0.) na= –1
		c=a–(b–a)/(fb–fa)*fa
		y=func(c)
		if(abs(y).lt.eps1) return
		ny=1
		if (y.lt.0.) ny= –1
		if(ny.eq.na) then
		a=c
		else
		b=c
		end if
		if (abs(b–a).gt.eps) goto 1
		return
		end

Как видно, данный пример подпрограммы определения корня почти полностью аналогичен предыдущему. Подпрограмма-функция func(x) здесь имеет тот же смысл, что и в методе бисекции.

Преимущество метод хорд над методом дихотомии имеет в том, что он требует меньшего (обычно вдвое – втрое) числа итераций для отыскания корня с той же погрешностью.

4.3. Метод касательных (метод Ньютона)

Рассмотрим еще один метод нахождения корней уравнений, предложенный Ньютоном [4, 6, 11]. Его графическая интерпретация приведена на рис. 4.4.

Пусть ξ есть корень уравнения $f(x) = 0$, единственный на отрезке [*a*, *b*] – области определения аргумента. Предположим, что каким-либо способом, например, графически, определено начальное приближение x_0 к корню. В этой точке вычисляем значения функции – левой части исходного уравнения и ее производной, то есть $f(x_0)$ и $f'(x_0)$. Значение производной численно равно *tg*(α), см. рис. 4.4. Следующее приближение к корню находим в точке x_1, где касательная к графику функции $f(x)$, проведенная из точки (x_0, $f(x_0)$), пересекает ось 0*x* (поэтому метод Ньютона и называют методом касательных). Эта точка x_1 принимается за новое начальное приближение, и процесс повторяется. Из рисунка видно, что процесс сходится к корню ξ. Процесс уточнения корня закончится, когда выполнится условие

$$|x_{i+1} - x_i| < \varepsilon, \quad i = 0,1,2,\ldots$$

где ε – допустимая погрешность определения корня.

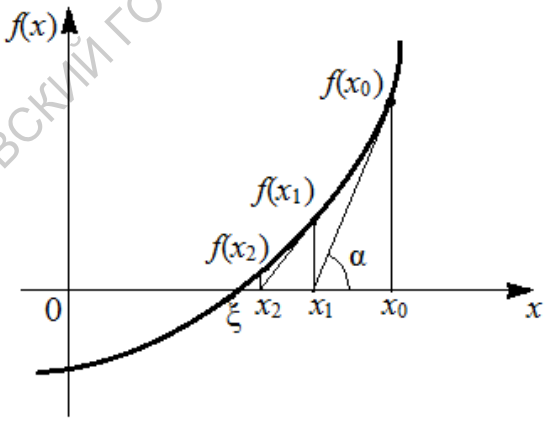


Рис. 4.4. Иллюстрация метода касательных (Ньютона)

Поскольку $f'(x_0) = tg(\alpha) \approx f(x_0)/(x_0 - x_1)$, расчетную формулу для метода Ньютона можно представить в виде:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \qquad (4.3)$$

Метод Ньютона обладает высокой скоростью сходимости. Обычно абсолютная погрешность решения 10^{-5} … 10^{-6} достигается за 5 – 6 итераций. Существенным недостатком этого метода является необходимость вычислять производную функции на каждом шаге итерационного процесса. Это может явиться серьезным препятствием, когда функция вычисляется по очень сложному или трудоемкому алгоритму.

Несколько уменьшив скорость сходимости, можно ограничиться вычислением производной только на первой итерации, а затем вести вычисления по формуле:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_0)}.$$

Структура программы решения уравнений методом Ньютона во многом напоминает структуру программ для методов дихотомии и хорд. Главное отличие в том, что здесь не нужно делать выбор между левой и правой частями отрезка, на которые его делит касательная. Кроме того, нет необходимости задавать полосу шума функции, так как по разности двух последовательных приближений $x_{i+1} - x_i$ можно сразу оценивать и величину отношения $f(x)/f'(x)$. Для предотвращения возможного "зацикливания" программы в случае неудачного выбора начального приближения или неправильно заданных параметров, рекомендуется предусмотреть счетчик числа итераций.

Замечание. Отметим без доказательства две особенности метода Ньютона [4, 5].

- Метод Ньютона имеет квадратичную сходимость, т.е. в отличие от метода половинного деления (и метода хорд) его погрешность на следующей итерации пропорциональна *квадрату* погрешности на предыдущей итерации. У метода дихотомии такая зависимость линейна.
- Быстрая сходимость метода Ньютона гарантируется лишь при ***близких*** к точному решению начальных приближениях, Если начальное приближение выбрано неудачно, то метод может сходиться медленно, либо не сойдется вообще.

4.4. Метод секущих

Этот метод получается из метода Ньютона заменой производной $f'(x_i)$ ее приближенным значением – разделенной разностью $[f(x_i) - f(x_{i-1})] / (x_i - x_{i-1})$, вычисленной по известным значениям x_i и x_{i-1}. Графической иллюстрацией этой замены является замена касательной к графику функции $f'(x_i)$ секущей, которая соединяет два достаточно близких соседних приближения к корню уравнения, см. рис. 4.5. Для того, чтобы начать итерационный процесс, необходимо знать ***два*** начальных приближения к корню: x_0 и x_1. В методе секущих применяются следующие расчетные соотношения [4, 11], вытекающие из указанной выше замены:

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

или, что эквивалентно,

$$r_0 = x_1 - x_0;\ x_{i+1} = x_i + r_i;$$
$$r_{i+1} = \frac{r_i}{f(x_{i-1}) - f(x_i)} \cdot f(x_i). \qquad (4.4)$$

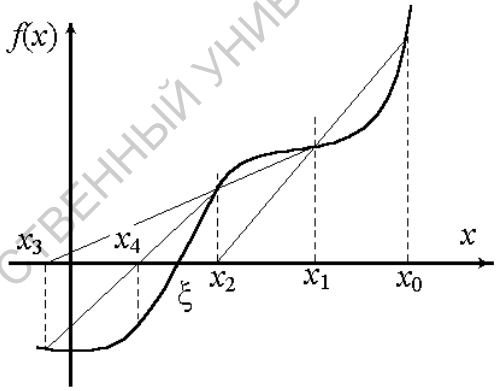


Рис. 4.5. Геометрическая иллюстрация метода секущих

Метод секущих несколько уступает методу Ньютона, однако не требует вычисления производных. От метода хорд этот метод отличается тем, что начальные приближения могут располагаться не только с разных сторон от корня, но и с одной стороны. Кроме того, при уточнении корня не проверяются знаки функции $f(x)$.

Вот один из возможных вариантов подпрограммы для реализации метода секущих:

Метки Позиции 1 – 5	6	Операторы. Позиции 7 – 72
		Subroutine secant(x0,x1,eps,ksi)
C		x0, x1 – начальные приближения к корню
C		ksi – найденное значение корня
C		func(x) – функция, определяющая уравнение; задается головной программой
C		eps – погрешность вычисления корня
		real*8 x0,x1,eps,ksi,z,f0,f1
		i=0
		z=x1–x0
		f0=func(x0)
1		i=i+1
		f1=func(x1)
		z=z/(f0–f1)*f1
		f0=f1
		x1=x1+z
		if(abs(z).gt.eps) goto 1
		if(i.eq.5000) goto 2
		ksi=x1
		return
		ksi=–777777.
		print 10
10		format(‘ корень не найден за 5000 итераций’)
		return
		end

Здесь используется тот же вариант задания функции func(x), что и ранее.

Еще отметим, что в данном примере введен счетчик итераций: если за 5000 итераций значение корня не найдено, выполнение подпрограммы прекращается, а значению корню присваивается символическое число ksi= –777777.0, см. Замечание к предыдущему параграфу.

4.5. Метод простых итераций (последовательных приближений)

От исходного нелинейного уравнения $f(x)=0$ тождественными алгебраическими преобразованиями перейдем к эквивалентной записи в виде $x=\varphi(x)$.

Выберем какое-то начальное приближение x_0 к корню ξ рассматриваемого уравнения при условии, что на отрезке определения $[a,b]$ этот корень единственный. Подставив его в правую часть последнего уравнения получим первое

приближение $x_1=\varphi(x_0)$, затем, действуя аналогично, получаем втрое приближение $x_2=\varphi(x_1)$, и так далее: $x_{i+1}=\varphi(x_i)$.

Возникает вопрос, при каких условиях данный итерационный процесс будет сходиться к корню ξ уравнения $x=\varphi(x)$, то есть когда $lim(x_i)_{i\to\infty}=\xi$?

Без доказательства приведем ответ на поставленный вопрос [5, 12].

Пусть $x\in[a,b]$ и имеется только один корень ξ исходного уравнения, принадлежащий области $[a,b]$.

Тогда если:

1) функция $\varphi(x)$ определена и непрерывно дифференцируема на $[a,b]$, (4.5,а)
2) все приближения $x_0, x_1, x_2, \ldots x_i, \ldots$ принадлежат отрезку $[a,b]$, (4.5,б)
3) на $[a,b]$ выполнено неравенство

$$max\left|\frac{d\varphi}{dx}\right|_{a\le x\le b}=M<1, \qquad (4.5,в)$$

то процесс последовательных приближений отыскания корня сходится к решению $lim(x_i)_{i\to\infty}=\xi$.

Оценка погрешности приближения на итерации с номером i определяется неравенством:

$$|\xi-x_i|\le\frac{M}{1-M}|x_i-x_{i-1}|.$$

Сходимость метода итераций считается хорошей, если $M<1/2$, при этом $M/(1-M)<1$. Поэтому, если в двух последовательных приближениях совпадают, например, три десятичных знака после запятой, то ошибка последнего приближения не превосходит 0,001.

Выполнение условия сходимости можно обеспечить путем рационального выбора вида функции $\varphi(x)$. Рассмотрим один из общих алгоритмов такого выбора.

Умножим левую и правую части уравнения $f(x)=0$ на произвольную постоянную C и потом добавим к обеим частям неизвестное x. Получим:

$$x + C\cdot f(x) = x.$$

При этом, очевидно, корни исходного уравнения не изменятся.

Введем обозначение

$$\varphi(x)=x + C\cdot f(x). \qquad (4.6)$$

Выбором константы C можно обеспечить выполнение условия сходимости. Следовательно, необходимо выбрать C так, чтобы $-1<\varphi'(x)<1$. При этом, чем меньше $|\varphi'(x)|$, тем лучше сходимость итерационного процесса.

Если функция $\varphi(x)$ выбрана в виде (4.6), то ее производная выражается формулой

$$\varphi'(x) = 1 + C f'(x).$$

Наибольшую скорость сходимости получим при $\varphi'(x) = 0$, тогда

$$C = -1/f'(x), \quad \varphi(x) = x - f(x)/f'(x), \quad x_{i+1} = x_i - f(x_i)/f'(x_i),$$

и формула метода простых итераций переходит в формулу Ньютона (4.3).

Пример. Уравнение

$$f(x) = x^3 - x - 1 = 0 \tag{4.7}$$

имеет в интервале $1 < x < 2$ один корень ξ, так как $f(1) = -1 < 0$ и $f(2) = 5 > 0$. Это уравнение можно представить в виде $x = x^3 - 1$. Здесь $\varphi(x) = x^3 - 1$, $\varphi'(x) = 3x^2$. Поэтому при $1 \le x \le 2$ $3 \le M \le 6$, где $\varphi'(x) = M$, и, следовательно, условие (4.5) сходимости процесса итераций не выполнено.

Но если записать уравнение (4.7) в виде

$$x = \sqrt[3]{x+1}, \tag{4.8}$$

то будем иметь: $\varphi(x) = \sqrt[3]{x+1}$, $\varphi'(x) = 1/\left(3 \cdot \sqrt[3]{(x+1)^2}\right)$. Отсюда при $1 \le x \le 2$ $0{,}16 < M < 0{,}21$, значит процесс итераций для уравнения (4.7) быстро сойдется. Определите сами, за сколько итераций можно получить решение уравнения (4.7) с погрешностью не хуже 10^{-3}.

Замечание. Метод простых итераций можно применять и для решения системы нелинейных уравнений. Например [12], пусть задана система нелинейных уравнений:

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

Необходимо найти действительное решение данной системы ξ_x и ξ_y. Точка (ξ_x, ξ_y) единственная в области определения системы: $a \le x \le b$, $c \le y \le d$.

В соответствии с общей схемой исходную систему необходимо переписать в виде

$$\begin{cases} x = \varphi_1(x,y); \\ y = \varphi_2(x,y), \end{cases} \tag{4.9}$$

где на функции φ_1 и φ_2 накладываются условия, аналогичные сформулированным ранее для функции φ (4.5).

Далее, используя начальное приближение (x_0, y_0), строим итерационный процесс:

$$\begin{cases} x_i = \varphi_1(x_{i-1}, y_{i-1}); \\ y_i = \varphi_2(x_{i-1}, y_{i-1}). \end{cases}$$

При этом все (x_0, x_1,…x_i) принадлежат отрезку [a, b], а все (y_0, y_1,…y_i) – отрезку [c, d].

Если в области определения {[a, b], [c, d]} выполняются неравенства

$$\begin{aligned} \left|\frac{\partial\varphi_1}{\partial x}\right| + \left|\frac{\partial\varphi_2}{\partial x}\right| \le M_1 < 1; \\ \left|\frac{\partial\varphi_1}{\partial y}\right| + \left|\frac{\partial\varphi_2}{\partial y}\right| \le M_2 < 1; \end{aligned} \qquad (4.10,а)$$

или

$$\begin{aligned} \left|\frac{\partial\varphi_1}{\partial x}\right| + \left|\frac{\partial\varphi_1}{\partial y}\right| \le M_1 < 1; \\ \left|\frac{\partial\varphi_2}{\partial x}\right| + \left|\frac{\partial\varphi_2}{\partial y}\right| \le M_2 < 1, \end{aligned} \qquad (4.10,б)$$

то итерационный процесс сходится к решению

$$lim(x_i)_{i\to\infty} = \xi_x \ , \quad lim(y_i)_{i\to\infty} = \xi_y \ .$$

Выберем из чисел M_1 и M_2 максимальное значение:

$$M = max(M_1, M_2).$$

Тогда оценка погрешности приближения на итерации с номером i будет определяться неравенством:

$$|\xi_x - x_i| + |\xi_y - y_i| \le \frac{M}{1-M}\{|x_i - x_{i-1}| + |y_i - y_{i-1}|\}.$$

Как и ранее, желательно, чтобы $M < ½$.

Сравните материал данного замечания с материалом параграфа 3.3 предыдущего раздела.

***Задания к разделу* 4.**

1. Найти **положительный** корень уравнения $f(x) = x^3 - 0{,}2\,x^2 - 0{,}2\,x - 1{,}2 = 0$. Точное значение корня $\xi = 1{,}20$.

2. Методом половинного деления на отрезке [0, 1] определить корень уравнения $f(x) = x^4 + 2x^3 - x - 1 = 0$ с погрешностью не хуже 10^{-3}. ($\xi \approx 0{,}867$)

3. Методом простых итераций с погрешностью не хуже 10^{-3} найти корни уравнений:

$x = Sin(x) + 0{,}25$ ($\xi \approx 1{,}172$); $e^x - x^2 = 0$ ($\xi \approx -0{,}703$) .

Решить те же уравнения методом половинного деления.

Попробуйте провести вычисления «ручным» способом (с помощью калькулятора) и сравните затраты времени на решение и сложность реализации каждого метода.

4. В физических задачах часто встречается характеристическое уравнение вида

$$\frac{J_0(x)}{J_1(x)} = \frac{x}{A},$$

где $J_0(x), J_1(x)$ – функции Бесселя первого рода нулевого и первого порядка соответственно, A – некоторая положительная постоянная. Указанные функции Бесселя определяются формулами (3.22) главы 3 настоящего пособия. Кроме того, в Приложении В приведены подпрограммы-функции их вычисления. Определить первые три корня данного характеристического уравнения с точностью четыре верных знака после десятичной запятой для A = 1,0. Значения корней:

$$\xi_1 = 1{,}2558;\ \xi_2 = 4{,}0795;\ \xi_3 = 7{,}1558.$$

5. Методом последовательных приближений решить следующую систему [12] с точностью три значащие цифры после десятичной запятой:

$$sin(x–0{,}6) – y = 1{,}6\ ;$$
$$3x – cos(y) = 0{,}9\ .$$

Указания. Исходную систему привести к виду

$$x = cos(y)/3 + 0{,}3 = \varphi_1(x, y)\ ;$$
$$y = sin(x–0{,}6) – 1{,}6 = \varphi_2(x, y)\ .$$

Область расположения корней системы определить графически. Убедиться, что корни расположены в области $0 < x < 0{,}3;\ –2{,}2 < y < –1{,}8$.

С помощью (4.10) определить значение числа M и убедиться в сходимости итерационного процесса.

Приближенные значения корней: $\xi_x \approx 0{,}151;\ \xi_y \approx –2{,}034$.

5. МЕТОДЫ ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ

Во многих практических задачах приходится вычислять определенные интегралы:

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

Если функция $f(x)$ интегрируема на отрезке [a, b] и известна ее первообразная $F(x)$, то искомый интеграл просто вычисляется по формуле Ньютона - Лейбница:

$$J=F(b) - F(a).$$

Однако часто первообразная $F(x)$ не определяется или является слишком сложной, что делает вычисление интеграла J затруднительным или даже невозможным.

Кроме того, на практике подынтегральная функция $f(x)$ может быть задана таблично. Тогда само понятие первообразной теряет смысл.

Поэтому приходится искать методы приближенного вычисления определенных интегралов, к которым в первую очередь относятся *численные методы интегрирования*[5].

Обычно заменяют подынтегральную функцию $f(x)$ на такую аппроксимирующую или интерполирующую функцию $\varphi(x) \approx f(x)$, чтобы интеграл от нее легко вычислялся в элементарных функциях, то есть:

$$J = \int_a^b f(x)dx = \int_a^b \varphi(x)dx + O = S + O, \qquad (5.1)$$

где S – приближенное значение интеграла; O – погрешность вычисления интеграла.

Различные способы приближения подынтегральной функции соответствуют различным методам численного интегрирования. Независимо от выбранного метода в ходе численного интегрирования необходимо вычислить приближенное значение S интеграла и грамотно оценить погрешность O. Понятно, что O желательно сделать минимально возможной.

Все методы численного интегрирования можно классифицировать в зависимости от примененного вида аппроксимации подынтегрального выражения. Формулы, определяющие приближенное значение вычисляемого однократного интеграла, называются *квадратурными формулами*.

[5] Численное вычисление однократного интеграла называется *механической квадратурой*.

Общий подход к решению поставленной задачи будет следующим. Определенный интеграл J из (5.1) представляет собой площадь, ограниченную кривой $f(x)$, осью $0X$ и прямыми $x = a$ и $x = b$. Мы будем пытаться вычислить этот интеграл, разбивая отрезок интегрирования $[a, b]$ на некоторое количество меньших отрезков, находя площадь каждой полоски, получающейся при таком разбиении, и суммируя площади этих полосок.

Необходимо отметить и такие случаи, когда один или оба предела интегрирования бесконечны, или когда подынтегральная функция имеет особенности внутри отрезка интегрирования или на его концах. Данные случаи требуют особого исследования, и к методу численного интегрирования в таких ситуациях нужно относиться очень внимательно.

5.1. Интегрирование по формулам прямоугольников

Эту группу методов можно рассматривать как результат замены подынтегральной функции набором полиномов нулевой степени, то есть подынтегральная функция приближается кусочно-постоянной функцией.

Идея интегрирования по формулам прямоугольников иллюстрируется на рис. 5.1. Промежуток интегрирования $[a,b]$ разбиваем на n *частичных* отрезков *узловыми точками* $x_0, x_1, x_2, \ldots, x_n$, причем $x_0 = a$, $x_n = b$. Заметим, что узловые точки на отрезке $[a,b]$ могут располагаться *не обязательно равномерно,* хотя на рисунке 5.1 показано именно равномерное разбиение. На каждом частичном отрезке $[x_i, x_{i+1}]$ заменим $f(x)$ *постоянным* значением f_i. Очевидно, что такая замена не является однозначной, поскольку константу можно взять равной любому значению $f(x)$ в пределах частичного отрезка. В соответствии с выбранным способом замены мы получаем тот или иной вариант метода прямоугольников. На рис. 5.1 представлен вариант метода *средних* прямоугольников:

$$f_i = f(x_{i-0,5}) = f(\bar{x}_i), \quad \text{где} \quad x_{i-0,5} = \bar{x}_i = x_i + (x_i - x_{i-1})/2, \quad i=1, 2,\ldots n.$$

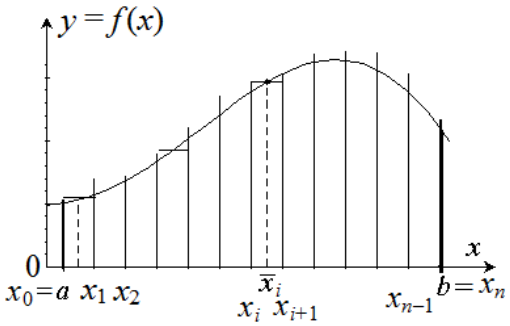


Рис. 5.1. Иллюстрация метода средних прямоугольников

Приближенное значение интеграла на каждом частичном отрезке $[x_{i-1}, x_i]$ выражается теперь как площадь прямоугольника:

$$S_i = (x_i - x_{i-1}) \cdot f_i .$$

Поэтому искомое значение интеграла приближенно представляется следующей суммой:

$$\int_a^b f(x)dx \cong \sum_{i=1}^{n} (x_i - x_{i-1}) f_i . \qquad (5.2)$$

Кроме метода *средних* прямоугольников очевидным образом можно получить метод *правых* и метод *левых* прямоугольников, выбирая в качестве f_i либо $f(x_{i+1})$, либо $f(x_i)$. Однако оказывается [2, 4], что именно метод средних прямоугольников обеспечивает наименьшую погрешность при одинаковой густоте разбиения отрезка $[a,b]$.

Для равномерного разбиения промежутка интегрирования имеем

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

где $h = const$ – шаг разбиения. Тогда

$$S_i = \int_{x_i}^{x_i+h} f(x) \cdot dx \cong h \cdot f(\overline{x_i}), \quad J = \sum_{i=1}^{n} S_i + O = \sum_{i=1}^{n} f(\overline{x_i}) + O, \qquad (5.3)$$

где $h = \dfrac{b-a}{n}$; $\overline{x_i} = x_i - h/2$; $i = 1,2..n$; $O = J - \sum_{i=1}^{n} S_i$.

Проведем для этого случая оценку погрешности O [5]. Разложим $f(x)$ в ряд Тейлора в окрестности средней точки $\overline{x_i}$ отрезка с номером i :

$$f(x) = f(\overline{x_i}) + f'(\overline{x_i}) \cdot (x - \overline{x_i}) + \frac{(x - \overline{x_i})^2}{2!} \cdot f''(\overline{x_i}) + ... \qquad (5.4)$$

После интегрирования $f(x)$ на i-м отрезке с учетом (5.4) получим:

$$J_i = \int_{x_i}^{x_i+h} f(x)dx = f(\overline{x_i}) \cdot h + f'(\overline{x_i}) \cdot \frac{(x - \overline{x_i})^2}{2}\Bigg|_{x_i}^{x_i+h} + f''(\overline{x_i}) \cdot \frac{(x - \overline{x_i})^3}{2 \cdot 3}\Bigg|_{x_i}^{x_i+h} + ... =$$

$$= f(\overline{x}_i) \cdot h + f''(\overline{x}_i) \frac{h^3}{24} + ...$$

Отметим, что после подстановки пределов интегрирования все члены суммы-результата, содержащие четные степени разности $(x - \overline{x})$, обращаются в нуль. Действительно:

$$\left(x-\bar{x}\right)\Big|^{x_i+h} = \left(x_i + h - x_i - h/2\right) = h/2,$$

$$\left(x-\bar{x}\right)\Big|_{x_i} = \left(x_i - x_i - h/2\right) = -h/2.$$

Следовательно, $\left(\frac{h}{2}\right)^{2n} - \left(-\frac{h}{2}\right)^{2n} = 0$, $\left(\frac{h}{2}\right)^{2n+1} - \left(-\frac{h}{2}\right)^{2n+1} = 2\left(\frac{h}{2}\right)^{2n+1}$.

Поэтому, ограничиваясь слагаемым со второй производной[6], получим следующую оценку главного члена погрешности на *i*-ом отрезке:

$$O_i = J_i - S_i = \frac{h^3}{24} \cdot f''\left(\overline{x_i}\right). \tag{5.5}$$

Тогда для всего отрезка $[a, b] = [x_0, x_n]$

$$O = \sum_{i=1}^{n} O_i = \frac{h^2}{24} \sum_{i=1}^{n} h \cdot f''\left(\overline{x_i}\right) = \frac{h^2}{24} \cdot \int_{x_0}^{x_n} f''(x)dx. \tag{5.6}$$

Последнее выражение записано с использованием метода средних прямоугольников для функции $f''(x)$ «наоборот»[7].

Так как по теореме о среднем значении

$$\int_{x_0}^{x_n} f''(x)dx = f''(\xi)\cdot(b-a),\ \xi \in [a,b],$$

то, обозначая $max_{\xi\in[a,b]}\left|f''(\xi)\right| = M_{\text{ПР}}$, перепишем формулу (5.6) в виде:

$$|O| = h^2 \frac{b-a}{24} M_{\text{ПР}}. \tag{5.6,а}$$

Полученная формула дает теоретическую оценку погрешности вычисления интеграла методом средних прямоугольников. Эта оценка является *априорной*, так как не использует вычисленное значение интеграла. Такая оценка неудобна для практического применения, но очень полезна для установления *структуры* главного члена погрешности *O* (остальные отброшены). Степень шага *h*, которой пропорциональна величина *O*, называется *порядком точности метода интегрирования.* Так как *O* оказалась пропорциональной (h^2), то говорят, что метод средних прямоугольников имеет второй порядок точности.

Подчеркнем, что формула (5.6,а) дает только оценку ошибки, но не ее верхнюю границу.

[6] Величина шага *h* считается достаточно малой. Поэтому $h > h^2 > h^3 > \ldots$

[7] Или с помощью теоремы о среднем значении определенного интеграла.

Действуя аналогично, можно показать, что для метода *левых* прямоугольников справедлива следующая априорная оценка погрешности:

$$|O| = \frac{h}{2} \cdot \left| \int_a^b f'(x)dx \right| = h\frac{b-a}{2} \cdot max_{\xi \in [a,b]} |f''(\xi)|, \qquad (5.7)$$

то есть имеет место *первый* порядок точности. Следовательно, погрешность этого метода *больше*, чем в методе средних прямоугольников.

5.2. Метод трапеций

В этом методе подынтегральная функция $f(x)$ на каждом частичном отрезке $[x_i, x_i+h]$ приближается кусочно-линейной зависимостью (см. рис. 5.2),то есть интерполяционным полиномом 1-й степени:

$$\int_{x_i}^{x_i+h} f(x)dx = \frac{h}{2}\left[f(x_i+h) + f(x_i)\right] + O_i. \qquad (5.8)$$

Формула (5.8) дает приближенное значение интеграла на частичном отрезке $[x_i, x_i+h]$ как величину площади соответствующей трапеции (отсюда и название). Тогда для всего отрезка интегрирования $[a, b]$ при постоянном шаге h его разбиения на n равных участков, $h = (b - a)/n$, приближенное значение интеграла выражается следующим образом:

$$\begin{aligned} J &= \frac{h}{2}\left(f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n)\right) + O = \\ &= \frac{b-a}{n}\left[\frac{f(x_0) - f(x_n)}{2} + \sum_{i=1}^{n-1} f(x_i)\right] + O. \end{aligned} \qquad (5.9)$$

Здесь, как и ранее, $f(x_0) = f(a)$, $f(x_n) = f(b)$.

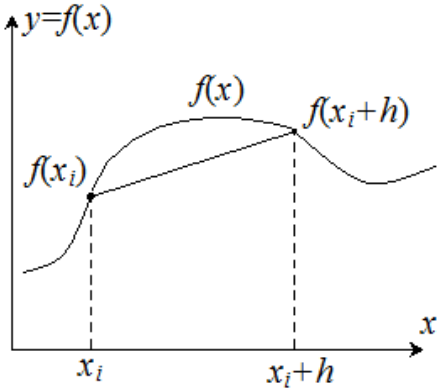


Рис. 5.2. Иллюстрация метода трапеций

Общая погрешность метода трапеций на всем отрезке [*a*,*b*] в два раза *больше*, чем у метода средних прямоугольников [5]:

$$|O| = \frac{h^2}{12} \cdot \left| \int_{x_0}^{x_n} f''(x)dx \right| \le h^2 \cdot \frac{b-a}{12} \cdot max_{\xi \in [a,b]} |f''(\xi)| \,. \qquad (5.10)$$

Это означает, что такой способ аппроксимации не является оптимальным для численного интегрирования. Следовательно, среди методов численного интегрирования, имеющих второй порядок погрешности, наилучшим следует признать метод средних прямоугольников.

5.3. Метод Симпсона (метод парабол)

В этом методе с целью уменьшения погрешности интегрирования подынтегральную функцию заменяют квадратичной параболой. Для такой замены необходимо уже три узловых точки: x_{i-1}, x_i и x_{i+1} (см. рис. 5.3). Проведя необходимые преобразования, можно получить, что

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx \approx \left(f_{i-1} + 4 \cdot f_i + f_{i+1}\right) \cdot \frac{h}{3}. \qquad (5.11)$$

Здесь величина h имеет прежний смысл: $x_{i+1} - x_i = x_i - x_{i-1} = h$.

Отметим, что формула (5.11) является точной, если $f(x)$ является любым полиномом третьей степени [5, 7]: $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$.

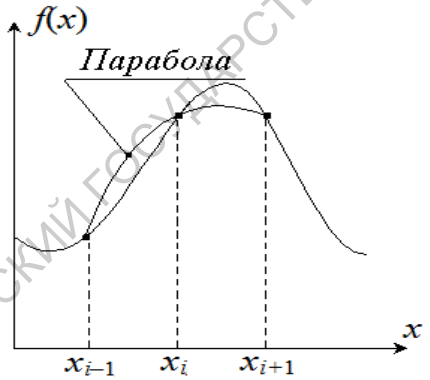


Рис. 5.3. Параболическая аппроксимация функции в методе Симпсона

Для интегрирования методом Симпсона по всему отрезку [*a*,*b*] отрезок нужно разбить на ***четное*** число малых интервалов [x_{i-1}, x_i], на каждой паре которых применяется формула (5.11) для узловых точек x_{i-1}, x_i, x_{i+1} . Тогда, если h – шаг в разбиении отрезка [*a* ,*b*], $n = (b - a)/h$ – ***<u>четное</u>*** число малых интервалов, то значение интеграла на всем отрезке по методу Симпсона имеет вид:

$$J \approx \frac{h}{3}\cdot\left[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+2f(x_4)+\ldots+\right.$$
$$\left.+2f(x_{n-2})+4f(x_{n-1})+f(x_n)\right], \qquad (5.12)$$

где, как и ранее, $f(x_0) = f(a)$, $f(x_n) = f(b)$. При этом полная погрешность интегрирования выразится следующим образом:

$$|O| = \frac{h^4(b-a)}{2880}\cdot max_{x\in[a,b]}\left|f^{\mathrm{IV}}(x)\right|,\ b-a = h\cdot n.$$

Следовательно, формула Симпсона имеет четвертый порядок точности.

Формула Симпсона обеспечивает высокую точность интегрирования, если четвертая производная подынтегральной функции не слишком велика. В противном случае методы второго порядка точности могут дать меньшую погрешность, чем метод парабол.

Например [11], пусть $f(x)= -25x^4+45x^2-7$. Необходимо вычислить интеграл $\int_{-1}^{+1} f(x)dx$. При вычислении по формуле трапеций, разбивая отрезок [–1, 1] на два промежутка ($n=2$), получается точный результат, равный 4, тогда как при вычислении по формуле Симпсона получается результат, не совпадающий даже по знаку: (–8/3).

5.4. Формула Ньютона - Котеса

Пусть функция $y = f(x)$ задана таблицей значений $(x_i, y_i = f(x_i))$ в n +1 равномерно разнесенных узловых точках $x_i = x_0 + i\cdot h$ (i = 1, 2, ...n) на отрезке [a, b], причем $f(a) = f(x_0)$, $f(b) = f(x_n)$. Требуется найти приближенное значение интеграла (5.1).

Выше рассмотренные методы численного интегрирования приближают подынтегральную функцию с помощью либо кусочно-постоянной, либо кусочно-линейной, либо кусочно-параболической интерполяции. Но по заданным значениям функции в узловых точках мы можем построить интерполяционный полином Лагранжа $L_n(x)$ степени n, см. формулу (3.5). Если узлы интерполяции равноотстоящие, то заменой переменных $q = (x - x_0)/h$ полином Лагранжа нетрудно привести к виду [7]:

$$L_n(x) = \sum_{i=0}^{n}\frac{(-1)^{n-i}}{i!(n-i)!}\cdot\frac{q(q-1)(q-2)\ldots(q-n)}{q-i}y_i.$$

Теперь заменим в (5.1) подынтегральную функцию интерполирующим полиномом Лагранжа. Получим:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} \frac{(-1)^{n-i}}{i!(n-i)!} \cdot \int_a^b y_i \frac{q(q-1)(q-2)\ldots(q-n)}{q-i} dx =$$

$$= (b-a)\sum_{i=0}^{n} y_i \cdot \frac{(-1)^{n-i}}{i!(n-i)!} \cdot \frac{1}{n}\int_0^n q(q-1)\ldots(q-i+1)(q-i-1)\ldots(q-n)dq\,. \qquad (5.13)$$

В (5.13) учтено, что $h = (b-a)/n$ и $dx = h \cdot dq$.

Выражение (5.13) носит название *формулы* ***Ньютона - Котеса***. Привлекательной особенностью формулы Ньютона - Кортеса является то, что в выражении (5.13) все множители при y_i *не зависят* от вида функции $f(x)$, а зависят только от степени n полинома Лагранжа (или от числа узлов интерполяции без одного). Эти множители называются коэффициентами Ньютона - Котеса:

$$H_i = \frac{(-1)^{n-i}}{i!(n-i)!} \cdot \frac{1}{n}\int q(q-1)\ldots(q-i+1)(q-i-1)\ldots(q-n)dq.$$

Коэффициенты H_i вычислены заранее для различного числа n и сведены в справочную таблицу. Здесь представлена такая таблица для $n \le 7$ (табл. 5.1) [7]:

Таблица 5.1

Коэффициенты Ньютона – Котеса для $n \le 7$

n	H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7
1	1/2	1/2	–	–	–	–	–	–
2	1/6	2/3	1/6	–	–	–	–	–
3	1/8	3/8	3/8	1/8	–	–	–	–
4	7/90	16/45	2/15	16/45	7/90	–	–	–
5	19/288	25/96	25/144	25/144	25/96	19/288	–	–
6	41/840	9/35	9/280	34/105	9/280	9/35	41/840	–
7	$\frac{751}{17280}$	$\frac{3577}{17280}$	$\frac{1323}{17280}$	$\frac{2989}{17280}$	$\frac{2989}{17280}$	$\frac{1323}{17280}$	$\frac{3577}{17280}$	$\frac{751}{17280}$

С введением коэффициентов Ньютона - Котеса формула (5.13) приобретает окончательный и очень простой вид:

$$J = \int_a^b f(x)dx \approx (b-a)\sum_{i=0}^{n} y_i \cdot H_i. \qquad (5.14)$$

Если положить $n = 1$, то формула Ньютона - Котеса переходит в формулу трапеций для отрезка $[a, b]$, а при $n = 2$ – в формулу Симпсона. При $n = 3$ получается так называемое правило трех восьмых:

$$\int_a^b f(x)dx = \frac{3h}{8}\left(y_0 + 3y_1 + 3y_2 + y_3\right). \qquad (5.15)$$

Погрешность этого выражения оценивается соотношением [5, 7]

$$|O| = h^5 \cdot \frac{3}{80} \cdot \left| f^{\,\mathrm{IV}}(\xi) \right|_{\xi \in [a,b]}.$$

Следовательно, правило трех восьмых имеет тот же порядок точности, что и метод Симпсона.

В общем случае увеличение степени интерполирующего полинома приводит к повышению точности вычислений. Можно показать [5 – 7], что для интерполирующего полинома степени n имеет место следующая оценка погрешности интегрирования:

$$|O| \le \frac{max\left| f^{(n+1)}(x) \right|}{(n+1)!} \cdot \int_a^b (x - x_0)(x - x_1)\ldots(x - x_n)dx\,.$$

Это означает, что с ростом n погрешность $|\,O\,|$ должна уменьшаться. Кроме того, наличие симметрии в формулах интегрирования рассмотренного типа повышает их точность [5, 7]. Формула интегрирования называется симметричной, если:

1) n четное,
2) узлы расположены симметрично середины отрезка интегрирования,
3) формула интегрирования имеет попарно одинаковые коэффициенты при симметрично расположенных слагаемых.

Именно поэтому на практике формула средник прямоугольников часто приводит к более точному результату, чем формула трапеции, а метод Симпсона по точности не хуже правила трех восьмых. Естественно, что формулы Ньютона - Котеса для $n > 3$ превосходят по точности метод Симпсона.

Однако на практике формулы Ньютона - Котеса с $n \ge 8$ используются редко из-за их так называемой численной неустойчивости. Дело в том, что для $n \ge 8$ коэффициенты Ньютона - Котеса имеют как положительные, так и отрицательные значения, и при вычислениях на компьютере приходится сталкиваться с ситуацией вычитания двух близких по величине чисел.

5.5. Метод Гаусса

Все формулы интегрирования, рассмотренные выше, предполагали, что их узлы заданы заранее. Однако оказывается, что при надлежащем выборе уз-

лов можно получить формулы интегрирования, имеющие повышенную точность по сравнению с формулами типа Ньютона - Котеса при том же порядке интерполирующего полинома. Такие формулы называются формулами Гаусса или формулами наивысшей алгебраической степени точности.

На отрезке интегрирования [–1, 1] формула Гаусса выглядит следующим образом [3, 5, 7]:

$$\int_{-1}^{1} f(x)dx \approx c_1 f(x_1) + c_2 f(x_2) + \ldots + c_n f(x_n). \qquad (5.16)$$

Поставим задачу: как нужно выбрать узловые точки интерполяции x_i и коэффициенты c_i, чтобы формула (5.16) была точной для всех полиномов наивысшей возможной степени N: $f(x) = P_N(x)$.

Пусть в нашем распоряжении имеется $2n$ постоянных x_i и c_i (i = 1, 2, ... n). Поскольку полином степени $2n$ –1 определяется $2n$ коэффициентами, то искомая наивысшая степень в общем случае равна $N = 2n–1$. Таким образом, $2n$ неизвестных величины (x_i, c_i) и условие, что формула (5.16) является точной, требуют, чтобы $f(x)$ являлась полиномом степени $2n–1$. Кроме того, налагается жесткое условие на расположение величин $x_1, x_2, \ldots x_n$ в пределах отрезка интегрирования. Именно поэтому этот отрезок и выбирают жестко определенным: [–1, 1].

В таблице 5.2 приведены без вывода числовые выражения для узловых точек интерполяции x_i и коэффициентов c_i (i = 1, 2, ... n) на отрезке интегрирования [–1, 1] с числом узловых значений от 1 до 8 (n = 1, 2 ... 8). Для проведения вычислений по формуле (5.16) требуется рассчитать все $y_i = f(x_i)$ для выбранного значения n и подставить в (5.16).

Для вычисления интеграла общего вида $\int_a^b f(x)dx$ методом Гаусса следует произвести замену переменной $z_i = \frac{b+a}{2} + \frac{b-a}{2} x_i$, i = 1, 2, ... n. Тогда формула Гаусса примет вид:

$$\int_a^b f(x)dx = \frac{b-a}{2}\left[c_1 f(z_1) + c_2 f(z_2) + \ldots + c_n f(z_n)\right]. \qquad (5.17)$$

Точность вычислений по формуле Гаусса на порядок выше точности вычислений по формулам типа Ньютона - Котеса [3, 5, 7] при одном и том же n . Кроме того, все коэффициенты в формуле Гаусса (5.16) положительны для любого n. Поэтому формулы (5.16) и (5.17) вычислительно устойчивы для любого

числа узловых точек (в отличии от формул Ньютона - Котеса). Наиболее полно теория численного интегрирования по методу Гаусса представлена в монографии [13].

Итак, метод Гаусса позволяет достичь большей точности, чем интегрирование по формулам типа Ньютона - Котеса при том же количестве узлов интерполяции. С другой стороны, узловые точки в методе Гаусса жестко определены и не зависят от желания исследователя. При численном интегрировании приходится делать выбор, например, между простотой метода Симпсона и возможной экономией машинного времени при использовании метода Гаусса. На практике чаще используют метод Симпсона, особенно если компьютер, на котором проводятся вычисления, достаточно производительный.

Таблица 5.3

Коэффициенты и абсциссы для формулы интегрирования Гаусса при *n* от 1 до 8

$n = 1$	$x_1 = 0{,}5$	$c_1 = 2$
$n = 2$	$-x_1=x_2=0{,}577350$	$c_1=c_2=1$
$n = 3$	$-x_1=x_3=0{,}774597$; $x_2=0$	$c_1=c_3=0{,}555555$; $c_2=0{,}888889$
$n = 4$	$-x_1=x_4=0{,}861136$; $-x_2=x_3=0{,}339981$	$c_1=c_4=0{,}347855$; $c_2=c_3=0{,}652145$
$n = 5$	$-x_1=x_5=0{,}906180$; $-x_2=x_4=0{,}538470$; $x_3=0$	$c_1=c_5=0{,}236927$; $c_2=c_4=0{,}478629$; $c_3=0{,}568889$
$n = 6$	$-x_1=x_6=0{,}932470$; $-x_2=x_5=0{,}661210$; $-x_3=x_4=0{,}238620$	$c_1=c_6=0{,}171324$; $c_2=c_5=0{,}360761$; $c_3=c_4=0{,}467914$
$n = 7$	$-x_1=x_7=0{,}949108$; $-x_2=x_6=0{,}741531$; $-x_3=x_5=0{,}405845$; $x_4=0$	$c_1=c_7=0{,}129485$; $c_2=c_6=0{,}279705$; $c_3=c_5=0{,}381830$; $c_4=0{,}417960$
$n = 8$	$-x_1=x_8=0{,}960290$; $-x_2=x_7=0{,}796666$; $-x_3=x_6=0{,}525532$; $-x_4=x_5=0{,}183434$	$c_1=c_8=0{,}101228$; $c_2=c_7=0{,}222381$; $c_3=c_6=0{,}313707$; $c_4=c_5=0{,}362684$

5.6. Сравнение некоторых методов численного интегрирования

Проведем на простом примере сравнение трех методов численного интегрирования: метода трапеций, метода Симпсона и метода Гаусса. Приводимый пример взят из монографии [2].

Рассмотрим интеграл вероятностей в виде (без множителя $1/\sqrt{2\pi}$):

$$J = \int_{-2}^{2} exp\left(-\frac{x^2}{2}\right) \cdot dx .$$

Точное значение этого интеграла J=2,3925 (для пяти значащих цифр).

Проведем вычисления данного интеграла указанными тремя способами при различном числе значений подынтегральной функции. При этом необходимо помнить, что в методах трапеции и Симпсона отрезок интегрирования [–2, 2], а в методе Гаусса [–1, 1], точки абсцисс подынтегральной функции жестко закреплены и мы должны использовать формулу (5.17).

Результаты вычислений сведены в таблицу 5.4.

Таблица 5.4

Сравнение методов численного интегрирования

	Шаг	***Значение интеграла***	***Абсолютная ошибка***
Правило трапеций	h = 1,0;	2,3484	0,0441
	h = 0,5;	2,3813	0,0112
	h = 0,25;	2,3898	0,0027
Метод Симпсона	h = 1,0;	2,3743	0,0182
	h = 0,5;	2,3923	0,0002
	h = 0,25;	2,3926	0,0001
Метод Гаусса (понятие «шаг» отсутствует)	***Число точек***		
	n = 2	2,0536	0,3389
	n = 3	2,4471	–0,0546
	n = 4	2,3859	0,0066
	n = 5	2,3931	–0,0006
	n = 6	2,3925	0,0000

Какие выводы можно сделать на основе данного примера? (Хотя на основе одного примера делать глобальные выводы некорректно.)

а) Метод Симпсона при n значениях функции дает примерно ту же точность, что и метод трапеций при 2n значениях подынтегральной функции.

б) Метод Гаусса при n значениях функции дает примерно ту же точность, что и метод Симпсона при 2n значениях подынтегральной функции.

Какой же метод предпочтительнее?

При одинаковом числе заданных значений функции метод Гаусса точнее. Однако если потребуется увеличить число значений функции для повышения точности интегрирования, предыдущие значения в методе Гаусса уже использовать нельзя, а в методе Симпсона можно, поскольку для повышения точности часто используется способ уменьшения шага вдвое.

Но самое главное заключается в том, что для интегрирования функции, определяемой экспериментальными данными, метод Гаусса можно использовать лишь тогда, когда расположение абсцисс узлов подходит для метода Гаусса, что случается редко. Формулами трапеций и средних прямоугольников можно пользоваться при любом расположении узлов. В принципе, параболическую аппроксимацию метода Симпсона нетрудно

распространить на случай нерегулярного расположения абсцисс узлов. При этом формула Симпсона обеспечивает достаточную точность при умеренном количестве узловых точек.

Наконец, мы ничего не говорили о способах численного интегрирования, основанных на методах сплайн-интерполяции подынтегральной функции. Эти методы находят все более широкое применение и входят во многие математические пакеты в связи с резким увеличением производительности и памяти вычислительной техники.

Исследователю необходимо самому решить, какой метод интегрирования более всего подходит для решения его задачи.

Программирование формул (5.3), (5.9), (5.12), (5.14) и (5.17) труда не составляет – обычное вычисление конечной суммы с известными слагаемыми. Требуемую точность можно обеспечить путем пересчета суммы при удвоении числа n разбиений отрезка интегрирования (кроме метода Гаусса) и сравнении результатов вычислений.

При интегрировании по методу Гаусса заданную точность придется обеспечивать пересчетом суммы (5.17) с последовательным увеличением числа узловах точек и сравнением результатов, при этом меняются не только значения подынтегральной функции $f(z_i)$, но и коэффициенты c_i в (5.17).

5.7. Вычисление интегралов от сильно осциллирующих функций

В физических задачах часто требуется вычислить интегралы вида:

$$\int_a^b f(x)\cos(\mu x)dx, \quad \int_a^b f(x)\sin(\mu x)dx. \qquad (5.18)$$

При больших значениях параметра μ здесь *невыгодно* использовать формулы интегрирования, основанные на кусочно-полиномиальной интерполяции всей подынтегральной функции, так как потребуется большое количество узловых точек из-за их сильной осцилляции. Намного более выгодно кусочно-полиномиальным приближением описывать только $f(x)$ – в этом случае интегралы типа (5.18) вычисляются аналитически, и при таком вычислении не теряется информация об осцилляции подынтегрального выражения.

Действительно, пусть $P_m(x)$ – полином степени m, приближающий функцию $f(x)$ на отрезке $[a, b]$. Тогда вычисление интегралов (5.18) сводится к вычислению интегралов

$$\int_a^b P_m(x)\cos\mu x dx, \quad \int_a^b P_m(x)\sin\mu x dx. \qquad (5.18,а)$$

Это нетрудно сделать, поскольку интегралы $\int x^k \cos(\mu x)dx$ и $\int x^k \sin(\mu x)dx$, где k – целое число, табличные.

Если отрезок $[a, b]$ большой или функцию $f(x)$ приходится приближать полиномом слишком высокой степени (что нежелательно), отрезок $[a, b]$ можно разбить на ряд малых интервалов, для которых и вычисляются интегралы (5.18,а).

В частности, пусть функция $f(x)$ задана $n + 1$ своим значением:
$f_0 = f(x_0) = f(a)$; $f_1 = f(x_1)$; $f_2 = f(x_2)$; ... $f_n = f(x_n) = f(b)$.
Тогда для кусочно-линейной интерполяции функции $f(x)$

$$\int_{x_0}^{x_n} f(x)\cos(\mu x)dx \approx -\frac{1}{\mu} f_0 \sin(\mu x_0) + \frac{1}{\mu} f_n \sin(\mu x_n) + \\ + \frac{1}{\mu^2}\sum_{j=1}^{n} \frac{\cos(\mu x_j) - \cos(\mu x_{j-1})}{h_j}\left(f_j - f_{j-1}\right); \qquad (5.19)$$

$$\int_{x_0}^{x_n} f(x)\sin(\mu x)dx \approx \frac{1}{\mu} f_0 \cos(\mu x_0) - \frac{1}{\mu} f_n \cos(\mu x_n) + \\ + \frac{1}{\mu^2}\sum_{j=1}^{n} \frac{\sin(\mu x_j) - \sin(\mu x_{j-1})}{h_j}\left(f_j - f_{j-1}\right). \qquad (5.20)$$

Здесь $h_j = x_j - x_{j-1}$. Формулы (5.19) и (5.20) получены в приближении функции $f(x)$ кусочно-линейным образом, как и в методе трапеций. Показано [10], что погрешность вычисления интегралов (5.19) и (5.20) для любого μ не превосходит величины

$$\frac{b-a}{8} h_{max}^2 \cdot max\left|f''(x)\right|_{x\in[a,b]}.$$

Аналогичным образом можно получить формулы для вычисления интегралов типа (5.18) при замене функции $f(x)$ кусочно-полиномиальным образом. Так, если функция $f(x)$ задана $n + 1$ своим значением и n четное, то нетрудно организовать вычисление такого интеграла, использовав метод Симпсона кусочно-параболического приближения $f(x)$. Часто используют кусочно-кубическое приближение для $f(x)$, используя метод кубических сплайн-функций.

Замечание. Пусть имеется определенный интеграл типа $\int_a^b \rho(x) f(x)dx$, где $\rho(x)$ - заданная интегрируемая функция, а $f(x)$ – некоторая достаточно гладкая функция. Формулы приближенного интегрирования, основанные на замене функции $f(x)$ таким интерполирующим выражением, что указанный интеграл можно вычислить аналитически, называются *формулами интерполяционного типа*. Очевидно, что в предыдущих параграфах данной главы мы рассматривали именно такие формулы, полагая $\rho(x) \equiv 1$.

Вопрос о том, какие формулы применять при компьютерных вычислениях интегралов, решается исследователем самостоятельно в зависимости от сложности подынтегрального выражения, наличия в нем особых точек, длины отрезка интегрирования (в том числе и проблема вычисления *несобственных* интегралов) и т.п. Интересующихся отсылаем к классическим монографиям [13, 14].

***Задания к разделу* 5.**

1. Вычислить функцию e^x, представленную рядом Фурье при $x \in [-\pi, \pi]$:

$$e^x = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx). \qquad (5.21)$$

Коэффициенты a_n и b_n определяются формулами:

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} e^x dx = 7{,}35215588;$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} e^x \cos(nx)dx = \frac{(-1)^n(e^{\pi} - e^{-\pi})}{\pi(1+n^2)};$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} e^x \sin(nx)dx = \frac{(-1)^{n+1} n \cdot (e^{\pi} - e^{-\pi})}{\pi(1+n^2)}.$$

С помощью формулы (5.21) вычислить e^x для $x = 1$ с погрешностью не хуже 10^{-4}, значение числа $e = 2{,}71828$. Выход из цикла при суммировании бесконечного ряда должен производиться при достижении указанной точности. Значения a_n и b_n определить аналитически.

Провести те же расчеты, вычислив коэффициенты a_n и b_n с помощью соотношений (5.19) и (5.20). Для этого функцию e^x представить массивом из 21 значения, разбив отрезок $[-\pi, \pi]$ на 20 равных интервалов.

3) Вычислить a_3 и b_3 с помощью метода Симпсона и сравнить результат с точным значением. Определить число узловых точек, необходимых для вычислений a_3 и b_3 по Симпсону с погрешностью не хуже 0,01%.

2. Интеграл ошибок Гаусса имеет следующее представление:

$$erf(z) = \frac{2}{\sqrt{\pi}} \int_0^z exp(-t^2) dt .$$

В справочнике по специальным функциям [15] приводится его аппроксимация следующего вида:

для $0 \le x \le 8$ и при $|\varepsilon| \le 2{,}5 \cdot 10^{-5}$

$$erf(z) \cong 1 - (a_1 t + a_2 t^2 + a_3 t^3) \exp(-z^2) + \varepsilon ,$$

где $t = 1/(1 + 0{,}47047 \cdot z)$; $a_1 = 0{,}3480242$; $a_2 = -0{,}0958798$; $a_3 = 0{,}7478556$.

Написать программу расчета интеграла ошибок по его приближенному и интегральному представлениям. Провести вычисления с шагом 0,5 изменения аргумента, результаты сравнить между собой и с табличными значениями. Вычисление интеграла производить с указанной здесь погрешностью ε.

3. С помощью различных методов интегрирования проверить данные таблицы 5.4.

4. Известно, что $\int_0^{\pi} sin(x) \cdot dx = 2{,}0$. Вычислить данный интеграл численно с погрешностью не хуже 10^{-5}, использовав все разобранные в данной главе методы численного интегрирования. Сравнить трудоемкость вычислений. Определить оптимальный метод интегрирования данного выражения.

5. Полный эллиптический интеграл первого рода определяется формулой:

$$K(\theta) = \int_0^{\pi/2} \frac{d\varphi}{\sqrt{1 - sin^2\theta \, sin^2\varphi}}.$$

Вычислить значение $K(30^o)$, используя рассмотренные формулы интегрирования, с погрешностью не хуже 10^{-4}. Значение $K(30^o) = 1{,}6858$.

6. Имеются экспериментальные результаты, согласно которым y является некоторой непрерывной функцией от x:

x	1,0	1,2	1,4	1,6	1,8	2,0	2,2	2,4
y	1,00	1,82	2,08	3,18	3,52	4,70	5,12	6,38

Вычислить среднее значение величины $y = f(x)$ на отрезке изменения переменной x. Указание: использовать теорему о среднем значении.

ЗАКЛЮЧЕНИЕ

Действующий учебный план физического факультета по направлению 03.03.02 “Физика” предусматривает цикл учебных дисциплин[8], который имеет целью получение студентами знаний по вопросам вычислительной физики, формирование у них навыков и умений практического применения современных персональных компьютеров для целей математического моделирования процессов. Указанный цикл прорабатывается студентами в течение практически всего времени обучения и включает значительное количество учебных предметов. Первой дисциплиной в этом цикле является «Вычислительная физика». Именно в рамках этой дисциплины студенты знакомятся с основными типами задач вычислительной физики, наиболее распространенными методами их решения и не только в теории, но и на практике, развивают знания, умения и навыки в области научного программирования и практической работе на компьютере.

Настоящее учебное пособие призвано помочь студентам освоить материал названной учебной дисциплины, ибо в рамках недостатка лекционных часов и часов практических занятий, выделяемых на данную дисциплину, преподавателю чрезвычайно сложно изложить достаточно широкий класс типовых задач и методов их решения.

Автор стремился изложить материал просто и понятно, иллюстрируя рассматриваемые задачи и методы примерами, часто решаемыми «вручную». Изложение каждого из методов вычислений включает его элементарную теорию, описание алгоритма и, как правило, текст работающей программы или процедуры, реализующей метод. По мнению автора, в настоящем учебном пособии представлен минимальный перечень вопросов, которые должны быть освещены в рамках курса «Вычислительная физика». За пределами учебного пособия остались такие важные вопросы, как вычисление несобственных интегралов, решение системы дифференциальных уравнений, приближение двумерных функций и их численное интегрирование, конечноразностные методы решения краевых задач, описание методов Монте-Карло и др.

[8]«Вычислительная физика (практикум на ЭВМ)», « Численные методы и математическое моделирование», «Методы математической физики», «Моделирование физических процессов».

Список использованных источников

1. Турчак Л.И. Основы численных методов: Учеб. пособие. – М.: Наука, 1987.
2. Мак-Кракен Д., Дорн У. Численные методы и программирование на ФОРТРАНе / Пер. с англ. под ред. Б.Н. Наймарка. – М.: Мир, 1977.
3. Хемминг Р.В. Численные методы для научных работников и инженеров. – М.: Наука, 1972.
4. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений. – М.: Мир, 1980.
5. Самарский А.А., Гулин А.В. Численные методы. – М.: Наука. Гл. ред. физ-мат. лит., 1989.
6. Калиткин Н.Н. Численные методы. – М.: Наука. Гл. ред. физ-мат. лит., 1978.
7. Демидович Б.П., Марон И.А. Основы вычислительной математики. – М.: Наука. Гл. ред. физ-мат. лит., 1970.
8. Справочник по специальным функциям с формулами, графиками и математическими таблицами / Под ред. М. Абрамовица и И. Сигала. – М.: Наука. Гл. ред. физ-мат. лит., 1979.
9. Копченова Н.В., Марон И.А. Вычислительная математика в примерах и задачах. - М.: Наука, 1972.
10. Завьялов Ю.С., Квасов Б.И., Мирошниченко В.Л. Методы сплайн-функций. – М.: Наука, 1980.
11. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. – Томск: МП “РАСКО”, 1991.
12. Численные методы / Н.И. Данилина, Н.С. Дубровская, О.П. Кваша, Г.Л. Смирнов, Г.И. Феклисов. – М.: Высшая школа, 1976. – 368 с.
13. Крылов В.И., Бобков В.В., Монастырный П.И. Начала теории вычислительных методов. Интерполирование и интегрирование. – Минск: Наука и техника, 1983.
14. Крылов В.И., Бобков В.В., Монастырный П.И. Вычислительные методы. - Т. 1. – М.: Наука, 1976. То же. – Т. 2. – М.: Наука, 1977.
15. Янке Э., Эмде Ф., Лёш Ф. Специальные функции: формулы, графики, таблицы / Пер. с нем. под ред. Л.И. Серова. - М.: Наука. Гл. ред. физ-мат. лит., 1977.

ПРИЛОЖЕНИЕ А

ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

1. Для приведенных здесь числовых рядов:

$$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{2n-1}=\frac{\pi}{4};\qquad \sum_{n=1}^{\infty}\frac{1}{(2n-1)^2}=\frac{\pi^2}{8};\qquad \sum_{n=1}^{\infty}\frac{1}{(2n-1)^3}=1{,}05179979;$$

$$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{n}=ln\,2;\qquad \sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{(2n-1)^3}=\frac{\pi^2}{32}$$

определить число слагаемых, необходимых для вычисления данных рядов с погрешностью не хуже 10^{-3}, 10^{-4}, 10^{-5} и 10^{-6}. Оценить возрастание вычислительных затрат с ростом точности вычислений. Принять π=3,14159265.

2. Решить систему уравнений методом Гаусса, проводя вычисления вручную и с помощью компьютерной программы:

$$\begin{aligned} 2x_1 + 2x_2 - x_3 + x_4 &= 4,\\ 4x_1 + 3x_2 - x_3 + 2x_4 &= 6,\\ 8x_1 + 5x_2 - 3x_3 + 4x_4 &= 12,\\ 3x_1 + 3x_2 - 2x_3 + 2x_4 &= 6. \end{aligned}$$

Ответ: $x_1 = 1,\ x_2 = 1,\ x_3 = -1,\ x_4 = -1.$

3. Для системы уравнений

$$\begin{aligned} 1{,}781x_1 + 3{,}008x_2 - 4{,}880x_3 &= -7{,}704\\ 4{,}632x_1 - 1{,}064x_2 - 2{,}274x_3 &= -6{,}359\\ -3.387x_1 + 9{,}814x_2 - 4{,}779x_3 &= 3{,}946 \end{aligned}$$

известно следующее приближенное решение: x_1^0= 2,0; x_2^0= 3,0; x_3^0= 4,0.

Необходимо уточнить это решение в соответствии с материалом параграфа 2.2.

4. Итерационным методом Гаусса-Зейделя решить следующую систему:

$$\begin{aligned} 10x_1 + 2x_2 + 6x_3 &= -28\\ x_1 + 10x_2 + 9x_3 &= 7\\ 2x_1 - 7x_2 - 10x_3 &= -17. \end{aligned}$$

Итерационный процесс продолжать до тех пор, пока разница между двумя последовательными приближениями для всех x по модулю не станет меньше 0,02.

5. На основе метода Гаусса - Зейделя решить с погрешностью не хуже 0,01 следующие две системы уравнений, первоначально приведя их к виду, удобному для итераций:

а) $\begin{aligned} 4x_1 - x_2 + x_3 &= 4,\\ x_1 + 6x_2 + 2x_3 &= 9,\\ -x_1 - 2x_2 + 5x_3 &= 2. \end{aligned}$

Точное решение: $x_1 = x_2 = x_3 = 1{,}0.$

Указание: положить $x_1^0 = x_2^0 = x_3^0 = 0.$

б) $8{,}7x_1 - 3{,}1x_2 + 1{,}8x_3 - 2{,}2x_4 = -9{,}7$
$2{,}1x_1 + 6{,}7x_2 - 2{,}2x_3 = 13{,}1$
$3{,}2x_1 - 1{,}8x_2 - 9{,}5x_3 - 1{,}9x_4 = 6{,}9$
$1{,}2x_1 + 2{,}8x_2 - 1{,}4x_3 - 9{,}9x_4 = 25{,}1$

Ответ: $x_1 = -0{,}72$; $x_2 = 1{,}88$; $x_3 = -0{,}92$; $x_4 = -1{,}94$.

6. Как показано в параграфе 3.1.3, минимальность величины $\Omega = S_m / (n - m)$ определяет наилучший аппроксимирующий полином, где значение S_m определено в соответствии с (3.7).

Рассмотрим следующие исходные данные:

i	**0**	**1**	**2**	**3**	**4**
x	0	1	2	3	4
y	0	20	40	50	70

Построить для этих данных аппроксимирующие линейный и квадратичный полиномы, нарисовать их графики и сравнить по критерию Гаусса. Также построить для этих данных интерполирующий полином Лагранжа и сравнить поведение интерполирующей и аппроксимирующих зависимостей в интервалах между узловыми точками.

7. Предположим, что имеются экспериментальные результаты, согласно которым величина f является некоторой функцией переменной x :

x	1,0	1,2	1,4	1,6	1,8	2,0	2,2	2,4	2,6	2,8	3,0
***f*(*x*)**	1,0	1,82	2,08	3,18	3,52	4,70	5,12	6,38	6,98	8,22	9,0

Проанализируйте данную таблицу значений. Полиномом какой степени выгодно аппроксимировать функцию *f*, чтобы быть уверенным в третьем знаке после запятой? Постройте данный аппроксимирующий полином

8. Составить таблицу значений функций Бесселя первого рода нулевого и первого порядков при 4-х значащих цифрах после запятой с шагом 0,5 для $0 \le x \le 8$. Сверить найденные значения с данными [15], таблица 45(стр.201).

9. Дан интеграл вероятностей в виде:

$$P(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} exp\left(-\frac{t^2}{2}\right) dt .$$

Его значения для некоторых x приведены в таблице [15]:

x	0	0,1	0,2	0,3	0,4	0,5	1,0	2,0	3,0
***P*(*x*)**	0,50	0,53983	0,57926	0,61791	0,65542	0,69146	0,84134	0,97725	0,98650

Написать программу и провести вычисления данного интеграла с точностью до 5-ти знаков поле запятой и сравнить полученные значения с табличными.

Дополнить таблицу значениями $P(x)$ для x от 0,5 до 0,9 с шагом 0,1.

10. Вычислить интегралы с погрешностью не хуже 10^{-6}:

1) $\int_0^{1/3} \frac{dx}{\sqrt[3]{1-x^2}}$. 2) $\int_0^{0.5} \sqrt{2-x^3}\,dx$. 3) $\int_0^{0.5} \sqrt[3]{3-2\cdot x^2}\,dx$. 4) $\int_0^{\pi/2} \sqrt{2+\cos(x)}\,dx$.

$\int_0^{\pi/2} \sqrt{1-0{,}1\cdot \sin^2(x)}\,dx$. 6) $\int_0^{\pi/3} \sqrt{x+\cos x}\,dx$. 7) $\int_0^{\pi/4} \frac{\sin x}{1+x^2}\,dx$. 8) $\int_0^{\pi/6} \frac{x}{1+tgx}\,dx$.

$\int_0^{0.5} \frac{arctgx}{x}\,dx$. 10) $\int_0^{\pi} \frac{\sin x}{x}\,dx$. 11) $\int_0^{1} \frac{shx}{x}\,dx$. 12) $\int_0^{1/4} \frac{\arcsin x}{x}\,dx$.

13) $S(x)=\int_0^{x} \sin\left(\frac{\pi}{2}t^2\right)dt$, где $|x| \le 2{,}55$.

11. В данной группе заданий все корни ищутся с погрешностью не хуже 10^{-3}... 10^{-4}.

1) Методом простых итераций найти корни уравнений:

$x = Sin(x) + 0{,}25$ $(\xi \approx 1{,}172)$; $e^x - x^2 = 0$ $(\xi \approx -0{,}703)$.

Решить те же уравнения методом половинного деления.

Попробуйте провести вычисления «ручным» способом (с помощью калькулятора) и сравните затраты времени на решение и сложность реализации каждого метода.

2) Методом итераций найти положительный наибольший корень уравнения $x^3 + x = 1000$. $\varepsilon \le 10^{-4}$. $(\xi = 9{,}9667)$

3) Найти корень уравнения $x^3 - x - 1 = 0$, расположенный вблизи значения $x_0 = 1{,}3$. Точное значение корня 1,3247179.

4) Методом последовательных приближений и методом деления отрезка пополам решить систему уравнений:

$$x^2 + y = 3;$$

$$x + y^2 = 5.$$

Решение данной системы: $x = 1{,}0$; $y = 2{,}0$.

5) Методами бисекции и хорд определить первые 6 корней характеристического уравнения (ограничить вычисления четырьмя верными цифрами после запятой):

$$ctg\mu = \frac{\mu}{1{,}5}.$$

Значения корней для проверки:

$$\mu_1 = 0{,}9882; \quad \mu_2 = 3{,}5422; \quad \mu_3 = 6{,}5097;$$
$$\mu_4 = 9{,}5801; \quad \mu_5 = 12{,}6841; \quad \mu_6 = 15{,}8026.$$

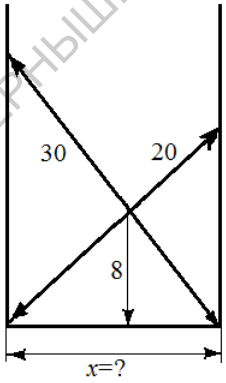
6) Для системы

$$\begin{cases} 2x^2 - xy - 5x + 1 = 0, \\ x + 3lg(x) - y^2 = 0 \end{cases}$$

найти положительные корни с четырьмя значащими цифрами.

Ответ: $\xi_x \approx 3{,}487; \ \xi_y = 2{,}262$.

12. Следующая задача (задача египетских жрецов) приводит к уравнению четвертого порядка.



Две лестницы, одна в 20, а другая в 30 футов длиной поставлены поперек улицы, как показано на рисунке. Лестницы опираются своими противоположными концами на противостоящие дома. Определить ширину улицы x, если точка пересечения лестниц находится на высоте 8 футов над землей.

Показать, что задача сводится к решению уравнения:

$$y^4 - 16y^3 + 500y^2 - 8000y + 32000 = 0,$$

тогда $x = \sqrt{400 - y^2}$.

ПРИЛОЖЕНИЕ Б

ПРОГРАММА РЕШЕНИЯ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МЕТОДОМ ПОСЛЕДОВАТЕЛЬНЫХ ИСКЛЮЧЕНИЙ ГАУССА С ВЫБОРОМ ВЕДУЩЕГО ЭЛЕМЕНТА

метки позиции 1 - 5	6	Операторы (позиции 7 – 72)
		subroutine gauss(num,n,a,b,ier,ip)
c		подпрограмма решения системы линейных
c		алгебраических уравнений с выбором ведущего элемента
c		ip – вектор ведущих элементов
c		
		real a(num,n),b(n)
		integer ip(n)
c		
c		a(num,n) – исходная матрица коэффициентов
c		b(n) – столбец свободных членов и решение
c		num–глобальная размерность матрицы A, задать в головной программе
c		n – требуемая размерность матрицы A
c		ier – параметр ошибки, ier=1–решение найдено
c		ier=0 и ier=–55 – матрица A вырожденная или плохо обусловленная
c		
		a1=0.
		t=0.
		do 2 j=1,n
		a1=a1+abs(b(j))
		do 1 i=1,n
		t=t+abs(a(i,j))
1		continue
2		continue
		a2=a1+t
		if(a2.lt.0.1e–20) go to 14
		ip(n)=1
		if(n.eq.1) go to 13
		nm1=n–1
		do 7 k=1,nm1

```
      kp1=k+1
      m=k
    do 3 i=kp1,n
      if(abs(a(i,k)).gt.abs(a(m,k))) m=i
3   continue
      ip(k)=m
      if(n.ne.k) ip(n)=–ip(n)
      t=a(m,k)
      a(m,k)=a(k,k)
      a(k,k)=t
     if(abs(t).le.0.1e–20) go to 7
    do 4 i=kp1,n
      a(i,k)=–a(i,k)/t
4    continue
    do 6 j=kp1,n
      t=a(m,j)
      a(m,j)=a(k,j)
      a(k,j)=t
     if(abs(t).le.0.1e–20) go to 6
    do 5 i=kp1,n
      a(i,j)=a(i,j)+a(i,k)*t
5        continue
6      continue
7   continue
     do 8 k=1,n
      if(abs(a(k,k)).lt.0.1e–20) go to 15
8    continue
    do 10 k=1,nm1
      kp1=k+1
      m=ip(k)
      t=b(m)
      b(m)=b(k)
      b(k)=t
    do 9 i=kp1,n
      b(i)=b(i)+a(i,k)*t
9    continue
10  continue
```

```
      do 12 kb=1,nm1
        km1=n–kb
        k=km1+1
        b(k)=b(k)/a(k,k)
        t=-b(k)
      do 11 i=1,km1
        b(i)=b(i)+a(i,k)*t
11    continue
12    continue
13   ier=1
        b(1)=b(1)/a(1,1)
        return
14   ier=0
      go to 16
15    ier=–55
16    do 17 k=1,n
        b(k)=0.
17    continue
        return
        end
```

Внимание! Исходная матрица коэффициентов и столбец свободных членов при выполнении подпрограммы не сохраняются!

Если матрица A вырождена или плохо обусловлена (ier=0 или ier=–55), вектору-решению b присваиваются нулевые значения.

ПРИЛОЖЕНИЕ В

ПОДПРОГРАММЫ-ФУНКЦИИ ВЫЧИСЛЕНИЙ ФУНКЦИЙ БЕССЕЛЯ ПЕРВОГО РОДА НУЛЕВОГО И ПЕРВОГО ПОРЯДКОВ

В представленных подпрограммах-функциях вычисления значений функций Бесселя для $x < 8{,}5$ производится по формулам (3.22), а для $x > 8{,}5$ – по асимптотическим разложениям Ганкеля [15, с. 220]. Для $x < 8{,}5$ вычисления с помощью рядов (3.22) производятся с погрешностью $0{,}1 \cdot 10^{-8}$ с выходом из цикла суммирования по условию выполнения теоремы Лейбница о сходимости абсолютно сходящихся знакопеременных рядов: ошибка не превосходит абсолютной величины первого из отбрасываемых слагаемых.

Метки Позиции 1 – 5	6	Операторы Позиции 7 - 72
		function bess0(x)
c		подпрограмма расчета функции Бесселя
c		первого рода нулевого порядка Jo(x)
c		
		if(x.gt.8.5) goto 3
		a=1.
		s=a
		do 1 n=1,40000
		y=x/2./n
		an=–a*y*y
		a=an
		s=s+an
		if(abs(an).lt.0.1e–08) goto 2
1		continue
2		bess0=s
		return
3		p0=1.+0.0703125/x/x*(–1.+1.595052082/x/x*(1.+
	+	5.1046875/x/x*(-1.+10.60965401/x/x*(1.+18.1126736/x/x*
	*	(–1.+27.61470168/x/x)))))
		q0=0.125/x*(-1.+0.5859375/x/x*(1.+33.10078125/x/x*(–1.+
	+	7.607514879/x/x*(1.+14.11132812/x/x*(–1.+22.61137784/x/x)))))
		s=x-0.785398162
		bess0=sqrt(0.636619773/x)*(p0*cos(s)–q0*sin(s))
		return
		end
c		

```
                function bess1(x)
c               подпрограмма расчета функции Бесселя
c               первого рода порядка один       J1(x)
c
                  if (x.gt.8.5) goto 3
                     x2=x/2.
                     a0=x2
                     s=a0
                 do 1 k=1,40000
                     j=k+1
                     ak=–a0*x2/k*x2/j
                     s=s+ak
                     if(abs(ak).lt.0.1e–08) goto 2
                     a0=ak
      1          continue
      2           bess1=s
                     return
      3          p1=1.+0.1171875/x/x*(1.+1.23046875/x/x*(–1.+4.6921875/x/x*
               * (1.+ 10.17438615/x/x*(–1.+17.6640625/x/x*(1.–27.15731532/x/x)))))
                   q1=0.375/x*(1.+0.2734375/x/x*(–1.+2.70703125/x/x*(1.+
               + 7.181919642/ /x/x*(-1.+13.66861979/x/x*(1.+22.16036931/x/x)))))
                     s=x–2.356194487
                  bess1=sqrt(0.636619773/x)*(p1*cos(s)-q1*sin(s))
                  return
                end
```

Для сравнения приводим подпрограммы-функции вычисления значений $J_0(x)$ и $J_1(x)$, полностью основанные на аппроксимации многочленами [8, с. 191].

Для $0 \le x \le 3{,}0$ применяются аппроксимации с погрешностями не хуже $5 \cdot 10^{-8}$, а для $x > 3{,}0$ погрешность не хуже $9 \cdot 10^{-8}$.

Метки Позиции 1 – 5	6	Операторы Позиции 7 - 72
		function bess0(x)
c		аргумент x>=0.
		if(x.gt.3.) goto 1
		r=x*x/9.
		y=((.21e-3*r-.39444e-2)*r+.444479e-1)*r
		z=(((y-.3163866)*r+1.2656208)*r–2.2499997)*r
		bess0=1.+z
		return

```
      1     r=3./x
            f0=(((((r*.14476e-3–.72805e-3)*r+.137237e-2)*r–
     +      .9512e-4)*r–.55274e-2)*r–.77e-6)*r+.79788456
            q=(((((.13558e-3*r–.29333e-3)*r–.54125e-3)*r+
     +      262573e-2)*r–.3954e-4)*r–.4166397e-1)*r–.78539816
            c=cos(q)/sqrt(x)
            bess0=f0*c
             return
            end
c
            function bess1(x)
c           аргумент x>=0.
            if(x.gt.3.) goto 1
            r=x*x/9.
            y=(((((r*.1109e-4-.31761e-3)*r+.443319e-2)*r–
     +      .3954289e-1)*r+.21093573)*r-.56249985)*r+.5
             bess1=y*x
             return
      1     r=3./x
             f=(((((.113653e-2–.20033e-3*r)*r–.249511e-2)*r+
     +      .17103e-3)*r+.1659667e-1)*r+.156e-5)*r+.79788456
            q=(((((.798244e-3–.29166e-3*r)*r+.74348e-3)*r–
     +      .637879e-2)*r+.565e-4)*r+.12499612)*r–2.35619449+x
            bess1=cos(q)*f/sqrt(x)
             return
            end
```