МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Э.В. Денисова

А.В. Кучер

ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

Учебно-методическое пособие

Санкт-Петербург

2010

СОДЕРЖАНИЕ

Глава 1
ПРАВИЛА ПРИБЛИЖЕННЫХ ВЫЧИСЛЕНИЙ И ОЦЕНКА ПОГРЕШНОСТЕЙ ПРИ ВЫЧИСЛЕНИЯХ

§ 1. Приближенные числа, их абсолютные и относительные погрешности

Расчеты, как правило, производятся с приближенными значениями величин — *приближенными числами.* Уже исходные данные для расчета обычно даются с некоторыми погрешностями; в процессе расчета еще накапливаются погрешности от округления, от применения приближенных формул и т. п. Разумная оценка погрешности при вычислениях позволяет указать оптимальное количество знаков, которые следует сохранять при расчетах, а также в окончательном результате.

Погрешность приближенного числа *a,* т. е. разность $a - a_0$ между ним и точным значением a_0, обычно неизвестна.

Под *оценкой погрешности* приближенного числа *a* понимают установление неравенства вида

$$|a - a_0| \le \Delta_a \tag{1.1}$$

Число Δ_a называется *абсолютной погрешностью приближенного числа a* (иногда употребляют термин «предельная абсолютная погрешность»). Это число определяется неоднозначно: его можно увеличить. Обычно стараются указать, возможно, меньшее число Δ_a, удовлетворяющее неравенству (1.1).

Абсолютные погрешности записывают не более чем с двумя-тремя значащими цифрами (при подсчете числа значащих цифр не учитывают нулей, стоящих слева; например, в числе 0,010030 имеется 5 значащих цифр). В приближенном числе *a* не следует сохранять те разряды, которые подвергаются округлению в его абсолютной погрешности Δ_a.

ПРИМЕР 1.1. Длина и ширина комнаты, измеренные с точностью до 1 *см,* равны a = 5,43 *м* и b = 3,82 *м.* Оценить погрешность в определении площади комнаты S = *ab* = 20,7426 м^2.

Решение. По условию задачи $\Delta_a = 0{,}01$ *м,* 0, $\Delta_b = 0{,}01$ *м.*

Крайние возможные значения площади равны

$$(a + 0{,}01)\,(b + 0{,}01) = 20{,}8352\ м^2,$$
$$(a - 0{,}01)\,(b - 0{,}01) = 20{,}6502\ м^2;$$

сравнивая их с подсчитанным выше значением *S,* получаем оценку

$$|S - S_0| \le 0{,}0926,$$

что дает возможность указать абсолютную погрешность числа S в виде $\Delta_S = 0{,}0926\ м^2$.

Здесь разумно округлить значение Δ_S, например, так: $\Delta_S = 0{,}093\ м^2$ при $\Delta_S = 0{,}10\ м^2$ (абсолютные погрешности округляют в большую сторону). При этом приближенное значение площади можно записать в виде $S = 20{,}743\ м^2$, или S = 20,74 $м^2$, или даже S = 20,7 $м^2$.

ПРИМЕР 1.2. В некоторую вычислительную машину мы можем ввести числа только с тремя значащими цифрами. С какой точностью мы можем ввести в нее числа π и 1/3?

Решение. Полагаем $\pi \approx 3{,}14 = a$ вместо π = 3,141592..., погрешность числа *a* можно оценить числом $\Delta_a = 0{,}0016$. Полагаем $1/3 \approx 0{,}333 = b$; погрешность числа *b* можно оценить числом $\Delta_b = 0{,}00034$ или $\Delta_b = 0{,}0004$.

Относительной погрешностью $\delta_S = \dfrac{0{,}0926}{20{,}7426} = \dfrac{926}{207426} = 0{,}0045 = 0{,}45\%$. *приближенного числа a* называется отношение его абсолютной погрешности Δ_a к абсолютной величине числа *a,* т. е.

$$\frac{\Delta_a}{|a|} \tag{1.2}$$

$(a \ne 0)$. Относительная погрешность обычно выражается в процентах, и ее принято записывать не более чем с двумя-тремя значащими цифрами (знаками).

Иногда под относительной погрешностью понимают отношение, $\dfrac{\Delta_a}{|a_0|}$, где a_0 — точное (но неизвестное) значение числа; если относительная погрешность числа *a* не превышает 5%, то различие между

отношениями $\frac{\Delta_a}{|a|}$ и $\frac{\Delta_a}{|a_0|}$ сказывается только на втором знаке погрешности, что не существенно.

П р и м е р 1.3. Определить относительную погрешность числа S в примере 1.1.

Р е ш е н и е . S = 20,7426, Δ_S = 0,0926, поэтому

$$\delta_S = \frac{0,0926}{20,7426} = \frac{926}{207426} = 0,0045 = 0,45\%.$$

Во многих технических приложениях принято характеризовать точность приближенных чисел их относительной погрешностью.

Относительная погрешность приближенного числа связана с количеством его верных знаков. *Количество верных знаков* числа отсчитывается от первой значащей цифры числа до первой значащей цифры его абсолютной погрешности: например, число S = 20,7426 с абсолютной погрешностью Δ_S = 0,0926 имеет три верных знака (2, 0, 7); остальные знаки — сомнительные.

Ориентировочно можно считать, что наличие только одного верного знака соответствует относительной погрешности порядка 10%, двух верных знаков — погрешности порядка 1%, трех верных знаков — погрешности порядка 0,1% и т. д.

В математических таблицах все числа округлены до верных знаков, причем абсолютная погрешность не превосходит половины единицы последнего оставленного разряда. Например, если в таблице указано e=2,718, то абсолютная погрешность не превосходит $0,5\cdot10^{-3}$.

В окончательных результатах вычислений обычно оставляют, кроме верных, один сомнительный знак.

В промежуточных результатах вычислений обычно сохраняют два-три сомнительных знака, чтобы не накапливать лишних погрешностей от округлений.

П р и м е р 1.4. Округлить число S = 20,7426 в примере 1.1 до верных знаков.

Р е ш е н и е . Так как в числе S три верных знака, то естественно записать S=20,7.

Однако при этом к абсолютной погрешности Δ_S=0,0926 приходится добавить еще величину 0,0426, отброшенную при округлении. Новая абсолютная погрешность Δ_S = 0,136 заставляет считать сомнительным уже третий знак числа *S,* и, следовательно, число 5 приходится округлять до двух знаков:

S=21 (Δ_S= 0,44 < 0,5).

Этот пример показывает, что округление результатов расчета до верных знаков не всегда целесообразно.

Примечание. В этом примере, как это обычно принято, применено *правило дополнения при округлении:* если первая отбрасываемая цифра больше или равна 5, то последняя сохраняемая цифра увеличивается на 1.

З А Д А Ч И

1. Округляя следующие числа до трех значащих цифр, определить абсолютную Δ и относительную δ погрешности полученных приближенных чисел.

а) 2,1514, б) 0,16152, в) 0,01204, г) 1,225, д) -0,0015281, е) -392,85, ж) 0,1545, з) 0,003922, и) 625,55, к) 94,525.

2. Определить абсолютную погрешность следующих приближенных чисел по их относительным погрешностям.

а) a=13267, δ=0,1%, б) a=2,32, δ= 0,7%, в) a = 35,72, δ= 1%, г) a =0,896, δ=10%, д) a= 232,44, δ=1%.

3. При измерении некоторых углов получили числа α_1 = 21°37'3", α_2 = 45°, α_3=1°10", α_4 = 75°20'44".

Определить относительные погрешности чисел $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, полагая абсолютную погрешность измерения равной 1".

4. Определить количество верных знаков в числе *x,* если известна его абсолютная погрешность.

а) $x = 0,3941$, $\Delta_x = 0,25\cdot10^{-2}$ б) x = 0,1132, $\Delta_x = 0,1\cdot10^{-3}$, в) $\Delta_x = 38,2543$, $\Delta_x = 0,27\cdot10^{-2}$,

г) x = 293,481, Δ_x = 0,1, д) x = 2,325, $\Delta_x = 0,1\cdot10^{-1}$, е) x = 14,00231, $\Delta_x = 0,1\cdot10^{-3}$,

ж) x = 0,0842, $\Delta_x = 0,15\cdot10^{-2}$, з) x = 0,00381, $\Delta_x = 0,1\cdot10^{-4}$, и) $x = -32,285$, $\Delta_x = 0,2\cdot10^{-2}$,

к) $x = -0,2113$, $\Delta_x = 0,5\cdot10^{-2}$.

5.Определить количество верных знаков в числе *a,* если известна его относительная погрешность.

а) $a = 1,8921$, $\delta_a = 0,1\cdot10^{-2}$ б) $a = 0,2218$, $\delta_a = 0,2\cdot10^{-1}$, в) $a = 22,351$, $\delta_a = 0,15$,

г) $a = 0,02425$, $\delta_a = 0,5\cdot10^{-2}$, д) $a = 0,000135$, $\delta_a = 0,15$, е) $a = 9,3598$, $\delta_a = 0,1\%$,

ж) $a = 0,11452$, $\delta_a = 10\%$, з) $a = 48361$, $\delta_a = 1\%$, и) $a = 592,8$, $\delta_a = 2\%$, к) $a = 14,9360$, $\delta_a = 1\%$.

§ 2. Устойчивость. Корректность. Сходимость

1. Устойчивость. Рассмотрим погрешности исходных данных. Поскольку это так называемые неустранимые погрешности и вычислитель не может с ними бороться, то нужно хотя бы иметь представление об их влиянии на точность окончательных результатов. Конечно, мы вправе надеяться на то, что погрешность результатов имеет порядок погрешности исходных данных. Всегда ли это так? К сожалению, нет. Некоторые задачи весьма чувствительны к неточностям в исходных данных. Эта чувствительность характеризуется так называемой устойчивостью.

Пусть в результате решения задачи по исходному значению величины x находится значение искомой величины y. Если исходная величина имеет абсолютную погрешность Δx, то решение имеет погрешность Δy. Задача называется *устойчивой* но исходному параметру x, если решение y непрерывно от него зависит, т. е. малое приращение исходной величины Δx приводит к малому приращению искомой величины Δy. Другими словами, малые погрешности в исходной величине приводят к малым погрешностям в результате расчетов.

Отсутствие устойчивости означает, что даже незначительные погрешности в исходных данных приводят к большим погрешностям в решении или вовсе к неверному результату. О подобных неустойчивых задачах также говорят, что они *чувствительны* к погрешностям исходных данных.

Примером такой задачи является отыскание действительных корней многочленов вида

$$(x-a)^n = \varepsilon,\ \ 0 < \varepsilon << 1.$$

Изменение правой части па величину порядка ε приводит к погрешности корней порядка $\varepsilon^{1/n}$.

Интересной иллюстрацией неустойчивой задачи является так называемый *пример Уилкинсона.* Рассматривается многочлен

$$P(x) = (x-1)(x-2)...(x-20) = x^{20} - 210x^{19} + ...$$

Очевидно, что корнями этого многочлена являются $x_1 = 1, x_2 = 2, ..., x_{20} = 20$.

Предположим, что один из коэффициентов многочлена вычислен с некоторой малой погрешностью. Например, коэффициент -210 при x^{19} увеличим на 2^{-23} (около 10^{-7}). В результате вычислений даже с точностью до 11 значащих цифр получим существенно другие значения корней. Приведем для наглядности эти значения, округленные до трех знаков:

$x_1 = 1{,}00,$	$x_9 = 8{,}92,$
$x_2 = 2{,}00,$	$x_{10,11} = 10{,}1 \pm 0{,}644i,$
$x_3 = 3{,}00,$	$x_{12,13} = 11{,}8 \pm 1{,}65i,$
$x_4 = 4{,}00,$	$x_{14,15} = 14{,}0 \pm 2{,}52i,$
$x_5 = 5{,}00,$	$x_{16,17} = 16{,}7 \pm 2{,}81i,$
$x_6 = 6{,}00,$	$x_{18,19} = 19{,}5 \pm 1{,}94i,$
$x_7 = 7{,}00,$	$x_{20} = 20{,}8$
$x_8 = 7{,}01,$	

Таким образом, изменение коэффициента -210 при x^{i9} на -210 $+10^{-7}$ привело к тому, что половина корней стали комплексными. Причина такого явления — неустойчивость самой задачи; вычисления выполнялись очень точно (11 разрядов), а погрешности округлений не могли привести к таким последствиям.

2. Корректность. Задача называется *поставленной корректно,* если для любых значений исходных данных из некоторого класса ее решение существует, единственно и устойчиво по исходным данным.

Рассмотренные выше примеры неустойчивых задач являются некорректно поставленными. Применять для решения таких задач численные методы, как правило, нецелесообразно, поскольку возникающие в расчетах погрешности округлений будут сильно возрастать в ходе вычислений, что приведет к значительному искажению результатов.

Вместе с тем отметим, что в настоящее время развиты методы решения некоторых некорректных задач. Это в основном так называемые *методы регуляризации.* Они основываются на замене исходной задачи корректно поставленной задачей. Последняя содержит некоторый параметр, при стремлении которого к нулю решение этой задачи переходит в решение исходной задачи.

3. Неустойчивость методов. Иногда при решении корректно поставленной задачи может оказаться неустойчивым метод ее решения. В частности, по этой причине при вычислении синуса большого аргумента был получен результат, не имеющий смысла.

Рассмотрим еще один пример неустойчивого алгоритма. Достроим численный метод вычисления интеграла

$$I_n = \int_0^1 x^n e^{x-1} dx\ ,\ \text{n} = 1, 2, \ldots$$

Интегрируя по частям, находим

$$I_1 = \int_0^1 xe^{x-1}dx = xe^{x-1}\Big|_0^1 - \int_0^1 e^{x-1}dx = \frac{1}{e}$$

$$I_2 = \int_0^1 x^2 e^{x-1}dx = x^2 e^{x-1}\Big|_0^1 - 2\int_0^1 xe^{x-1}dx = 1 - 2I_{n-1}$$

……

$$I_2 = \int_0^1 x^n e^{x-1}dx = x^n e^{x-1}\Big|_0^1 - n\int_0^1 x^{n-1}e^{x-1}dx = 1 - nI_1$$

.

Пользуясь полученным рекуррентным соотношением, вычисляем

$I_1 = 367879$, $I_6 = 0{,}127120$,
$I_2 = 0{,}263242$, $I_7 = 0{,}110160$,
$I_3 = 0{,}207274$, $I_8 = 0{,}118720$,
$I_4 = 0{,}170904$, $I_9 = -0{,}0684800$.
$I_5 = 0{,}145480$,

Значение интеграла I_9 не может быть отрицательным, поскольку подынтегральная функция на всем отрезке интегрирования [0, 1] неотрицательна. Исследуем источник погрешности. Видим, что округление в I_1 дает погрешность, равную примерно лишь $4.4*10^{-7}$. Однако на каждом этапе эта погрешность умножается на число, модуль которого больше единицы (-2, -3, ..., -9), что в итоге дает 9. Это и приводит к результату, не имеющему смысла. Здесь снова причиной накопления погрешностей является алгоритм решения задачи, который оказался неустойчивым.

Численный алгоритм (метод) называется *корректным* в случае существования и единственности численного решения при любых значениях исходных данных, а также в случае устойчивости этого решения относительно погрешностей исходных данных.

4. Понятие сходимости. При анализе точности вычислительного процесса одним из важнейших, критериев является *сходимость* численного метода. Она означает близость получаемого численного решения задачи к истинному решению. Строгие определения разных оценок близости могут быть даны лишь с привлечением аппарата функционального анализа. Здесь мы ограничимся некоторыми понятиями сходимости, необходимыми для понимания последующего материала.

Рассмотрим понятие *сходимости итерационного процесса.* Этот процесс состоит в том, что для решения некоторой задачи и нахождения искомого значения определяемого параметра (например, корня нелинейного уравнения) строится метод последовательных приближений. В результате многократного повторения этого процесса (или *итераций)* получаем последовательность значений $x_1, x_2, ..., x_n$. Говорят, что эта последовательность *сходится* к точному решению $x = a$, если при неограниченном возрастании числа итераций предел этой последовательности существует и равен a: $\lim\limits_{n\to\infty} x_n = a$. В этом случае имеем сходящийся численный метод.

Другой подход к понятию сходимости используется в методах дискретизации. Эти методы заключаются в замене задачи с непрерывными параметрами на задачу, в которой значения функций вычисляются в фиксированных точках. Это относится, в частности, к численному интегрированию, решению дифференциальных уравнений и т. п. Здесь под *сходимостью метода* понимается стремление значений решения дискретной модели задачи к соответствующим значениям решения исходной задачи при стремлении к нулю параметра дискретизации (например, шага интегрирования).

При рассмотрении сходимости важными понятиями являются ее вид, порядок и другие характеристики. С общей точки зрения эти понятия рассматривать нецелесообразно; к ним будем обращаться при изучении численных методов.

Таким образом, для получения решения задачи с необходимой точностью ее постановка должна быть корректной, а используемый численный метод должен обладать устойчивостью и сходимостью.

§ 3. Сложение и вычитание приближенных чисел

1. Абсолютная погрешность алгебраической суммы нескольких приближенных чисел равна сумме абсолютных погрешностей слагаемых: если

$$S = a_1 + a_2 + \ldots + a_n, \text{ то}$$

$$\Delta_S = \Delta_{a_1} + \Delta_{a_2} + \ldots + \Delta_{a_n}. \qquad (1.3)$$

При большом количестве слагаемых оценка абсолютной погрешности суммы по формуле (1.3) оказывается сильно завышенной, так как обычно происходит частичная компенсация погрешностей разных знаков.

Если среди слагаемых имеется одно число, абсолютная погрешность которого значительно превосходит

абсолютные погрешности остальных слагаемых, то абсолютная погрешность суммы считается равной этой наибольшей погрешности. При этом в сумме целесообразно сохранять столько десятичных знаков, сколько их в слагаемом с наибольшей абсолютной погрешностью.
Покажем на примере, как производится сложение приближенных чисел и оценка погрешности.

Пример 1.5. Найти сумму приближенных чисел 0,348; 0,1834; 345,4; 235,2; 11,75**;** 9,27; 0,0849; 0,0214; 0,000354, считая в них все знаки верными, т. е. считая, что абсолютная погрешность каждого слагаемого не превосходит половины единицы младшего оставленного разряда.
Решение. Наибольшую абсолютную погрешность $\Delta = 0{,}05$ имеют два числа: 345,4 и 235,2. Поэтому можно считать, что абсолютная погрешность суммы составляет $2\Delta = 0{,}10$. Так как количество слагаемых невелико, то в расчетах сохраняем только один запасной знак, т. е. округляем слагаемые до 0,01:

345,4
235,2
11,75
9,27
0,35
0,18
0,08
0,02
0,00
S = 602,25.

В окончательном результате запасной знак отбрасываем:

S = 602,2

При этом к указанной выше абсолютной погрешности 0,10 добавляем погрешность округления 0,05, что дает

$$\Delta_S = 0{,}15 \text{ или } \Delta_S = 0{,}2.$$

Заметим, что в этом примере полный учет всех погрешностей слагаемых по формуле (1.3) только усложнил бы расчет, не внося существенных уточнений в результат.
2. Относительная погрешность δ_S суммы нескольких чисел одного и того же знака заключена между наименьшей и наибольшей из относительных погрешностей слагаемых:

$$\min \delta_{a_k} \le \delta_S \le \max \delta_{a_k} \ (a_k > 0,\ k = 1, 2, \ldots, n) \qquad (1.4)$$

Пример 1.6. Оценить относительную погрешность суммы чисел в примере 1.5 и сравнить ее с относительными погрешностями слагаемых.
Решение. Относительная погрешность суммы S равна

$$\delta_S = \frac{0{,}10}{602{,}25} = 0{,}017\%$$

Относительные погрешности слагаемых составляют соответственно

$\frac{0{,}5}{348} = 0{,}15\%$; $\frac{0{,}5}{1834} = 0{,}027\%$; $\frac{0{,}5}{3454} = 0{,}015\%$

$\frac{0{,}5}{2352} = 0{,}022\%$; $\frac{0{,}5}{1175} = 0{,}043\%$; $\frac{0{,}5}{927} = 0{,}054\%$;

$\frac{0{,}5}{849} = 0{,}059\%$; $\frac{0{,}5}{214} = 0{,}24\%$; $\frac{0{,}5}{354} = 0{,}15\%$

ЗАДАЧИ

1. Найти суммы приближенных чисел и указать их погрешность
а) 0,145 + 321 +78,2 (все знаки верные),
б) 0,301 + 193,1 + 11,58 (все знаки верные),
в) 398,5 - 72,28 + 0,34567 (все знаки верные),
г) $x_1 + x_2 - x_3$, где $x_1 = 197{,}6$, $\Delta_{x_1} = 0{,}2$, $x_2 = 23{,}44$, $\Delta_{x_2} = 0{,}22$, $x_3 = 201{,}55$, $\Delta_{x_3} = 0{,}17$.

§ 4. Умножение и деление приближенных чисел

При умножении и делении приближенных чисел их относительные погрешности складываются (а не абсолютные). Относительная погрешность выражения

$$r = \frac{a_1 a_2 \ldots a_m}{b_1 b_2 \ldots b_n} \qquad (1.5)$$

Оценивается величиной

$$\delta_r = \delta_{a_1} + \delta_{a_2} + \ldots + \delta_{a_m} + \delta_{b_1} + \delta_{b_2} + \ldots + \delta_{b_n} \qquad (1.6)$$

При большом числе $m+n$ выгоднее пользоваться *статистической оценкой,* учитывающей частичную компенсацию погрешностей разных знаков: если все числа a_i, b_j ($i = 1, 2, \ldots, m;\ j=1,2, \ldots, n$) имеют примерно одинаковую относительную погрешность δ, то относительная погрешность выражения (1.5) принимается равной

$$\delta_r = \sqrt{3(n+m)}\delta \qquad (n+m>10) \qquad (1.7)$$

Если у одного из чисел !!!a_h *bj !!!* относительная погрешность значительно превышает относительные погрешности остальных чисел, то относительная погрешность выражения (1.3) считается равной этой наибольшей погрешности. При этом в результате целесообразно сохранять столько знаков (значащих цифр), сколько их в числе с наибольшей относительной погрешностью.

Абсолютная погрешность выражения (1.5) вычисляется по его относительной погрешности:

$$\Delta_r = |r|\delta_r .$$

Покажем на примере, как производится умножение и деление приближенных чисел и *оценка* погрешности результата.

П р и м е р 1.7. Вычислить выражение

$$r = \frac{3,2 \cdot 356,7 \cdot 0,04811}{7,1948 \cdot 34,56},$$

считая, что все числа даны с верными знаками, т. е. что их абсолютные погрешности не превосходят половины единицы младшего оставленного разряда.

Р е ш е н и е. Наибольшую относительную погрешность имеет число а = 3,2, которое содержит всего два верных знака (против четырех-пяти верных знаков в остальных числах):

$$\delta_a = \frac{0,5}{32} = 1,6\% .$$

Поэтому можно считать, что относительная погрешность результата составляет δ_a= 1,6%, т. е. что результат содержит не более двух верных знаков. Так как количество'данных чисел невелико, то в расчетах сохраняем один запасной знак, округляя все числа до трех знаков:

$$r = \frac{3,2 \cdot 357 \cdot 0,0481}{7,19 \cdot 34,6} = 0,221$$

Абсолютную погрешность результата вычисляем по его относительной погрешности и найденному численному значению:

$$\Delta_r = r\delta_r = 0,221 \cdot 0,016 = 0,0036 .$$

Округляя результат до верных знаков, отбрасываем запасной знак и получаем

$$r = 0,22$$

с абсолютной погрешностью $\Delta_r < 0,005$.

ЗАДАЧИ

1. Найти произведения приближенных чисел и указать их погрешности (считая в исходных данных все знаки верными).

а) 3,49*8,6, б) 25,1*1,743, в) 0,02*16,5, г) 0,253*654*83,6, д) 1,78*9,1*1,183, е) 482,56*7256*0,0052.

2. Найти частное приближенных чисел.

а) 5,684/5,032, б) 0,144/1,2, в) 216/4, г) 726,676/829, д) 754,9367/36,5, е) 7,3/4491.

Стороны прямоугольника равны (4,02 ± 0,01) *м,* (4,96 ± 0,01) *м.* Вычислить площадь прямоугольника.

Катеты, прямоугольного треугольника равны (12,10 + 0,01) *см,* (25,21 + 0,01) *см.* Вычислить тангенс угла, противолежащего первому катету.

При измерении радиуса *R* круга с точностью до 0,5 *см* получилось число 12 *см.* Найти абсолютную и относительную погрешности при вычислении площади круга.

Каждое ребро куба, измеренное с точностью до 0,02 *см,* оказалось равным 8 *см.* Найти абсолютную и относительную погрешности при вычислении объема куба.

Высота *h* и радиус основания *R* цилиндра измерены с точностью до 0,5%. Какова предельная относительная погрешность при вычислении объема цилиндра?

§ 5. Погрешности вычисления значений функции

1. Функции одной переменной. Абсолютная погрешность дифференцируемой функции $y = f(x)$, вызываемая достаточно малой погрешностью аргумента A_x, оценивается величиной

$$\Delta_y = \left|f'(x)\right|\Delta_x \tag{1.8}$$

Если значения функции *f(x)* положительны, то для относительной погрешности имеет место оценка

$$\delta_y = \frac{\left|f'(x)\right|}{f(x)}\Delta_x = \left|\left[\ln f(x)\right]'\right|\Delta_x \tag{1.9}$$

В частности, для основных элементарных функций получаем следующие правила.

а) Степенная функция $y = x^a$. Абсолютная погрешность степенной функции равна

$$\Delta_y = ax^{a-1}\Delta_x \tag{1.10}$$

Относительная погрешность степенной функции равна

$$\delta_y = \left|a\right|\delta_x \tag{1.11}$$

Например, относительная погрешность квадрата x^2 вдвое больше относительной погрешности основания x, относительная погрешность квадратного корня $\sqrt{x}$ вдвое меньше относительной погрешности подкоренного числа *x,* относительная погрешность обратной величины 1/x равна относительной погрешности самого числа x.

б) Показательная функция $y = a^x \ (a > 0)$. Абсолютная погрешность показательной функции равна

$$\Delta_y = a^x \ln a \cdot \Delta_x \tag{1.12}$$

Относительная погрешность показательной функции равна

$$\delta_y = \Delta_x \ln a \tag{1.13}$$

Заметим, что здесь относительная погрешность функции пропорциональна абсолютной погрешности аргумента. Для функции $y = e^x$ отсюда получаем

$$\delta_y = \Delta_x \tag{1.14}$$

в) Логарифмическая функция $y = \ln x$. Абсолютная погрешность натурального логарифма числа равна относительной погрешности самого числа:

$$\Delta_y = \frac{1}{x}\Delta_x = \delta_x \tag{1.15}$$

Для десятичного логарифма $y = \lg x$ имеем

$$\Delta_y = 0,4343\delta_x \tag{1.16}$$

откуда следует, что при расчетах с числами, имеющими *m* верных знаков, надо пользоваться *(m*+1)-значными таблицами логарифмов.

г) Тригонометрические функции. Абсолютные погрешности синуса и косинуса не превосходят абсолютных погрешностей аргумента:

$$\Delta_{\sin x} = \left|\cos x\right|\Delta_x \le \Delta_x \qquad \Delta_{\cos x} = \left|\sin x\right|\Delta_x \le \Delta_x \tag{1.17}$$

Абсолютная погрешность тангенса и котангенса всегда больше абсолютной погрешности аргумента:

$$\Delta_{tgx} = (1+tg^2x)\Delta_x \ge \Delta_x, \quad \Delta_{ctgx} = (1+ctg^2x)\Delta_x \ge \Delta_x \tag{1.18}$$

П р и м е р 1.8. Диаметр круга, измеренный с точностью до 1 *мм,* оказался равным *d* = 0,842 *м.* Вычислить площадь круга.

Р е ш е н и е . Площадь круга $S = \pi d^2/4$. Так как число π мы можем взять для расчета с любой точностью, то погрешность вычисления площади определяется погрешностью вычисления. Относительная погрешность d^2 равна

$$\delta_{d^2} = 2\delta_d = 2\cdot\frac{1}{842} = 0,24\%.$$

Чтобы при округлении числа π не увеличить относительную погрешность

$$\delta_S = \delta\left(\frac{\pi}{4}\right) + 2\delta_d,$$

надо взять число π по крайней мере с четырьмя верными **знаками,** еще лучше с пятью. Тогда получим

$$S = \frac{3,1416}{4}\cdot 0,842^2\ м^2 = 0,7854\text{-}0,7090\ м^2 = 0,5568\ м^2.$$

Абсолютная погрешность результата составляет

$$\Delta_S = S\delta_S = 0,557\cdot 0,0024 = 0,0014.$$

Округляем результат до трех знаков (отбрасывая запасной знак и пользуясь правилом дополнения):

$$S = 0,557\ м^2, \qquad \Delta_S = 0,002.$$

П р и м е р 1.9. Угол x = 25°20' измерен с точностью до 1'. Определить $\sin x$ и его абсолютную

погрешность.

Р е ш е н и е . Вычислим сначала абсолютную погрешность $\sin x$ по формуле (1.17) для этого надо еще перевести 1' в радианы: 1'=0,000291 — и подсчитать

$$\Delta_{\sin x} = \cos x \cdot \Delta_x = \cos 25^\circ 20' = 0,4279$$

Поэтому для вычисления *sin(x)* надо взять четырехзначные таблицы тригонометрии ческих функций, что дает sin *x* = sin 25°20' = 0,4279.

2. **Функции нескольких переменных.** Абсолютная погрешность дифференцируемой функции $y = f(x_1, x_2, \dots x_n)$, вызываемая достаточно малыми погрешностями Δ_{x_1}, Δ_{x_2}, ..., Δ_{x_n} аргументов x_1, x_2, ..., x_n, оценивается величиной

$$\sqrt{y} = \sqrt{4,4} = 2,10 \tag{1.19}$$

Если значения функции положительны, то для относительной погрешности имеет место оценка

$$\delta_y = \sum_{i=1}^{n} \frac{1}{f}\left|\frac{\partial f}{\partial x_i}\right| \Delta_{x_i} = \sum_{i=1}^{n} \left|\frac{\partial \ln f}{\partial x_1}\right| \tag{1.20}$$

П Р И М Е Р 1.10. Вычислить значение функции $u = xy^2z^3$, если $x = 37,1$, $y = 9,87$, $z = 6,052$, причем $\Delta_x = 0,3$, $\Delta_y = 0,11$, $\Delta_z = 0,016$.

Р е ш е н и е . Здесь относительные погрешности аргументов равны

$$\delta_x = \frac{3}{371} = 0,81\%, \qquad \delta_y = \frac{11}{987} = 1,12\%, \qquad \delta_z = \frac{16}{6052} = 0,26\%.$$

Относительная погреишость функции равна

$$\delta_u = \delta_x + 2\delta_y + 3\delta_z = 3,8\%;$$

поэтому значение функции следует вычислять не более чем с двумя-тремя знаками:

$$u = 801 \cdot 10^3$$

(нельзя писать 801 000, это имело бы другой смысл). Абсолютная погрешность при этом составляет

$$\Delta_u = u\delta_u = 801 \cdot 10^3 \cdot 0,038 = 30 \cdot 10^3.$$

Здесь целесообразно округлить результат до двух знаков:

$$u = 8,0 \cdot 10^5, \qquad \Delta_u = 0,3 \cdot 10^5.$$

П Р И М Е Р 1.11. Вычислить значение $z = \ln(10,3 + \sqrt{4,4})$, считая верными все знаки приближенных чисел x=10,3 и y==4,4.

Р е ш е н и е . Число *y* имеет относительную погрешность $\delta_y = \frac{0,5}{44} = 1,2\%$, поэтому $\sqrt{y}$ имеет относительную погрешность 0,6% и его следует записать с тремя знаками:

$$\sqrt{y} = \sqrt{4,4} = 2,10;$$

при этом абсолютная погрешность этого корня равна $\Delta_{\sqrt{y}} = 2,10 \cdot 0,006 = 0,013$.

Абсолютная погрешность суммы $x + \sqrt{y} = 10,3 + 2,10 = 12,4$ оценивается величиной 0,05 + 0,013 = 0,063, ее относительная погрешность равна $\frac{0,63}{124} = 0,5\%$.

По формуле (1.15) такова же будет абсолютная погрешность натурального логарифма, т. е. $\Delta_z = 0,005$. Поэтому $z = \ln(10,3 + 2,10) = \ln 12,40 = 2,517$.

Здесь результат имеет три верных знака; округление до верных знаков нецелесообразно, так как при этом надо писать значение Δ_z с учетом погрешности округления:

$$z = 2,52, \qquad \Delta_z = 0,008.$$

ЗАДАЧИ

1. Углы *x* измерены с предельной абсолютной погрешностью Δ_x. Определить абсолютную и относительную погрешности функций *y = sinx, y =* cos*x* и *y=tgx*. Найти по таблицам значения функций, сохранив в результате лишь верные цифры.

а) $x = 11^\circ 20'$, $\Delta_x = 1'$, б) x = 48°42'31", Δx = 5", в) x = 45°, $\Delta_x = 1'$, г) x = 50°10', Δx = 0,05°,

д) x = 0,45, $\Delta x = 0{,}5\cdot 10^{-2}$, е) x = 1,115, $\Delta x = 0{,}1\cdot 10^{-3}$.

2. Для следующих функций вычислить значения при указанных значениях x и указать абсолютную и относительную погрешности результатов.

а) $y = x^3 \sin x$ при $x = \sqrt{2}$, полагая $\sqrt{2} \approx 1{,}414$,

б) $y = x \ln x$ при $x = \pi$, полагая $\pi \approx 3{,}142$,

в) $y = e^x \cos x$ при $x = \sqrt{3}$, полагая $\sqrt{3} \approx 1{,}732$.

3. Для следующих функций вычислить значения при указанных значениях переменных. Указать абсолютную и относительную по грешности результатов, считая все знаки исходных данных верными.

а) $u = \ln(x_1 + x_2^2)$, $x_1 = 0{,}97$, $x_2 = 1{,}132$,

б) $u = \dfrac{x_1 + x_2^2}{x_3}$, $x_1 = 3{,}28$, $x_2 = 0{,}932$, $x_3 = 1{,}132$,

в) $u = x_1x_2 + x_1x_3 + x_2x_3$, $x_1 = 2{,}104$, $x_2 = 1{,}935$, $x_3 = 0{,}845$.

4. Определить относительную погрешность при вычислении полной поверхности усеченного конуса, если радиусы его оснований R и r и образующая /, измеренные с точностью до 0,01 *см,* равны

R =23,64 *см,* r =17,31 *см,* l =10,21 *см.*

§ 6. Определение допустимой погрешности аргументов по допустимой погрешности функции

Эта задача имеет однозначное решение только для функции одной переменной $y = f(x)$: если эта функция дифференцируема и $f'(x) \neq 0$, то

$$\Delta_x = \frac{1}{|f'(x)|}\Delta_y \tag{1.21}$$

Для функции нескольких переменных $y=f(x_1, x_2, ..., x_n)$ задача решается только при введении каких-либо дополнительных ограничений. Например, если значение одного из аргументов значительно труднее измерить или вычислить с большой точностью, чем значения остальных аргументов, то погрешность именно этого аргумента надо согласовать с требуемой погрешностью функции.

Если значения всех аргументов можно одинаково легко определить с любой точностью, то обычно применяют *принцип равных влияний,* считая, что в формуле (4.12)!!!(в источнике ссылается на формулу, которая 1.19)!!!! все слагаемые $\left|\dfrac{\partial f}{dx_i}\right|\Delta_{x_i}$ равны между собой; это дает формулу

$$\Delta_{x_i} = \frac{\Delta_y}{n\left|\dfrac{\partial f}{dx_i}\right|} \qquad (i = 1, 2, \ ..., n) \tag{1.22}$$

На практике часто встречаются задачи промежуточного типа между указанными крайними случаями. Мы рассмотрим соответствующие примеры.

Пример 1.12. С какой точностью следует измерить угол x в первой четверти, чтобы получить значение sinx с пятью верными знаками?

Решение. Если известно, что угол $x > 6°$, так что $\sin x > 0{,}1$, то надо определить Δ_x так, чтобы выполнялось неравенство $\Delta_{\sin x} < 0{,}5\cdot 10^{-5}$. Для этого в соответствии с формулой (1.17) достаточно взять $\Delta_x < 0{,}5\cdot 10^{-5}$, т. е. измерить угол x с точностью до 1". Если, сверх того, известно, что угол $x > 60°$ и, значит, $\cos x < 0{,}5$, то стоит воспользоваться формулой (1.21), откуда $\Delta_x = \dfrac{1}{\cos x}\Delta_{\sin x} > 2\cdot 0{,}5\cdot 10^{-5} = 10^{-5}$,

т. е. достаточно измерить угол x с точностью всего до 2".

Но если угол $x < 6°$, например, $1° < x < 6°$, то $0{,}01 < \sin x < 0{,}1$ и для обеспечения пяти верных знаков в значении sin л: придется обеспечить неравенство $\Delta_{\sin x} < 0{,}5\cdot 10^{-6}$, для чего придется измерять угол x с точностью до 0,1".

Пример 1.13. С какой точностью следует определить радиус основания R и высоту H цилиндрической банки, чтобы ее вместимость можно было определить с точностью до 1%?

Решение. В формуле $V = \pi R^2 H$ число я можно взять с любым числом верных знаков, так что его погрешность не скажется на результате, и поэтому можно считать $\delta_V = 2\delta_R + \delta_H$. Если можно обеспечить

любую точность определения *R* и *H,* то можно воспользоваться принципом равных влияний, откуда па долю $2\delta_R$ и δ_H приходится по 0,5%. Таким образом, но принципу равных влияний надо определить радиус с относительной погрешностью 0,25%, а высоту — с относительной погрешностью 0,5%. На практике чаще встречаются такие случаи, когда, наоборот, радиус банки определяется с меньшей точностью, чем высота. Например, если радиус определяется с точностью вдвое меньшей, чем высота, то полагаем $\delta_R = 2\delta_H$ и из условия

$$2\delta_R + \delta_H = 5\delta_H = 1\%$$

находим

$$\delta_H = 0,2\%, \quad \delta_R = 0,4\%.$$

Что касается числа π, то во всех указанных случаях надо брать его с относительной погрешностью порядка 0,01%, чтобы эту погрешность можно было не учитывать в окончательном результате. Это означает, что можно положить π = 3,142 с относительной погрешностью $\frac{0,4}{3142} = 0,013\%$, но не следует полагать π=3,14 с относительной погрешностью $\frac{0,16}{314} = 0,051\%$.

П Р И М Е Р 1.14. Найти допустимую абсолютную погрешность приближенных величин *x* = 15,2, *y* = 57°, для которых возможно найти значение функции

$$u = 6x^2(\lg x - \sin 2y)$$

с точностью до двух десятичных знаков (после запятой).

Р е ш е н и е . Находим u= $6x^2$ (lg x-sin *2y)* = 6 $(15,2)^2$ (lg 15,2 —sin 114°) = **371,9,**

$$\frac{du}{dx} = 12x(\lg x - \sin 2y) + 6x \cdot \lg e = 88,54$$

$$\frac{du}{dy} = -12x^2 \cos 2y = 1127,7.$$

По условию $\Delta_u = 0,005$. Тогда согласно принципу **равных влияний** по формуле (1.22) находим

$$\Delta_x = \frac{\Delta_u}{2\left|\frac{du}{dx}\right|} = \frac{0,005}{2 \cdot 88,54} = 0,28 \cdot 10^{-4},$$

$$\Delta_y = \frac{\Delta_u}{2\left|\frac{du}{dy}\right|} = \frac{0,005}{2 \cdot 1127,7} = 0,22 \cdot 10^{-5} = 0'',45.$$

ЗАДАЧИ

1. С какой точностью следует взять приближенные числа x, чтобы значения sinx могли быть найдены с указанным числом *m* верных знаков?

а) x = 1°, *m* = 3, б) x = 25°, *m* = 4, в) x = 30,75°, *m=3,* г) x=1,05, *m* = 2, д) *x* = 0,075, *m* =2.

2. С какой точностью определены углы x по значениям sin *x,* взятым из пятизначной таблицы"функций?

а) *x* = 2°1', б) x = 15°30', в) x = 44°, г) x = 50°18', д) x = 65°23',

е) x = 87^0.

3. С какой точностью может быть определено число *x* по логарифму с помощью пятизначной таблицы логарифмов, если число находится в указанных пределах?

а) 300<x<400, б) 35<x<40, в) $1,5 < x < 1,7$, г) $3,25 < x < 3,29$, д) $5000 < x < 6000$.

4. С каким числом верных знаков следует взять значение аргумента *x,* чтобы получить значения указанных функций с точностью до 0,1-10^{-5}?

а) $y = x^3 \sin x$, $x = \sqrt{2}$, б) $y = x \ln x$, $x = \pi$, в) $y = e^x \cos x$, $x = \sqrt{3}$.

5. С каким числом верных знаков должен быть известен свободный член уравнения x^2 — 2x + lg2 = 0, чтобы получить корни с четырьмя верными знаками?

6. Найти допустимые абсолютные погрешности аргументов, которые позволяют вычислять значения данных функций с четырьмя верными знаками.

а) $u = \ln(x_1 + x_2^2)$, $x_1 = 0,9731$, $x_2 = 1,13214$,

б) $u = \dfrac{x_1 + x_2^2}{x_3}$, $x_1 = 3,2835$, $x_2 = 0,93221$, $x_3 = 1,13214$,

в) $u = x_1x_2 + x_1x_3 + x_2x_3$, $x_1 = 2,10415$, $x_2 = 1,93521$, $x_3 = 0,84542$.

7. Для определения модуля Юнга по прогибу стержня прямоугольного сечения применяется формула

$$E = \frac{1}{4}\frac{l^3P}{d^3bs}$$

где l — длина стержня, *b* и *d* — основание и высота поперечного сечения, s— стрела прогиба, *P*—нагрузка. С какой точностью следует измерить длину l и стрелу s, чтобы погрешность E не превышала 5,5% при условии, что P известна с точностью до 0,1%, величины *b* и *d* известны с точностью до 1%, $l \approx 50$ *см,* $s \approx 2,5$ *см?*

Глава 2
ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ФУНКЦИИ

При вычислении с помощью счётных машин значений функций, заданных формулами, далеко не безразлично, в каком виде записана соответствующая формула. Математически эквивалентные выражения часто оказываются неравноценными с точки зрения практики вычислений. Дело в том, что основными операциями большинства вычислительных машин являются сложение, вычитание, умножение и деление. Поэтому возникает необходимость представить рассматриваемую математическую задачу в виде последовательности этих элементарных операций. Учитывая ограниченность объёма памяти машины и необходимость экономии машинного времени, желательно эти операции разбить на повторяющиеся циклы и выбрать соответствующий алгоритм. Ниже мы рассмотрим приёмы, сводящие вычисление некоторых функций к таким циклам из элементарных операций.

§ 1. Вычисление значений многочлена. Схема Горнера

Пусть дан многочлен n-й степени

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

с действительными коэффициентами $a_k \ (k = 0,1,\ldots,n)$, и пусть требуется найти значение этого многочлена при $x = \xi$

$$P(\xi) = a_0 \xi^n + a_1 \xi^{n-1} + \ldots + a_n \tag{2.1}$$

Вычисление значения $P(\xi)$ удобнее всего производить следующим образом. Представим выражение $P(\xi) = a_0 \xi^n + a_1 \xi^{n-1} + \ldots + a_n$ (2.1) в виде

$$P(\xi) = \Big(\Big(\Big(\big(a_0 \xi + a_1\big)\xi + a_2\Big)\xi + a_3\Big)\xi + \ldots + a_n\Big).$$

Если ввести числа

$$\left.\begin{array}{ll} & b_0 = a_0, \\ c_1 = b_0 \xi, & b_1 = a_1 + c_1, \\ c2 = b_1 \xi, & b_2 = a_2 + c_2, \\ \ldots\ldots\ldots & \ldots\ldots\ldots \\ c_n = b_{n-1} \xi, & b_n = a_n + c_n, \end{array}\right\} \tag{2.2}$$

то $b_n = P(\xi)$.

Таким образом, вычисление значения многочлена $P(x)$ при $x = \xi$ сводится к повторению следующей совокупности элементарных операций:

$$c_k = b_{k-1} \xi \quad b_k = a_k + c_k \quad (k = 1,2,\ldots,n).$$

Нетрудно показать, что числа $b_0 = a_0, b_1, \ldots, b_{n-1}$ являются коэффициентами многочлена $Q(x)$, полученного в качестве частного при делении данного многочлена $P(x)$ на двучлен $x - \xi$, а $b_n = P(\xi)$ – остаток от деления.

Таким образом, можно, не производя деления, определять коэффициенты частного $Q(x)$, а также остаток $P(\xi)$. Числа $b_0, b_1, \ldots, b_n$ обычно находят, пользуясь известной схемой *Горнера*

$$\begin{array}{ccccccc} a_0 & a_1 & a_2 & \ldots & a_n & \Big| & \xi \\ + & b_0\xi & b_1\xi & \ldots & b_{n-1}\xi & & \\ \hline b_0 = a_0 & b_1 & b_2 & \ldots & \underline{b_n = P(\xi)} & & \end{array}$$

Вычисление значений многочлена $P_n(x)$ по схеме Горнера требует выполнения n умножений и $n-k$ сложений, где k – число коэффициентов a_i, равных нулю. Если $a_0 = 1$, то требуется выполнить $n-1$ умножений. Показано, что для многочленов общего вида нельзя построить схему более экономную в смысле числа операций, чем схема Горнера.

П р и м ер 1.1. Вычислить при $x=-1,5$ значение многочлена

$$P(x)=x^7-2x^6+x^5-3x^4+4x^3-x^2+6x-1.$$

Р е ш е н и е. Пользуясь схемой Горнера, получим

	1	–2	1	–3	4	–1	6	–1	$\lfloor\xi$
+	–1,5	–1,5	5,25	–9,375	18,5625	–33,8438	52,2657	–87,3985	
	1	–3,5	6,25	–12,375	22,5625	–34,8438	58,2657	–88,3985	$=P(-1,5).$

Таким образом,

$$P(-1,5)=-88,3985.$$

З А Д А Ч И

1. Дан многочлен

$$P(x)=a_0x^4+a_1x^3+a_2x^2+a_3x+a_4.$$

Найти значение $P(3,25)$ для коэффициентов a_0, a_1, a_2, a_3, a_4, приведённых в Таблица 2.1

Таблица 2.1

	a_0	a_1	a_2	a_3	a_4		a_0	a_1	a_2	a_3	a_4
а)	7,54	11,08	3,82	0,44	-0,48	д)	2,79	9,85	14,15	5,38	7,24
б)	9,36	12,69	14,39	0,79	-094	е)	3,45	-2,91	3,79	-6,75	-2,38
в)	12,78	14,35	17,19	1,34	-1,72	ж)	4,79	5,38	-2,86	7,31	4,55
г)	15,65	17,58	21,7	2,78	1,34	з)	8,34	-7,75	4,53	-9,29	5,79

2. Дан многочлен

$$P(x)=0,22x^5-3,27x^4-2,74x^3-2,81x^2-3,36x+2.$$

Найти значения $P(\xi)$, где $\xi=0,80+0,05k;\ k=0,1,2,\ldots,20.$

§ 2. Вычисление значений некоторых трансцендентных функций с помощью степенных рядов

Здесь рассматриваются только такие трансцендентные функции, которые являются суммами своих рядов Маклорена

$$f(x)=\sum_{k=0}^{\infty}\frac{f^{(k)}(0)}{k!}x^k. \qquad (2.3)$$

Беря сумму нескольких первых членов ряда Маклорена, получаем приближённую формулу

$$f(x)\approx P_n(x),$$

$$P_n(x)=\sum_{k=0}^{\infty}\frac{f^{(k)}(0)}{k!}x^k.$$

При этом остаток ряда

$$R_n(x)=f(x)-P_n(x)$$

Представляет ошибку при замене $f(x)$ многочленом $P_n(x)$. Оценка остатка позволяет определить требуемое число слагаемых, т.е. степень n многочлена $P_n(x)$.

Заметим, что так как расчёт суммарной погрешности представляет собой трудоёмкую операцию, то на практике для обеспечения заданий точности все промежуточные вычисления проводят с одним или двумя запасными знаками.

1. Вычисление значений показательной функции.

Для показательной функции справедливо разложение

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \quad \left(-\infty < x < \infty\right). \tag{2.4}$$

Вычисления удобно вести, пользуясь следующей рекуррентной записью:

$$e^x = \sum_{k=0}^{\infty} u_k \quad u_k = \frac{x}{k} u_{k-1}, \quad S_k = S_{k-1} + u_k \quad \left(k = 1,2,\ldots,n\right).$$

Где $u_0 = 1,\ S_0 = 1.$ Число $S_n = \sum_{k=0}^{n} \frac{x^k}{k!}$ приближённо даёт искомый результат e^x.

Для остатка ряда может быть получена следующая оценка :

$$\left|R_n\left(x\right)\right| < \left|u_n\right| \text{ при } 0 < 2\left|x\right| \le n;$$

поэтому процесс суммирования может быть прекращён, как только очередной вычисленный член ряда u_k будет по модулю меньше заданной допустимой погрешности ε :

$$\left|u_n\right| < \varepsilon \qquad \left(\textit{если только } \left|x\right| \le \frac{\pi}{2}\right).$$

При больших по модулю значениях x ряд $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \quad \left(-\infty < x < \infty\right).$ (2.4) малопригоден для вычислений. В этих случаях обычно поступают так: представляют x в виде суммы

$$x = E\left(x\right) + q,$$

где $E\left(x\right) -$ целая часть x и $q -$ дробная его часть, $0 \le q < 1.$ Тогда

$$e^x = e^{E(x)} \cdot e^q.$$

Первый множитель $e^{E(x)}$ находится с помощью умножения

$$e^{E(x)} = \underbrace{e \cdot e \ldots e}_{E(x)\ \textit{раз}}, \text{ если } E\left(x\right) > 0,$$

и

$$e^{E(x)} = \underbrace{\frac{1}{e} \cdot \frac{1}{e} \ldots \frac{1}{e}}_{-E(x)\ \textit{раз}}, \text{ если } E\left(x\right) < 0,$$

Второй множитель вычисляется с помощью степенного разложения

$$e^q = \sum_{n=0}^{\infty} \frac{q^n}{n!}.$$

При $0 \le q < 1$ этот ряд быстро сходится, так как

$$0 \le R_n\left(q\right) < \frac{1}{n!n} q^{n+1}.$$

П р и м е р 2.1. Найти $\sqrt{e}$ с точностью до 10^{-5}.

Р е ш е н и е. Пользуемся формулой

$$e^{1/2} = \sum_{k=0}^{n} u_k + R_n\left(\frac{1}{2},\right) \tag{2.5}$$

где $u_0 = 1,\ u_k = \frac{u_{k-1}}{2k}$ $\left(k = 1,2,\ldots,n\right)$. Слагаемые подсчитываем с двумя запасными десятичными

знаками.
Последовательно имеем

$$u_0 = 1,$$

$$u_1 = \frac{u_0}{2} = 0,5000000,$$

$$u_2 = \frac{u_1}{4} = 0,1250000,$$

$$u_3 = \frac{u_2}{6} = 0,0208333,$$

$$u_4 = \frac{u_3}{8} = 0,0026042,$$

$$u_5 = \frac{u_4}{10} = 0,0002604,$$

$$u_6 = \frac{u_5}{12} = 0,0000217,$$

$$u_7 = \frac{u_6}{14} = \underline{0,0000016,}$$

$$S_7 = 1,6487212.$$

Округляя сумму до пяти десятичных знаков после запятой, получим

$$\sqrt{e} = 1,64872.$$

Для вычисления значений показательной функции $a^x \left(a > 0\right)$ можно использовать формулу $a^x = e^{x \ln a}$.

2. Вычисление значений синуса и косинуса.

Для вычисления значений функции $\sin x$ и $\cos x$ пользуемся степенными разложениями

$$\sin x = \sum_{k=0}^{\infty} \left(-1\right)^k \frac{x^{2k+1}}{\left(2k+1\right)!} \quad \left(-\infty < x < \infty\right), \tag{2.6}$$

$$\cos x = \sum_{k=0}^{\infty} \left(-1\right)^k \frac{x^{2k}}{\left(2k\right)!} \quad \left(-\infty < x < \infty\right), \tag{2.7}$$

Ряды (2.6) и (2.7**Ошибка! Источник ссылки не найден.**) при больших x сходятся медленно, но, учитывая периодичность функций sin(x) и cos(x) и формулы приведения тригонометрических функций, легко заключить, что достаточно уметь вычислять sin(x) и cos(x) для промежутка $0 \le x \le \frac{\pi}{4}$. При этом можно использовать следующие рекуррентные формулы:

$$\left.\begin{aligned} \sin x &= \sum_{k=1}^{n} u_k + R_n\left(x\right), \\ u_1 = x, \;\; u_{k+1} &= -\frac{x^2}{2k\left(2k+1\right)} u_k \;\; \left(k = 1,2,\ldots,n-1\right); \end{aligned}\right\} \tag{2.8}$$

$$\left.\begin{aligned} \cos x &= \sum_{k=1}^{n} \upsilon_k + R_n(x), \\ \upsilon_1 = x,\ \ \upsilon_{k+1} &= -\frac{x^2}{2k(2k-1)} u_k \ \ (k = 1,2,\ldots,n-1); \end{aligned}\right\} \tag{2.9}$$

Так как в промежутке $\left(0, \frac{\pi}{4}\right)$ ряд (2.8) знакочередующийся с монотонно убывающими по модулю членами, то для его остатка R_n справедлива оценка

$$|R_n| \le \frac{|x|^{2n+1}}{(2n+1)!} = |u_{n+1}|.$$

Аналогично для ряда (2.5) $|R_n| \le |\upsilon_{n+1}|$. Следовательно, процесс вычисления $\sin x$ *и* $\cos x$ можно прекратить, как только очередной полученный член ряда по модулю будет меньше допустимой погрешности ε.

П р и м е р 2.2. Вычислить $\sin 23^\circ 54'$ с точностью до 10^{-4}.

Р е ш е н и е. Переводим аргумент в радианы, сохраняя один запасной знак: $x = \text{arc}\, 23^\circ 54' = 0,41714.$ Применяя формулу, получим

$$u_1 = x = +0,41714,$$

$$u_2 = -\frac{x^2 u_1}{2 \cdot 3} = -0,01210,$$

$$u_3 = -\frac{x^2 u_2}{4 \cdot 5} = +0,00011,$$

$$u_4 = -\frac{x^2 u_3}{6 \cdot 7} = -0,00000.$$

Отсюда $\sin 23^\circ 54' = 0,40515 \approx 0,4052.$

П р и м е р 2.3. Вычислить $\cos 17^\circ 24'$ с точностью 10^{-5}.

Р е ш е н и е. $x = \text{arc}\, 17^\circ 24' = 0,30369$. Применяя формулу (2.7), будем иметь

$$\upsilon_1 = 1,000000,$$

$$\upsilon_2 = -\frac{x^2}{1 \cdot 2} \upsilon_1 = -0,046114,$$

$$\upsilon_3 = -\frac{x^2}{3 \cdot 4} \upsilon_2 = +0,000354,$$

$$\upsilon_4 = -\frac{x^2}{5 \cdot 6} \upsilon_3 = -0,000001.$$

Отсюда

$$\cos 17^\circ 24' = 0,95424.$$

3. Вычисление значений гиперболического синуса и гиперболического косинуса.

Пользуемся степенными разложениями

$$\operatorname{sh} x = \sum_{k=1}^{\infty} \frac{x^{2k-1}}{(2k-1)!} \qquad (-\infty < x < \infty), \tag{2.10}$$

$$ch\, x = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!} \qquad (-\infty < x < \infty) \tag{2.11}$$

и рекуррентной записью

$$\left.\begin{aligned} \operatorname{sh} x &= \sum_{k=1}^{\infty} u_k + R_n, \\ u_1 = x,\ u_{k+1} &= \frac{x^2}{2k(2k+1)} u_k; \end{aligned}\right\} \tag{2.12}$$

$$\left.\begin{aligned} ch\, x &= \sum_{k=0}^{\infty} \upsilon_k + R_n^{\bullet}, \\ \upsilon_0 = 1,\ \upsilon_{k+1} &= \frac{x^2}{(2k+1)(2k+2)} \upsilon_k; \end{aligned}\right\} \tag{2.13}$$

При $n \geq |x| > 0$ имеют место оценки $R_n < \frac{1}{3}|u_n|$ и $R_n^* < \frac{2}{3}\upsilon$.

Пример 2.4. Вычислить $\operatorname{sh} 1,4$ с точностью до 10^{-5}.

Решение. Применяя формулу (2.10), получим

$$u_1 = 1,400000,$$

$$u_2 = -\frac{x^2}{2 \cdot 3} u_1 = -0,4573333,$$

$$u_3 = -\frac{x^2}{4 \cdot 5} u_2 = +0,0448187,$$

$$u_4 = -\frac{x^2}{6 \cdot 7} u_3 = -0,0020915,$$

$$u_5 = -\frac{x^2}{8 \cdot 9} u_4 = 0,0000569,$$

$$u_6 = -\frac{x^2}{10 \cdot 11} u_5 = 0,0000010.$$

Отсюда

$$\operatorname{sh} 1,4 = 1,904301.$$

4. Вычисление значений логарифмической функции.

Пользуемся разложением по степеням $\frac{1-z}{1+z}$:

$$\ln z = -2 \sum_{k=1}^{\infty} \frac{1}{2k-1} \left(\frac{1-z}{1+z} \right)^{2k-1} \qquad (0 < z < +\infty).$$

Пусть $x -$ положительное число. Представим его в виде

$$x = 2^m z,$$

где $m -$ целое число и $1/2 \leq z < 1;$ тогда, полагая

$$\frac{1-z}{1+z}=\xi,$$

получим

$$\ln x=\ln 2^m z=m\ln 2+\ln z=m\ln 2-2\sum_{k=1}^{\infty}\frac{1}{2k-1}\xi^{2k-1},$$

где $0<\xi\le 1/2.$ Обозначив

$$u_k=\frac{\xi^{2k-1}}{2k-1}\qquad (k=1,2,\ldots,n),$$

получаем рекуррентную запись

$$\left.\begin{aligned}\ln x&=m\ln 2-2\sum_{k=1}^{n}u_k+R_n,\\ u_1&=\xi,\ u_{k+1}=\frac{(2k-1)\xi^2}{2k+1}u_k.\end{aligned}\right\}\qquad (2.14)$$

Процесс суммирования прекращается, как только выполнится неравенство $u_n<4\varepsilon,$ где $\varepsilon-$ допустимая погрешность.

ПРИМЕР 2.5. Найти ln5 с точностью до 10^{-6}.

Р е ш е н и е. Вычисления будем производить с двумя запасными знаками. Положим $5=2^3*0{,}625$. Следовательно, z=0,625 и $\xi=\frac{1-z}{1+z}=\frac{0{,}375}{1{,}625}=0{,}23076923.$

Выпишем первые слагаемые:

$$\begin{aligned}u_1&=\xi=0{,}23076923,\\ u_2&=\frac{\xi^3}{3}=0{,}00409650,\\ u_3&=\frac{\xi^5}{5}=0{,}00013089,\\ u_4&=\frac{\xi^7}{7}=0{,}00000498,\\ \hline \textit{сумма}&=0{,}23500160.\end{aligned}$$

По формуле (2.12) получим

$$\ln 5=3\cdot 0{,}69314718-2\cdot 0{,}23500160=1{,}609438.$$

З А Д А Ч И

1. Пользуясь разложением в степенной ряд, составить с указанной точностью ε таблицы значений следующих функций.

а) $e^x,\quad x=0{,}300+0{,}002k\quad (k=0,1,\ldots,14),\quad \varepsilon=10^{-5},$

б) $e^x,\quad x=2{,}500+0{,}002k\quad (k=0,1,\ldots,14),\quad \varepsilon=10^{-4},$

в) $e^{-x},\quad x=1{,}35+0{,}01k\quad (k=0,1,\ldots,14),\quad \varepsilon=10^{-5},$

г) $e^{-x},\quad x=0{,}505+0{,}005k\quad (k=0,1,\ldots,15),\quad \varepsilon=10^{-5},$

д) $e^{x^2},\quad x=0{,}50+0{,}02k\quad (k=0,1,\ldots,15),\quad \varepsilon=10^{-5},$

е) $e^{-x^2},\quad x=1{,}30+0{,}01k\quad (k=0,1,\ldots,15),\quad \varepsilon=10^{-5},$

ж) $\frac{1}{2\sqrt{n}}e^{\frac{-x^2}{2}}$, $x=1,78+0,03k$ $(k=0,1,\ldots,15)$, $x=2,545+0,005k$ $(k=0,1,\ldots,15)$.

2. Пользуясь разложением $\sin x$ *и* $\cos x$ в степенной ряд, составить таблицы значений следующих функций с точностью до 10^{-5}.

а) $\sin x$, $x=0,345+0,005k$ $(k=0,1,\ldots,15)$,

б) $\sin x$, $x=1,75+0,01k$ $(k=0,1,\ldots,15)$,

в) $\cos x$, $x=0,745+0,005k$ $(k=0,1,\ldots,15)$,

г) $\cos x$, $x=1,75+0,01k$ $(k=0,1,\ldots,15)$,

д) $\frac{\sin x}{x}$, $x=0,4+0,01k$ $(k=0,1,\ldots,15)$,

е) $\frac{\cos x}{x}$, $x=0,25+0,01k$ $(k=0,1,\ldots,15)$.

3. Пользуясь разложением $\operatorname{sh} x$ *и* $\operatorname{ch} x$ в степенной ряд, составить таблицы значений следующих функций с точностью до ε.

а) $\operatorname{sh} x$, $x=0,23+0,01k$ $(k=0,1,\ldots,15)$,

$\varepsilon=10^{-5}$, $x=23,0+0,05k$ $(k=0,1,\ldots,15)$, $\varepsilon=10^{-4}$,

б) $\operatorname{ch} x$ для тех же значений x.

§ 3. Некоторые многочленные приближения

Вычисление с помощью рядов Тейлора даёт достаточно быструю сходимость, вообще говоря, только при малых значениях $|x-x_0|$. Однако часто бывает нужно с помощью многочлена сравнительно невысокой степени подобрать приближение, которое давало бы достаточную точность для всех точек заданного отрезка. В этих случаях применяются разложения функций, полученные с помощью полиномов Чебышева на заданном отрезке. Ниже приводятся примеры таких разложений, указываются промежутки, в которых их следует использовать, а также соответствующие абсолютные погрешности ε. Для вычисления значений многочлена можно использовать схему Горнера.

1. Вычисление значений показательно функции на отрезке $[-1,1]$.

Пользуемся следующим многочленным приближением:

$$e^x \approx \sum_{k=0}^{7} a_k x^k \quad (|x|\le 1), \quad \varepsilon = 2\cdot 10^{-7}, \tag{2.15}$$

$a_0=0,9999998$, $a_1=1,0000000$, $a_2=0,5000063$, $a_3=0,1666674$,

$a_4=0,0416350$, $a_5=1,0083298$, $a_6=0,0014393$, $a_7=0,0002040$.

2. Вычисление значений логарифмической функции.

Имеет место формула

$$\ln(1+X) \approx \sum_{k=1}^{7} a_k x^k \quad (0 \le x \le 1), \quad \varepsilon = 2{,}2 \cdot 10^{-7}, \tag{2.16}$$

$a_1 = 0{,}999981028, \quad a_2 = -0{,}499470150 \quad a_3 = 0{,}328233122,$
$a_4 = -0{,}225873284, \quad a_5 = -0{,}134639267, \quad a_6 = -0{,}055119959,$
$a_7 = 0{,}010757369.$

3. Вычисление значений тригонометрических функций.
Пользуемся следующими многочленными приближениями:

$$\sin x \approx \sum_{k=0}^{4} a_{2k+1} k^{2k+1} \quad \left(|x| \le \frac{\pi}{2}\right) \quad \varepsilon = 6 \cdot 10^{-9}, \tag{2.17}$$

$a_1 = 1{,}000000002 \quad a_3 = -0{,}166666589 \quad a_5 = 0{,}008333075,$
$a_7 = -0{,}000198107, \quad a_9 = 0{,}000002608;$

$$\cos x \approx \sum_{k=0}^{5} a_{2k} x^{2k} \quad (|x| \le 1) \quad \varepsilon = 2 \cdot 10^{-9}, \tag{2.18}$$

$a_0 = 1{,}000000000000, \quad a_2 = -0{,}499999999942,$
$a_4 = 0{,}041666665950, \quad a_6 = -0{,}001388885683,$
$a_8 = 0{,}000024795132, \quad a_{10} = -0{,}000000269591;$

$$\operatorname{tg} x \approx \sum_{k=0}^{6} a_{2k+1} x^{2k+1} \quad \left(|x| \le \frac{\pi}{4}\right) \quad \varepsilon = 2 \cdot 10^{-8}, \tag{2.19}$$

$a_1 = 1{,}00000002, \quad a_3 = 0{,}33333082, \quad a_5 = 1{,}3339762,$
$a_7 = 0{,}05935836, \quad a_9 = 0{,}02457096, \quad a_{11} = 0{,}00294045,$
$a_{13} = 0{,}00947324.$

ПРИМЕР 2.6 Пользуясь многочленным приближением, найти значение $\sqrt{e}$ с точностью до 10^{-6}.
Р е ш е н и е. Вычисления проводим по формуле (2.15) пользуясь схемой Горнера при $x = 0{,}5$ (см. Таблица 2.2).

Таблица 2.2

0,0002040	0,0014393	0,0083298	0,0416350
+	0,0001020	0,0007706	0,0045502
0,0002040	0,0015413	0,0091004	0,00461852

0,1666674	0,5000063	1,0000000	0,9999998 | 0,5
0,0230926	0,0948800	0,2974431	0,6487216
0,1897600	0,5948863	1,2974431	$1{,}6487214 = P(0{,}5)$

Округляя до шести знаков после запятой, получим $e^{\frac{1}{2}} \approx 1{,}648721$ (ср. пример 2.1).

ПРИМЕР 2.7 Пользуясь многочленным приближением найти значение $\sin 0{,}5$ с точностью до 10^{-8}.
Р е ш е н и е. Вычисления можно провести по формуле (2.19), пользуясь схемой Горнера при $x = 0{,}5$. Но так как многочлен в формуле (2.19) содержит только нечётные степени x, удобнее переписать его в виде

$$\sum_{k=0}^{4} a_{2k+1}x^{2k+1} = \left(a_1 + a_3x^2 + a_5x^4 + a_7x^6 + a_9x^8\right)x$$

и применить схему Горнера к многочлену

$$P(\xi) = a_1 + a_3\xi + a_5\xi^2 + a_7\xi^3 + a_9\xi^4, \qquad \xi = x^2 = 0,25. \tag{2.20}$$

Вычисление значения $P(0,25)$ проведено в Таблица 2.3. Умножая полученное значение $P(0,25) = 0,958851087$ на $x = 0,5$ и округляя, получим искомое значение $\sin 0,5 \approx 0,47942554$.

Таблица 2.3

$0,000002608$	$-0,000198107$	$0,008333075$	$-0,166666589$	$0,000000002 \quad \vert 0,25$
$+$	$0,000000652$	$-0,000049364$	$0,002070928$	$-0,041148915$
$0,000002608$	$-0,000197445$	$0,008283711$	$-0,164595661$	$0,958851087 = P(0,25)$

З А Д А Ч И

1. Составить таблицы значений следующих функций с точностью до ε для указанных значений x.

а) e^x, $x = 0,725 + 0,001k \quad (k = 0,1,2,\ldots,15)$, $\varepsilon = 10^{-4}$,

$x = 0,213 + 0,002k \quad (k = 0,1,2,\ldots,15)$, $\varepsilon = 10^{-5}$,

б) e^x, $x = 0,213 + 0,003k \quad (k = 0,1,2,\ldots,15)$, $\varepsilon = 10^{-5}$,

$x = 1,27 + 0,02k \quad (k = 0,1,2,\ldots,15)$, $\varepsilon = 10^{-5}$,

в) $e^{\frac{1}{x}}$, $x = 0,2 + 0,05k \quad (k = 0,1,2,\ldots,15)$, $\varepsilon = 10^{-5}$,

г) $\dfrac{1}{2\sqrt{\pi}} e^{-\frac{x^2}{2}}$, $x = 0,4 + 0,02k \quad (k = 0,1,2,\ldots,15)$, $\varepsilon = 10^{-5}$,

$x = 1,2 + 0,1k \quad (k = 0,1,2,\ldots,15)$, $\varepsilon = 10^{-5}$.

2. Составить таблицы для следующих функций с точностью до 10^{-5} для указанных значений x.

а) $\sin x$, $x = 0,055 + 0,003k \quad (k = 0,1,2,\ldots,15)$, $x = 0,80 + 0,05k \quad (k = 0,1,2,\ldots,15)$,

б) $\cos x$ для тех же значений x, в) $\operatorname{tg} x$ для тех же значений x.

§ 4. Применение цепных дробей для вычисления значений трансцендентных функций

1. **Вводные замечания**.

Пусть $a_0, a_1, a_2, \ldots, a_n, \ldots, b_1, b_2, \ldots, b_n, \ldots$ – две последовательности. Выражение вида

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ddots}}} = \left[a_0; \frac{b_1}{a_1}, \frac{b_2}{a_2}, \frac{b_3}{a_3}, \ldots\right] \tag{2.21}$$

называется *цепной* или *непрерывной* дробью, отвечающей заданным последовательностям

$\{a_k\}_{k=0}^{\infty}$, $\{b_k\}_{k=1}^{\infty}$. В общем случае элементы цепной дроби a_0, a_k, b_k $(k=1,2,\ldots)-$ вещественные или комплексные числа или функции одной или нескольких переменных. Выражения

$$a_0+\frac{b_1}{a_1},\ a_0+\frac{b_1}{a_1+\dfrac{b_2}{a_2}}a_0,\ a_0+\frac{b_1}{a_1+\dfrac{b_2}{a_2+\dfrac{b_3}{a_3}}},\ldots,\ a_0+\cfrac{b_1}{a_1+\cfrac{b_2}{a_2+\ddots\ a_{n-1}+\cfrac{b_n}{a_n}}}$$

называются соответственно первой, второй, третьей, … , $n-$й *подходящей дробью* Для данной бесконечной дроби и обычно обозначаются $\frac{P_1}{Q_1}$, $\frac{P_2}{Q_2}$, … ,$\frac{P_n}{Q_n}$. При работе на ЭВМ подходящие цепные дроби удобно находить с помощью следующей последовательности операций:

$$c_1=\frac{b_n}{a_n},\qquad d_1=a_{n-1}+c_1,$$
$$c_2=\frac{b_{c-1}}{d_1},\quad d_2=a_{n-2}+c_2,$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$c_k=\frac{b_{n-k+1}}{d_{k-1}},\quad d_k=a_{n-k}+c_k,$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$c_n=\frac{b_1}{d_{n-1}},\quad d_n=a_0+c_n=\frac{P_n}{Q_n}.$$

Указанная последовательность действий легко программируется.
Для числителей и знаменателей подходящих дробей имеют место следующие рекуррентные формулы:

$$P_n=a_nP_{n-1}+b_nP_{n-2},\qquad Q_n=a_nQ_{n-1}+b_nQ_{n-2},\qquad (n=1,2,\ldots),$$

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

Цепная дробь (2.21) называется *сходящейся*, если существует конечный предел A подходящей дроби P_n/Q_n при $n\to\infty$:

$$A=\lim_{n\to\infty}\frac{P_n}{Q_n},$$

причём число A принимается значение этой дроби. Если же предел не существует, то цепная дробь (2.21) называется *расходящейся*. Для сходящейся цепной дроби подходящая дробь P_n/Q_n является её приближённым значением.

2. Разложение e^x в цепную дробь. Пользуемся следующим разложением:

$$e^x=\left[0,\ \frac{1}{1},\ \frac{-2x}{2+x},\ \frac{x^2}{6},\ \frac{x^2}{10},\ldots,\ \frac{x^2}{4n+2},\ldots\right]. \qquad (2.22)$$

Для любого x эта дробь сходящаяся. Подходящими дробями для данной функции являются

$$\frac{P_1}{Q_1}=\frac{1}{1},\qquad \frac{P_2}{Q_2}=\frac{2+x}{2-x},\qquad \frac{P_3}{Q_3}=\frac{12+6x+x^2}{12-6x+x^2},$$

$$\frac{P_4}{Q_4}=\frac{120+60x+12x^2+x^3}{120-60x+12x^2-x^3},$$

$$\frac{P_5}{Q_5}=\frac{1680+840x+180x^2+20x^3+x^4}{1680-840x+180x^2-20x^3+x^4},$$

П р и м е р 2.8. Пользуясь разложением e^x в цепную дробь, вычислить $1/e$ с точностью до 10^{-5}.

Р е ш е н и е. Вычислим четвёртую и пятую подходящие дроби при $x=-1$:

$$\frac{P_4}{Q_4}=\frac{120-60+12-1}{120+60+12+1}=\frac{71}{193}\approx 0,367876.$$

$$\frac{P_5}{Q_5}=\frac{1680-840+180-20+1}{1680+840+180+20+1}=\frac{1001}{2721}\approx 0,367879.$$

Сравнивая полученные приближения, замечаем, что у них совпадают пять десятичных знаков, поэтому можно положить $e^{-1}\approx 0,36788.$

3. Разложение $\mathrm{tg}x$ в цепную дробь.

Имеет место следующее разложение:

$$\mathrm{tg}\,x=\left[0,\ \frac{x}{1},\ -\frac{x^2}{3},\ -\frac{x^2}{5},\ldots,\ 0\frac{x^2}{2n+1},\ldots\right]. \tag{2.23}$$

Это разложение справедливо во всех точках непрерывности $\mathrm{tg}\,x$. Первыми подходящими дробями будут

$$\frac{P_1}{Q_1}=\frac{1}{1},\qquad \frac{P_4}{Q_4}=\frac{105x-10x^3}{105-45x^2+x^4},$$

$$\frac{P_2}{Q_2}=\frac{3x}{3-x^2},\qquad \frac{P_5}{Q_5}=\frac{945x-105x^3+x^5}{945-420x^2+15x^4},$$

$$\frac{P_3}{Q_3}=\frac{15x-x^3}{15-6x^2},$$

З А Д А Ч И

1. Пользуясь разложениями в цепную дробь, составить таблицы значений следующих функций с точностью до ε.

а) e^x, $x=0,155+0,005k$ $\left(k=0,1,2,\ldots,15\right)$, $\varepsilon=10^{-5}$, $x=0,30+0,03k$ $\left(k=0,1,2,\ldots,15\right)$, $\varepsilon=10^{-4}$,

б) $\mathrm{tg}\,x$, $x=0,47+0,005k$ $\left(k=0,1,2,\ldots,15\right)$, $\varepsilon=10^{-5}$

§ 5. Применение метода итераций для приближённого вычисления значений функций

Всякую функцию $y=f\left(x\right)$ можно различными способами задавать неявно, т.е. некоторым уравнением

$$F\left(x,y\right)=0. \tag{2.24}$$

Часто бывает, что решение уравнения (2.24) относительно y каким-либо итерационным методом сводится к однообразным операциям, легко реализуемым на ЭВМ. Тогда, очевидно, целесообразно применить метод итераций.

Один из возможных итерационных процессов для вычисления $y\left(x\right)$ можно построить следующим образом.

Пусть y_n – приближённое значение y. Применив формулу Лагранжа, получим

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

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

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

Причём значение $\overline{y}_n$ нам не известно.

Полагая приближённо $\overline{y}_n\approx y_n$, получим следующую формулу для вычисления $y\approx y_{n+1}$:

$$y_{n+1}=y_n-\frac{F(x,y_n)}{F_y'(x,y_n)} \qquad (n=0,1,2,\ldots). \tag{2.25}$$

Если $F_y'(x,y)$ и $F_{yy}''(x,y)$ существуют и сохраняют постоянные знаки в рассматриваемом интервале, содержащем корень $y(x)$, то итерационный процесс сходится к $y(x)$.

Начальное приближение $y_0(x)$ выбирают так, чтобы оно легко вычислялось и было, по возможности, близким к истинному значению $y(x)$.

Процесс итераций продолжается до тех пор, пока в пределах заданной точности два последовательных значения y_{n+1} и y_n не совпадут между собой, после чего приближённо полагают

$$y(x)\approx y_{n+1}.$$

1. Вычисление обратной величины. Пусть $y=1/x\ (x>0)$. Положим $F(x,y)=x-1/y=0.$ Применив формулу (2.25), получим

$$y_{n+1}=y_n(2-xy_n). \tag{2.26}$$

Вычисление y_{n+1} по полученной итерационной формуле (2.26) содержит лишь действия умножения и вычитания. Таким образом, можно находить 1/x на вычислительных машинах, в которых нет операции деления. Начальное значение y_0 выбирается обычно следующим образом. Записывают аргумент x в двоичной системе:

$$x=2^m x_1,$$

где m – целое число и $1/2\le x_1<1$. Полагают $y_0=2^{-m};$ при таком выборе начального значения y_0 сходимость итерационного процесс довольно быстрая.

Замечание. Так как частное a/b есть произведение a на $1/b$, то деление на машинах, в которых нет операции деления, можно реализовать в два этапа:

1) Вычисление $y=1/b$ (обратной величины делителя),

2) Умножение y на делимое a.

ПРИМЕР 2.9 С помощью формулы (2.26) найти значение функции $y=1/x$ при $x=5$ с точностью до 10^{-4}.

Решение. Запишем аргумент x в виде $x=2^3\cdot\frac{5}{8}$. Полагаем $y_0=2^{-3}=1/8$. По формуле (2.23) будем иметь

$$y_1=\frac{1}{8}\left(2-\frac{5}{8}\right)=\frac{11}{64}=0{,}1718, \qquad y_2=\frac{11}{64}\left(2-\frac{55}{64}\right)=\frac{803}{4096}=0{,}1960,$$

$$y_3 = 0{,}1960\left(2 - 0{,}9800\right) = 0{,}1960 \cdot 1{,}0200 = 0{,}19992.$$

Мы видим, что здесь уже третье приближение даёт $y\left(x\right) \approx 0{,}1999$ с точностью до 10^{-4}.

2. Вычисление квадратного корня. Пусть $y = \sqrt{x}\ \ \left(x > 0\right)$. Преобразуем это уравнение к виду

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

Применяя формулу (2.22), получим

$$y_{n+1} = \frac{1}{2}\left(y_n + \frac{x}{y_n} \right) \qquad \left(n = 0,1,2,\ldots\right). \tag{2.27}$$

Эта формула называется *формулой Герона.*

Пусть аргумент x записан в двоичной системе:

$$x = 2^m x_1,$$

где $m -$ целое число и $1/2 \le x_1 < 1$. Тогда обычно полагают

$$y_0 = 2^{E\left(m/2\right)},$$

Где $E\left(m/2\right) -$ целая часть числа $m/2$.

Итерационный процесс по формуле Герона легко реализуется на машине, имеющей деление в качестве элементарной операции; при этом процесс итераций сходится при любом выборе $y_0 > 0$ (в этом примере легко проверить выполнение указанных выше условий сходимости, так как $F_y' = 2y > 0$ и $F_{yy}'' = 2y > 0$).

Если $0{,}01 \le x \le 1$, то за начальное приближение можно брать $y_0 = ax + b;$ соответствующие коэффициенты a и b приведены в Таблица 2.4.

Таблица 2.4 Коэффициенты для начального приближения в формуле Герона

Интервал	a	b	Интервал	a	b
$\left(0{,}01;0{,}02\right)$	4,1	0,060	$\left(0{,}18;0{,}30\right)$	1,0	0,247
$\left(0{,}02;0{,}03\right)$	3,2	0,078	$\left(0{,}30;0{,}60\right)$	0,8	0,304
$\left(0{,}03;0{,}08\right)$	2,2	0,110	$\left(0{,}60;1{,}00\right)$	0,6	0,409
$\left(0{,}08;0{,}018\right)$	1,4	0,174			

При таком выборе начального приближения y_0 уже вторая итерация y_2 даёт значение $\sqrt{x}$ с восемью десятичными знаками после запятой, причём при вычислении y_0 можно брать значение x лишь с тремя десятичными знаками.

ПРИМЕР 2.10 Найти $\sqrt{7}$ с точностью до 10^{-5}.

Р е ш е н и е. Здесь $x = 7 = 2^3 \cdot \frac{7}{8}$. Следовательно, начальное приближение имеет вид

$$y_0 = 2^{E\left(3/2\right)} = 2.$$

По формуле (2.24) последовательно находим

$$y_1 = \frac{1}{2}\left(2 + \frac{7}{2}\right) = 2,75000,$$

$$y_2 = \frac{1}{2}\left(\frac{11}{4} + \frac{28}{11}\right) = 2,64772,$$

$$y_3 = \frac{1}{2}\left(\frac{233}{88} + \frac{616}{233}\right) = 2,64575,$$

$$y_4 = \frac{1}{2}\left(2,64575 - \frac{7}{2,64575}\right) = 2,64575.$$

Заметим, что в значениях y_3 и y_4 совпадают пять десятичных знаков, приближённо полагаем $\sqrt{7} \approx 2,64575$.

Замечание. Если вычисление ведётся на вычислительной машине, система команд которой не содержит операции деления, то можно пользоваться другой итерационной формулой, а именно:

$$y_{n+1} = y_n\left(\frac{3}{2} - \frac{y_n^2}{2x}\right) \qquad (n = 0,1,2,\ldots). \tag{2.28}$$

Извлечение квадратного корня сводится по этой формуле к однократному вычислению обратной величины $\frac{1}{2x}$ и затем к итерационному процессу, каждый этап которого содержит лишь действия умножения и вычитания. Формула (2.28) соответствует преобразованию исходного уравнения к виду

$$F(x,y) \equiv \frac{1}{y^2} - \frac{1}{x} = 0.$$

3. Вычисление обратной величины квадратного корня. Пусть имеем

$$y = \frac{1}{\sqrt{x}} \qquad (x > 0).$$

Итерационная формула для вычисления обратной величины квадратного корня имеет вид

$$y_{n+1} = \frac{3}{2}y_n - \frac{1}{2}xy_n^3 \qquad (n = 0,1,2\ldots). \tag{2.29}$$

Формула (2.29) получается при преобразовании исходного уравнения $y = \frac{1}{\sqrt{x}}$ к виду

$$F(x,y) \equiv \frac{1}{y^2} - x = 0.$$

В качестве начального приближения обычно берут

$$y_0 = 2^{-E(m/2)},$$

где $x = 2^m x_1,\ 1/2 \le x_1 < 1$.

Мы имеем здесь итеративный процесс также «без деления».

4. Вычисление кубического корня. Пусть имеем $y = \sqrt[3]{x}$. Применив формулу (2.25) к уравнению $F(x,y) \equiv y^3 - x = 0$, получим итерационную формулу для вычисления кубического корня в виде

$$y_{n+1} = \frac{1}{3}\left(\frac{2y_n^2 + x}{y_n^2} \right); \tag{2.30}$$

начальное приближение

$$y_0 = 2^{E(m/2)},$$

где $x = 2^m x_1,\ m -$ целое число и $1/2 \le x_1 < 1.$

ПРИМЕР 2.11 Вычислить $\sqrt[3]{5}$ с точностью до 10^{-3}.

Р е ш е н и е. Здесь $x = 5 = 2^3 \cdot \frac{5}{8}$. Начальное приближение

$$y_0 = 2^{E(3/3)} = 2,$$

$$y_1 = \frac{1}{3}\left(4 + \frac{5}{4} \right) = \frac{21}{12} = 1,7500.$$

Дальнейшие вычисления сведены в Таблица 2.5.

Таблица 2.5 Вычисление $\sqrt[3]{5}$

n	y_n	y_n^2	$3y_n^2$	y_n^3	$2y_n^3 + 5$
0	2	4	12	8	21
1	1,7500	3,0625	9,1875	5,3594	15,7188
2	1,7100	2,9241	8,7723	5,0002	15,0004
3	1,7100				

Таким образом, $\sqrt[3]{5} \approx 1,710.$

5. Вычисление корня $p -$ й степени. Пусть

$$y = \sqrt[n]{x},$$

где $x > 0$ и $p > 0 -$ целое число.

Применив формулу (2.22) и уравнению $F(x, y) \equiv 1 - \frac{x}{y^F} = 0,$ получим

$$y_{n+1} = y_n \left[\left(1 + \frac{1}{p} \right) - \frac{y_n^p}{px} \right]. \tag{2.31}$$

Итерационный процесс будет сходящимся, если только начальное приближение $y_0 > 0$ выбрать настолько малым, чтобы $y_0^p < (p+1)x.$

6. **Формула Ньютона для вычисления корня p -й степени.** Пусть $y = \sqrt[p]{x};$ тогда имеет место формула

$$y_{n+1}=\frac{1}{p}\left[(p-1)y_n+\frac{x}{y_n^{p-1}}\right], \tag{2.32}$$

получающаяся из формулы (2.28) при $F(x,y)\equiv y^p-x.$

Начальное приближение y_0 можно подобрать с точностью до одной-двух значащих цифр.

При p=2 из формулы Ньютона получаем формулу Герона.

ПРИМЕР 2.12. Вычислить $y=\sqrt[7]{277234}$ с точностью до 10^{-6}.

Р е ш е н и е. Возьмём $y_0=6;$ по формуле (5.8) последовательно вычисляем:

$$y_1=6\left[\left(1+\frac{1}{7}\right)-\frac{6^7}{7\cdot 277234}\right]=5,99164605,$$

$$y_2=5,99169225,\quad y_3=5,99169225.$$

Ответ: $\sqrt[7]{277234}\approx 5,991692.$

ПРИМЕР 2.13. Вычислить $y=\sqrt[4,78]{16234}$ с точностью до 10^{-8}.

Решение. Возьмём $y_0=7;$ тогда по формуле (2.28) будем иметь

$$y_1=\frac{1}{4,78}\left[(4,78-1)\cdot 7+\frac{16234}{7\cdot 3,78}\right]=7,70590133,$$

$$y_2=7,60319046,\quad y_3=7,60050180,\quad y_4=7,60050001,\quad y_5=7,60050001.$$

Таким образом, с точностью до 10^{-6} получаем

$$\sqrt[4,78]{16234}\approx 7,600500.$$

З А Д А Ч И

1. Пользуясь методом итераций, составить таблицы значений следующих функций с точностью до 10^{-6}.

а) $1/x,\quad x=3+2k\quad (k=0,1,2,\ldots,15)$, б) $1/x^2$ для тех же значений x, в) $1/x^3$ для тех же значений x, г) $\dfrac{x}{1+x}$, $x=0,007+0,003k\ (k=0,1,2,\ldots 15).$

2. Пользуясь методом итераций, составить таблицы значений следующих функций с точностью до 10^{-5}.

а) $\sqrt{x},\quad x=2+k\quad (k=0,1,2,\ldots,15),$

б) $x\sqrt{x}$ для тех же значений x,

в) $\sqrt{1+x^2},\quad x=0,3+0,002k\quad (k=0,1,2,\ldots,15),$

г) $\sqrt{x^2+\dfrac{1}{x}}$ для тех же значений x,

3. Пользуясь методом итераций поставить таблицы значений следующих функций с точностью до 10^{-5}.

а) $1/\sqrt{x},\ x-3+3k\quad (k=0,1,2,\ldots,15),$

б) $1/\sqrt{2+x^2},\quad x=0,3+0,002k\quad (k=0,1,2,\ldots,15),$

в) $(2x+1)/\sqrt{x^2},\quad x=3,1+0,005k\quad (k=0,1,2,\ldots,15),$

г) $1/\sqrt{x(x+1)},\ x=2,3+0,002k \quad (k=0,1,2,\ldots,15),$

4. Пользуясь методом итераций, составить таблицы значений функций с точностью до 10^{-6}.

а) $\sqrt[3]{x}, \quad x=3+k \quad (k=0,1,2,\ldots 15),$

б) $1/\sqrt[3]{x}$ для тех же значений x.

5. Пользуясь методом итераций, составить таблицы значений следующих функций с точностью до 10^{-6}.

а) $\sqrt[4]{x}, \quad x=0,05+0,02k \quad (k=0,1,2,\ldots,15),$

б) $\sqrt[5]{x}$ для тех же значений x,

в) $\sqrt[6]{x}$ для тех же значений x,

г) $\sqrt[7]{x}$ для тех же значений x.

Глава 3
РЕШЕНИЕ НЕЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Пусть f- полином или трансцендентная функция одного переменного, действительного или комплексного. Задача состоит в том, чтобы найти один или более *нулей* f, т. е. решений уравнения f(x)=0. Нахождение формул для нулей полиномов было одним из важнейших разделов итальянской математики эпохи Ренессанса. Для полиномов 2-й, 3-й и 4-й степеней ещё несколько столетий назад были найдены алгоритмы, выражающие корни посредством конечного числа квадратичных или кубичных радикалов и рациональных операций. Но только в тридцатых года прошлого века Галуа доказал невозможность подобных алгоритмов для полиномов 5-й или более высокой степени, даже если допустить в формулах радикалы с показателем n.

§ 1. Уравнения с одним неизвестным. Метод деления пополам. Метод хорд. Метод касательной. Метод простой итерации.

1.1 Уравнения с одним неизвестным

Вводные замечания

Нахождения корней нелинейных уравнений вида

$$F(x)=0$$

1.1.1 Метод деления пополам

Пусть дано уравнение

$$F(x)=0 \tag{3.1}$$

где функция *f(x)* непрерывна на *[a, b]* и $f(a)f(b) < 0$. Для нахождения корня уравнения (3.1), принадлежащего отрезку *[a, b],* делим этот отрезок пополам. Если $f(\frac{a+b}{2})=0$, т $\xi=\frac{a+b}{2}$ является корнем уравнения. Если $f=\frac{a+b}{2}\neq 0$, то выбираем ту из половин $\left[a,\frac{a+b}{2}\right]$ или $\left[\frac{a+b}{2},b\right]$, на концах которой функция *f(x)* имеет противоположные знаки. Новый суженный отрезок $[a_1,b_1]$ снова делим пополам и проводим то же рассмотрение и т. д. В результате получаем на каком-то этапе или точный корень уравнения (3.1), или же бесконечную последовательность вложенных друг в друга отрезков $[a_1,b_1]$, $[a_2,b_2]$, . . . ,$[a_n,b_n]$, . . . таких, что

$$f(a_n)f(b_n)<0\,(n=1,2,\ldots) \tag{3.2}$$

$$b_n-a_n=\frac{1}{2^n}(b-a). \tag{3.3}$$

Так как левые концы $a_1,a_2,\ldots a_n,\ldots$ образуют монотонную неубывающую ограниченную последовательность, а правые концы $b_1,b_2,\ldots b_n,\ldots$ —монотонную невозрастающую ограниченную последовательность, то в силу равенства (
(3.3) существует общий предел

$$\xi=\lim_{n\to\infty}a_n=\lim_{n\to\infty}b_n$$

Переходя к пределу при $n\to\infty$ в неравенстве (3.2), в силу непрерывности функции *f(x)* получим $[f(\xi)]^2\le 0$. Отсюда $f(\xi)=0$, т. е. ξ является корнем уравнения , причем, очевидно,

$$0\le\xi-a_n\le\frac{1}{2^n}(b-a) \tag{3.4}$$

Если корни уравнения (3.1) не отделены на отрезке [a, b], то таким способом можно найти один из корней уравнения (3.3).

Метод половинного деления практически удобно применять для грубого нахождения корня данного уравнения, так как при увеличении точности значительно возрастает объем вычислительной работы.

Заметим, что метод половинного деления легко реализуется на электронных счетных машинах. Программа вычисления составляется так, чтобы машина находила значение правой части уравнения (3.3) в середине каждого из отрезков $[a_n,b_n]\ (n=1,2,\ldots)$ и выбирала соответствующую половину его.

Пример. Методом половинного деления уточнить корень уравнения

$$f\left(x\right)\equiv x^4+x^2-x-1=0,$$

лежащий на отрезке [0,1].

Решение. Последовательно имеем:

f(0)=-1; f(1)=1;
f(0,5)=0,06+0,25-0,5-1=-1,19;
f(0,75)=0,32+0,84-0,75-1=-0,59;
f(0,875)=0,59+1,072-0,812-1=-0,304;
f(0,8438)=0,507+1,202-0,844-1=-0,135;
f(0,8594)=0,546+1,270-0,859-1=-0,043 и т.д.

Можно принять

$$\xi=\frac{1}{2}\left(0,859+0,875\right)=0,867.$$

1.1.2 Метод хорд.

Укажем более быстрый способ нахождения корня ξ уравнения *f(x) = 0,* лежащего на заданном отрезке [a, b] таком, что *f(a)f(b)* < 0.

Пусть для определенности *f(a)*< 0 и *f(b)*> 0. Тогда, вместо того чтобы делить отрезок *[a,b]* пополам, более естественно разделить его в отношении —*f(a):f(b).* Это дает нам приближённое значение корня

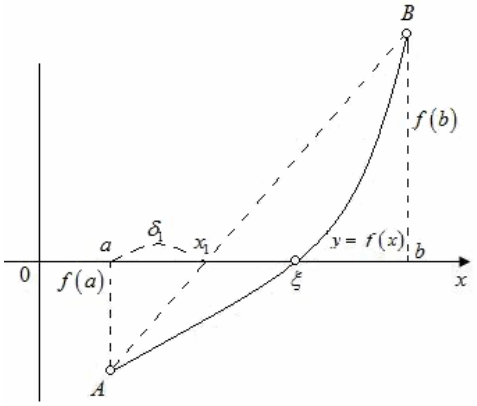


Рисунок 3.1

$$x_1=a+h_1, \tag{3.5}$$

где

$$h_1=\frac{-f(a)}{-f(a)+f(b)}(b-a)=-\frac{f(a)}{f(b)-f(a)}(b-a) \tag{3.6}$$

Далее, применяя этот прием к тому из отрезков [a,x_1] или [x_1, b], на концах которого функция *f*(x) имеет противоположные знаки**,** получим второе приближение корня x_2 и т. д.

Геометрически способ пропорциональных частей эквивалентен замене кривой *y=f(x)* хордой, проходящей через точки (**Рисунок 3.1**). В самом деле, уравнение хорды *AB* есть

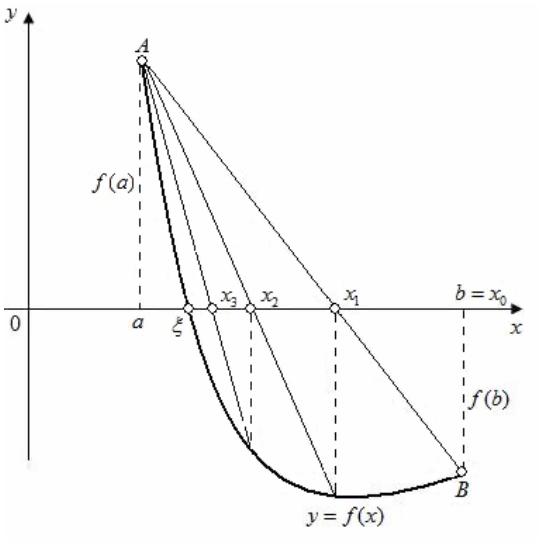


Рисунок 3.2

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

Отсюда, полагая x = x_1 и y=0 , получим:

$$x_1=a-\frac{f(a)}{\gamma(b)-\gamma(a)}(b-a). \tag{3.7}$$

Формула (3.7) полностью эквивалентна формулам (3.3) и (3.4). Для доказательства сходимости процесса предположим,что корень отделен и вторая производная *f''(x)* сохраняет постоянный знак на отрезке [a, b]. Пусть для определенности *f''(x)> 0* при $a \leq x \leq b$ (случай *f''(x)<0* сводится к нашему, если записать уравнение в виде – *f(x)=0*).

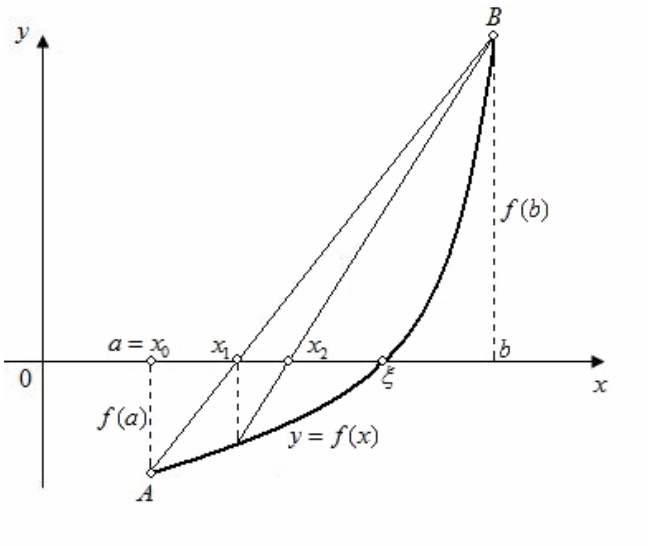


Рисунок 3.3

Тогда кривая *y=f(x)* будет выпукла вниз и, следовательно, расположена ниже своей хорды AB. Возможны два случая: 1) *f*(a)>0 (рис. 3.3) и 2) *f*(a)<0 (рис. 3.2).

В первом случае конец *a* неподвижен и последовательные приближения: $x_0 = b$;

$$x_{n+1}=x_n-\frac{f(x_n)}{\gamma(x_n)-\gamma(a)}(x_n-a)x \qquad (n=0,1,2...) \tag{3.8}$$

образуют ограниченную монотонно убывающую последовательность, причем

$$a<\xi<...<x_{n+1}<x_n<...<x_1<x_0$$

Во втором случае неподвижен конец *b*, а последовательные приближения: $x_0 = a$;

$$x_{n+1} = x_n - \frac{f(x_n)}{\gamma(b) - \gamma(x_n)}(b - x_n) \tag{3.9}$$

образуют ограниченную монотонно возрастающую последовательность, причем

$$x_0 < x_1 < x_2 < \ldots < x_n < x_{n+1} < \ldots < \xi < b.$$

Обобщая эти результаты, заключаем: 1) неподвижен тот конец, для которого знак функции f(x) совпадает со знаком ее второй производной f''(x); 2) последовательные приближения x_n лежат по ту сторону корня ξ, где функция f(x) имеет знак, противоположный знаку ее второй производной f''(x). В обоих случаях каждое следующее приближение x_{n+1} ближе к корню ξ, чем предшествующее x_n. Пусть

$$\bar{\xi} = \lim_{n\to\infty} x_n \qquad (a < \bar{\xi} < b)$$

(предел существует, так как последовательность $\{x_n\}$ ограничена и монотонна). Переходя к пределу в равенстве (3.8), для первого случая будем иметь:

$$\bar{\xi} = \bar{\xi} - \frac{f(\bar{\xi})}{\gamma(\bar{\xi}) - \gamma(a)}(\bar{\xi} - a);$$

Отсюда $f(\bar{\xi}) = 0.$ Так как по предположению уравнение f(x)=0 имеет единственный корень ξ на интервале (a, b), то, следовательно, $\bar{\xi} = \xi$, что и требовалось доказать.

Совершенно так же переходом к пределу в равенстве (3.9) доказывается, что $\bar{\xi} = \xi$ для второго случая.
Для оценки точности приближения можно воспользоваться формулой:

$$|x_n - \xi| \le \frac{|f(x_n)|}{m_1}$$

где $|f'(x)| \ge m_1$ при $a \le x \le b$.

Приведем еще формулу, позволяющую оценивать абсолютную погрешность приближенного значения x_n, если известны два последовательных приближения x_{n-1} и x_n.
Будем предполагать, что производная f'(x) непрерывна на отрезке [a, b], содержащем все приближения, и сохраняет постоянный знак, причем

$$0 < m_1 \le |f'(x)| \le M_1 < +\infty. \tag{3.10}$$

Примем для определенности, что последовательные приближения x_n точного корня ξ вырабатываются по формуле (3.8) (рассмотрение формулы (3.9) аналогично)

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{\gamma(x_n - 1) - \gamma(a)}(x_{n-1} - a)$$

(n= 1, 2, ...), где конец a является неподвижным. Отсюда, учитывая, что $f(\xi) = 0$ будем иметь:

$$f(\xi) - f(x_{n-1}) = \frac{f(x_{n-1}) - f(a)}{x_{n-1} - a}(x_n - x_{n-1}).$$

Применяя теорему Лагранжа о конечном приращении функции, получим:

$$(\xi - x_{n-1}) f'(\xi_{n-1}) = (x_n - x_{n-1}) f'(\bar{x}_{n-1}),$$

где $\xi_{n-1} \in (x_{n-1}, \xi)$ и $\bar{x}_{n-1} \in (a, x_{n-1}).$
Следовательно,

$$|\xi - x_n| = \frac{|f'(\bar{x}_{n-1}) - f'(\xi_{n-1})|}{|\gamma'(\xi_{n-1})|}|x_n - x_{n-1}|. \tag{3.11}$$

Так как f'(x) сохраняет постоянный знак на отрезке [a, b], причем $\bar{x}_{n-1} \in [a, b]$ и $\xi_{n-1} \in [a, b]$, очевидно, имеем:

$$|f'(\bar{x}_{n-1}) - f'(\xi_{n-1})| \le M_1 - m_1.$$

Поэтому из формулы (3.11) выводим:

$$|\xi - x_n| \le \frac{M_1 - m_1}{m_1}|x_n - x_{n-1}|, \qquad (3.12)$$

где за m_1 и M_1 могут быть взяты соответственно наименьшее и наибольшее значения модуля производной *f'(x)* на отрезке *[a, b]*. Если отрезок *[a, b]* столь узок, что имеет место неравенство

$$M_1 \le 2m_1,$$

то из формулы (3.7) получаем:

$$|\xi - x_n| \le |x_n - x_{n-1}|.$$

Таким образом, в этом случае, как только будет обнаружено, что

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

где ξ — заданная предельная абсолютная погрешность, то гарантировано, что

$$|\xi - x_n| < \varepsilon.$$

Пример. Найти положительный корень уравнения

$$f(x) \equiv x^3 - 0,2x^3 - 0,2x - 1,2 = 0$$

с точностью до 0,002.

Решение. Прежде всего отделяем корень. Так как

$$f(1) = -0,6 < 0 \text{ и } f(2) = 5,6 > 0,$$

то искомый корень ξ лежит в интервале (1, 2). Полученный интервал велик, поэтому разделим его пополам. Так как

$$f(1,5) = 1,425, \qquad 1 < \xi < 1,5.$$

Последовательно применяя формулы (1) и (2), будем иметь:

$$x_1 = 1 + \frac{0,6}{1,425 + 0,6}(1,5 - 1) = 1 + 0,15 = 1,15;$$

$$f(x_1) = -0,173;$$

$$x_2 = 1,15 + \frac{0,173}{1,425 + 0,073}(1,5 - 1,15) = 1,15 + 0,040 = 1,190;$$

$$f(x_2) = -0,036;$$

$$x_3 = 1,190 + \frac{0,036}{1,425 + 0,036}(1,5 - 1,190) = 1,190 + 0,008 = 1,198;$$

$$f(x_3) = -0,0072.$$

Так как $f'(x) = 3x^2 - 0,4x - 0,2$ и при $x_3 < x < 1,5$ имеем

$$f'(x) = 3x^2 - 0,4x - 0,2 = 3 \cdot 1,43 - 0,8 = 3,49,$$

то можно принять:

$$0 < \xi - x_3 < \frac{0,0072}{3,49} \approx 0,002.$$

Таким образом, $\xi = 1,198 + 0,020$, где $0 < \theta \le 1.$

Заметим, что точный корень уравнения (3.10) есть $\xi = 1,2.$

1.1.3 Метод касательной.

Его отличие от предыдущего метода состоит в том, что на *k*-й итерации вместо хорды проводится касательная к кривой *y = F(x)* при $x = c_{k-1}$ и ищется точка пересечения касательной с осью абсцисс. При этом не обязательно задавать отрезок [*a, b*], содержащий корень уравнения , а достаточно лишь найти некоторое начальное приближение корня $x = c_0$ (Рисунок 3.4).

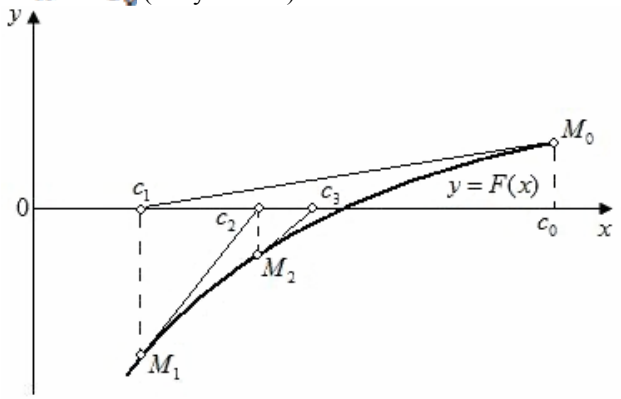


Рисунок 3.4

Уравнение касательной, проведенной к кривой *y = F(x)* в точке M_0 с координатами c_0 и $F(c_0)$, имеет вид

$$y - F(c_0) = F'(c_0)(x - c_0).$$

Отсюда найдем следующее приближение корня c_1 как абсциссу точки пересечения касательной с осью *x* (*y* = 0):

$$c_1 = c_0 - F(c_0)/F'(c_0).$$

Аналогично могут быть найдены и следующие приближения как точки пересечения с осью абсцисс касательных, проведенных в точках M_1, M_2 и т. д. Формула для *k*-го приближения имеет вид

$$c_k = c_{k-1} - F(c_{k-1})/F'(c_{k-1}), \qquad k = 1,2,\ldots \tag{3.13}$$

При этом необходимо, чтобы $F'(c_{k-1})$ не равнялась нулю.

Из формулы следует, что на каждой итерации объем вычислений в методе Ньютона больший, чем в рассмотренных ранее методах, поскольку приходится находить значение не только функции *F(x),* но и ее производной. Однако скорость сходимости здесь значительно выше, чем в других методах.

Остановимся на некоторых вопросах, связанных со сходимостью метод Ньютона и его использованием. Имеет место следующая теорема**.**

Теорема. Пусть x=c - корень уравнения , т. е. F(c) =0, а F'(c)≠ 0 и F"(x) непрерывна. Тогда существует окрестность D корня с (с $\in$ D) такая, что если начальное приближение c_0 принадлежит этой окрестности, то для метода Ньютона последовательность значений $\{c_k\}$ сходится к с при $k \to \infty$. При этом для погрешности корня $\varepsilon_k = c - c_k$ имеет место соотношение

$$\lim_{k\to\infty}\left|\frac{\varepsilon_k}{\varepsilon_{k-1}^2}\right| = \left|\frac{F''(c)}{2F'(c)}\right|.$$

Фактически это означает, что на каждой итерации погрешность возводится в квадрат, т. е. число верных знаков корня удваивается. Если

$$\left|\frac{F''(c)}{2F'(c)}\right| \approx 1,$$

то легко показать, что при $|\varepsilon_0| \le 0.5$ пяти-шести итераций достаточно для получения минимально возможной погрешности при вычислениях с двойной точностью. Действительно, погрешность теоретически станет в этом случае величиной порядка 2^{-64}, что намного меньше, чем максимальная погрешность округления при вычислениях с двойной точностью, равная 2^{-52}. Заметим, что для получения столь малой погрешности в методе деления отрезка пополам потребовалось бы более 50 итераций.

ПРИМЕР. Для иллюстрации рассмотрим уравнение $x^2 - 0.25 = 0$ и найдем методом Ньютона один из его корней, например *x*= *c*= 0.5. Для данного уравнения *F''(c)/F'(c)* = 1. Выберем c_0 = 1, тогда ε_0 = -0.5. Проводя вычисления с двойной точностью, получим следующие значения погрешностей:

ε_1 = -1.25$\cdot 10^{-1}$, ε_3 = -1.52$\cdot 10^{-4}$, ε_5 = -5.55 $\cdot 10^{-16}$,
ε_2 = -1.25$\cdot 10^{-2}$, ε_4 = -2.32$\cdot 10^{-8}$, ε_6 = 0.
Таким образом, после шести итераций погрешность в рамках арифметики с двойной точностью исчезла.
Трудность в применении метода Ньютона состоит в выборе начального приближения, которое должно находиться в окрестности *D.* При неудачном выборе начального приближения итерации могут расходиться.
ПРИМЕР. Для уравнения arctg x = 0 (корень x= c = 0) при начальном приближении c_0= 1.5 первые шесть итераций приводят к погрешностям
ε_1 = 1.69, ε_3 = 5.11, ε_5 = 1.58 $\cdot 10^3$,
ε_2 = -2.32, ε_4 = -32.3, ε_6 = -3.89 $\cdot 10^6$.
Очевидно, что итерации здесь расходятся.
Для предотвращения расходимости иногда целесообразно использовать смешанный алгоритм. Он состоит в том, что сначала применяется всегда сходящийся метод (например, метод деления отрезка пополам), а после некоторого числа итераций — быстро сходящийся метод Ньютона.

1.1.4 Метод простой итерации.

Для использования этого метода исходное нелинейное уравнение записывается в виде

$$x = f(x) \tag{3.14}$$

Пусть известно начальное приближение корня x=c_0. Подставляя это значение в правую часть уравнения (3.14), получаем новое приближение

$$c_1 = f(c_0).$$

Подставляя каждый раз новое значение корня в $x = f(x)$ (3.14), получаем последовательность значений

$$c_k = f(c_{k-1}), \qquad k = 1,2,...$$

Итерационный процесс прекращается, если результаты двух последовательных итераций близки, т. е. если выполнено неравенство $|c_k - c_{k-1}| < \varepsilon$. Заметим, что в методе простой итерации для невязки, полученной на *k-й* итерации, выполнено соотношение

$$\gamma_k = c_k - f(c_k) = c_k - c_{k+1}.$$

Таким образом, условие малости невязки на *k-й* итерации оказывается эквивалентным условию близости k-го и k +1-го приближений.
Достаточное условие сходимости метода простой итерации дается следующей теоремой.
Теорема.
Пусть x=c - корень уравнения $x = f(x)$ (3.14), т. е. c= f(c), а $|f'| < 1$ и $f'(x)$ непрерывна. Тогда существует окрестность D корня c (c $\in$ D) такая, что если начальное приближение c_0 принадлежит этой окрестности, то для метода простой итерации последовательность значений { c_k} сходится к c при к$\rightarrow \infty$.
Метод простой итерации рассмотрен нами для уравнения (3.14). К такому виду можно привести и более общее уравнение , аналогично тому, как это делалось при решении систем линейных уравнений:

$$\begin{aligned} F(x) &= 0 \\ \tau F(x) &= 0 \\ x &= x - \tau F(x) \end{aligned} \tag{3.15}$$

Здесь τ≠0 - некоторое число. Уравнение (3.15 эквивалентно 3.14 функцие $f(x) = x - \tau F(x)$. За счет выбора значения параметра τ можно добиваться сходимости метода простой итерации и повышения скорости сходимости. Например, если на некотором отрезке, содержащем корень уравнения, производная $F'(x)$ ограничена константами m и M:

$$0 < m < F'\{x) < M,$$

то для производной $f'(x)$ будет справедливо неравенство

$$1 - \tau M < f'(x) < 1 - \tau m.$$

Выбирая $\tau = 2/(M + m)$, получаем

$$-\frac{M-m}{M+m} < f'(x) < \frac{M-m}{M+m},$$

г. е. $|f'(x)| < 1$, что обеспечивает сходимость метода простой итерации.
Параметр τ в (3.15 можно выбирать и переменным, зависящим от

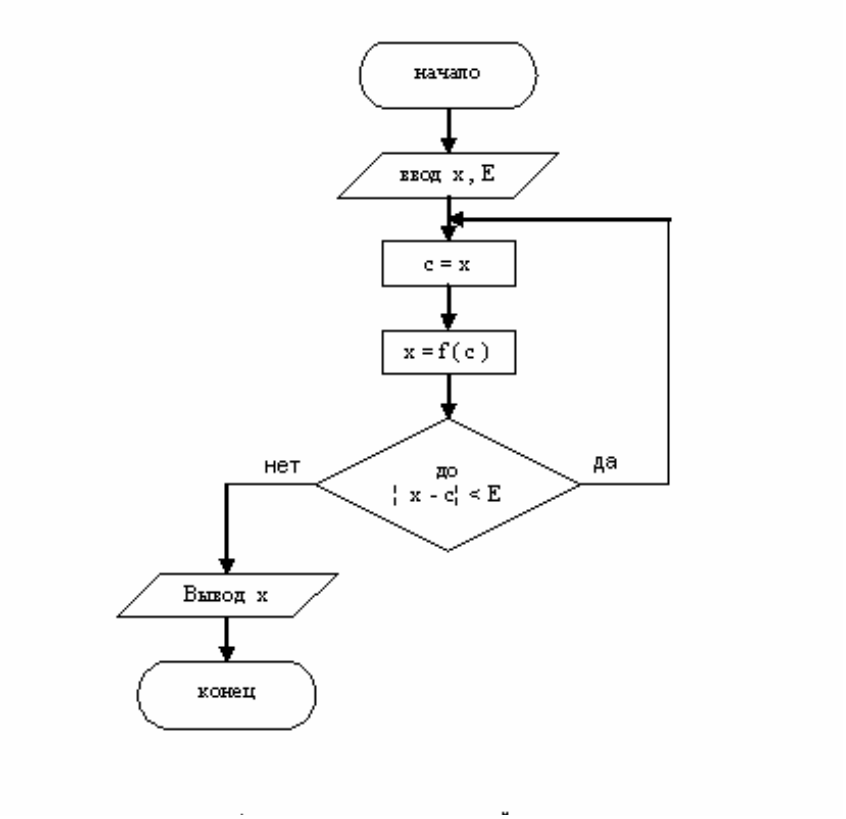


Рисунок 3.5 !!!!!!!

номера итерации. Так, если положить $\tau_k = 1/F'(c_{k-1})$, то метод простой итерации для уравнения (3.15 примет вид

$$c_k = c_{k-1} - F(c_{k-1})/F'(c_{k-1}).$$

Это соотношение совпадает с формулой метода Ньютона . Следовательно, метод Ньютона можно трактовать как частный случай метода простой итерации с переменным τ.

На рисунке 3.5 представлен алгоритм решения нелинейного уравнения 3.14 методом простой итерации. Здесь x - начальное приближение корня, а в дальнейшем — значение корня после каждой итерации, c - результат предыдущей итерации. В данном алгоритме предполагалось, что итерационный процесс сходится. Если такой уверенности нет, то необходимо ограничить число итераций и ввести для них счетчик.

§2. Действительные и комплексные корни алгебраических уравнений.

Действительные корни.

Рассмотренные выше методы решения нелинейных уравнений пригодны как для трансцендентных, так и для алгебраических уравнений. Вместе с тем при нахождении корней многочленов приходится сталкиваться с некоторыми особенностями. В частности, при рассмотрении точности вычислительного процесса отмечалась чувствительность к погрешностям значений корней многочлена. С другой стороны, по сравнению с трансцендентными функциями многочлены имеют то преимущество, что заранее известно число их корней.

Напомним некоторые известные из курса алгебры свойства алгебраических уравнений с действительными коэффициентами вида

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0. \qquad (3.16)$$

1. Уравнение степени n имеет всего n корней с учетом кратности, среди вторых могут быть как действительные, так и комплексные.
2. Комплексные корни образуют комплексно - сопряженные пары, т. е. каждому корню $x = c + id$ соответствует корень $x = c - id$.

Одним из способов решения уравнения 3.16 является *метод понижения порядка.* Он состоит в том, что после нахождения какого - либо корня $x = c$ данное уравнение можно разделить на $x - c$, понизив его порядок до n — 1. Правда, при таком способе нужно помнить о точности, поскольку даже небольшая погрешность в значении первого корня может привести к накапливанию погрешности в дальнейших вычислениях.

Рассмотрим применение метода Ньютона к решению уравнения (3.16) в соответствии с формулой (3.13) итерационный процесс для нахождения корня нелинейного уравнения (3.14) имеет вид

$$x_k = x_{k-1} - \frac{F(x_{k-1})}{F'(x_{k-1})},$$

$$F(x) = a_0 + a_1 x + \ldots + a_n x^n, \qquad F'(x) = a_1 + 2a_2 x + \ldots + n a_n x^{n-1}.$$

Для вычисления значений многочленов *F(x)* и *F'(x)* в точке $x=x_{k-1}$ может быть использована схема Горнера.

Естественно, при использовании метода Ньютона должны выполняться условия сходимости. При их соблюдении в результате численного решения получается значение того корня, который находится вблизи заданного начального приближения x_0.

Заметим, что для уменьшения погрешностей лучше сначала находить меньшие по модулю корни многочлена и сразу удалять их из уравнения, приводя его к меньшей степени. Поэтому, если отсутствует информация о величинах корней, в качестве начальных приближений принимают числа 0, ±1 и т. д.

Комплексные корни.

При использовании компьютера имеется возможность работать с комплексными числами; поэтому изложенный метод Ньютона может быть использован (с необходимым обобщением) и для нахождения комплексных корней многочленов. При этом, если в качестве начального приближения x_0 взять комплексное число, то последующие приближения и окончательное значение корня могут оказаться комплексными. Ниже рассмотрим другой подход к отысканию комплексных корней.

Комплексные корни попарно сопряженные, и при их исключении порядок уравнения уменьшается на два, поскольку оно делится сразу на квадратный трехчлен, т. е.

$$F(x) = (x^2 + px + q)(b_n x^{n-2} + \cdots + b_2) + b_1 x + b_0 \qquad (3.17)$$

Линейный остаток $b_1 x + b_0$ равен нулю, если *p, q* выражаются с помощью найденных корней:

$$p = -2c, \quad q = c^2 + d^2, \quad x = c \pm id.$$

Представление (3.17) может быть также использовано для нахождения *p, q,* а значит, и для определения корней. Эта процедура лежит в основе *метода Лина.* Суть этого метода состоит в следующем. Предположим, что коэффициенты b_0, b_1 равны нулю. Тогда, сравнивая коэффициенты при одинаковых степенях *x* многочлена *F(x)* в выражениях (3.16) и (3.17), можно получить (для упрощения выкладок $b_n = a_n = 1$)

$$\begin{aligned} b_{n-1} &= a_{n-1} - p, \\ b_{n-2} &= a_{n-2} - p b_{n-1} - q \end{aligned} \qquad (3.18)$$

$$\ldots\ldots\ldots$$

$$b_2 = a_2 - p b_3 - p b_4 ;$$

$$p = (a_1 - \frac{q b_3}{b_2}, \quad q = a_0 / b_2 \qquad (3.19)$$

В соотношения (3.19) входят коэффициенты b_2 и b_3 которые являются функциями *p* и *q*. Действительно, задав значения *p* и *q,* из соотношений (3.18) можно последовательно найти b_2 и b_3. Поэтому соотношения (3.19) представляют собой систему двух нелинейных уравнений относительно *p* и *q:*

$$p = f_1(p, q),$$
$$q = f_2(p, q).$$

Такая система в методе Лина решается методом простой итерации: задаются начальные приближения для *p, q* которые используются для вычисления коэффициентов $b_{n-1}, b_{n-2}, \ldots, b_2,$ затем из уравнений (3.19) уточняются значения *p, q.* Итерационный процесс вычисления этих величин продолжается до тех пор, пока их изменения в двух последовательных итерациях не станут малыми.

Широко распространен также другой метод, основанный на выделении квадратичного множителя $x^2 + px + q$, — *метод Бэрстоу.* Он использует метод Ньютона для решения системы двух уравнений.

§3 Системы уравнений. Метод простой итерацию. Метод Ньютона.

3.1 Системы уравнений.

Вводные замечания.

Многие практические задачи сводятся к решению *системы нелинейных уравнен*

Пусть для вычисления неизвестных $x_1, x_2, \ldots, x_n$ требуется решить систему йных уравнений

$$\begin{aligned} F_1(x_1, x_2, \ldots, x_n) &= 0, \\ F_2(x_1, x_2, \ldots, x_n) &= 0, \\ \ldots\ldots\ldots\ldots \\ F_n(x_1, x_2, \ldots, x_n) &= 0. \end{aligned} \qquad (3.20)$$

В векторной форме эту систему можно записать как

$$\mathbf{F(x) = O,}$$

где

$$\mathbf{F}=\{F_1, F_2, \dots, F_n\}, \quad x=\{x_1, x_2, \dots, x_n\}.$$

В отличие от систем линейных уравнений не существует прямых методов решения нелинейных систем общего вида. Лишь в отдельных случаях систему (3.20) можно решить непосредственно. Например, для случая двух уравнений иногда удается выразить одно неизвестное через другое и таким образом свести задачу к решению одного нелинейного уравнения относительно одного неизвестного.

Для решения систем нелинейных уравнений обычно используются итерационные методы. Ниже будут рассмотрены некоторые из них: метод простой итерации, метод Зейделя и метод Ньютона.

3.2 Метод простой итерации.

Систему уравнений (3.20) представим в виде

$$\begin{aligned} x_1 &= f_1(x_1, x_2, \dots, x_n), \\ x_2 &= f_2(x_1, x_2, \dots, x_n), \\ &\dots\dots\dots\dots \\ x_n &= f_n(x_1, x_2, \dots, x_n) \end{aligned} \qquad (3.21)$$

Для решения этой системы можно использовать *метод простой итерации,* аналогичный соответствующему методу для одного уравнения. Значения неизвестных на *k-й* итерации будут найдены с использованием их значений на предыдущей итерации $x_1^{(k-1)}, x_2^{(k-1)}, \dots, x_n^{(k-1)}$ как

$$x_i^{(k)} = f_i(x_1^{(k-1)}, x_2^{(k-1)}, \dots, x_n^{(k-1)}), i = 1,2,\dots,n \qquad (3.22)$$

Систему (3.21) можно решать и *методом Зейденя,* напоминающим метод Гаусса-Зейделя решения систем линейных уравнений. Значение $x_i^{(k)}$ находится из *i*-го уравнения системы (3.21) с использованием уже вычисленных на текущей итерации значений неизвестных. Таким образом, значения неизвестных на *k-й.* итерации будут находиться не с помощью (3.22), а с помощью соотношения

$$x_i^{(k)} = f_i(x_1^{(k)}, \dots, x_{i-1}^{(k)}, x_i^{(k-1)}, \dots, x_n^{(k-1)}), i = 1,2,\dots,n$$

Итерационный процесс в обоих методах продолжается до тех пор, пока изменения всех неизвестных в двух последовательных итерациях не станут малыми.

При использовании метода простой итерации и метода Зейделя успех во многом определяется удачным выбором начальных приближений неизвестных: они должны быть достаточно близкими к истинному решению. В Противном случае итерационный процесс может не сойтись.

3.3 Метод Ньютона.

Этот метод обладает гораздо более быстрой сходимостью, чем метод простой итерации и метод Зейделя. В случае одного уравнения *F(x)*= 0 алгоритм метода Ньютона был легко получен путем записи уравнения касательной к кривой *y =F(x).* По сути для нахождения нового приближения функция *F(x)* заменялась линейной функцией, т. е. раскладывалась в ряд Тейлора, при этом член, содержащий вторую производную, отбрасывался (как и все последующие члены). Та же идея лежит в основе метода Ньютона для системы уравнений: функции $F_i(x_1, x_2, \dots, x_n)$ раскладываются в ряд Тейлора, причем в разложении отбрасываются члены, содержащие вторые (и более высоких порядков) производные.

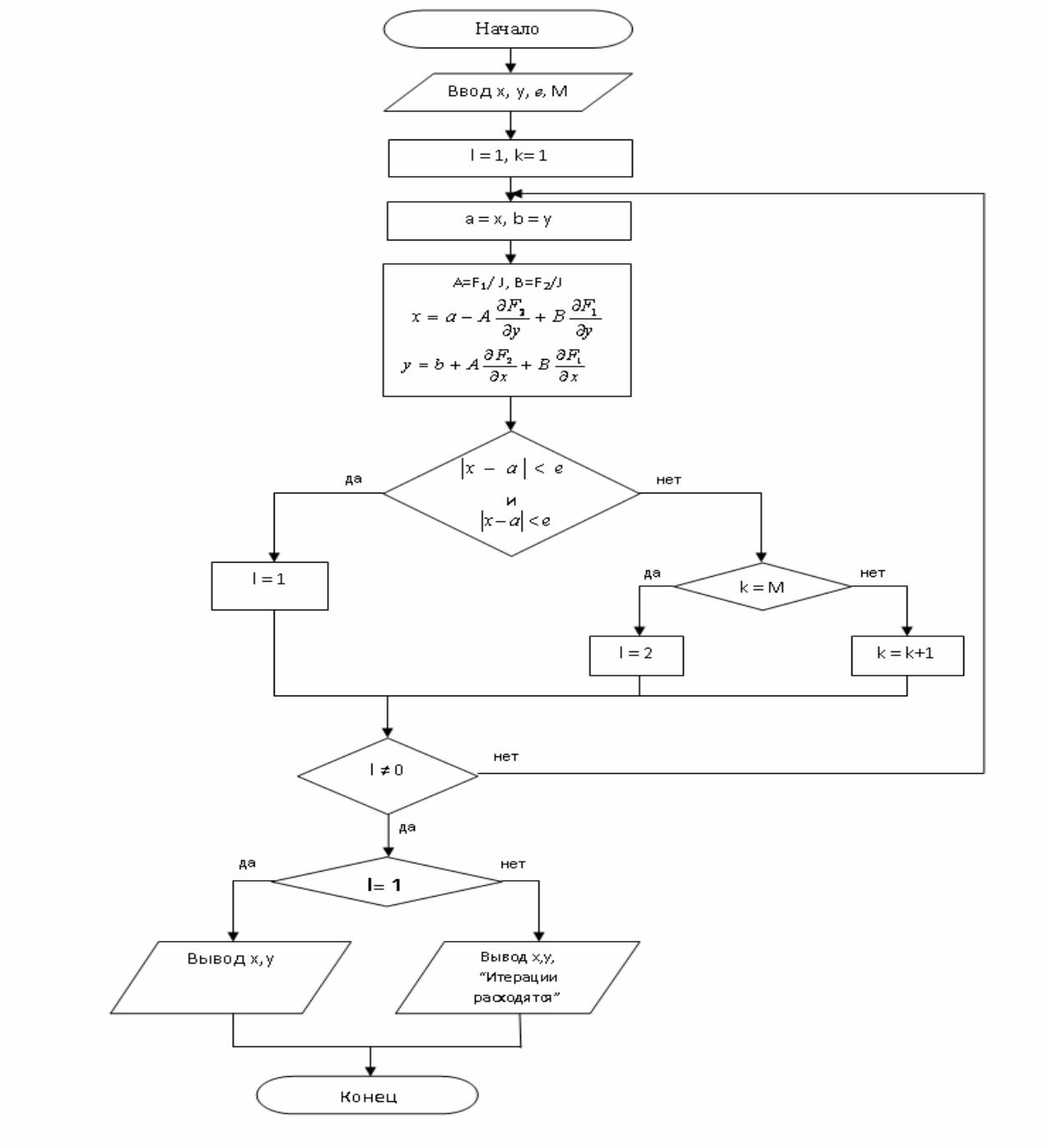


Рисунок 3.6 !!!!!!!

Пусть приближенные значения неизвестных системы (3.20), полученные на предыдущей итерации, равны соответственно $x_1^{(k-1)}, x_2^{(k-1)}, \dots, x_n^{(k-1)}$. Задача состоит в нахождении приращений (поправок) к этим значениям $\Delta x_1, \Delta x_2, \dots, \Delta x_n$, благодаря которым следующее приближение к решению системы (3.20) запишется в виде

$$\begin{aligned} x_1^{(k)} &= x_1^{(k-1)} + \Delta x_1, \\ x_2^{(k)} &= x_2^{(k-1)} + \Delta x_2, \\ &\dots \\ x_n^{(k)} &= x_n^{(k-1)} + \Delta x_n, \end{aligned} \qquad (3.23)$$

Проведем разложение левых частей уравнений (3.20) в ряд Тейлора, ограничиваясь лишь линейными членами относительно приращений:

$$F_1(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \approx F_1 + \frac{\partial F_1}{\partial x_1}\Delta x_1 + \dots + \frac{\partial F_1}{\partial x_n}\Delta x_n,$$

$$F_2(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}) \approx F_2 + \frac{\partial F_2}{\partial x_1}\Delta x_1 + \ldots + \frac{\partial F_2}{\partial x_n}\Delta x_n, \qquad (3.24)$$

. . . . . .

$$F_n(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}) \approx F_n + \frac{\partial F_n}{\partial x_1}\Delta x_1 + \ldots + \frac{\partial F_n}{\partial x_n}\Delta x_n.$$

В правых частях этих соотношений значения $F_1, F_2, \ldots, F_n$ и их производных вычисляются в точке $x^{(k-1)} = (x_1^{(k-1)}, x_2^{(k-1)}, \ldots, x_n^{(k-1)})$

Поскольку в соответствии с (3.20) левые части (3.24) должны обращаться в нуль, приравняем нулю и правые части, т. е. найдем новое приближение из условия равенства нулю разложений функций F_i. Получим следующую систему линейных алгебраических уравнений относительно приращений:

$$\frac{\partial F_1}{\partial x_1}\Delta x_1 + \frac{\partial F_1}{\partial x_2}\Delta x_2 + \ldots + \frac{\partial F_1}{\partial x_n}\Delta x_n = -F_1,$$

$$\frac{\partial F_2}{\partial x_1}\Delta x_1 + \frac{\partial F_2}{\partial x_2}\Delta x_2 + \ldots + \frac{\partial F_2}{\partial x_n}\Delta x_n = -F_2,$$

. . . . . .

$$\frac{\partial F_n}{\partial x_1}\Delta x_1 + \frac{\partial F_n}{\partial x_2}\Delta x_2 + \ldots + \frac{\partial F_n}{\partial x_n}\Delta x_n = -F_n, \qquad (3.25)$$

Определителем системы (3.25) является *якобиан.*

Для существования единственного решения системы (3.25) он должен быть отличным от нуля на каждой итерации.

Таким образом, итерационный процесс решения системы уравнений (3.20) методом Ньютона состоит в определении приращений $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$ к значениям неизвестных на каждой итерации посредством решения системы (3.25). В методе Ньютона также важен удачный выбор начального приближения для обеспечения хорошей сходимости. Сходимость ухудшается с увеличением числа уравнений системы. В качестве примера рассмотрим использование метода Ньютона для решения системы двух уравнений

$$F_1(x, y) = 0, F_1(x, y) = 0 \qquad (3.26)$$

Пусть приближенные значения неизвестных равны *a, b.* Предположим, что якобиан системы (3.26) при *x=a, y=b* отличен от нуля, т. е.

$$J = \begin{vmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial y} \end{vmatrix} \neq 0$$

Тогда следующие приближения неизвестных можно записать в виде

$$x = a - \frac{1}{J}(F_1\frac{\partial F_2}{\partial y} - F_2\frac{\partial F_1}{\partial y}),$$

$$y = b + \frac{1}{J}(F_1\frac{\partial F_2}{\partial x} - F_2\frac{\partial F_1}{\partial x}).$$

Величины, стоящие в правых частях, вычисляются при *x= a, y= b.*

Алгоритм метода Ньютона для решения системы двух уравнений изображен на рис. 3.6. В качестве исходных данных задаются начальные приближения неизвестных *x, y,* погрешность ε и допустимое число итераций *M.* В условной конструкции, проверяющей выполнение критерия завершения итерации, используется логическая операция «и», имеющаяся в современных языках программирования. Если итерации сойдутся, то выводятся значения *x, y;* в противном случае — текущие значения *x, y* и соответствующее сообщение.

Глава 4
ПРЕДСТАВЛЕНИЕ МАТРИЦ И МНОГОМЕРНЫХ МАССИВОВ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

§ 1. Представление матриц и многомерных массивов на языках С, С++.

Специального типа данных матрица или многомерный массив в Си (Си++) нет, однако, можно использовать массив элементов типа массив. Например, переменная a представляет матрицу размера 3×3 с вещественными элементами:

```
double a[3][3];
```

Элементы матрицы располагаются в памяти последовательно по строкам: сначала идут элементы строки с индексом 0, затем строки с индексом 1, в конце строки с индексом 2 (в программировании отсчет индексов всегда начинается с нуля, а не с единицы!). При этом выражение

```
a[i],
```

где i -- целая переменная, представляет собой указатель на начальный элемент i-й строки и имеет тип double*.

Для обращения к элементу матрицы надо записать его индексы в квадратных скобках, например, выражение

```
a[i][j],
```

представляет собой элемент матрицы a в строке с индексом i и столбце с индексом j. Элемент матрицы можно использовать в любом выражении как обычную переменную (например, можно читать его значение или присваивать новое).

Такая реализация матрицы удобна и максимально эффективна с точки зрения времени доступа к элементам. У нее только один существенный недостаток: так можно реализовать только матрицу, размер которой известен заранее. Язык Си не позволяет описывать массивы переменного размера, размер массива должен быть задан до начала стадии компиляции.

Пусть нужна матрица, размер которой определяется во время работы программы. Тогда пространство под нее надо захватывать в динамической памяти с помощью функции malloc языка Си или оператора new языка С++. При этом в динамической памяти захватывается линейный массив и возвращается указатель на него. Рассмотрим вещественную матрицу размером n строк m на столбцов. Захват памяти выполняется с помощью функции malloc языка **Си**

```
double *a;
. . .
a = (double *) malloc(n * m * sizeof(double));
```

или с помощью оператора new языка С++:

```
double *a;
int m, n;
. . .
a = new double[n * m];
```

При этом считается, что элементы матрицы будут располагаться в массиве следующим образом: сначала идут элементы строки с индексом 0, затем элементы строки с индексом 1 и т.д., последними идут элементы строки с индексом m − 1. Каждая строка состоит из n элементов, следовательно, индекс элемента строки i и столбца j в линейном массиве равен

```
i * n + j
```

(действительно, поскольку индексы начинаются с нуля, то i равно количеству строк, которые нужно пропустить, i * n - суммарное количество элементов в пропускаемых строках; число j равно смещению внутри последней строки). Таким образом, элементу матрицы в строке i и столбце j соответствует выражение

```
*(a + i * n + j)
```

Этот способ представления матрицы удобен и эффективен. Его основное преимущество состоит в том, что элементы матрицы хранятся в *непрерывном* отрезке памяти. Во-первых, это позволяет оптимизирующему компилятору преобразовывать текст программы, добиваясь максимального быстродействия; во-вторых, при выполнении программы максимально используется механизм кеш-памяти, сводящий к минимуму обращения к памяти и значительно ускоряющий работу программы.

В некоторых книгах по Си рекомендуется реализовывать матрицу как массив указателей на ее строки, при этом память под каждую строку захватывается отдельно в динамической памяти:

```
double **a; // Адрес массива указателей
int m, n;   // Размеры матрицы: m строк, n столбцов
int i;
. . .
// Захватывается память под массив указателей
a = (double **) malloc(m * sizeof(double *));

for (i = 0; i < m; ++i) {
    // Захватывается память под строку с индесом i
    *(a+i) = (double *) malloc(n * sizeof(double));
}
```

После этого к элементу a_{ij} можно обращаться с помощью выражения

```
*(*(a+i)+j)
```

Несмотря на всю сложность этого решения, никакого выигрыша нет, наоборот, программа проигрывает в скорости! Причина состоит в том, что матрица не хранится в непрерывном участке памяти, это мешает как оптимизации программы, так и эффективному использованию кеш-памяти. Так что лучше не применять такой метод представления матрицы.

Многомерные массивы реализуются аналогично матрицам. Например, вещественный трехмерный массив размера 4 × 4 × 2 описывается как

```
double a[4][4][2];
```

обращение к его элементу с индексами x, y, z осуществляется с помощью выражения

```
a[x][y][z]
```

Многомерные массивы переменного размера с числом индексов большим двух встречаются в программах довольно редко, но никаких проблем с их реализацией нет: они реализуются аналогично матрицам. Например, пусть надо реализовать трехмерный вещественный массив размера m × n × k. Захватывается линейный массив вещественных чисел размером m * n * k:

```
double *a;
    . . .
    a = (double *) malloc(m * n * k * sizeof(double));
```

Доступ к элементу с индексами x, y, z осуществляется с помощью выражения

```
a[(x * n + y) * k + z]
```

§ 2. Представление матриц и многомерных массивов на языке Pascal

В Паскале двумерный массив представляется массивом, элементами которого являются одномерные массивы. Два следующих описания двумерных массивов тождественны:

```
Type
Vector = array [1..5] of <тип_элементов>;
Matrix= array [1..10] of vector;
Var m: matrix;
Type
Matrix= array [1..5] of array [1..10] of < тип элементов >;
```

или еще проще:

```
type
matrix = array [1..5, 1..10] of <тип элементов>;
```

Чаще всего при описании двумерного массива используют второй способ. Доступ к каждому отдельному элементу осуществляется обращением к имени массива с указанием индексов (первый индекс – номер строки, второй индекс – номер столбца).

```
m [i, j]
```

Только для инициализации двумерного массива используется вложенный цикл for.

Например:

```
. . . .
For i:= 1 to 10 do
For j:= 1 to 20 do

m[i, j] := 0;
```

Многомерные массивы реализуются аналогично матрицам. N-мерный массив характеризуется N индексами. Формат описания такого типа данных:

```
Type

<Имя типа> = Array[ <диапазон индекса1>, <диапазон индекса2>,...
<диапазон индекса N> ] Of <тип компонент>;
```

Отдельный элемент именуется так:

```
<Имя массива>[<Индекс 1>,<Индекс 2>,...,<Индекс N>]
```

§ 3. Пример приведения матрицы к ступенчатому виду методом Гаусса на языке С.

В качестве примера работы с матрицами рассмотрим алгоритм Гаусса приведения матрицы к ступенчатому виду. Метод Гаусса - один из основных результатов линейной алгебры и аналитической геометрии, к нему сводятся множество других теорем и методов линейной алгебры (теория и вычисление определителей, решение систем линейных уравнений, вычисление ранга матрицы и обратной матрицы, теория базисов конечномерных векторных пространств и т.д.).
Напомним, что матрица A с элементами a_{ij} называется ступенчатой, если она обладает следующими двумя свойствами:
1. если в матрице есть нулевая строка, то все строки ниже нее также нулевые;
2. пусть a_{ij} не равное 0 -- первый ненулевой элемент в строке с индексом i, т.е. элементы $a_{il} = 0$ при $l < j$.

Тогда все элементы в j-м столбце ниже элемента a_{ij} равны нулю, и все элементы левее и ниже a_{ij} также равны нулю: $a_{kl} = 0$ при $k > i$ и $l =< j$.
Ступенчатая матрица выглядит примерно так: !!!!СДЕЛАТЬ СТУПЕНЧАТУЮ матрицу формулой

Здесь тёмными квадратиками отмечены первые ненулевые элементы строк матрицы. Белым цветом изображаются нулевые элементы, серым цветом - произвольные элементы.
Алгоритм Гаусса использует элементарные преобразования матрицы двух типов.

- *Преобразование первого рода*: две строки матрицы меняются местами, и при этом знаки всех элементов одной из строк изменяются на противоположные.
- *Преобразование второго рода:* к одной строке матрицы прибавляется другая строка, умноженная на произвольное число.

Элементарные преобразования сохраняют определитель и ранг матрицы, а также множество решений линейной системы. Алгоритм Гаусса приводит произвольную матрицу элементарными преобразованиями к ступенчатому виду. Для ступенчатой квадратной матрицы определитель равен произведению диагональных элементов, а ранг - числу ненулевых строк (рангом по определению называется размерность линейной оболочки строк матрицы).
Метод Гаусса в математическом варианте состоит в следующем:
1. ищем сначала *ненулевой* элемент в первом столбце. Если все элементы первого столбца нулевые, то переходим ко второму столбцу, и так далее. Если нашли ненулевой элемент в k-й строке, то при помощи элементарного преобразования первого рода меняем местами первую и k-ю строки, добиваясь того, чтобы первый элемент первой строки был отличен от нуля;
2. используя элементарные преобразования второго рода, обнуляем все элементы первого столбца, начиная со второго элемента. Для этого от строки с номером k вычитаем первую строку, умноженную на коэффициент a_{k1}/a_{11} .
3. переходим ко второму столбцу (или j-му, если все элементы первого столбца были нулевыми), и в дальнейшем рассматриваем только часть матрицы, начиная со второй строки и ниже. Снова повторяем пункты 1) и 2) до тех пор, пока не приведем матрицу к ступенчатому виду.

Программистский вариант метода Гаусса имеет три отличия от математического:
1. индексы строк и столбцов матрицы начинаются с нуля, а не с единицы;
2. недостаточно найти просто *ненулевой* элемент в столбце. В программировании все действия с вещественными числами производятся приближенно, поэтому можно считать, что точного равенства вещественных чисел вообще не бывает. Некоторые компиляторы даже выдают предупреждения на каждую операцию проверки равенства вещественных чисел. Поэтому вместо проверки на равенство нулю числа a_{ij} следует сравнивать его абсолютную величину a_{ij} с очень маленьким числом ε (например, ε = 0.00000001). Если a_{ij} =< ε, то следует считать элемент a_{ij} нулевым;
3. при обнулении элементов j-го столбца, начиная со строки i + 1, мы к k-й строке, где k > i, прибавляем i-ю строку, умноженную на коэффициент $r = -a_{kj}/a_{ij}$:

$r = -a_{kj}/a_{ij}$.

$a_k = a_k + r * a_i$

Такая схема работает нормально только тогда, когда коэффициент r по абсолютной величине не превосходит единицы. В противном случае, ошибки округления умножаются на большой коэффициент и, таким образом, экспоненциально растут. Математики называют это явление *неустойчивостью* вычислительной схемы. Если вычислительная схема неустойчива, то полученные с ее помощью результаты не имеют никакого отношения к исходной задаче. В нашем случае схема устойчива, когда коэффициент $r = -a_{kj}/a_{ij}$ не превосходит по модулю единицы. Для этого должно выполняться неравенство

$a_{ij} >= a_{kj}$ при $k > i$

Отсюда следует, что при поиске разрешающего элемента в j-м столбце необходимо найти не первый попавшийся ненулевой элемент, а *максимальный по абсолютной величине*. Если он по модулю не превосходит ε, то считаем, что все элементы столбца нулевые; иначе меняем местами строки, ставя его на вершину столбца, и затем обнуляем столбец элементарными преобразованиями второго рода.
Ниже дан полный текст программы на Си, приводящей вещественную матрицу к ступенчатому виду. Функция, реализующая метод Гаусса, одновременно подсчитывает и ранг матрицы. Программа вводит размеры матрицы и ее элементы с клавиатуры и вызывает функцию приведения к ступенчатому виду. Затем программа печатает ступенчатый вид матрицы и ее ранг. В случае квадратной матрицы также вычисляется и печатается определитель матрицы, равный произведению диагональных элементов ступенчатой матрицы.
При реализации метода Гаусса используется схема построения цикла с помощью инварианта,. В цикле меняются две переменные -- индекс строки i, 0 =< i < m - 1, и индекс столбца j, 0 =< j < n - 1. Инвариантом цикла является утверждение о том, что часть матрицы (математики говорят *минор*) в столбцах 0,1,...j - 1 приведена к ступенчатому виду и что первый ненулевой элемент в строке i - 1 стоит в столбце с индексом меньшим j. В теле цикла рассматривается только минор матрицы в строках i,...,m - 1 и столбцах j,...,n - 1. Сначала ищется максимальный по модулю элемент в j-м столбце. Если он по абсолютной величине не превосходит ε, то j увеличивается на единицу (считается, что столбец нулевой). Иначе перестановкой строк разрешающий элемент ставится на вершину j-го столбца минора, и затем столбец обнуляется элементарными преобразованиями второго рода. После этого оба индекса i и j увеличиваются на единицу. Алгоритм завершается, когда либо i = m, либо j = n. По окончании алгоритма значение переменной i равно числу ненулевых строк ступенчатой матрицы, т.е. рангу исходной матрицы.
Для вычисления абсолютной величины вещественного числа x типа double мы пользуемся стандарной математической функцией fabs(x), описанной в стандартном заголовочном файле "math.h.

```
#include <stdio.h>  // Описания функций ввода-вывода
#include <math.h>   // Описания математических функций
#include <stdlib.h> // Описания функций malloc и free

// Прототип функции приведения матрицы
// к ступенчатому виду.
// Функция возвращает ранг матрицы
int gaussMethod(
    int m,          // Число строк матрицы
    int n,          // Число столбцов матрицы
    double *a,      // Адрес массива элементов матрицы
    double eps      // Точность вычислений
);

int main() {
    int m, n, i, j, rank;
    double *a;
    double eps, det;

    printf("Введите размеры матрицы m, n: ");
```

```
    scanf("%d%d", &m, &n);

    // Захватываем память под элементы матрицы
    a = (double *) malloc(m * n * sizeof(double));

    printf("Введите элементы матрицы:\n");
    for (i = 0; i < m; ++i) {
        for (j = 0; j < n; ++j) {
            // Вводим элемент с индексами i, j
            scanf("%lf", (a+i*n + j));
        }
    }

    printf("Введите точность вычислений eps: ");
    scanf("%lf", &eps);

    // Вызываем метод Гаусса
    rank = gaussMethod(m, n, a, eps);

    // Печатаем ступенчатую матрицу
    printf("Ступенчатый вид матрицы:\n");
    for (i = 0; i < m; ++i) {
        // Печатаем i-ю строку матрицы
        for (j = 0; j < n; ++j) {
            printf(         // Формат %10.3lf означает 10
                "%10.3lf ", // позиций на печать числа,
                *(a+i*n + j)  // 3 знака после точки
            );
        }
        printf("\n");   // Перевести строку
    }

    // Печатаем ранг матрицы
    printf("Ранг матрицы = %d\n", rank);

    if (m == n) {
        // Для квадратной матрицы вычисляем и печатаем
        //     ее определитель
        det = 1.0;
        for (i = 0; i < m; ++i) {
            det *= *(a+i*n + i);
        }
        printf("Определитель матрицы = %.3lf\n", det);
    }

    free(a);    // Освобождаем память
    return 0;   // Успешное завершение программы
}

// Приведение вещественной матрицы
// к ступенчатому виду методом Гаусса с выбором
// максимального разрешающего элемента в столбце.
// Функция возвращает ранг матрицы
int gaussMethod(
    int m,          // Число строк матрицы
    int n,          // Число столбцов матрицы
    double *a,      // Адрес массива элементов матрицы
    double eps      // Точность вычислений
) {
    int i, j, k, l;
    double r;

    i = 0; j = 0;
    while (i < m && j < n) {
```

```
        // Инвариант: минор матрицы в столбцах 0..j-1
        //   уже приведен к ступенчатому виду, и строка
        //   с индексом i-1 содержит ненулевой эл-т
        //   в столбце с номером, меньшим чем j

        // Ищем максимальный элемент в j-м столбце,
        // начиная с i-й строки
        r = 0.0;
        for (k = i; k < m; ++k) {
            if (fabs(a[k*n + j]) > r) {
                l = k;      // Запомним номер строки
                r = fabs(*(a+k*n + j)); // и макс. эл-т
            }
        }
        if (r <= eps) {
            // Все элементы j-го столбца по абсолютной
            // величине не превосходят eps.
            // Обнулим столбец, начиная с i-й строки
            for (k = i; k < m; ++k) {
                *(a+k*n + j) = 0.0;
            }
            ++j;      // Увеличим индекс столбца
            continue; // Переходим к следующей итерации
        }

        if (l != i) {
            // Меняем местами i-ю и l-ю строки
            for (k = j; k < n; ++k) {
                r = a[i*n + k];
                *(a+i*n + k) = *(a+l*n + k);
                *(a+l*n + k) = (-r); // Меняем знак строки
            }
        }

        // Утверждение: fabs(*(a+i*n + k)) > eps

        // Обнуляем j-й столбец, начиная со строки i+1,
        // применяя элем. преобразования второго рода
        for (k = i+1; k < m; ++k) {
            r = (-(*(a+k*n + j)) / *(a+i*n + j));

            // К k-й строке прибавляем i-ю, умноженную на r
            *(a+k*n + j) = 0.0;
            for (l = j+1; l < n; ++l) {
                *(a+k*n + l) += r * (*(a+i*n + l));
            }
        }

        ++i; ++j;   // Переходим к следующему минору
    }

    return i; // Возвращаем число ненулевых строк
}
```

Приведем два примера работы этой программы. В первом случае вводится вырожденная матрица размера 4 × 4:

```
Введите размеры матрицы n, m: 4 4
Введите элементы матрицы:
1 2 3 4
4 3 2 1
5 6 7 8
8 7 6 5
```

```
Введите точность вычислений eps: 0.00001
Ступенчатый вид матрицы:
     8.000      7.000      6.000      5.000
     0.000      1.625      3.250      4.875
     0.000      0.000      0.000      0.000
     0.000      0.000      0.000      0.000
Ранг матрицы = 2
Определитель матрицы = 0.000
```

Во втором случае вводится матрица размера 3 × 4 максимального ранга:

```
Введите размеры матрицы n, m: 3 4
Введите элементы матрицы:
1 0 2 1
2 1 0 -1
1 0 1 0
Введите точность вычислений eps: 0.00001
Ступенчатый вид матрицы:
     2.000      1.000      0.000     -1.000
     0.000      0.500     -2.000     -1.500
     0.000      0.000     -1.000     -1.000
Ранг матрицы = 3
```

Глава 5
РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

Линейные системы. К решению систем линейных уравнений сводятся многочисленные практические задачи. Можно с полным основанием утверждать, что решение линейных систем является одной из самых распространенных и важных задач вычислительной математики.

Запишем *систему n линейных алгебраических уравнений с n неизвестными:*

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n. \end{aligned} \qquad (5.1)$$

Совокупность коэффициентов этой системы запишем в виде таблицы:

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}.$$

Данная таблица n^2 элементов, состоящая из n строк и n столбцов, называется *квадратной матрицей* порядка . Если подобная таблица содержит mn элементов, расположенных в m строках и n столбцах, то она называется *прямоугольной матрицей.*

Используя понятие матрицы *A,* систему уравнений (5.1) можно записать в матричном виде:

$$Ax = B \qquad (5.2)$$

где *x и b* — вектор-столбец неизвестных и вектор-столбец правых частей соответственно:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{pmatrix},$$

или, в более компактной записи,

$$x = \{x_1, x_2, \ldots x_n\}., \qquad b = \{b_1, b_2, \ldots b_n\}.$$

В ряде случаев получаются системы уравнений с некоторыми специальными видами матриц. Вот некоторые примеры таких матриц:

$$A=\begin{pmatrix} 2 & 1 & -1 \\ 1 & 3 & 2 \\ -1 & 2 & 4 \end{pmatrix}, \qquad B=\begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & 1 \\ 0 & 0 & 2 \end{pmatrix},$$

$$C=\begin{pmatrix} 1 & 2 & 1 & 0 & 0 & 0 \\ 2 & -1 & 2 & 0 & 0 & 0 \\ 3 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & -1 & 1 \\ 0 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 2 & 1 & 1 \end{pmatrix}, \qquad F=\begin{pmatrix} 3 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 3 & -2 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 0 & -1 & 3 \end{pmatrix},$$

$$E=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad O=\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Здесь:

A — симметрическая матрица (ее элементы расположены симметрично относительно главной диагонали ($a_{ij}=a_{ji}$));

B — верхняя треугольная матрица с равными нулю элементами, расположенными ниже диагонали;

C — клеточная матрица (ее ненулевые элементы составляют отдельные группы (клетки));

F — ленточная матрица (ее ненулевые элементы составляют «ленту», параллельную диагонали (в данном случае ленточная матрица *D* одновременно является тфакже трехдиагональной));

E — единичная матрица (частный случай диагональной);

O — нулевая матрица.

Определителем (детерминантом) матрицы *A* n-го порядка называется число *D*, равное

$$D=\det A=\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}=\sum(-1)^k a_{1\alpha}a_{2\beta}\dots a_{n\omega}. \tag{5.3}$$

Здесь индексы $\alpha, \beta, \dots, \omega$ пробегают все возможные n! перестановок номеров 1, 2, ..., n; k — число инверсий в данной перестановке.

Необходимым и достаточным условием существования единственного решения системы линейных уравнений является условие $D\neq 0$. В случае равенства нулю определителя системы матрица называется *вырожденной,* при этом система линейных уравнений (5.1) либо не имеет решения, либо имеет их бесчисленное множество.

Все эти случаи легко проиллюстрировать геометрически для системы

$$\begin{aligned} a_1x+b_1y&=c_1, \\ a_2x+b_2y&=c_2. \end{aligned} \tag{5.4}$$

Каждое уравнение описывает прямую на плоскости; координаты точки пересечения указанных прямых являются решением системы (5.4).

Рассмотрим три возможных случая взаимного расположения двух прямых на плоскости:

1) прямые пересекаются — коэффициенты системы (5.4) не пропорциональны:

$$\frac{a_1}{a_2}\neq\frac{b_1}{b_2}; \tag{5.5}$$

2) прямые параллельны — коэффициенты системы (5.4) подчиняются условиям

$$\frac{a_1}{a_2}=\frac{b_1}{b_2}\neq\frac{c_1}{c_2}; \tag{5.6}$$

3) прямые совпадают — все коэффициенты (5.4) пропорциональны:

$$\frac{a_1}{a_2}=\frac{b_1}{b_2}=\frac{c_1}{c_2}. \qquad (5.7)$$

Запишем определитель D системы (5.4) в виде

$$D=\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}.$$

Отметим, что при выполнении условия (5.5) $D\neq 0$, и система (5.4) имеет единственное решение. В случаях отсутствия решения или при бесчисленном множестве решений имеют место соответственно соотношения (5.6) или (5.7), из которых получаем $D=0$.

На практике, особенно при вычислениях на ЭВМ, когда происходят округление или отбрасывание младших разрядов чисел, далеко не всегда удается получить точное равенство определителя нулю. При $D\approx 0$ прямые могут оказаться почти параллельными (в случае системы двух уравнений); координаты точки пересечения этих прямых весьма чувствительны к изменению коэффициентов системы.

Таким образом, малые погрешности вычислений пли исходных данных могут привести к существенным погрешностям в решении. Такие системы уравнений называются *плохо обусловленными.*

Заметим, что условие $D\approx 0$ является необходимым для плохой обусловленности системы линейных уравнений, но не достаточным. Например, система уравнений n-го порядка с диагольной матрицей с элементами $a_{ii}=0.1$ не является плохо обусловленной, хотя ее определитель мал ($D=10^{-n}$).

Геометрическая иллюстрация системы двух уравнений; при малом изменении параметров одной из прямых координаты точки пересечения мало изменяются в случае a и заметно изменяются в случае b

Приведенные соображения справедливы и для любого числа уравнений системы (5.1) хотя в случае $n>3$ нельзя привести простые геометрические иллюстрации. При $n = 3$ каждое уравнение описывает плоскость в пространстве, и в случае почти параллельных плоскостей или линий их опарного пересечения получаем плохо обусловленную систему трех уравнений.

О методах решения линейных систем. Методы решения систем линейных уравнений делятся на две группы — прямые и итерационные. *Прямые методы* используют конечные соотношения (формулы) для вычисления неизвестных. Они дают решение после выполнения заранее известного числа операций. Эти методы сравнительно просты и наиболее универсальны, т. е. пригодны для решения широкого класса линейных систем.

Вместе с тем прямые методы имеют и ряд недостатков. Как правило, они требуют хранения в оперативной памяти компьютера сразу всей матрицы, и при больших значениях n расходуется много места в памяти. Далее, прямые методы обычно не учитывают структуру матрицы — при большом числе нулевых элементов в разреженных матрицах (например, клеточных или ленточных) эти элементы занимают место в памяти машины, и над ними проводятся арифметические действия. Существенным недостатком прямых методов является также накапливание погрешностей в процессе решения, поскольку вычисления на любом этапе используют результаты предыдущих операций. Это особенно опасно для больших систем, когда резко возрастает общее число операций, а также для плохо обусловленных систем, весьма чувствительных к погрешностям. В связи с этим прямые методы используются обычно для сравнительно небольших ($n \leq 200$) систем с плотно заполненной матрицей и не близким к нулю определителем.

Отметим еще, что прямые методы решения линейных систем иногда называют *точными,* поскольку решение выражается в виде точных формул через коэффициенты системы. Однако точное решение может быть получено лишь при выполнении вычислений с бесконечным числом разрядов (разумеется, при точных значениях коэффициентов системы). На практике при использовании ЭВМ вычисления проводятся с ограниченным числом знаков, определяемым разрядностью машины. Поэтому неизбежны погрешности в окончательных результатах.

Итерационные методы — это методы последовательных приближений. В них необходимо задать некоторое приближенное решение — *начальное приближение.* После этого с помощью некоторого алгоритма проводится один цикл вычислений, называемый *итерацией.* В результате итерации находят новое приближение. Итерации проводятся до получения решения с требуемой точностью. Алгоритмы решения линейных систем с использованием итерационных методов обычно более сложные по сравнению с прямыми методами. Объем вычислений заранее определить трудно.

Тем не менее итерационные методы в ряде случаев предпочтительнее. Они требуют хранения в памяти машины не всей матрицы системы, а лишь нескольких векторов с n компонентами. Иногда элементы матрицы можно совсем не хранить, а вычислять их по мере необходимости. Погрешности окончательных результатов при использовании итерационных методов не накапливаются, поскольку точность вычислений в каждой итерации определяется лишь результатами предыдущей итерации и практически не зависит от ранее выполненных вычислений. Эти достоинства итерационных методов делают их особенно полезными в случае большого числа уравнений, а также плохо обусловленных систем. Следует отметить, что при этом сходимость итераций может быть очень медленной; поэтому ищутся эффективные пути ее ускорения.

Итерационные методы могут использоваться для уточнения решений, полученных с помощью прямых методов. Такие смешанные алгоритмы обычно довольно эффективны, особенно для плохо обусловленных систем. В последнем случае могут также применяться методы регуляризации.

Другие задачи линейной алгебры. Кроме решения систем линейных уравнений существуют другие задачи линейной алгебры — вычисление определителя, обратной матрицы, собственных значений матрицы и др.
Легко вычисляются лишь определители невысоких порядков и некоторые специальные типы определителей. В частности, для определителей второго и третьего порядков соответственно имеем

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{12} - a_{12}a_{21},$$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{21}a_{32}a_{13} - a_{31}a_{22}a_{13} - a_{21}a_{12}a_{33} - a_{32}a_{23}a_{11}.$$

Определитель треугольной матрицы равен произведению ее элементов, расположенных на главной диагонали: $D=a_{11}a_{22}\ldots a_{nn}$. Отсюда также следует, что определитель единичной матрицы равен единице, а нулевой — нулю: detE=1, det O=0.
В общем случае вычисление определителя оказывается значительно более трудоемким. Определитель *D* порядка n имеет вид (5.3)

$$D = \sum (-1)^k a_{1\alpha}a_{2\beta}\ldots a_{n\omega}.$$

Из этого выражения следует, что определитель равен сумме *n*\ слагаемых, каждое из которых является произведением *n* элементов. Поэтому для вычисления определителя порядка *n* (без использования специальных приемов) требуется *(n*— 1)*n*! умножений и *n!* — 1 сложений, т. е. общее число арифметических операций равно

$$N = n \cdot n! - 1 \approx n \cdot n! \qquad (5.8)$$

Оценим значения *N* в зависимости от порядка *n* определителя:

n	3	10	20
N	17	$3.6 \cdot 10^7$	$5 \cdot 10^{19}$

Можно подсчитать время вычисления таких определителей на компьютере с заданным быстродействием. Примем для определенности среднее быстродействие равным 10 млн. операций в секунду. Тогда для вычисления определителя 10-го порядка потребуется около 3.6 сек, а при *n* = 20—свыше 150 тыс. лет.
Приведенные оценки указывают на необходимость разработки и использования экономичных численных методов, позволяющих эффективно проводить вычисления определителей.
Матрица $\boldsymbol{A}^{-1}$ называется, *обратной* по отношению к квадратной матрице *A,* если их произведение равно единичной матрице: $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = E$. В линейной алгебре доказывается, что всякая невырожденная матрица *A* (т. е. с отличным от нуля определителем *D)* имеет обратную. При этом

$$\det A^{-1} = 1/D.$$

Запишем исходную матрицу в виде

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1j} & \ldots & a_{1n} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{i1} & \ldots & a_{ij} & \ldots & a_{in} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & \ldots & a_{nj} & \ldots & a_{nn} \end{pmatrix}.$$

Минором элемента a_{ij} называется определитель (n-1)-го порядка, образованный из определителя матрицы *A* зачеркиванием i-й строки и j-го столбца.
Алгебраическим дополнением A_{ij} элемента a_{ij} называется его минор, взятый со знаком плюс, если сумма *i+j* номеров строки *i* и столбца *j* четная, и со знаком минус, если эта сумма нечетная, т. е.

$$A_{ij} = (-1)^{i+j} \begin{vmatrix} a_{11} & \dots & a_{1,j-1} & a_{1,j+1} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i-1,1} & \dots & a_{i-1,j-1} & a_{i-1,j+1} & \dots & a_{i-1,n} \\ a_{i+1,1} & \dots & a_{i+1,j-1} & a_{i+1,j+1} & \dots & a_{i+1,n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & \dots & a_{n,j-1} & a_{n,j+1} & \dots & a_{nn} \end{vmatrix}.$$

Каждый элемент z_{ij} (i,j = l, ..., *n)* обратной матрицы $Z=A^{-1}$ равен отношению алгебраического дополнения A_{ij} элемента a_{ii} (не a_{ii}) исходной матрицы *A* к значению ее определителя *D:*

$$Z = A^{-1} = \begin{pmatrix} \frac{A_{11}}{D} & \frac{A_{21}}{D} & \dots & \frac{A_{n1}}{D} \\ \frac{A_{12}}{D} & \frac{A_{22}}{D} & \dots & \frac{A_{n2}}{D} \\ \dots & \dots & \dots & \dots \\ \frac{A_{1n}}{D} & \frac{A_{2n}}{D} & \dots & \frac{A_{nn}}{D} \end{pmatrix}$$

Здесь, как и выше, можно также подсчитать число операций, необходимое для вычисления обратной матрицы без использования специальных методов. Это число равно сумме числа операций, с помощью которых вычисляются n^2 алгебраических дополнений, каждое из которых является определителем (*n*-1)-го порядка, и n^2 делений алгебраических до получений на определитель *D.* Таким образом, общее число операций для вычислений обратной матрицы равно

$$N = \left[(n-1)\cdot(n-1)!-1\right]n^2 + n^2 + n\cdot n!-1 = n^2\cdot n!-1.$$

Важной задачей линейной алгебры является также вычисление собственных значений матрицы.

§1. Прямые методы. Метод Гаусса. Метод главных диагоналей. Определитель и обратная матрица. Метод прогонки.

1.1 Прямые методы

Вводные замечания. Одним из способов решения системы линейных уравнений является *правило Крамера,* согласно которому каждое неизвестное представляется в виде отношения определителей. Запишем его для системы

$$a_1x + b_1y = c_1$$
$$a_2x + b_2y = c_2$$

Тогда

$$x = D_1 / D,\ \ y = D_1 / D,$$

$$D = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}, \quad D_1 = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}, \quad D_2 = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}.$$

Можно попытаться использовать это правило для решения систем уравнений произвольного порядка. Однако при большом числе уравнений потребуется выполнить огромное число арифметических операций, поскольку для вычислений *n* неизвестных необходимо найти значения определителей, число которых n+1. Количество арифметических операций можно оценить с учетом формулы (8). При этом предполагаем, что определители вычисляются непосредственно — без использования экономичных методов. Тогда получим

$$N = (n+1)(n\cdot n!-1) + n.$$

Поэтому правило Крамера молено использовать лишь для решения систем, состоящих из нескольких уравнений.

Известен также метод решения линейной системы с использованием обратной матрицы. Система записывается в виде $Ax = b$. Тогда, умножая обе части этого векторного уравнения слева на обратную матрицу A^{-1}, получаем x= A^{-1}b. Однако если не использовать экономичных схем для вычисления обратной матрицы, этот способ также непригоден для практического решения линейных систем при больших значениях *n* из-за большого объема вычислений.

Наиболее распространенными среди прямых методов являются метод исключения Гаусса и его модификации.

Ниже рассматривается применение метода исключения для решения систем линейных уравнений, а также для вычисления определителя и нахождения обратной матрицы.

1.2 Метод Гаусса.

Он основан на приведении матрицы системы к треугольному виду. Это достигается последовательным исключением неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается x_i из всех последующих уравнений системы. Затем с помощью второго уравнения исключается x_2 из третьего и всех последующих уравнений. Этот процесс, называемый *прямым ходом метода Гаусса,* продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным x_n, т. е. матрица системы будет приведена к треугольному виду. (Заметим, что к такому виду приводится лишь невырожденная матрица, в противном случае метод Гаусса неприменим).

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее сравнение, находим единственное неизвестное x_n. Далее, используя это значение, из предыдущего уравнения вычисляем x_{n-1} и т. д. Последним найдем x_1 из первого уравнения.

Рассмотрим применение метода Гаусса для системы

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{33}x_3 &= b_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3. \end{aligned} \tag{5.9}$$

Для исключения x_1 из второго уравнения прибавим к нему первое, умноженное на $-a_{21}/\ a_{11}$. Затем, умножив первое уравнение на $-a_{31}/\ a_{11}$ и прибавив результат к третьему уравнению, также исключаем из него x_1. Получив равносильную систему уравнений вида

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1, \\ &\quad a'_{22}x_2 + a'_{23}x_2 = b'_2, \\ &\quad a'_{32}x_2 + a'_{33}x_2 = b'_3; \\ &a'_{ij} = a_{ij} - \frac{a_{i1}}{a_{11}}a_{1j}, \quad i,j = 2,3, \\ &b'_i = b_i - \frac{a_{i1}}{a_{11}}b_1, \qquad i = 2,3. \end{aligned} \tag{5.10}$$

Теперь из третьего уравнения системы (5.10) нужно исключить x_2. Для этого умножим второе уравнение на $-a'_{32}/\ a'_{22}$ и прибавим результат к третьему. Получим

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1, \\ &a'_{22}x_2 + a'_{23}x_3 = b'_2, \\ &a''_{33}x_3 = b''_3; \\ &a''_{33} = a'_{33} - \frac{a'_{32}}{a'_{22}}a'_{23},\ b''_3 = b'_3 - \frac{a'_{32}}{a'_{22}}b'_2. \end{aligned} \tag{5.11}$$

Матрица системы (5.11) имеет треугольный вид. На этом заканчивается прямой ход метода Гаусса.

Заметим, что в процессе исключения неизвестных приходится выполнять операции деления на коэффициенты a_{11}, a'_{22} и т. д. Поэтому они должны быть отличными от нуля; в противном случае необходимо соответственным образом переставить уравнения системы. Перестановка уравнений должна быть предусмотрена в вычислительном алгоритме при его реализации на компьютере.

Обратный ход начинается с решения третьего уравнения системы (5.11)

$$x_3 = b''_3 / a''_3.$$

Используя это значение, можно найти x_2 из второго уравнения, а затем x_1 из первого:

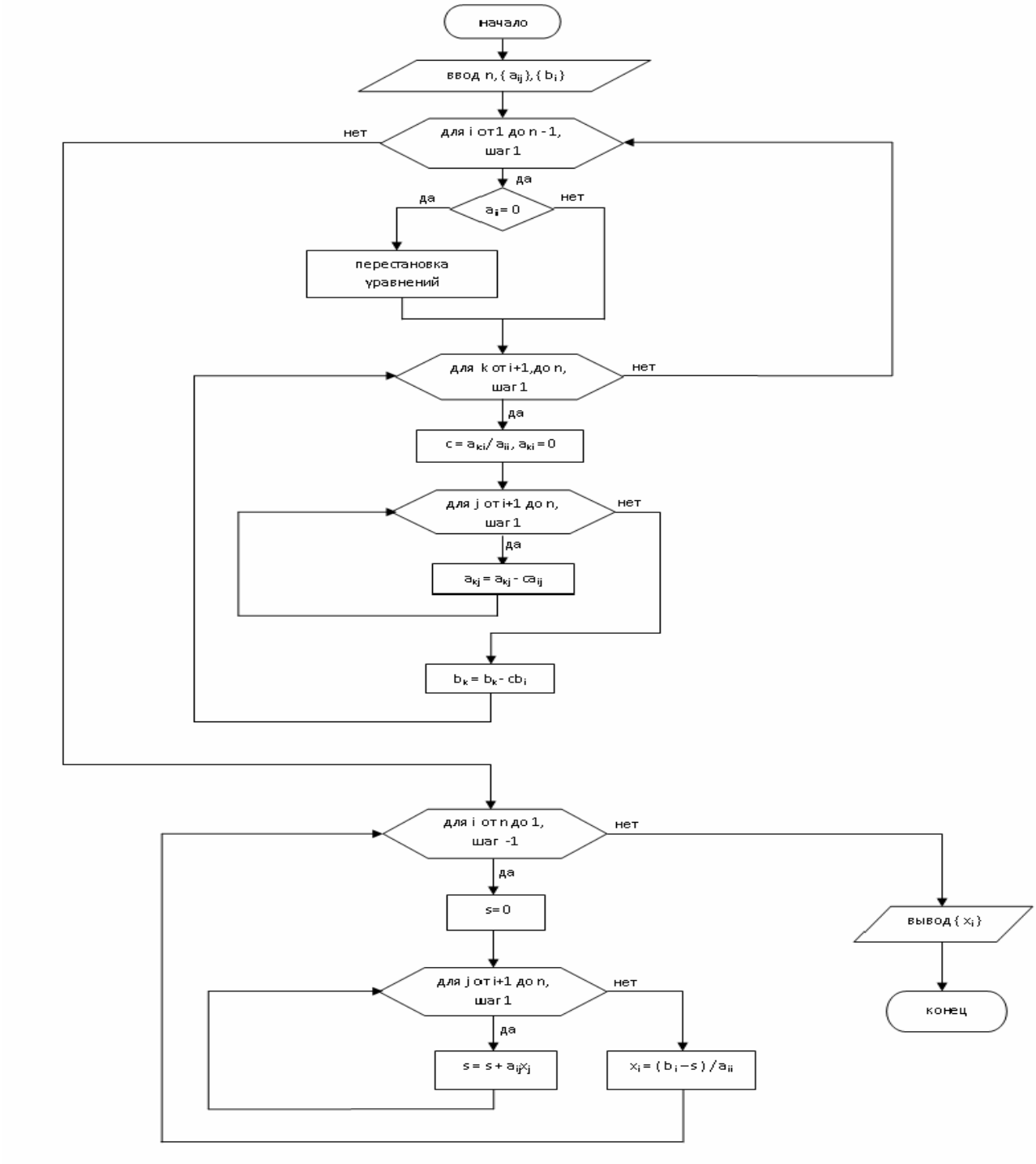


Рисунок 5.1 !!!!!

$$x_2 = \frac{1}{a_{22}^{'}}(b_2^{'} - a_{23}^{'}x_3),$$

$$x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3).$$

Аналогично строится вычислительный алгоритм для линейной системы с произвольным числом уравнений.

Левая часть блок-схемы соответствует прямому ходу. Поясним смысл индексов: i — номер уравнения, из которого исключается неизвестное x_k; j-номер столбца; k — номер неизвестного, которое исключается из оставшихся n-k уравнений (а также номер того уравнения, с помощью которого исключается из оставшихся n-k уравнений). Операция перестановки уравнений (т. е. перестановки соответствующих коэффициентов) служит для предотвращения деления на нулевой элемент. Правая часть блок-схемы описывает процесс обратного хода. Здесь i - номер неизвестного, которое определяется из i-го уравнения; j = i + 1, i + 2, ...-

номера уже найденных неизвестных.

Одной из модификаций метода Гаусса является *схема с выбором главного элемента*. Она состоит в том, что требование неравенства нулю диагональных элементов a_{kk}, на которые происходит деление в процессе исключения, заменяется более жестким: из всех оставшихся в k-м столбце элементов нужно выбрать наибольший по модулю и переставить уравнения так, чтобы этот элемент оказался на месте элемента a_{kk}.

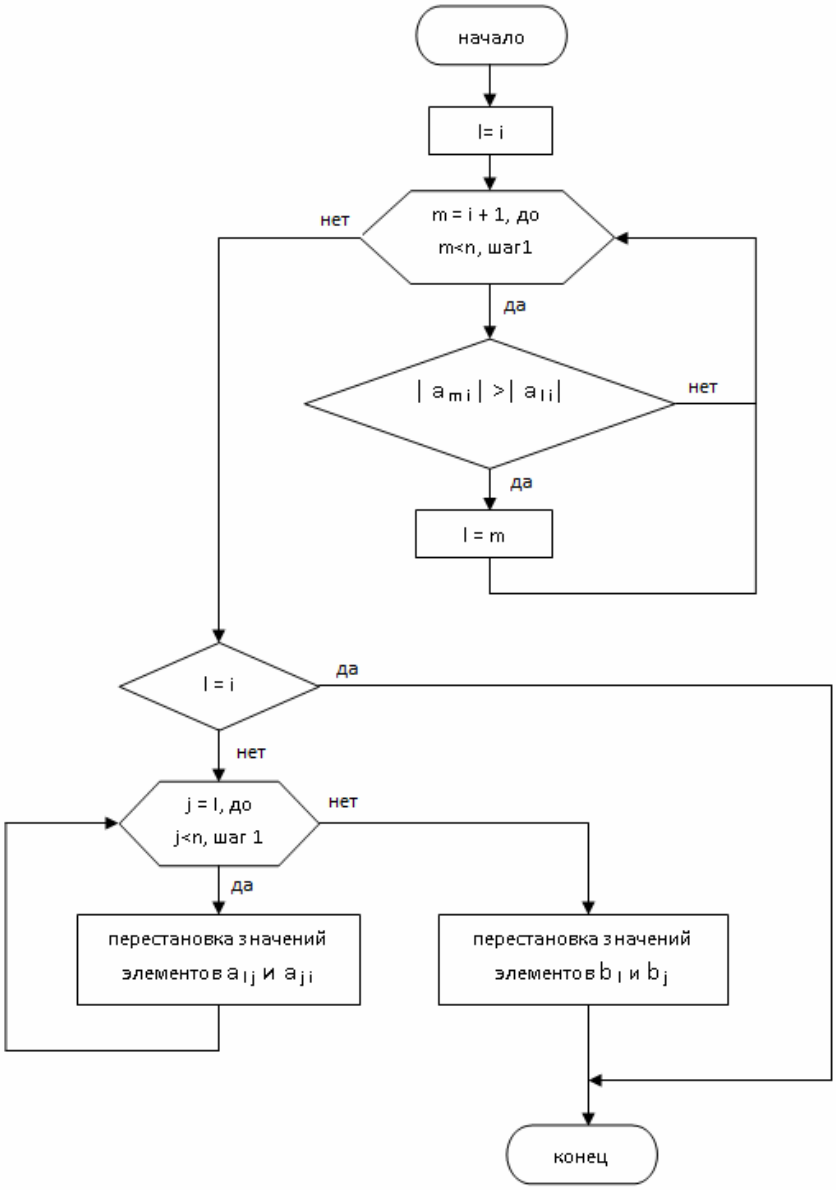


Рисунок 5.2 !!!!!

Блок-схема алгоритма выбора главного элемента приведена на Рисунок 5.1 . Она дополняет блок-схему метода Гаусса

Здесь введены новые индексы: *l*— помер наибольшего по абсолютной величине элемента матрицы в столбце с номером *k* (т. е. среди элементов a_{kk}, …, a_{km},…, a_{kn}); m — текущий номер элемента, с которым происходит сравнение. Заметим, что диагональные элементы матрицы называются *ведущими* элементами; ведущий элемент a_{kk} — это коэффициент при *k-м* неизвестном в k-м уравнении на *k*-м шаге исключения.

Благодаря выбору наибольшего по модулю ведущего элемента уменьшаются множители, используемые для преобразовании уравнений, что способствует снижению погрешностей вычислений. Поэтому метод Гаусса с выбором главного элемента обеспечивает приемлемую точность решения для сравнительно небольшого числа (n< 100) уравнений. И только для плохо обусловленных систем решения, полученные по этому методу, ненадежны.

Метод Гаусса целесообразно использовать для решения систем с плотно заполненной матрицей. Все элементы матрицы и правые части системы уравнений находятся в оперативной памяти машины. Объем вычислений определяется порядком системы n: число арифметических операций примерно равно (2/3) n^3.

Пример. Рассмотрим алгоритм решения линейной системы методом Гаусса и некоторые особенности этого метода для случая трех уравнений:

$$\begin{aligned} 10x_1 - 7x_2 &= 7, \\ -3x_1 + 3x_2 + 6x_3 &= 4, \\ 5x_1 - x_2 + 5x_3 &= 6. \end{aligned}$$

Исключим x_1 из второго и третьего уравнений. Для этого сначала умножим первое уравнение на 0.3 и результат прибавим ко второму, а затем умножим первое же уравнение на -0.5 и результат прибавим к третьему. Получим

$$\begin{aligned} 10x_1 - 7x_2 &= 7, \\ -0.1x_2 + 6x_3 &= 6.1, \\ 2.5x_2 + 5x_3 &= 2.5. \end{aligned}$$

Прежде чем исключать x_2 из третьего уравнения, заметим, что коэффициент при x_2 во втором уравнении (ведущий элемент) мал; поэтому было бы лучше переставить второе и третье уравнения. Однако мы проводим сейчас вычисления в рамках точной арифметики и погрешности округлении не опасны, поэтому продолжим исключение. Умножим второе уравнение на 25 и результат сложим с третьим уравнением. Получим систему в треугольном виде:

$$\begin{aligned} 10x_1 - 7x_2 &= 7, \\ -0.1x_2 + 6x_3 &= 6.1, \\ 155x_3 &= 155. \end{aligned}$$

На этом заканчивается прямой ход метода Гаусса.

Обратный ход состоит в последовательном вычислении x_3, x_2, x_1 соответственно из третьего, второго, первого уравнений. Проведем эти вычисления:

$$x_3 = \frac{155}{155} = 1, \;\; x_2 = \frac{6x_3 - 6.1}{0.1} = -1, \;\; x_1 = \frac{7x_2 + 7}{10} = 0.$$

Подстановкой в исходную систему легко убедиться, что (0, - 1, 1) и есть ее решение.

Изменим теперь слегка коэффициенты системы таким образом, чтобы сохранить прежним решение и вместе с тем при вычислениях использовать округления. Таким условиям, в частности, соответствует система

$$\begin{aligned} 10x_1 - 7x_2 &= 7, \\ -3x_1 + 2.099x_2 + 6x_3 &= 3.901, \\ 5x_1 - x_2 + 5x_3 &= 6. \end{aligned}$$

Здесь изменены коэффициент при x_2 и правая часть второго уравнения. Будем снова вести процесс исключения, причем вычисления проведем в рамках арифметики с плавающей точкой, сохраняя пять разрядов числа. После первого шага исключения получим

$$\begin{aligned} 10x_1 - 7x_2 &= 7, \\ -0.001x_2 + 6x_3 &= 6.001, \\ 2.5x_2 + 5x_3 &= 2.5. \end{aligned}$$

Следующий шаг исключения проводим при малом ведущем элементе (-0.001). Чтобы исключить x_2 из третьего уравнения, мы вынуждены умножить второе уравнение на 2500. При умножении получаем число 15002,5, которое нужно округлить до пяти разрядов. В результате получаем третье уравнение в виде

$$15005x_3 = 15006.$$

Отсюда x_3 = 15 006/15 005 = 1.0001. Из второго и первого уравнений найдем

$$x_2 = \frac{6.001 - 6 \cdot 1.0001}{-0.001} = -0.4, \;\; x_1 = \frac{7 + 7 \cdot (-0.4)}{10} = 0.42.$$

Вычисления проводились с усечением до пяти разрядов по аналогии с процессом вычислений на компьютере. В результате этого было получено решение (0.42, - 0.4, 1.0001) вместо (0, - 1,1).

Такая большая неточность результатов объясняется малой величиной ведущего элемента. В подтверждение этому переставим сначала уравнения системы:

$$\begin{aligned} 10x_1 - 7x_2 &= 7, \\ 2.5x_2 + 5x_3 &= 2.5, \\ -0.001x_2 + 6x_3 &= 6.001. \end{aligned}$$

Исключим теперь x_2 из третьего уравнения, прибавив к нему второе, умноженное на 0.0004 (ведущий элемент

здесь равен 2.5). Третье уравнение примет вид

$$6.002x_3 = 6.002.$$

Отсюда находим $x_3 = 1$. С помощью второго и первого уравнений вычислим x_2, x_1:

$$x_2 = \frac{2.5 - 5 \cdot 1}{2.5} = -1, \;\; x_1 = \frac{7 + 7 \cdot (-1)}{10} = 0.$$

Таким образом, в результате перестановки уравнений, т. е. выбора наибольшего по модулю из оставшихся в данном столбце элементов, погрешность решения в рамках данной точности исчезла.

Рассмотрим подробнее вопрос о погрешностях решения систем линейных уравнений методом Гаусса. Запишем систему в матричном виде: $Ax = b$. Решение этой системы можно представить в виде $x = A^{-1}b$. Однако вычисленное по методу Гаусса решение X^* отличается от этого решения из-за погрешностей округлений, связанных с ограниченностью разрядной сетки машины.

JСуществуют две величины, характеризующие степень отклонения полученного решения от точного. Одна из них — *погрешность* ΔX, равная разности этих значений, другая — *невязка* τ, равная разности между правой и левой частями уравнений при подстановке в них решения:

$$\Delta x = x - x_*, \quad \tau = Ax_* - b.$$

Можно показать, что если одна из этих величин равна нулю, то и другая должна равняться нулю. Однако из малости одной не следует малость другой. При $\Delta X \approx 0$ обычно $\tau \approx 0$, но обратное утверждение справедливо не всегда. В частности, для плохо обусловленных систем при $\tau \approx 0$ погрешность решения может быть большой.

Вместе с тем в практических расчетах, если система не является плохо обусловленной, контроль точности решения осуществляется с помощью невязки. Можно отметить, что метод Гаусса с выбором главного элемента в этих случаях дает малые невязки.

1.3 Определитель и обратная матрица.

Ранее уже отмечалось, что непосредственное нахождение определителя требует большого объема вычислений. Вместе с тем легко вычисляется определитель треугольной матрицы: он равен произведению ее диагональных элементов.

Для приведения матрицы к треугольному виду может быть использован *метод исключения,* т. е. прямой ход метода Гаусса. В процессе исключения элементов величина определителя не меняется. Знак определителя меняется на противоположный при перестановке его столбцов или строк. Следовательно, значение определителя после приведения матрицы A к треугольному виду вычисляется по формуле

$$\det A = (-1)^k \prod_{i=1}^{n} a_{ii}.$$

Здесь диагональные элементы a_{ii} берутся из преобразованной (а не исходной) матрицы. Знак зависит от того, четной или нечетной была суммарная перестановка строк (или столбцов) матрицы при ее приведении к треугольному виду (для получения ненулевого или максимального по модулю ведущего элемента на каждом этапе исключения). Благодаря методу исключения можно вычислять определители 1000-го и большего порядков, и объем вычислений значительно меньший, чем в проведенных ранее оценках.

Теперь найдем обратную матрицу A^{-1}. Обозначим ее элементы через z_{ij}. Запишем равенство $AA^{-1} = E$ в виде

$$\begin{pmatrix} a_{11} & a_{12} \ldots a_{1n} \\ a_{21} & a_{22} \ldots a_{2n} \\ \cdot & \cdot \\ a_{n1} & a_{n2} \ldots a_{nn} \end{pmatrix} \cdot \begin{pmatrix} z_{11} & z_{12} \ldots z_{1n} \\ z_{21} & z_{22} \ldots z_{2n} \\ \cdot & \cdot \\ z_{n1} & z_{n2} \ldots z_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \ldots & 1 \end{pmatrix}. \qquad (5.12)$$

Отсюда следует, что

$$Az_i = e_j, \;\; j = 1, 2, \ldots, n,$$

Где z_1 и e_i-j-е столбцы матриц A^{-1} и E соответственно. Таким образом, для нахождения j-го столбца обратной матрицы нужно решить систему уравнений (5.12). Решив n таких систем для j=1,2,…, n, мы найдём все столбцы z_i и, следовательно, саму обратную матрицу.

Поскольку при разных j матрица A системы (5.12) не меняется, исключение неизвестныхпри использовании метода Гаусса (прямой ход) проводится только один раз, причем сразу для всех правых частей – столбцов e_i. Затем для каждой из систем (5.12) делается обратный ход в соответствующей преобразованной правой частью.

Оценки показывают, что это весьма экономичный способ обращения матрицы. Он требует примерно лишь в три раза больше действий, чем при решении одной системы уравнений.

Степень отклонения вычисленной обратной матрицы A_*^{-1} от её точного значения характеризуется погрешностью $\Delta \mathbf{A}^{-1}$ и невязкой R:

$$\Delta A^{-1} = A^{-1} - A_*^{-1}, \;\; R = AA_*^{-1} - E.$$

1.4 Метод прогонки

Он является модификацией метода Гаусса для частного случая разреженных систем — системы уравнений с трехдиагональной матрицей. Такие системы получаются при моделировании некоторых инженерных задач, а также при численном решении краевых задач для дифференциальных уравнений.

Запишем систему уравнений в виде

$$\begin{aligned}
&b_1x_1 + c_1x_2 &= d_1,\\
&a_2x_1 + b_2x_2 + c_2x_3 &= d_2,\\
&\quad a_3x_2 + b_3x_3 + c_3x_4 &= d_3,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\
&\qquad a_{n-1}x_{n-2} + b_{n-1}x_{n-1} + c_{n-1}x_n &= d_{n-1},\\
&\qquad\qquad a_nx_{n-1} + b_nx_n &= d_n.
\end{aligned} \tag{5.13}$$

На главной диагонали матрицы этой системы стоят элементы b_1, b_2, ..., b_n, над ней — элементы c_1, c_2, ..., c_{n-1} под ней — элементы a_2, a_3, ..., a_n. (при этом обычно все коэффициенты b_i не равны нулю.

Метод прогонки состоит из двух этапов — *прямой прогонки* (аналога прямого хода метода Гаусса) и *обратной прогонки* (аналога обратного хода метода Гаусса). Прямая прогонка состоит в вычислении прогоночных коэффициентов A_i,B_i, с помощью которых каждое неизвестное x_i выражается через x_{i+1}:

$$x_i = A_ix_{i+1} + B_i,\ i = 1,2,...,n-1. \tag{5.14}$$

Из первого уравнения системы (5.13) найдем

$$x_1 = -\frac{c_1}{b_1}x_2 + \frac{d_1}{b_1}. \tag{5.15}$$

С другой стороны, по формуле (5.14) $x_1=A_1x_2 + B_1$. Приравнивая коэффициенты в обоих выражениях для x_1 получаем

$$A_1 = -\frac{c_1}{b_1},\ \ B_1 = \frac{d_1}{b_1}.$$

Подставим во второе уравнение системы (5.13) вместо x_1 его выражение через x_2 по формуле (5.14)

$$a_2(A_1x_2 + B_1) + b_2x_2 + c_2x_3 = d_2.$$

Выразим отсюда x_2 через x_3:

$$x_2 = \frac{-c_2x_3 + d_2 - a_2B_1}{a_2A_1 + b_2}, \tag{5.16}$$

или

$$x_2 = A_2x_3 + B_2,$$

$$A_2 = -\frac{c_2}{e_2},\ B_2 = \frac{d_2 - a_2B_1}{e_2},\ e_2 = a_2A_1 + b_2.$$

Аналогично можно вычислить прогоночные коэффициенты для любого номера i:

$$A_i = -\frac{c_i}{e_i},\ B_i = \frac{d_i - a_iB_{n-1}}{e_i},$$

$$e_i = a_iA_{i-1} + b_i,\ i = 2,3,...,n-1.$$

Обратная прогонка состоит в последовательном вычислении неизвестных x_i. Сначала нужно найти x_n. Для этого воспользуемся выражением (5.14) при i=n-1 и последним уравнением системы (5.13). Запишем их:

$$x_{n-1} = A_{n-1}x_n + B_{n-1},$$

$$a_nx_{n-1} + b_nx_n = d_n.$$

Отсюда, исключая x_{n-1}, находим

$$x_n = \frac{d_n - a_nB_{n-1}}{b_n + a_nA_{n-1}}.$$

Далее, используя формулы (5.14) и вычисленные ранее по формулам (5.15), (5.16) прогоночные коэффициенты, последовательно вычисляем все неизвестные x_{n-1}, x_{n-2}, …, x_1 Алгоритм решения системы линейных уравнений вида (5.13) методом прогонки представлен на рисунке 5.3.

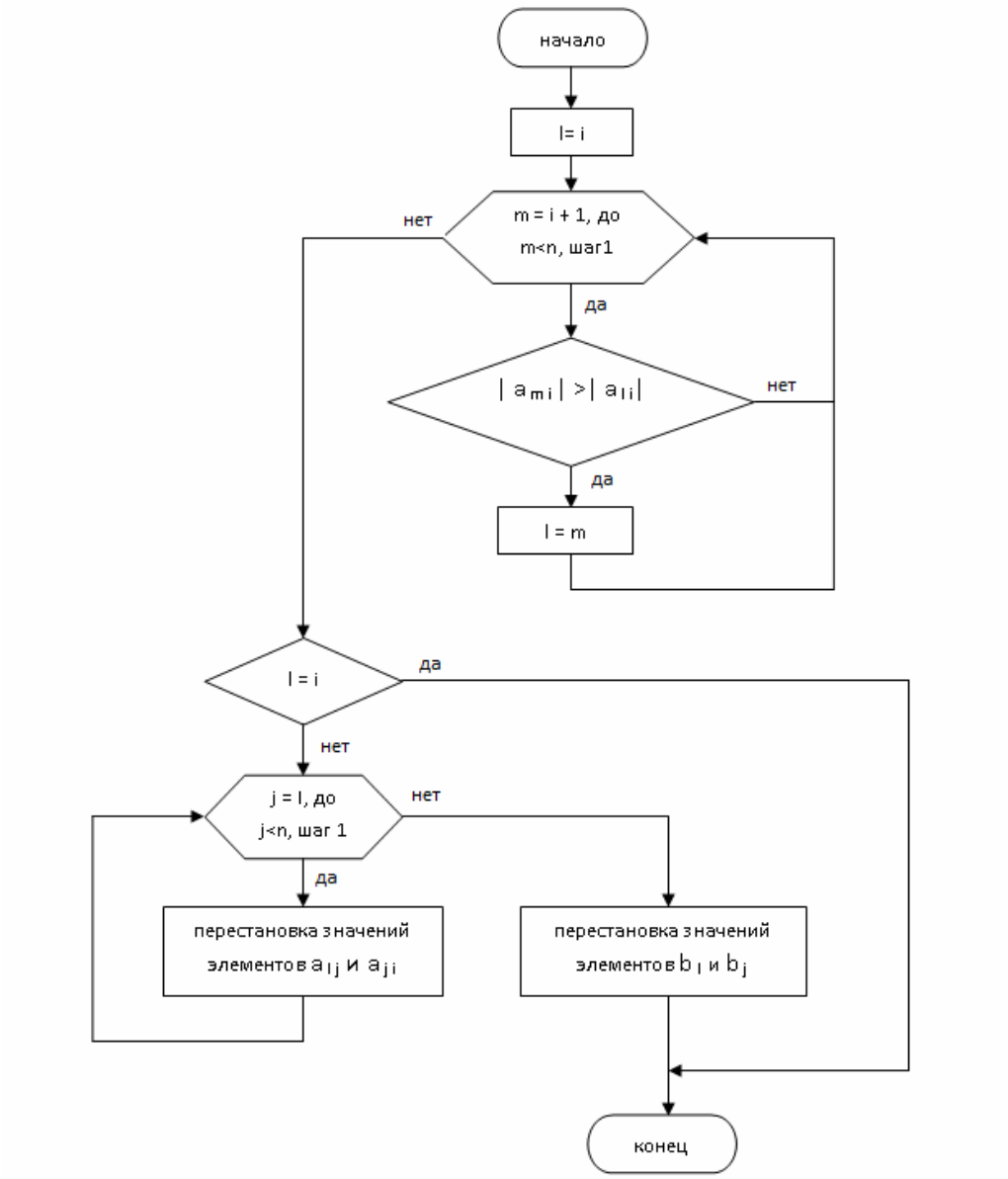


Рисунок 5.3 !!!!

При анализе алгоритма метода прогонки надо учитывать возможность деления на нуль в формулах (5.15), (5.16). Можно показать, что при выполнении условия преобладания диагональных элементов, т. е. если $|b_i| \geq |a_i| + |c_i|$, причем хотя бы для одного значения i имеет место строгое неравенство, деления на нуль не возникает, и система (5.13) имеет единственное решение.

Приведенное условие преобладания диагональных элементов обеспечивает также устойчивость метода прогонки относительно погрешностей округлений. Последнее обстоятельство позволяет использовать метод прогонки для решения больших систем уравнений. Заметим, что данное условие устойчивости прогонки является достаточным, но не необходимым. В ряде случаев для хорошо обусловленных систем вида (5.13) метод прогонки оказывается устойчивым даже при нарушении условия преобладания диагональных элементов.

§2. Итерационные методы. Уточнение решения. Метод простой итерации. Метод Гаусса-Зейделя

2.1 Уточнение решения.

Решения, получаемые с помощью прямых методов, обычно содержат погрешности, вызванные округлениями при выполнении операций над числами с плавающей точкой на компьютере с ограниченным числом разрядов. В ряде случаев эти погрешности могут быть значительными, и необходимо найти способ их уменьшения. Рассмотрим здесь один из методов, позволяющий уточнить решение, полученное с помощью прямого метода. Найдем решение системы линейных уравнений

$$Ax=b \tag{5.17}$$

Пусть с помощью некоторого прямого метода вычислено приближенное решение $x^{(0)}$ (т. е. приближенные значения неизвестных $x_1^{(0)}$, $x_2^{(0)}$,…, $x_n^{(0)}$), называемое начальным или нулевым приближением к решению. Подставляя это решение в левую часть системы (5.17) получаем некоторый столбец правых частей $b^{(0)}$, отличный от b:

$$Ax^{(0)}=b^{(0)} \tag{5.18}$$

Введем обозначения: $\Delta x^{(0)}$— погрешности значений неизвестных, $\tau^{(0)}$ невязка, т. е.

$$\Delta x^{(0)} = x - x^{(0)}, \quad \tau^{(0)} = Ax^{(0)} - b = b^{(0)} - b. \tag{5.19}$$

Вычитая равенство (5.18) из равенства (5.17), с учетом обозначений (5.19) получаем

$$A\Delta x^{(0)} = -\tau^{(0)} \tag{5.20}$$

Решая эту систему, находим значения погрешностей: $\Delta x^{(0)}$,; которое используем в качестве поправки к приближенному решению $x^{(0)}$, вычисляя таким образом новое приближенное решение $x^{(1)}$ (или следующее приближение к решению):

$$x^{(1)}=x^{(0)}+\Delta x^{(0)}.$$

Таким же способом можно найти новую поправку к решению $\Delta x^{(1)}$ и следующее приближение $x^{(2)}=x^{(1)}+\Delta x^{(1)}$ и т. д. Процесс продолжается до тех пор, пока все очередное значение погрешности (поправки) $\Delta x^{(k)}$ не станет достаточно малым, т. е. пока очередные приближенные значения неизвестных $x_1^{(k+1)}, x_2^{(k+1)}$, … , $x_n^{(k+1)}$ не будет мало отличаться от предыдущих значений $x_1^{(k)}$, $x_2^{(k)}$, … , $x_n^{(k)}$.

Рассмотренный процесс уточнения решения представляет собой фактически итерационный метод решения системы линейных уравнений. При этом заметим, что для нахождения очередного приближения, т. с. на каждой итерации, решаются системы уравнений вида (5.20) с одной и той же матрицей, являющейся матрицей исходной системы (5.17), при разных правых частях. Это позволяет строить экономичные алгоритмы. Например, при использовании метода Гаусса сокращается объем вычислений на этапе прямого хода.

Решение систем линейных уравнений с помощью рассмотренного метода (а также решение систем линейных уравнений иными итерационным методами, решение итерационными методами уравнений другого вида и их систем) сводится к следующему (Рисунок 5.4). Вводятся исходные данные, например, коэффициенты уравнений и допустимое значение погрешности. Необходимо также задать начальные приближения значений неизвестных (вектор-столбец $x^{(0)}$). Они либо водятся в компьютер, либо вычисляются каким-либо способом (в частности, путем решения системы уравнений с помощью прямого метода). Затем организуется циклический вычислительный процесс, каждый цикл которого представляет собой одну итерацию — переход от предыдущего приближениях $x^{(k-1)}$ к последующему $x^{(k)}$. Если оказывается, что с увеличением числа итераций приближенное решение стремится к точному:

$$\lim_{k\to\infty}^{(k)} = x,$$

то итерационный метод называют *сходящимся.*

На практике наличие сходимости и достижение требуемой точности обычно определяют приближенно, поступая следующим образом. При малом (с заданной допустимой погрешностью) изменении x на двух последовательных итерациях, т. е. при малом отличии $x^{(k)}$ от $x^{(k-1)}$, процесс прекращается, и происходит вывод значений неизвестных, полученных на последней итерации.

Возможны разные подходы к определению малости отличия x на двух последовательных итерациях.

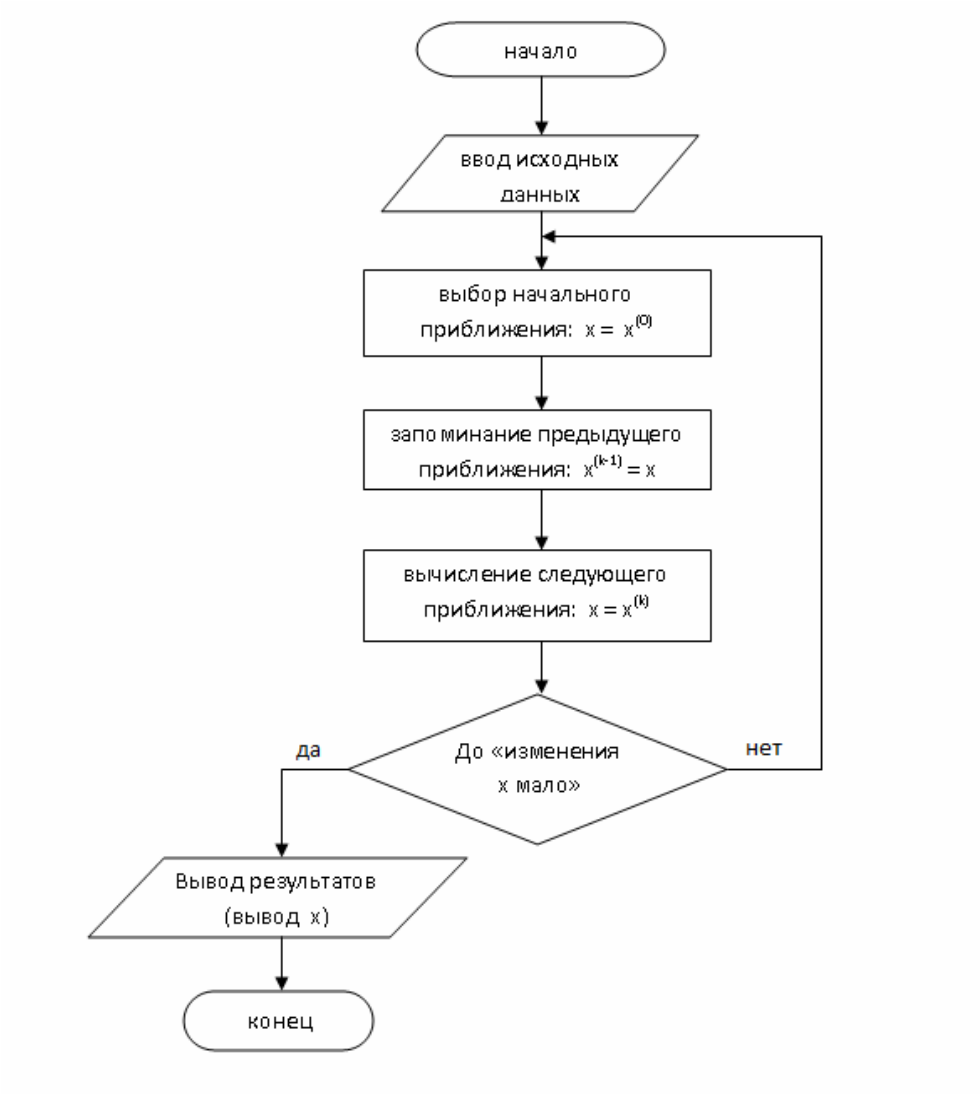


Рисунок 5.4 !!!!!

Например, если задана допустимая погрешность $\varepsilon > 0$, то критерием окончания итерационного процесса можно считать выполнение одного из трех неравенств:

$$| x^{(k)} - x^{(k-1)} | = \sqrt{\sum_{i=1}^{n} \left(x_i^{(k)} - x_i^{(k-1)} \right)^2} < \varepsilon, \tag{5.21}$$

$$\max_{1 \le i \le n} | x_i^{(k)} - x_i^{(k-1)} | < \varepsilon, \tag{5.22}$$

$$\max_{1 \le i \le n} \left| \frac{x_i^{(k)} - x_i^{(k-1)}}{x_i^{(k)}} \right| < \varepsilon, \ при \ | x_i | >> 1. \tag{5.23}$$

Здесь в первом случае отличие векторов $x^{(k)}$ и $x^{(k-1)}$ «на ε» понимался в смысле малости модуля их разности, во втором - в смысле малости разностей всех соответствующих компонент векторов, в третьем — в смысле малости относительных разностей компонент. Если система не является плохо обусловленной, то в качестве критерия окончания итерационного процесса можно использовать и условие малости невязки, например

$$| r^{(k)} | < \varepsilon. \tag{5.24}$$

Заметим, что в рассмотренном алгоритме не предусмотрен случай отсутствия сходимости. Для предотвращения непроизводительных затрат машинного времени в алгоритм вводят счетчик числа итераций и при достижении им некоторого заданного значения счет прекращают. Такой элемент будет в дальнейшем введен в структурограмму.

2.2 Метод простой итерации.

Этот метод широко используется для численного решения уравнений и их систем различных видов. Рассмотрим применение метода простой итерации к решению систем линейных уравнений.
Запишем исходную систему уравнений в векторно-матричном виде, выполним ряд тождественных преобразований:

$$\begin{aligned} &Ax = b; \quad 0 = b - Ax; \quad x = b - Ax + x; \\ &x = (b - Ax)\tau + x; \quad x = (E - \tau A)x + \tau b; \\ &x = Bx + \tau b, \end{aligned} \tag{5.25}$$

где $\tau \neq 0$ – некоторое число, E — единичная матрица, $B=E-\tau A$. получившаяся система (5.25) эквивалентна исходной системе и служит основой для построения метода простой итерации.
Выберем некоторое начальное приближение $x^{(0)}$ и подставим его в правую часть системы (5.25):

$$x^{(1)} = Bx^{(0)} + \tau b.$$

Поскольку $x^{(0)}$ не является решением системы, в левой части (5.25) получится некоторый столбец $x^{(1)}$, в общем случае отличный от $x^{(0)}$. Полученный столбец $x^{(1)}$, будем рассматривать в качестве следующего (первого) приближения к решению. Аналогично, по известному k - му приближению можно найти (k+1)-е приближение:

$$x^{(k+1)} = Bx^{(x)} + \tau b, \;\; k = 0,1,2,... \tag{5.26}$$

Формула (5.26) и выражает собой метод простой итерации. Для ее применения нужно задать неопределенный пока параметр τ. От значения τ зависит, будет ли сходиться метод, а если будет, то какова будет *скорость сходимости* т. е. как много итераций нужно совершить для достижения требуемой точности. В частности, справедлива следующая теорема.
Теорема. Пусть det A≠0. Метод простой итерации (5.26) сходится тогда и только тогда, когда все собственные числа матрицы B=A-τE по модулю меньше единицы.
Для некоторых типов матрицы A можно указать правило выбора τ, обеспечивающее сходимость метода и оптимальную скорость сходимости. В простейшем же случае τ можно положить равным некоторому постоянному числу, например, 1, 0.1 и т. д.

2.3 Метод Гаусса-Зейделя.

Одним из самых распространенных итерационных методов, отличающийся простотой и легкостью программирования, является *метод Гаусса-Зейделя.*
Проиллюстрируем сначала этот метод па примере решения системы

$$\begin{aligned} &a_{11}x_1 + a_{12}x_1 + a_{13}x_3 = b_1, \\ &a_{21}x_1 + a_{22}x_1 + a_{23}x_3 = b_2, \\ &a_{31}x_1 + a_{32}x_1 + a_{33}x_3 = b_3. \end{aligned} \tag{5.27}$$

Предположим, что диагональные элементы *ац, а22, озз* отличны от нуля (в противном случае можно переставить уравнения). Выразим неизвестные x_1, x_2, и x_3 в соответственно из первого, второго и третьего уравнений системы (5.27):

$$x_1 = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2 - a_{13}x_3\right), \tag{5.28}$$

$$x_2 = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1 - a_{23}x_3\right), \tag{5.29}$$

$$x_3 = \frac{1}{a_{33}}\left(b_3 - a_{31}x_1 - a_{32}x_2\right), \tag{5.30}$$

Зададим некоторые начальные (нулевые) приближения значений неизестных: $x_1 = x_1^{(0)}, x_2 = x_2^{(0)}, x_3 = x_3^{(0)}$. Подставляя эти значения в правую часть выражения (5.28), получаем новое (первое) приближение для x_1:

$$x_1^{(1)} = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)}\right).$$

Используя это значение для x_1 и приближение $x_3^{(0)}$ для x_3, находим из (5.29) первое приближение для x_2:

$$x_2^{(1)} = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(0)}\right).$$

И наконец, используя вычисленные значения $x_1 = x_1^{(0)}, x_2 = x_2^{(0)}$, находим с помощью выражения (5.30) первое приближение для x_3:

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

На этом заканчивается первая итерация решения системы (5.28) - (5.30). Используя теперь значения $x_1^{(1)}$, $x_2^{(1)}, x_2^{(1)}$, можно таким же способом провести вторую итерацию, в результате которой будут найдены вторые приближения к решению: $x_1 = x_1^{(2)}, x_2 = x_2^{(2)}, x_3 = x_2^{(2)}$ и т. д.

Приближение с номером *k* можно вычислить, зная приближение с номером *k-1,* как

$$x_1^{(k)} = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)}\right),$$

$$x_2^{(k)} = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k-1)}\right),$$

$$x_3^{(k)} = \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)}\right).$$

Итерационный процесс продолжается до тех пор, пока значения $x_1^{(k)}, x_2^{(k)}, x_3^{(k)}$ не станут близкими с заданной погрешностью к значениям $x_1^{(k-1)}, x_2^{(k-1)}, x_3^{(k-1)}$.

ПРИМЕР. Решить с помощью метода Гаусса-Зейделя следующую систему уравнений:

$$4x_1 - x_2 + x_3 = 4,$$
$$2x_1 + 6x_2 - x_3 = 7,$$
$$x_1 + 2x_2 - 3x_3 = 0.$$

Легко проверить, что решение данной системы следующее: $x_1 = x_2 = x_3 = 1$.

Решение. Выразим неизвестные x_1, x_2 и x_3 соответственно из первого, второго и третьего уравнений:

$$x_1 = \frac{1}{4}\left(4 + x_2 - x_3\right), \quad x_2 = \frac{1}{6}\left(7 - 2x_1 + x_3\right),$$

$$x_3 = \frac{1}{3}(x_1 + 2x_2).$$

В качестве начального приближения (как это обычно делается) примем $x_1^{(0)} = 0$, $x_2^{(0)} = 0$, $x_3^{(0)} = 0$. Найдем новые приближения неизвестных:

$$x_1^{(1)} = \frac{1}{4}(4 + 0 - 0) = 1, \quad x_2^{(1)} = \frac{1}{6}(7 - 2 \cdot 1 + 0) = \frac{5}{6},$$

$$x_3^{(1)} = \frac{1}{3}(1 + 2 \cdot \frac{5}{6}) = \frac{8}{9}.$$

Аналогично вычислим следующие приближения:

$$x_1^{(2)} = \frac{1}{4}(4 + \frac{5}{6} - \frac{8}{9}) = \frac{71}{72}, \quad x_2^{(2)} = \frac{1}{6}(7 - 2 \cdot \frac{71}{72} + \frac{8}{9}) = \frac{71}{72},$$

$$x_3^{(2)} = \frac{1}{3}(\frac{71}{72} + 2 \cdot \frac{71}{72}) = \frac{71}{72}.$$

Итерационный процесс можно продолжать до получения малой разности между значениями неизвестных в двух последовательных итерациях. Рассмотрим теперь систему *n* линейных уравнений с n неизвестными. Запишем ее в виде

$$a_{i1}x_1 + \ldots + a_{i,i-1}x_{i-1} + a_{ii}x_i + a_{i,i+1}x_{i+1} + \ldots + a_{in}x_n = b_i,$$
$$i = 1, 2, \ldots, n.$$

Здесь также будем, предполагать, что все диагональные элементы отличны от нуля. Тогда в соответствии с методом Гаусса-Зейделя *k-c* приближение к решению можно представить в виде

$$x_i^{(k)} = \frac{1}{a_{ii}}(b_i - a_{i1}x_1^{(k)} - \ldots - a_{i,i-1}x_{i-1}^{(k)} - a_{i,i+1}x_{i+1}^{(k-1)} - \ldots - a_{in}x_n^{(k-1)}),$$
$$i = 1,2,\ldots,n. \qquad (5.31)$$

Итерационный процесс продолжается до тех пор, пока все значения $x_i^{(k)}$ не станут близкими к $x_i^{(k-1)}$, т. е. в качестве критерия завершения итераций используется одно из условий (5.21) - (5.23), (5.24).

Для сходимости итерационного процесса (5.31) достаточно, чтобы модули диагональных коэффициентов для каждого уравнения системы были не меньше сумм модулей всех остальных коэффициентов (преобладание диагональных элементов):

$$|a_{ii}| \geq \sum_{j \neq i} |aij|, \quad i = 1,2,\ldots,n. \qquad (5.32)$$

При этом хотя бы для одного уравнения неравенство должно выполняться строго. Эти условия являются достаточными для сходимости метода, но они не являются необходимыми, т. е. для некоторых систем итерации сходятся и при нарушении условий (5.32).

Алгоритм решения системы *n* линейных уравнений методом Гаусса-Зейделя представлен на рисунке 5.5. В качестве исходных данных вводятся n, коэффициенты и правые части уравнений системы, погрешность ε, максимально допустимое число итераций M, а также начальные приближения переменных x_i *(i* = 1,2,...,n). Отметим, что начальные приближения можно не вводить в компьютер, а полагать их равными некоторым значениям (например, нулю). Критерием завершения итераций выбрано условие (22),

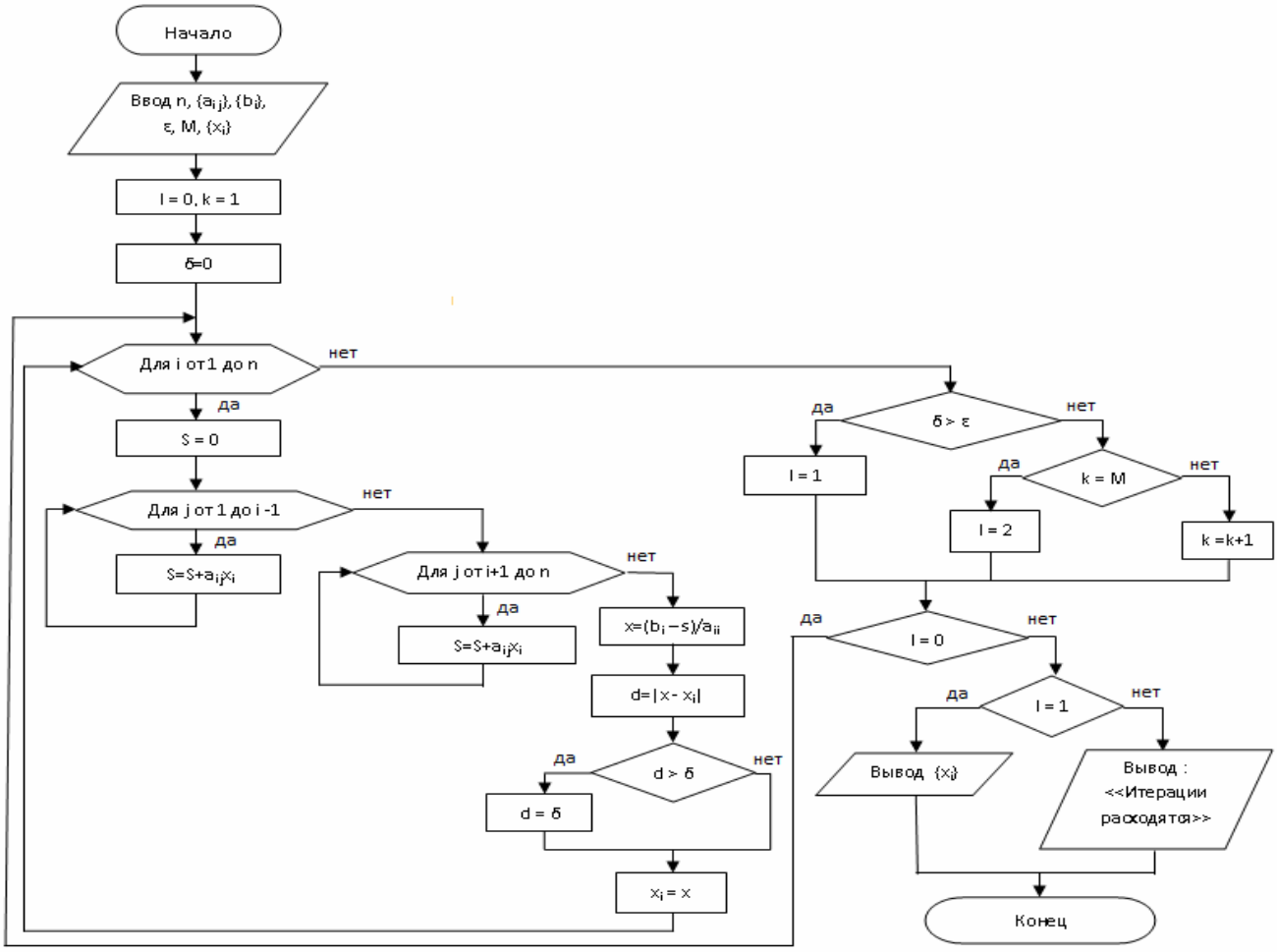


Рисунок 5.5 !!!!

в котором через δ обозначена максимальная абсолютная величина разности $x_i^{(k)}$ и $x_i^{(k-1)}$:

$$\delta = \max_{1 \leq i \leq n} |x_i^{(k)} - x_i^{(k-1)}| < \varepsilon$$

Для удобства чтения струкгурограммы объясним другие обозначения: *k*- порядковый номер итерации; *i* — номер уравнения, а также переменного, которое вычисляется в соответствующем цикле; *j* — номер члена вида в правой части соотношения (5.31). Итерационный процесс прекращается либо при $\delta < \varepsilon$, либо при *к* = *M*. В

последнем случае итерации не сходятся, о чем выдается сообщение. Для завершения цикла, реализующего итерационный процесс, используется переменная *l,* которая принимает значения 0, 1 и 2 соответственно при продолжении итераций, при выполнении условия $\delta < \varepsilon$ и при выполнении условия *k=M.*

§3. Задачи на собственные значения. Метод вращений. Трехдиагональные матрицы.

3.1 Задачи на собственные значения.

Основные понятия. Большое число научно-технических задач, а также некоторые исследования в области вычислительной математики требуют нахождения собственных значений и собственных векторов матриц. Введем некоторые определения, необходимые для изложения материала данного параграфа.

Рассмотрим, квадратную матрицу n-го

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}. \qquad (5.33)$$

Вектор x=$\{x_1, x_2, \dots, x_n\}$ называется *собственным вектором* матрицы A соответствующим *собственному значению* λ, если он удовлетворяет системе уравнений

$$Ax = \lambda x. \qquad (5.34)$$

Поскольку при умножении собственного вектора на скаляр он остается собственным вектором той же матрицы, его можно нормировать. В частности, каждую координату собственного вектора можно разделить на максимальную из них или на длину вектора; в последнем случае получится единичный собственный вектор.

Характеристической матрицей C данной матрицы *A* называется матрица вида

$$C = A - \lambda E = \begin{pmatrix} a_{11} - \lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - \lambda & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} - \lambda \end{pmatrix} \qquad (5.35)$$

где *E* — единичная матрица. Легко видеть, что систему (5.34) можно записать в виде

$$(A - \lambda E)x = 0 \;\; или \;\; Cx = 0. \qquad (5.36)$$

Если перейти к координатной форме записи вектора x, то с учетом (5.33) систему (5.36) можно записать в виде

$$\begin{aligned} &(a_{11} - \lambda)x_1 + a_{12}x_2 + \dots + a_{1n}x_n = 0, \\ &a_{21}x_1 + (a_{22} - \lambda)x_2 + \dots + a_{2n}x_n = 0, \\ &\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots, \\ &a_{n1}x_1 + a_{n2}x_2 + \dots + (a_{nn} - \lambda)x_n = 0. \end{aligned} \qquad (5.37)$$

Система (5.36) или (5.37) является однородной системой *n* линейных уравнений с *n* неизвестными. Она имеет ненулевые решения лишь тогда, когда ее определитель равен нулю: det *C* =0, причем решение не единственно.

Определитель матрицы *C* является многочленом n-й степени относительно λ:

$$\det C = c_0\lambda^n + c_1\lambda^{n-1} + \dots + c_{n-1}\lambda + c_n, \qquad (5.38)$$

называемым *характеристическим многочленом.* Корни этого многочлена являются собственными значениями матрицы *A*.

Для нахождения собственных векторов матрицы требуется решить систему линейных алгебраических уравнений, решение которой не единственно. Из линейной алгебры известно, что в этом случае структура общего решения системы имеет следующий вид: одно или несколько неизвестных, называемых *свободными,* могут принимать любые значения, а остальные неизвестные выражаются через свободные. Число свободных неизвестных равно числу уравнений системы, являющихся следствием остальных уравнений. На практике, если свободное неизвестное одно (что часто и бывает), его полагают равным некоторому числу, например единице. После этого остальные неизвестные (компоненты вектора) находятся однозначно из подсистемы линейно независимых уравнений, в которой отброшено уравнение, являющееся следствием остальных. Эта процедура не влияет на результат решения задачи, поскольку, как уже отмечалось, собственные векторы находятся с точностью до постоянного множителя.

Пример. Вычислить собственные числа и собственные векторы матрицы

$$A = \begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix}.$$

Решение. Составим характеристический многочлен

$$\begin{vmatrix} 3-\lambda & 1 \\ 2 & 4-\lambda \end{vmatrix} = (3-\lambda)(4-\lambda) - 2 = \lambda^2 - 7\lambda + 10.$$

Найдем корни этого многочлена второй степени:

$$\lambda^2 - 7\lambda + 10 = 0,\ \lambda_1 = 2,\ \lambda_2 = 5.$$

Дня нахождения собственных векторов x_1, x_2, соответствующих собственным значениям λ_1, λ_2 составим системы уравнений типа (5.36), (5.37) для каждого из них.

При $\lambda_1 = 2$ получим

$$\begin{pmatrix} 3-2 & 1 \\ 2 & 4-2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

или, в координатной форме,

$$x_1 + x_2 = 0,$$
$$2x_1 + 2x_2 = 0.$$

Замечаем, что уравнения линейно зависимы. Поэтому оставляем лишь одно из них.

Из первого уравнения следует, что $x_2 = -x_1$. Неизвестное x_1 можно считать свободным, полагаем $x_1 = 1$. Тогда $x_2 = -1$, и собственный вектор, соответствующий собственному значению $\lambda_1 = 2$, имеет вид $x_1 = \{1, -1\}$ или $x_1 = e_1 - e_2$, где e_1, e_2 единичные орты выбранной базисной системы.

Аналогично находим второй собственный вектор, соответствующий собственному значению $\lambda_2 = 5$. Опуская комментарии, получаем

$$\begin{pmatrix} 3-5 & 1 \\ 2 & 4-5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \begin{matrix} -2x_1 + x_2 = 0, \\ 2x_1 - x_2 = 0. \end{matrix}$$

Отсюда $x_1 = 1$, $x_2 = 2$, $x_2 = e_1 + 2e_2$.

Вектор x_1 нормирован; нормируем также вектор x_2, разделив его компоненты на наибольшую из них. Получим $x_2 = 0.5e_1 + e_2$. Можно также привести векторы к единичной длине, разделив их компоненты на значения модулей векторов. В этом случае

$$x_1 = \frac{1}{\sqrt{2}}(e_1 - e_2),\ x_2 = \frac{1}{\sqrt{5}}(e_1 + 2e_2).$$

Мы рассмотрели простейший пример вычисления собственных значений и собственных векторов для матрицы второго порядка. Нетрудно также провести подобное решение задачи для матрицы третьего порядка и для некоторых весьма специальных случаев.

В общем случае, особенно для матриц высокого порядка, задача о нахождении их собственных значений и собственных векторов, называемая *полной проблемой собственных значений,* значительно более сложная.

На первый взгляд может показаться, что вопрос сводится к вычислению корней многочлена (5.38). Однако здесь задача осложнена тем, что среди собственных значений часто встречаются кратные. И кроме того, для произвольной матрицы непросто вычислить сами коэффициенты характеристического многочлена.

Отметим некоторые *свойства собственных значений* для частных типов исходной матрицы.

1. Все собственные значения симметрической матрицы действительны.
2. Если собственные значения матрицы действительны и различны, то соответствующие им собственные векторы ортогональны и образуют базис рассматриваемого пространства. Следовательно, любой вектор в данном пространстве можно выразить через совокупность линейно независимых собственных векторов.
3. Если две матрицы *A* и *B подобны,* т. е. они связаны соотношением

$$B = P^{-1}AP, \tag{5.39}$$

то их собственные значения совпадают (здесь P — некоторая матрица). *Преобразование подобия* (5.39) можно использовать для упрощения исходной матрицы, а задачу о вычислении ее собственных значений свести к аналогичной задаче для более простой матрицы. Очевидно, самым лучшим упрощением матрицы (5.33) было бы приведение ее к треугольному виду

$$A' = \begin{pmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} \\ & a'_{22} & \dots & a'_{2n} \\ & & \dots & \dots \\ 0 & & & a'_{nn} \end{pmatrix}.$$

Тогда матрица (5.35) также имела бы треугольный вид. Как известно, определитель треугольной матрицы равен произведению ее диагональных элементов, поэтому характеристический многочлен (5.38) в этом случае имеет вид

$$\det C = (a'_{11} - \lambda)(a'_{22} - \lambda)\dots(a'_{nn} - \lambda).$$

Собственные значения матрицы, равные корням этого многочлена, можно получить сразу:

$$\lambda_1 = a'_{11},\ \lambda_2 = a'_{22},\ \dots,\ \lambda_n = a'_{nn}.$$

Таким образом, собственные значения треугольной матрицы равны ее диагональным элементам. То же самое, естественно, относится и к диагональной матрице, которая является частным случаем треугольной.

Некоторые типы матриц удается привести к треугольному виду с помощью преобразования подобия. В частности, симметрическую матрицу можно привести к диагональному виду. На практике часто используется приведение симметрической матрицы к трехдиагональному виду. Процедура вычисления собственных значений для полученной матрицы значительно упрощается по сравнению с задачей для исходной матрицы.

Существует ряд методов, основанных на использовании преобразования подобия, позволяющего привести исходную матрицу к более простой структуре. Мы рассмотрим ниже один из них — метод вращений.

3.2 Метод вращений.

Одним из эффективных методов, позволяющих привести исходную симметрическую матрицу n-го порядка к трехдиагональному виду, является *метод вращений.* Он основан на специально подбираемом вращении системы координат в n-мерном пространстве. Поскольку любое вращение можно заменить последовательностью элементарных (плоских) вращений, то решение задачи можно разбить на ряд шагов, на каждом из которых осуществляется плоское вращение. Таким образом, на каждом шаге выбираются две оси — *i*-я и j-я (i≠j), и поворот на угол φ производится в плоскости, проходящей через эти оси; остальные оси координат на данном шаге неподвижны. *Матрица вращения* мри этом имеет вид

$$P_{ij} = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & 0 & \\ & & \ddots & & & & \\ & & & p_{ii} & \cdots & p_{ij} & \\ & & & \vdots & \ddots & \vdots & \\ & & & p_{ji} & \cdots & p_{jj} & \\ & & & & & & \ddots \\ & 0 & & & & 1 & \\ & & & & & & 1 \end{pmatrix}. \tag{5.40}$$

$$p_{ii} = p_{jj} = p,\ \ p_{ij} = -p_{ji} = -q,\ \ p = \cos\varphi,\ \ q = \sin\varphi.$$

Здесь мы рассматриваем матрицы с вещественными элементами. В случае комплексных векторов для использования этого метода нужно изменить формулы (5.40).

Для осуществления преобразования подобия (5.39) необходимо найти обратную матрицу P_{ij}^{-1}. Можно показать, что она равна в рассматриваемом случае транспонированной матрице P_{ij}^{T}. Для получения обратной матрицы достаточно провести зеркальное отражение всех элементов исходной матрицы относительно ее диагонали. Другими словами, нужно поменять местами строки и столбцы исходной матрицы; элементы p_{ij} и p_{ji} при этом поменяются местами.

Угол поворота φ на каждом шаге выбирается таким, чтобы в преобразованной матрице обратился в нуль один элемент (в симметрической матрице — два). Процесс преобразования исходной матрицы путем элементарного вращения на любом k-м шаге можно представить в виде рекуррентных соотношений

$$A^{(k)} = P_{ij}^T A^{(k-1)} P_{ij},\ k = 1,2,... \tag{5.41}$$

Рассмотрим первый шаг преобразования. Сначала вычисляется произведение матриц $B = A^{(0)} P_{ij}$ (здесь $A^{(0)}$ - исходная матрица A*)*. Как видно из (5.40), в полученной матрице отличными от исходных являются элементы, стоящие в *i*-м и *j*-м столбцах; остальные элементы совпадают с элементами матрицы $A^{(0)}$, т.е.

$$\begin{aligned} &b_{li} = a_{li}^{(0)} p + a_{lj}^{(0)} q,\ b_{lj} = -a_{li}^{(0)} q + a_{lj}^{(0)} p, \\ &b_{lm=} a_{lm}^{(0)},\ m \neq i, j,\ l = 1,2,...,n. \end{aligned} \tag{5.42}$$

Затем находится преобразованная матрица $A^{(1)} = P_{ij}^T B$. Элементы полученной матрицы отличаются от элементов матрицы *B* только *i-й* и *j-и* строками. Они связаны соотношениями

$$\begin{aligned} &a_{im}^{(1)} = b_{im} p + b_{jm} q,\ a_{jm}^{(1)} = -b_{im} q + b_{jm} p, \\ &a_{lm}^{(1)} = b_{lm},\ l \neq i, j,\ m = 1,2,...,n. \end{aligned} \tag{5.43}$$

Таким образом, преобразованная матрица $A^{(1)}$отличается от $A^{(0)}$ элементами строк и столбцов с номерами *i* и *j*. Эти элементы пересчитываются по формулам (5.42), (5.43). В данных формулах пока не определенными остались параметры: *p* и *q;* при этом лишь один из них свободный, поскольку они подчиняются тождеству

$$p^2+q^2=1 \tag{5.44}$$

Недостающее одно уравнение для определения этих параметров получается из условия обращения в нуль некоторого элемента новой матрицы $A^{(1)}$.В зависимости от выбора этого элемента строятся различные алгоритмы метода вращений.

Одним из таких алгоритмов является последовательное обращение в нуль всех ненулевых элементов, лежащих вне трех диагоналей исходной симметрической матрицы. Это так называемый *прямой метод вращений.* В соответствии с этим методом обращение в нуль элементов матрицы производится последовательно, начиная с элементов первой строки (и первого столбца, так как матрица симметрическая).

Процесс вычислений поясним с использованием схематического изображения матрицы (Рисунок 5.6). Точками отмечены элементы матрицы. Наклонные линии указывают три диагонали матрицы, элементы на которых после окончания расчета отличны от нуля. Алгоритм решения задачи нужно построить таким образом, чтобы все элементы по одну сторону от этих трех диагоналей обратились в нуль; тогда симметрично расположенные элементы также станут нулевыми. Обращение элементов в нуль можно выполнять, например, в следующей последовательности: a_{13}, a_{14}, …, a_{1n}, a_{24}, a_{25}, …, a_{2n}, …, a_{n-2n}.

Рассмотрим сначала первый шаг данного метода, состоящий в обращении в нуль элемента a_{13} (и автоматически a_{31}*)*. Для осуществления элементарного вращения нужно выбрать две оси — i-ю и j-ю, так чтобы элемент a_{13} оказался в строке или столбце с номером i или *j*. Положим *i* = 2, *j*=3 и умножим матрицу $A^{(0)}$ справа на матрицу вращения P_{23} и слева на транспонированную матрицу P_{23}^T. Получим новые значения элементов матрицы, которые вычисляются по формулам (5.42), (5.43). Полагая в них l=1 и *m*=3, находим

$$a_{13}^{(1)} = b_{13} = -a_{12} q + a_{13} p = 0.$$

Учитывая тождество p2+q2=1 (5.44), получаем систему уравнений для определения параметров *p, q:*

$$\begin{aligned} &a_{13} p - a_{12} q = 0, \\ &p^2 + q^2 = 1. \end{aligned}$$

Решая эту систему, находим

$$p = \frac{a_{12}}{\sqrt{a_{12}^2 + a_{13}^2}},\ q = \frac{a_{12}}{\sqrt{a_{12}^2 + a_{13}^2}}.$$

Используя эти параметры p, q, можно по формулам (5.42), (5.43) вычислить значения элементов, стоящих в строках и столбцах с номерами i=2,3; j=2,3 (остальные элементы исходной матрицы не изменились).

Аналогично, выбирая для элементарного вращения i-ю и j-ю оси, можно добиться нулевого значения любого элемента $a_{i-1,j}^{(k)}$ на k-м шаге. В этом случае строится матрица вращения $P_{i,j,}$ параметры которой вычисляются по формулам, полученным из условия равенства нулю элемента $a_{i-1,j}^{(k)}$ и (5.44). Эти формулы имеют вид

$$p = \frac{a_{i-1,i}^{(k-1)}}{\sqrt{\left(a_{i-1,i}^{(k-1)}\right)^2 + \left(a_{i-1,j}^{(k-1)}\right)^2}},\ q = \frac{a_{i-1,j}^{(k-1)}}{\sqrt{\left(a_{i-1,i}^{(k-1)}\right)^2 + \left(a_{i-1,j}^{(k-1)}\right)^2}}.$$

Учитывая найденные значения параметров *p, q,* можно по формулам (5.42), (5.43) найти элементы преобразованной матрицы. Для иллюстрации вновь обратимся к Рисунок 5.6. Вертикальными линиями показаны столбцы с номерами *i* и *j,* соответствующими осям элементарного вращения, горизонтальными — строки с теми же номерами.

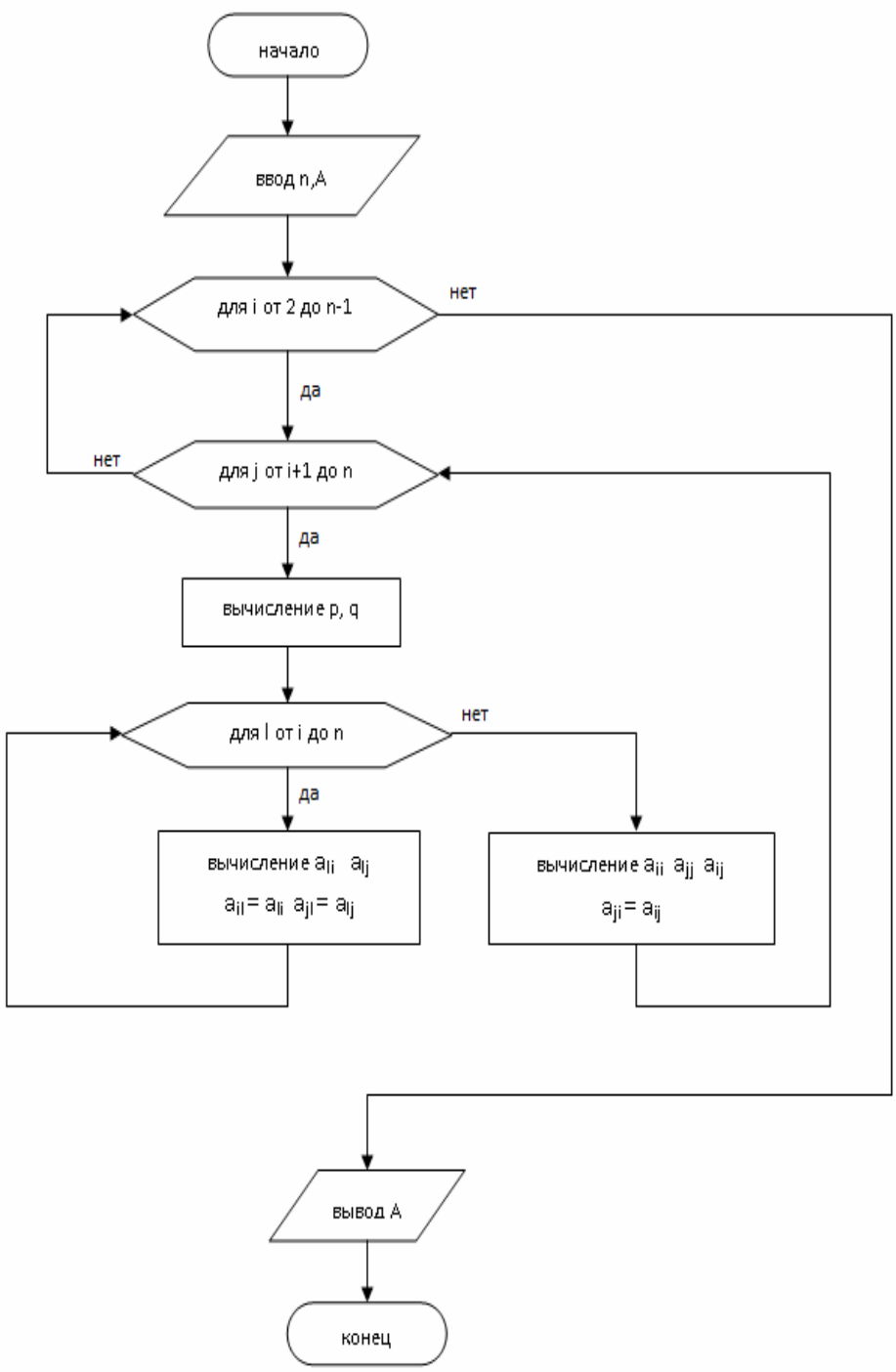


Рисунок 5.6 !!!!

На рассматриваемом шаге матрица преобразуется таким образом, чтобы отмеченные крестиками элементы обратились в нуль. Элементарное вращение (5.41) на каждом шаге требует пересчета всех элементов отмеченных столбцов и строк. Учитывая симметрию, можно вычислить лишь все элементы столбцов, а элементы получаются из условий симметрии. Исключение составляют лишь элементы, расположенные на пересечениях этих строк и столбцов. Они изменяются на каждом из двух этапов выполняемого шага. Таким образом, на каждом шаге преобразования симметрической матрицы для вычисления элементов столбцов используются формулы (5.42), а элементы, находящиеся на пересечениях изменяемых строк и столбцов, пересчитываются еще по формулам (5.43). При этом полученные ранее нулевые элементы не изменяются. Алгоритм приведения симметрической матрицы к трехдиагональному виду с помощью прямого метода вращений представлен на рисунке.

Собственные значения полученной трехдиагональной матрицы будут также собственными значениями исходной матрицы. Собственные векторы x_i исходной матрицы не равны непосредственно собственным векторам y_i трехдиагональной матрицы, а вычисляются с помощью соотношений

$$x_i = P_{23}P_{24}\ldots P_{n-1,n}y_i \tag{5.45}$$

3.3 Трехдиагональные матрицы.

Симметрическую матрицу можно привести с помощью преобразований подобия к трехдиагональному виду. Кроме того, трехдиагональные матрицы представляют самостоятельный интерес, поскольку они встречаются в вычислительной практике, и нередко требуется находить их собственные значения и собственные векторы. Рассмотрим трехдиагональную матрицу вида

$$A = \begin{pmatrix} b_1 & c_1 & & & & 0 \\ a_1 & b_2 & c_2 & & & \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & & & & a_n & b_n \end{pmatrix}. \tag{5.46}$$

Здесь элементы $b_1, b_2, \ldots, b_n$ расположены вдоль главной диагонали; $c_1, c_2, \ldots, c_{n-1}$ - над ней; $a_2, a_3, \ldots, a_n$ - под ней.

Для нахождения собственных значений нужно приравнять нулю определитель $D_n(\lambda)$ = dot(A - λ E), или

$$D_n(\lambda) = \begin{vmatrix} b_1-\lambda & c_1 & & & & 0 \\ a_2 & b_2-\lambda & c_2 & & & \\ & & & & & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & & & & a_n & b_n \end{vmatrix} = 0. \tag{5.47}$$

Произвольный определитель *n*-го порядка можно выразить через *n* миноров *(n*-1)-го порядка путем разложения его по элементам любой строки или любого столбца. Разложим определитель (5.47) по элементам последней строки, в которой всего два ненулевых элемента. Получим

$$D_n(\lambda) = (b_n-\lambda)D_{n-1}(\lambda) - a_nM_{n-1}(\lambda),$$

$$M_{n-1}(\lambda) = \begin{vmatrix} b_1-\lambda & c_1 & & & 0 \\ a_2 & b_2-\lambda & c_2 & & \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & & & a_{n-1} & c_{n-1} \end{vmatrix}. \tag{5.48}$$

Поскольку минор $M_{n-1}(\lambda)$ содержит в последнем столбце лишь один ненулевой элемент c_{n-1} то, разлагая его по элементам этого столбца, получаем

$$M_{n-1}(\lambda) = c_{n-1}D_{n-2}(\lambda).$$

Подставляя это выражение в формулу (5.48), получаем рекуррентные соотношения, выражающие минор высшего порядка через миноры двух низших порядков:

$$D_n(\lambda) = (b_n-\lambda)D_{n-1}(\lambda) - a_nc_{n-1}D_{n-2}(\lambda). \tag{5.49}$$

Положим $D_0(\lambda)$= 1. Минор первого порядка равен элементу a_{11} определителя, т. е. в данном случае $D_1(\lambda)$=$b_1 - \lambda$. Проверим, с учетом значений $D_0(\lambda)$, $D_1(\lambda)$ правильность формулы (5.49) при *n* = 2;

$$D_2(\lambda) = (b_2-\lambda)D_1(\lambda) - a_2c_1D_o(\lambda) = (b_2-\lambda)(b_1-\lambda) - a_2c_1. \tag{5.50}$$

Вычисляя минор второго порядка определителя (5.47), убеждаемся в справедливости выражения (5.50). Таким образом, используя рекуррентные соотношения (5.49), можно найти выражение для характеристического многочлена $D_n(\lambda)$ для любого $n \geq 2$. Вычисляя корни этого многочлена, получаем собственные значения $\lambda_1, \lambda_2, \ldots, \lambda_n$ трехдиагональной матрицы (5.46).

Будем считать, что собственные значения $\lambda_1, \lambda_2, \ldots, \lambda_n$ матрицы (5.46) вычислены. Найдем соответствующие им собственные векторы. Для любого собственного значения собственный вектор находится из системы уравнений (5.36).

$$(A-\lambda E)x = 0.$$

Перейдем от матричной формы записи этой системы к развернутой *(A*- матрица вида (5.46), x-{ $x_1, x_2, \ldots, x_n$}):

$$
\begin{aligned}
(b_1-\lambda)x_1 + c_1x_2 &= 0,\\
a_2x_1 + (b_2-\lambda)x_2 + c_2x_3 &= 0,\\
&\cdots\cdots\cdots\cdots\\
a_{n-1}x_{n-2} + (b_{n-1}-\lambda)x_{n-1} + c_{n-1}x_n &= 0,\\
a_nx_{n-1} + (b_n-\lambda)x_n &= 0.
\end{aligned}
\qquad (5.51)
$$

Матрица системы (5.51) вырожденная, поскольку ее определитель (5.47) равен нулю. Можно показать, что если $c_i \neq 0$ (i = 1,2,..., n-1), то последнее уравнение системы (5.51) является следствием остальных уравнений. Действительно, если отбросить первый столбец и последнюю строку в матрице A, то вместо (5.47) получится определитель вида

$$
\begin{vmatrix}
c_1 & & & 0\\
b_2-\lambda & c_2 & & \\
 & & & \\
0 & a_{n-1} & b_{n-1}-\lambda & c_{n-1}
\end{vmatrix} = c_1c_2\ldots c_{n-1} \neq 0. \qquad (5.52)
$$

Следовательно, все строки с первой по (n-1)-ю линейно независимы, последнее уравнение системы (5.51) — следствие остальных, и одно неизвестное этой системы является свободным. Отбрасывая последнее уравнение системы (5.51), записываем ее в виде

$$
\begin{aligned}
c_1x_2 &= -(b_1-\lambda)x_1,\\
(b_2-\lambda)x_2 + c_2x_3 &= -a_2x_1,\\
&\cdots\cdots\cdots\cdots\\
a_{n-1}x_{n-2} + (b_{n-1}-\lambda)x_{n-1} + c_{n-1}x_n &= 0.
\end{aligned}
\qquad (5.53)
$$

Считая компоненту x_1 свободным неизвестным и полагая ее равной любому ненулевому значению (иначе при $x_1 = 0$ остальные компоненты также будут нулевыми), можно из системы (5.53) найти последовательно все остальные компоненты: из первого уравнения легко вычислить x_2, из второго x_3 и т, д., из последнего x_n. Поскольку определитель (5.52) этой системы отличен от нуля, то она имеет единственное решение. Описанным способом могут быть найдены собственные векторы, соответствующие всем собственным значениям $\lambda_1, \lambda_2, \ldots, \lambda_n$ трехдиагональной матрицы (5.46).

Если трехдиагодальная матрица получена в результате последовательности преобразований подобия исходной симметричной матрицы, то, как уже отмечалось, все собственные значения трехдиагональной матрицы являются одновременно собственными значениями исходной матрицы, а собственные векторы пересчитываются по формулам (5.45). При этом вычислять произведения матриц $P_{n-1,n}, \ldots, P_{24}, P_{23}$ на собственные векторы трехдиагональной матрицы нецелесообразно, поскольку при умножении матрицы P_{ij}, на вектор x изменяются только две его компоненты: x_i и x_j. Поэтому в качестве этих компонент берем значения $px_i - qx_j$ и $qx_i + px_j$, что сокращает объем вычислений по сравнению с умножением матрицы P_{ij}, на вектор x.

Глава 6
ПРИБЛИЖЕНИЕ ФУНКЦИЙ

§1.Точечная аппроксимация. Равномерное приближение.

1.1 Точечная аппроксимация.

Одним из основных типов точечной аппроксимации является интерполирование. Оно состоит в следующем: для данной функции

$$y=f(x) \tag{6.1}$$

строим многочлен (6.1), принимающий в заданных точках x те же значения y_i, что и функция f(x), т.е i=0,1,....,n.

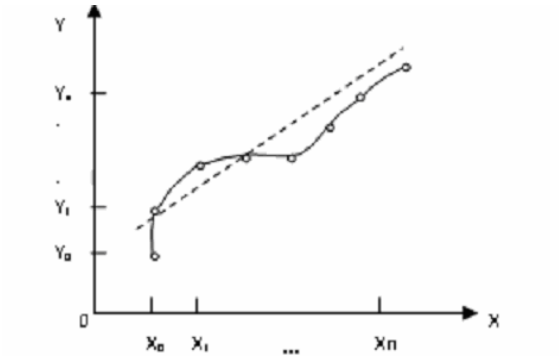


Рисунок 6.1

При этом предполагается, что среди значений x_i нет одинаковых, т.е. $x_i \neq x_k$ при $i \neq k$.

Точки x_i называются узлами интерполяции, а многочлен $\varphi(x)$ - интерполяционным многочленом.

Таким образом, близость интерполяционного многочлена к заданной функции состоит в том, что их значения совпадают на заданной системе точек (рисунок 6.1, сплошная линия).

Максимальная степень интерполяционного многочлена m=n, в этом случае говорят о глобальной интерполяции, поскольку один многочлен

$$\varphi(x) = a_0 + a_1 x + ... + a_n x^n \tag{6.2}$$

используется для интерполяции функции f(x) на всем рассматриваемом интервале изменения аргумента x. Коэффициенты a_i многочлена находятся из системы (6.2). Можно показать, что при $x_i \neq x_k$ $(i \neq k)$ эта система имеет единственное решение.

Интерполяционные многочлены могут строиться отдельно для разных частей рассматриваемого интервала изменения x. В этом случае имеем кусочную (или локальную) интерполяцию.

Как правило интерполяционные многочлены используется для аппроксимации функции в промежуточных точках между крайними узлами интерполяции, т.е. при $x_0 < x < x_n$. Однако они используются и для приближенного вычисления функции вне рассматриваемого отрезка ($x < x_0$, $x > x_n$). Это приближение называют экстраполяцией.

При интерполировании основным условием является прохождение графика интерполяционного многочлена через данные значения функции в узлах интерполяции. Однако в ряде случаев выполнение этого условия затруднительно или даже нецелесообразно.

Например, при большом числе узлов интерполяции получается высокая степень многочлена (6.2) в случае глобальной интерполяции, т.е. когда нужно иметь один интерполяционный многочлен для всего интервала изменения аргумента. Кроме того, табличные данные могли быть получены путем измерений и содержать ошибки. Построение аппроксимирующего многочлена с условием обязательного прохождения его графика через эти экспериментальные точки означало бы тщательное повторение допущенных при измерениях ошибок. Выход из этого положения может быть найден выбором такого многочлена, график которого проходит близко от данных точек (рисунок 6.1, штриховая линия). Понятие «близко» уточняется при рассмотрении разных видов приближения.

Одним из таких видов является среднеквадратичное приближение функции с помощью многочлена (6.1). При этом $m \leq n$; случай m=n соответствует интерполяции. На практике стараются подобрать аппроксимирующий многочлен как можно меньшей степени (как правило, m=1, 2. 3).

Мерой отклонения многочлена $\varphi(x)$ от заданной функции f(x) на множестве точек (x_i,y_i) (i=0,1, 2, …, n) при среднеквадратичном приближении является величина S, равная сумме квадратов разностей между значениями

многочлена и функции в данных точках:

$$S = \sum_{i=0}^{n} [\varphi(x_i) - y_i]^2 \qquad (6.3)$$

Для построения аппроксимирующего многочлена нужно подобрать коэффициенты a_0, a_1, …, a_m так, чтобы величина S была наименьшей. В этом состоит метод наименьших квадратов.

1.2 Равномерное приближение.

Во многих случаях, особенно при обработке экспериментальных данных, среднеквадратичное приближение вполне приемлемо, поскольку оно сглаживает некоторые неточности функции f(x) и дает достаточно правильное представление о ней. Иногда, однако, при построении приближения ставится более жесткое условие: требуется, чтобы во всех точках некоторого отрезка [a. b] отклонение многочлена $\varphi(x)$ от функции f(x) было по абсолютной величине меньшим заданной величины $\varepsilon > 0$:

$$|f(x) - \varphi(x)| < \varepsilon, a \le x \le b.$$

В этом случае говорят, что многочлен $\varphi(x)$ равномерно аппроксимирует функцию f(x) с точностью ε на отрезке [a, b].

Введем понятие абсолютного отклонения Δ многочлена $\varphi(x)$ от функции f(x) на отрезке [a, b]. Оно равно максимальному значению абсолютной величины разности между ними на данном отрезке:

$$\Delta = \max_{a \le x \le b} |f(x) - \varphi(x)|.$$

По аналогии можно ввести понятие среднеквадратичного отклонения $\overline{\Delta} = \sqrt{S/n}$ при среднеквадратичном приближении функций.

Возможность построения многочлена, равномерно приближающего данную функцию, следует из теоремы Вейерштрасса об аппроксимации:

Теорема. Если функция f(x) непрерывна на отрезке [a, b], то для любого $\varepsilon > 0$ существует многочлен $\varphi(x)$ степени $m = m(\varepsilon)$, абсолютное отклонение которого от функции f(x) на отрезке [a, b] меньше ε.

§2. Многочлены Чебышева. Вычисление многочленов. Рациональные приближения.

2.1 Многочлены Чебышева.

Одним из способов совершенствования алгоритма вычислений, позволяющих более равномерно распределить погрешность по всему интервалу, является использование многочленов Чебышева.

Многочлен Чебышева

$$T_n(x) = \frac{1}{2}[(x + \sqrt{x^2 - 1})^n + (x - \sqrt{x^2 - 1})^n], \quad -1 \le x \le 1 \qquad (6.4)$$

n=0,1,…

Легко показать, что (6.4) действительно является многочленом: при возведении в степень и последующих преобразованиях члены, содержащие корни, уничтожаются. Приведем многочлены Чебышева, полученные по формуле (6.3) при n=0,1,2,3

$$T_0(x) = 1,\ T_1(x) = x,\ T_2(x) = 2x^2 - 1,\ T_3(x) = 4x^3 - 3x.$$

Для вычисления многочлена Чебышева можно воспользоваться рекуррентным соотношением

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \qquad (6.5)$$

n=1,2,…

В ряде случаев важно знать коэффициент a_n при старшем члене многочлена Чебышева степени n

$$T_n(x) = a_0 + a_1 x + \ldots + a_n x^n.$$

Разделив этот многочлен на x^n, найдем

$$a_n = \frac{T_n(x)}{x^n} - \frac{a_0}{x^n} - \ldots - \frac{a_{n-1}}{x}.$$

Перейдем к пределу при $x \to \infty$ и воспользуемся формулой (6.4), получим

$$a_n = \lim_{x \to \infty} \frac{T_n(x)}{x^n} = \frac{1}{2} \lim_{x \to \infty} [(1 + \sqrt{1 - \frac{1}{x^2}})^n + ?(1 - \sqrt{1 - \frac{1}{x^2}})^n] = 2^{n-1}.$$

Многочлены Чебышева можно так же представить в тригонометрической форме:

$$T_n(x) = \cos(n \arccos x) \qquad (6.6)$$

n =0,1,…

С помощью этих выражений могут быть получены формулы (6.5).

Нули (корни) многочленов Чебышева на отрезке [-1,1] определяются формулой

$$x_k = \cos\frac{2k-1}{2n}\pi \text{ , k=1,2,…,n.}$$

Они расположены неравномерно на отрезке и сгущаются к его концам.

Вычисляя экстремумы многочлена Чебышева по обычным правилам (с помощью производных), можно найти его максимумы и минимумы:

$$x_k = \cos(k\pi / n) \text{ , k=1,2,…,n-1.}$$

В этих точках многочлен принимает поочередно значения $T_n(x_k)$=-(+)1, т.е. все максимумы равны 1, а минимумы -1. На границах отрезка значения многочленов Чебышева равны –(+)1.

Многочлены Чебышева широко используются при аппроксимации функций. Рассмотрим их применение для улучшения приближения функций с помощью степенных рядов, а именно для более равномерного распределения погрешностей аппроксимации по заданному отрезку [-π/2, π/2].

Отрезок [-π/2, π/2] является не совсем удобным при использовании многочленов Чебышева, поскольку они обычно рассматриваются на стандартном отрезке [-1,1]. Первый отрезок легко привести ко второму заменой переменной x на π/2. В этом случае ряд для аппроксимации синуса на отрезке [-1,1]примет вид:

$$\sin\frac{\pi x}{2} = \frac{\pi x}{2} - \frac{1}{3!}(\frac{\pi x}{2})^2 + \frac{1}{5!}(\frac{\pi x}{2})^5 - ... \qquad (6.7)$$

При использовании этого ряда погрешность вычисления функции в окрестности концов отрезка $x=\pm 1$ существенно возрастает и становится значительно больше, чем в окрестности точки x=0. Если вместо (6.6) использовать ряд

$$\sin(\frac{\pi x}{2}) = c_0 + c_1 T_1(x) + c_2 T_2(x) + ...,$$

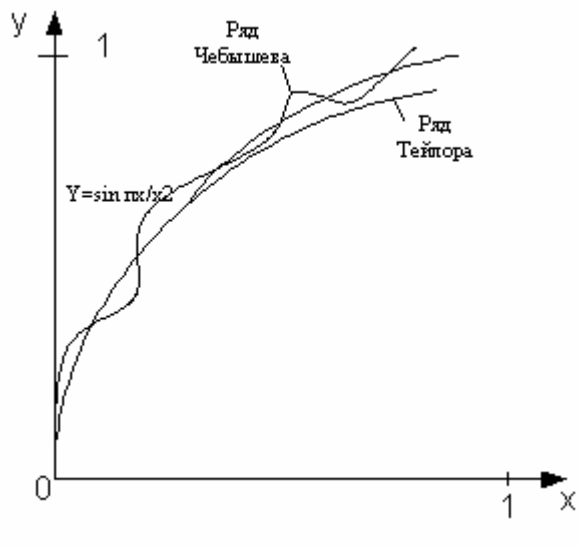


Рисунок 6.2

члены которого являются многочлены Чебышева, то погрешность будет распределена по всему отрезку (рисунок 6.2). В частности при использовании многочленов Чебышева до девятой степени включительно погрешность находится в интервале $(-5\div 5)*10^{-9}$. Причем погрешность ряда Тейлора для этой задачи на концах отрезка составляет $4*10^{-6}$.

Нахождение коэффициентов ряда Чебышева довольно сложно и здесь рассматриваться не будет. На практике часто используют многочлены Чебышева для повышения точности аппроксимации функций с помощью ряда Тейлора.

Пусть частичная сумма ряда Тейлора, представленная в виде многочлена, используется для приближения функции f(x) на стандартном отрезке [—1, 1], т. е.

$$f(x) \approx a_0 + a_1 x + ... + a_n x^n \qquad (6.8)$$

Если рассматриваемый отрезок [a, b] отличается от стандартного, то его всегда можно привести к стандартному заменой переменной

$$t = \frac{a+b}{2} + \frac{b-a}{2}x_2 \qquad -1 \le x \le 1.$$

Многочлен Чебышева $T_n(x)$ можно записать в виде

$$T_n(x) = b_0 + b_1 x + b_2 x^2 + ... + 2^{n-1}x^n.$$

Отсюда получаем

$$x^n = -2^{1-n}(b_0 + b_1 x + ... + b_{n-1}x^{n-1}) + 2^{1-n}T_n(x) \qquad (6.9)$$

Если отбросить последний член, то допущенную при этом погрешность Δ легко оценить: $|\Delta| \le 2^{1-n}$, поскольку $|T_n(x)| \le 1$. Таким образом, из (6.8) получаем, что x^n есть линейная комбинация более низких степеней x. Подставляя эту линейную комбинацию в (6.7), приходим к многочлену степени n- 1 вместо многочлена степени n. Этот процесс может быть продолжен до тех пор, пока погрешность не превышает допустимого значения.

Используем эту процедуру для повышения точности аппроксимации функции с помощью ряда (6.5). Будем учитывать члены ряда до 11-й степени включительно. Вычисляя коэффициенты при степенях x, получаем

$$\sin(\frac{\pi x}{2}) \approx 1.5707963x - 0.64596410x^3 + 0.79692626x^5 -$$
$$0.0046817541x^7 + 0.00016044118x^9 - 0.0000035988432x^{11} \qquad (6.10)$$

Многочлен Чебышева 11-й степени имеет вид

$$T_{11} = 1024x^{11} - 2816x^9 + 2816x^7 - 1232x^5 + 220x^3 - 11x.$$

Выразим отсюда x^{11} через более низкие степени:

$$x^{11} = 2\text{-}10\ (11x - 220x^3 + 1232x^5 - 2816x^7 + 2816x^9 + T_{11}).$$

Подставляя в (6.10) вместо x^{11} правую часть этого равенства и вычисляя новые значения коэффициентов, получаем:

$$\sin(\frac{\pi x}{2}) \approx 1.5707962x - 0.64596332x^3 + 0.079688296x^5 -$$
$$0.0046718573x^7 + 0.00015054436x^9 - 0.00000000351T_{11} \qquad (6.11)$$

Отбрасывая последний плен этого разложения, мы допускаем погрешность $|\Delta| \le 3.51 \cdot 10^{-9}$. Из-за приближенного вычисления коэффициентов при степенях x реальная погрешность больше. Здесь она оценивается величиной $|\Delta| \le 8 \cdot 10^{-9}$.Эта погрешность немного больше, чем для многочлена Чебышева ($5*10^{-9}$), и значительно меньше, чем для ряда Тейлора ($4*10^{-6}$).

Процесс модификации приближения можно продолжить. Если допустимое значение погрешности больше, чем при использовании выражения (6.12) (без последнего члена с T_{11}), то x^9 можно заменить многочленом седьмой степени, а член с T_9 отбросить; так продолжать до тех пор, пока погрешность остается меньше допустимой.

В заключение приведем некоторые формулы, необходимые при использовании многочленов Чебышева.

1. Многочлены Чебышева:

$$T_n(\mathrm{x}) = \frac{1}{2}\left[(x + \sqrt{x^2 - 1})^n + (x - \sqrt{x^2 - 1})^n\right] = \cos(n \arccos x),$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),\ \mathrm{n}=1,2,\ldots,$$

$$T_0(x) = 1,$$

$$T_1(x) = x,$$

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

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

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

$$T_5(x) = 16x^5 - 20x^3 + 5x,$$

$$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1,$$

$$T_7(x) = 64x^7 - 112x^5 + 56x^3 - 7x,$$

$$T_8(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1,$$

$$T_{10}(x) = 512x^{10} - 1280x^8 + 1120x^5 - 400x^4 + 50x^2 - 1,$$

$$T_{11}(x) = 1024x^{11} - 2816x^9 + 2816x^6 - 1232x^5 + 220x^3 - 11x,$$

2. Представление степеней x через многочлены $T_n(x)$:

$$x^0 = 1 = T_{01},$$

$$x = T_{11},$$

$$x^2 = \frac{1}{2}(T_0 + T_2),$$

$$x^3 = \frac{1}{4}(3T_1 + T_3),$$

$$x^4 = \frac{1}{8}(3T_0 + 4T_2 + T_4)$$

$$x^5 = \frac{1}{16}(10T_1 + 5T_3 + T_5)$$

$$x^6 = \frac{1}{32}(10T_1 + 5T_3 + T_5)$$

$$x^7 = \frac{1}{64}(35T_1 + 21T_3 + 7T_5 + T_7)$$

$$x^8 = \frac{1}{128}(35T_0 + 56T_2 + 28T_4 + 8T_6 + T_8)$$

$$x^9 = \frac{1}{256}(126T_1 + 84T_3 + 36T_5 + 9T_7 + T_9)$$

$$x^{10} = \frac{1}{512}(126T_0 + 210T_2 + 120T_4 + 45T_6 + 10T_8 + T_{10})$$

$$x^{11} = \frac{1}{1024}(462T_1 + 330T_3 + 165T_5 + 55T_7 + 11T_9 + T_{11})$$

3. Выражение x^n через более низкие степени

$x=T_{11}$

$$x^2 = \frac{1}{2}(1 + T_2),$$

$$x^3 = \frac{1}{4}(3x + T_3),$$

$$x^4 = \frac{1}{8}(8x^2 - 1 + T_4),$$

$$x^5 = \frac{1}{16}(20x^3 - 5x + T_5),$$

$$x^6 = \frac{1}{32}(48x^4 - 18x^2 + 1 + T_6),$$

$$x^7 = \frac{1}{64}(112x^5 - 56x^3 + 7x + T_7),$$

$$x^8 = \frac{1}{128}(256x^6 - 160x^4 + 32x^2 - 1 + T_8),$$

$$x^9 = \frac{1}{256}(576x^7 - 432x^5 + 120x^3 - 9x + T_9),$$

$$x^{10} = \frac{1}{512}(1280x^8 - 1120x^6 + 400x^4 - 50x^2 - 1 + T_{10}),$$

$$x^{11} = \frac{1}{1024}(2816x^9 - 2816x^7 + 1232x^5 - 220x^3 - 11x + T_{11})$$

2.2 Вычисление многочленов.

При аппроксимации функций, а также в некоторых других задачах приходится вычислять значения многочленов вида

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \qquad (6.12)$$

Если проводить вычислении «в лоб», т. е. находить значения каждого члена и суммировать их, то при больших n потребуется выполнить большое число операций($n^2 + n/2$ умножений и n сложений). Кроме того, это может привести к потере точности за счет погрешностей округления. В некоторых частных случаях, как это сделано при вычислении синуса, удается выразить каждый последующий член через предыдущий и таким образом значительно сократить объем вычислений.
Анализ многочлена (6.13) в общем случае приводит к тому, что для исключения возведения x в степень в каждом члене многочлен целесообразно переписать в виде

$$P(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + xa_n) + \ldots)). \quad (6.13)$$

Прием, с помощью которого многочлен представляется в таком виде, называется схемой Горнера. Этот метод требует n умножений на n сложении. Использование схемы Горнера для вычисления значений многочленов не только экономит машинное время, но и повышает точность вычислений за счет уменьшения погрешностей округления.

2.3 Рациональные приближения.

Рассмотрим другой вид аппроксимации функций — с помощью дробно- рационального выражения. Функцию представим в виде отношения двух многочленов некоторой степени. Пусть, например, это будут многочлены третьей степени, т. е. представим функцию f(x) d виде дробно- рационального выражения:

$$f(x) = \frac{b_0 + b_1x + b_2x^2 + b_3x^3}{1 + c_1x + c_2x^2 + c_3x^3} \quad (6.14)$$

Значение свободного члена в знаменателе $c_0 = 1$ не нарушает общности этого выражения, поскольку при $c_0 \neq 1$ числитель и знаменатель можно разделить на c_0. Перепишем выражение (6.13) в виде

$$b_0 + b_1x + b_2x^2 + b_3x^3 = (1 + c_1x + c_2x^2 + c_3x^3) f(x).$$

Используя разложение функции f(x) в ряд Тейлора:

$$f(x) = a_0 + a_1x + a_2x^2 + \ldots \quad (6.15)$$

и учитывая члены до шестой степени включительно, получаем

$$b_0 + b_1x + b_2x^2 + b_3x^3 = (1 + c_1x + c_2x^2 + c_3x^3) \times (a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6)$$

Преобразуем правую часть этого равенства, записав ее разложение по степеням x:

$$b_0 + b_1x + b_2x^2 + b_3x^3 = a_0 + x(a_1 + a_0c_1) + x^2(a_2 + a_1c_1 + a_0c_2) + x^3(a_3 + a_2c_1 + a_1c_2 + a_0c_3) +$$
$$+ x^4(a_4 + a_3c_1 + a_2c_2 + a_1c_3) + x^5(a_5 + a_4c_1 + a_3c_2 + a_2c_3) + x^6(a_6 + a_5c_1 + a_4c_2 + a_3c_3).$$

Приравнивая коэффициенты при одинаковых степенях x левой и правой частях, получаем следующую систему уравнений:

$$b_0 = a_0,$$
$$b_1 = a_1 + a_0c_1,$$
$$b_2 = a_2 + a_1c_1 + a_0c_2,$$
$$b_3 = a_3 + a_2c_1 + a_1c_2 + a_0c_3,$$
$$0 = a_4 + a_3c_1 + a_2c_2 + a_1c_3,$$
$$0 = a_5 + a_4c_1 + a_3c_2 + a_2c_3,$$
$$0 = a_6 + a_5c_1 + a_4c_2 + a_3c_3.$$

Решив эту систему, найдем коэффициенты b_0, b_1, b_2, b_3, c_1, c_2, c_3, необходимо для аппроксимации (6.13).
Пример.

Рассмотрим рациональное приближение для функции $f(x) = \sin(\frac{\pi x}{2})$. Воспользуемся представлением (6.13), которое в данном случае упрощается, поскольку функция sin x нечетная. В частности в частности, в числителе можем оставить только члены с нечетными степенями л, а в знаменателе — с четными; коэффициенты при других степенях x равны нулю: $b_1 = b_2 = c_1 == c_3 = 0$,

Коэффициенты b_1, b_2, c_2, найдем из системы уравнении (6.13), причем значения коэффициентов a_0, a_1, …, a_6 разложения функции в ряд Тейлора (6.12) $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ (6.12)можем взять из выражения (6.4), т. е.

$a_0 = 0$, $a_1 = \frac{\pi}{2}$, $a_2 = 0$, $a_3 = -\frac{\pi^3}{8 \cdot 3}!$, $a_4 = 0$, $a_5 = \frac{\pi^5}{32 \cdot 5!}$, $a_6 = 0$.

Система уравнений в данном случае примет вид

$b_1 = \frac{\pi}{2}$, 8 $0 = \frac{\pi^5}{32 \cdot 5!} - \frac{\pi^3}{8 \cdot 3!} c_2$.

Отсюда находим $b_1 = \frac{\pi}{2}$, $b_3 = -\frac{7\pi^3}{480}$, $c_2 = \frac{\pi^2}{80}$.

Таким образом, дробно- рациональное приближение (6.11) для функции $f(x) = \sin(\frac{\pi x}{2})$ примет вид

$$\sin(\frac{\pi x}{2}) = \frac{(\pi/2)x - (7\pi^3/480)x^3}{1 + (\pi^2/80)x^2} \tag{6.16}$$

Это приближение по точности равносильно аппроксимации (6.4) с учетом членов до пятого порядка включительно.

На практике с целью экономии числа операций выражение (6.11) представляется в виде цепной дроби. Представим в таком виде дробно-рациональное выражение (6.14). Сначала перепишем это выражение, вынося за скобки коэффициенты при x^3 и x^2. Получим

$$\sin(\frac{\pi x}{2}) = -\frac{7\pi}{6} \frac{x^3 - (60/7)(2/\pi)^2 x}{x^2 + 20(2/\pi)^2}.$$

Разделим числитель на знаменатель по правилу деления многочленов и введем обозначения для коэффициентов.

Получим

$$\sin(\frac{\pi x}{2}) = k_1 (x + \frac{k_2 x}{x^2 + k_3}),$$

$$k_1 = -\frac{7\pi}{6}, \quad k_2 = -\frac{200}{7}(\frac{2}{\pi})^2, \; k_3 = 20(\frac{2}{\pi})^2.$$

Полученное выражение можно записать в виде

$$\sin(\frac{\pi x}{2}) = k_1 \left(x + \frac{k_2}{x + \frac{k_3}{x}} \right) \tag{6.17}$$

Для вычисления значения функции по этой формуле требуется намного меньше операций (два деления, два сложения, одно умножение), чем для вычисления с помощью выражения (6.15) или усеченного ряда Тейлора (6.4) (далее с использованием правила Горнера).

Приведем формулы для приближения некоторых элементарных функций с помощью цепных дробей, указывая интервалы изменения аргумента и погрешности Δ:

$$e^x = 1 + \cfrac{x}{-\frac{x}{2} + \cfrac{k_0 + k_1 x^2}{1 + k_2 x^2}} \tag{6.18}$$

$$k_0 = 1.0000000020967, \; -\frac{1}{2}\ln 2 \le x \le \frac{1}{2}\ln 2, \; |\Delta| \le 10^{-10};$$

$$\ln(1+x) = k_0 + \cfrac{x}{k_1 + \cfrac{x}{k_2 + \cfrac{x}{k_3 + \cfrac{x}{k_4 + \cfrac{x}{k_5}}}}} \tag{6.19}$$

$$k_0 = 0.0000000894\,,\ k_1 = 1.0000091365,$$
$$k_2 = 2.0005859000,\ k_3 = 3.0311932666,$$
$$k_4 = 1.0787748225,\ k_5 = 8.8952784060,$$
$$0 \le x \le 1,\ |\Delta| \le 10^{-7};$$

$$tg\frac{\pi x}{4} = x\left(k_0 + \cfrac{x^2}{k_1 + \cfrac{x^2}{k_2 + \cfrac{x^2}{k_3}}}\right) \tag{6.20}$$

$$k_0 = 0.7853980289\,,\ k_1 = 6.1922344479,$$
$$k_2 = -0.6545887679,\ k_3 = 491.0013934779\,,$$
$$-1 \le x \le 1,\ |\Delta| \le 2\cdot 10^{-7}.$$

$$arctgx = x\left(k_0 + \cfrac{x^2}{k_1 + \cfrac{x^2}{k_2 + \cfrac{x^2}{k_3 + \cfrac{x^2}{k_4}}}}\right) \tag{6.21}$$

$$k_0 = 0.99999752\,,\ k_1 = -3.00064286,$$
$$k_2 = -0.55703890,\ k_3 = -17.03715998\,,$$
$$k_4 = -0.20556880,$$
$$-1 \le x \le 1,\ |\Delta| \le 2\cdot 10^{-7}.$$

§3. Интерполирование. Линейная и квадратичная интерполяция. Многочлен Лагранжа. Многочлен Ньютона. Кубические сплайны. Точность интерполяции.

3.1 Линейная квадратичная интерполяция.

Простейшим и часто используемым видом локальной интерполяции является линейная интерполяция. Она состоит в том, что заданные точки (x_i, y_i) (i=0,1,2,3,…,n)-соединяются прямолинейными отрезками, и функция f(x) приближается ломаной с вершинами в данных точках.

Уравнения каждого отрезка ломано разные. Поскольку имеется n интервалов (x_{i-1}, x_i), то для каждого из них в качестве уравнения интерполяционного многочлена не используется уравнение примой, проходящей через две точки. В частности, для i-го интервала можно написать уравнение прямой, проходящей через точки (x_{i-1}, y_{i-1}) и (x_i, y_i), в виде

$$\frac{y-y_{i-1}}{y_i-y_{i-1}}=\frac{x-x_{i-1}}{x_i-x_{i-1}}.$$

Отсюда

$$y=a_ix+b_i,\ x_{i-1}\le x\le x_i \qquad (6.22)$$

, $$a_i=\frac{y_i-y_{i-1}}{x_i-x_{i-1}},\ b_i=y_{i-1}-a_ix_{i-1}.$$

Следовательно, при использовании линейной интерполяции сначала нужно определить интервал, в который попадает значение аргумента x, а затем подставить его в формулу (6.21) и найти приближенное значении е функции в этой точке.

Рассмотрим теперь случай квадратичной интерполяции. В качестве интерполяционной функции на отрезке $[x_{i-1}, x_{i+1}]$ принимается квадратный трехчлен. Такую интерполяцию называют также параболической.

Уравнение квадратного трехчлена

$$y=a_ix^2+b_ix+c_i,\qquad x_{i-1}\le x\le x_{i+1} \qquad (6.23)$$

Содержит три неизвестных коэффициента a_i, b_i, c_i, для определения которых необходимо три уравнения. Ими служат условия прохождения параболы (6.22_через три точки (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}). Эти условия можно записать в виде

$$\begin{aligned}a_ix_{i-1}^2+b_i\,x_{i-1}+c_i&=y_{i-1},\\ a_ix_i^2+b_ix_i+c_i&=y_i\\ a_ix_{i+1}^2+b_i\,x_{i+1}+c_i&=y_{i+1}\end{aligned} \qquad (6.24)$$

Алгоритм вычисления приближенного значения функции с помощью квадратичной интерполяций можно представить в виде блок-схемы, как и для случая линейной интерполяции. Вместо формулы (6.21) нужно использовать (6.22) с учетом решения системы линейных уравнений (6.23). Интерполяция для любой точки $x\in[x_0,x_n]$ проводится по трем ближайшим к ней узлам.

ПРИМЕР. Найти приближенное значение функции y = f(x) при x = 0.32, если известна следующая таблица ее значений:

x	0.15	0.30	0.40	0.55
y	2.17	3.63	5.07	7.78

Воспользуемся сначала формулой линейной интерполяции (6.22) $y=a_ix+b_i,\ x_{i-1}\le x\le x_i$ (6.22). Значение X = 0.32 находится между узлами $x_i = 0.30$ и $x_i = 0.40$. В этом случае

$$a_i=\frac{y_i-y_{i-1}}{x_i-x_{i-1}}=\frac{5.07-3.63}{0.40-0.30}=14.4,$$

$$b_i=y_{i-1}-a_ix_{i-1}=3.63-14.4\cdot0.30=-0.69,$$

$$y\approx14.4x-0.69=14.4\cdot0.32-0.69=3.92.$$

Найдем теперь приближенное значение функции с помощью формулы квадратичной интерполяции (6.22). Составим систему уравнений (6.23) с учетом ближайших к точке x= 0.32 узлов: $x_{i-1}=0.15$, $x_i=0.30$, $x_{i+1}=0.40$. Соответственно $y_{i-1} = 2.17$, $y_i = 3.63$, $y_{i+1}=5.07$. Система (6.24) запишется в виде

$$0.15^2a_i+0.15b_i+c_i=2.17,$$

$$0.30^2a_i+0.30b_i+c_i=3.63,$$

$$0.40^2a_i+0.40b_i+c_i=5.07.$$

Решая эту систему, находим $a_i=18.67$, $b_i=1.33$, $c_i=1.55$. Искомое значение функции

$$y\approx18.67\cdot0.32^2+1.33\cdot0.32+1.55=3.89$$

3.2Многочлен Лагранжа.

Перейдем к случаю глобальной интерполяции, т. е. построению интерполяционного многочлена, единого для всего отрезка $[x_0, x_n]$. При этом, естественно, график интерполяционного многочлена должен проходить через все заданные точки.

Запишем искомый многочлен в виде

$$\varphi(x)=a_0+a_1x+\ldots+a_nx^n \qquad (6.25)$$

Из условий равенства значений этого многочлена в узлах x_i соответствующим заданным табличным значением y_i получим следующую систему уравнений для нахождения коэффициентов a0, a1, …,an:

$$\begin{aligned} a_0 + a_1 x_0 + ... + a_n x_0^n &= y_0 \\ a_0 + a_1 x_1 + ... + a_n x_1^n &= y_1, \\ &\ldots\ldots\ldots\ldots \\ a_0 + a_1 x_n + ... + a_n x_n^n &= y_n. \end{aligned} \tag{6.26}$$

Можно показать, что эта система имеет единственное решение, если среди узлов интерполяции нет совпадающих, т.е. если $x_i \neq x_j$ при $i \neq j$. Решив эту систему найдем коэффициенты интерполяционного многочлена (6.25). Заметим вместе с тем, что такой путь построения интерполяционного многочлена требует значительного объема вычислений, особенно при большом числе узлов. Существуют более простые алгоритмы построения интерполяционных многочленов.

Будем искать многочлен в виде линейной комбинации многочленов степени n:

$$L(x) = y_0 l_0(x) + y_1 l_1(x) + ... + y_n l_n(x) \tag{6.27}$$

При этом потребуем, чтобы каждый многочлен $l_i(x)$ обращался в 0 на всех узлах интерполяции, за исключением одного (i-го), где он должен равняться единице. Легко проверить, что этим условиям отвечает многочлен вида

$$l_0(x) = \frac{(x-x_1)(x-x_2)...(x-x_n)}{(x_0-x_1)(x_0-x_2)...(x_0-x_n)} \tag{6.28}$$

Действительно, $l_0(x_0)=1$ при $x=x_0$. При $x=x_1, x_2, \ldots, x_n$ числитель выражения (6.28) обращается в нуль. По аналогии с (6.28) получим

$$\begin{aligned} l_1(x) &= \frac{(x-x_0)(x-x_2)...(x-x_n)}{(x_1-x_0)(x_1-x_2)...(x_1-x_n)}, \\ l_2(x) &= \frac{(x-x_0)(x-x_1)(x-x_3)...(x-x_n)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)...(x_2-x_n)} \\ l_i(x) &= \frac{(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} \end{aligned} \tag{6.29}$$

Подставляя в (6.27) выражения (6.28), (6.29), находим

$$L(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} \tag{6.30}$$

Эта формула называется интерполяционным многочленом Лагранжа.

Покажем, что этот многочлен является единственным. Допустим противоположное: пусть существует еще один многочлен F(x) степени n, принимающий в узлах интерполяции заданные значения, т.е. $F(x_i)=y_i$. Тогда разность $R(x)=L(x)-F(x)$, являющийся многочленом степени n (или ниже), в узлах x_i равна

$R(x_i)=L(x_i)-F(x_i)=0$, i=0,1,…,n.

Это означает, что многочлен R(x) степени не больше n имеет n+1 корней. Отсюда следует, что R(x)=0 и L(x)=F(x).

Из формулы (6.30) можно получить выражение для линейной (n=1) и квадратичной (n=2) интерполяций:

$$L(x) = \frac{x-x_1}{x_0-x_1} y_0 + \frac{x-x_0}{x_1-x_0} y_1, \text{ n=1;}$$

$$L(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} y_2, \text{ n=2.}$$

Существует несколько обобщений интерполяционного многочлена Лагранжа. Например, довольно широко используются интерполяционные многочлены Эрмита. Здесь наряду со значениями функции y_i'. Здесь задача состоит в том, чтобы найти многочлен $\varphi(x)$ степени 2n+1, значения которого и его производной в узлах x_i удовлетворяют соответственно соотношениям

$$\varphi(x_i) = y_i, \ \varphi'(x_i) = y_i', \ i = 0,1,..., n.$$

В этом случае так же существует единственное решение, если все x_i различны.

3.3 Многочлен Ньютона

До сих не делалось никаких предположений о законе распределения узлов интерполяции. Теперь рассмотрим случай равноотстоящих значений аргумента, т.е. $x_i-x_{i-1}=h=\text{const}$ (i=1,2…,n). Величина h называется шагом.

Введем также понятие конечных разностей. Пусть известны значения функции в узлах x_i: $y_i=f(x_i)$. Составим разности значений функции:

$$\Delta y_0 = y - y_0 = f(x_0 + h) - f(x_0),$$

$$\Delta y_1 = y_2 - y_1 = f(x_0 + 2h) - f(x_0 + h),$$

……..

$$\Delta y_{n-1} = y_n - y_{n-1} = f(x_0 + nh) - f(x_0 + (n-1)h).$$

Эти значения называют первыми разностями (или разностями первого порядка) функции.
Можно составить вторые разности функции:

$$\Delta^2 y_0 = \Delta y_1 - \Delta y_0,\ \Delta^2 y_1 = \Delta y_2 - \Delta y_1, \ldots$$

Аналогично составляются разности порядка k:

$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i,\ \text{i=0, 1, …,n-1.}$$

Конечные разности можно выразить непосредственно через значения функций. Например,

$$\Delta^2 y_0 = \Delta\, y_1 - \Delta\, y_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0,$$

$$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0 = \ldots = y_3 - 3y_2 + 3y_1 - y_0.$$

Аналогично для любого k можно написать

$$\Delta^k y_0 = y_k - k y_{k-1} + \frac{k(k-1)}{2!} y_{k-2} + \ldots + (-1)^k y_0 \qquad (6.31)$$

Эту формула можно записать и для значения разности в узле x_i:

$$\Delta^k y_i = y_{k+1} - k y_{k+i-1} + \frac{k(k-1)}{2!} y_{k+i-2} + \ldots + (-1)^k y_i.$$

Используя конечные разности, можно определить y_k :

$$y_k = y_0 + k\Delta y_0 + \frac{k(k-1)}{2!}\Delta^2 y_0 + \ldots + \Delta^k y_0 \qquad (6.32)$$

Перейдем к построению интерполяционного многочлена. Ньютона. Этот многочлен будем искать в следующем виде:

$$N(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \qquad (6.33)$$

График многочлена должен проходить через заданные узлы, т.е. $N(x_i) = y_i$ (i=0, 1,…,n). Эти условия используем для нахождения коэффициентов многочлена:

$$N(x_0) = a_0 = y_0,$$

$$N(x_1) = a_0 + a_1(x_1 - x_0) = a_0 + a_1 h = y_1,$$

$$N(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = a_0 + 2a_1 h + 2a_2 h^2 = y_2$$

Найдем отсюда коэффициенты a_0, a_1, a_2:

$$a_0 = y_0,\ a_1 = \frac{y_1 - a_0}{h} = \frac{y_1 - y_0}{h} = \frac{\Delta y_0}{h},$$

$$a_2 = \frac{y_2 - a_0 - 2a_1 h}{2h^2} = \frac{y_2 - y_0 - 2\Delta y_0}{2h^2} = \frac{\Delta^2 y_0}{2h^2}.$$

Аналогично можно найти и другие коэффициенты. Общая формула имеет вид

$$a_k = \frac{\Delta^h y_0}{k! h^k},\ \text{k=0, 1, …, n.}$$

Подставляя эти выражения в формулы (6.33), получаем следующий вид интерполяционного многочлена Ньютона:

$$N(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2! h^2}(x - x_0)(x - x_1) + \ldots + \frac{\Delta^n y_n}{n! h^n}(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \qquad (6.34)$$

Конечные разности $\Delta^k y_0$ могут быть вычислены по формуле (6.31).

Интерполяционный многочлен Ньютона можно записать в виде

$$N(x_i + th) = y_i + t\Delta y_i + \frac{t(t-1)}{2!}\Delta^2 y_i + \ldots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_i \qquad (6.35)$$

i=0,1,…

Полученное выражение называется первым интерполяционным многочленом Ньютона для интерполирования вперед.

Интерполяционную формула (6.35) обычно используют для вычисления значений функции в точках левой половины рассматриваемого отрезка. Это объясняется следующим. Разности $\Delta^k yi$ вычисляются через значения функций $y_i, y_{i+1},\ldots,y_{i+k}$ причем $i+k \le n$; поэтому при больших значениях i мы не можем вычислить разности высших порядков ($k \le n-i$). Например, при i=n-3 в (6.35) можно учесть только $\Delta y, \Delta^2 y, \Delta^3 y$.

Для первой половины рассматриваемого отрезка разности лучше вычислять справа налево. В этом случае

$$t=(x-x_n)/h \qquad (6.36)$$

т.е. t<0, и интерполяционный многочлен можно получить в виде

$$N(x_n + th) = y_n + t\Delta y_{n-1} + \frac{t(t+1)}{2!}\Delta^2 y_{n-2} + \ldots + \frac{t(t+1)\ldots(t+n-1)}{n!}\Delta^n y_0 \qquad (6.37)$$

Полученная формула называется вторым интерполяционным многочленом Ньютона для интерполирования назад.

Рассмотрим пример применения интерполяционной формулы Ньютона при ручном счете.

Пример. Вычислить в точках x=0.1, 0.9 значения функции y=f(x), заданной таблице 6.1.

Процесс вычислений удобно свести в ту же Таблица 6.1. Каждая последующая конечная разность получается путем вычитания в предыдущей колонке верхней строки из нижней.

Таблица 6.1

X	Y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
0	1.2715	1.1937	-0.0146	0.0007	-0.0001	0.0000
0.2	2.4652	1.1791	—0.0139	0.0006	—0.0001	
0.4	3.6443	1.1652	—0.0133	0.0005		
0.6	4.8095	l.f519	—0.0128			
0.8	5.9614	1.1391				
1.0	7.1005					

При ч=0.1 имеем $t=(x-x_0)/h=(0.1-0)/0.2=0.5$ По формуле получим

$$f(0.1) \approx N(0.1) = 1.2715 + 0.5\cdot 1.1937 + \frac{0.5(0.5-1)}{2!}\cdot(-0.0146) +$$

$$+\frac{0.5(0.5-1)(0.5-2)}{3!}\cdot 0.0007 + \frac{0.5(0.5-1)(0.5-2)(0.5-3)}{4!}\cdot(-0.0001) = 1.2715 + 0.59685 +$$

$$+ 0.00004 + 0.000004 = 1.8702.$$

Для сравнения по формуле линейной интерполяции получаем

$$f(0.1) \approx 1.8684.$$

Значение функции в точке x=0.9. $t=(x-x_n)/h=(0.9-1)/0.2=-0.5$. Тогда

$$f(0.9) \approx N(0.9) = 7.1005 - 0.5\cdot 1.1391 - \frac{0.5(-0.5+1)}{2!}\cdot(-0.0128) - \frac{0.5(-0.5+1)(-0.5+2)}{3!}\cdot 0.0005 -$$

$$-\frac{0.5(-0.5+1)(-0.5+2)(-0.5+3)}{4!}\cdot(-0.0001) = 7.1005 - 0.5696 + 0.0016 - 0.00003 + 0.0000004 =$$

$$= 6.5325$$

Мы рассмотрели построение интерполяционного многочлена Ньютона для равноотстоящих узлов. Можно построить многочлен Ньютона и для произвольно расположенных узлов, как и в случае многочлена Лагранжа.

В заключении отметим, что разные способы построения многочленов Лагранжа и Ньютона дают тождественные

интерполяционные формулы при заданной таблицу значений функции. Это следует из единственности интерполяционного многочлена заданной степени (при отсутствии совпадающих узлов интерполяции).

3.4 Сплайны

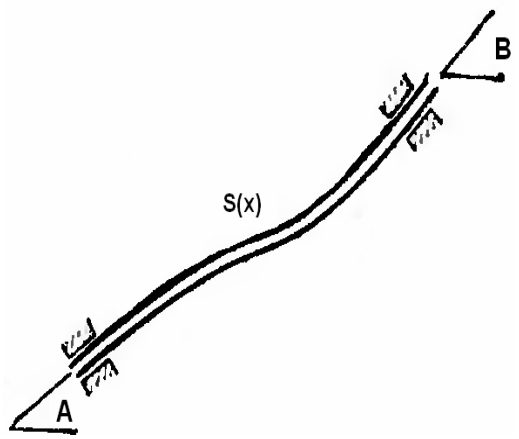


Рисунок 6.3

Сейчас широкое распространение для интерполяции получило использование кубических сплайн-функций — специальным образом построенных многочленов третьей степени. Они представляют собой некоторую математическую модель гибкого тонкого стержня из упругого материала. Если закрепить его в двух соседних узлах интерполяции с заданными углами наклонов A и B (Рисунок 6.3), то между точками закрепления этот стержень (механический сплайн) примет некоторую форму, минимизирующую его потенциальную энергию. Пусть форма этого стержня определяется функцией y=S(x). Из курса сопротивления материалов известно, что уравнение свободного равновесия имеет вид $S^{IV}(x)= 0$. Отсюда следует, что между каждой парой соседних узлов интерполяции функция S(x)является многочленом третьей степени. Запишем ее в виде

$$S(x) = a_i + b(x - x_{i-1}) + c(x - x_{i-1})^2 + d_i(x - x_{i-1})^3 \qquad (6.38)$$

$x_{i-1} \le x \le x_i$

Для определения коэффициентов a_i, b_i, c_i, d_i на всех n элементарных отрезках необходимо получить 4n уравнений. Часть из них вытекает из условий прохождения графика функции S(x) через заданные точки, т.е $S(x_{i-1})=y_{i-1}$. $S(x_i)=y_i$. Эти условия можно записать в виде

$$S(x_{i-1}) = a_i = y_{i-1} \qquad (6.39)$$

$$S(x_i) = a_i + b_i h_i + c_i h^2{}_i + d_i h_i^3 = y_i \qquad (6.40)$$

$$h_i = x_i - x_{i-1}, \text{ i=1, 2, , …, n.}$$

Эта система содержит 2n уравнении. Для получения недостающих уравнений зададим условия непрерывности первых и вторых производных в узлах интерполяции, т. е. условия гладкости кривой во всех точках. Вычислим производные многочлена (6.38).

$$S'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2,$$

$$S''(x) = 2c_i + 6d_i(x - x_{i-1}).$$

Приравнивая в каждом внутреннем узле $x = x_i$ значения этих производных, вычисленные в левом и правом от узла интервалах, получаем 2n—2 уравнений

$$b_{i+1} = b_i + 2c_i h_i + 3h_i^2 d_i \tag{6.41}$$

$$c_{i+1} + c_i + 3h_i d_i \tag{6.42}$$

i=1, 2, …, n-1.

Недостающие два соотношения получаются из условий закрепления концов сплайна.

В частности, при свободном закреплении концов можно приравнять нулю кривизну линии в этих точках. Такая функция, называемая свободным кубическим сплайном, обладает свойством минимальной кривизны, т. е. она самая гладкая среди всех интерполяционных функций данного класса. Из условий нулевой кривизны на концах следуют равенства нулю вторых производных в этих точках:

$$S''(x_0) = c_1 = 0, \qquad S''(x_n) = 2c_n + 6d_n h_n = 0 \tag{6.43}$$

Уравнения (6.39) – (6.43) составляют систему алгебраических уравнений для определения 4n коэффициентов a_i, b_i, c_i, d_i, (i=1,2, …,n).

Однако с целью аксиомой памяти ЭВМ и машинного времени эту систему можно привести к более удобному виду. Из условия (6.39) сразу можно найти все коэффициенты a_i. Далее из (6.42), (6.43) получим

$$d_i = \frac{c_{i+1} - c}{3h_i}, \qquad \text{i=1,2,…,n-1,} \qquad d_n = -\frac{c_n}{3h_n} \tag{6.44}$$

Подставим эти соотношения, а также значения $a_i=y_i$-1 в (6.40) и найдем отсюда коэффициенты

$$b_i = \frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i), \text{ i=1,2,…,n-1.}$$

$$b_n \frac{y_n - y_{n-1}}{h_n} - \frac{3}{2} h_n c_n \tag{6.45}$$

Учитывая выражения (6.44) и (6.45), исключаем из уравнения (6.40) коэффициенты d_i и b_i. Окончательно получим следующую систему уравнений только для коэффициентов c_i:

$$c_1 = 0, \qquad c_{n+1} = 0,$$

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}}\right) \tag{6.46}$$

i=1,2,…,n.

Матрица этой системы трехдиагональная, т. е. ненулевые элементы находятся лишь на главной и двух соседних с ней диагоналях, расположенных сверху и снизу. Для ее решения целесообразно использовать метод прогонки. По найденным из системы (6.46) коэффициентам c_i легко вычислить коэффициенты d_i и b_i.

3.5 Точность интерполяции

График интерполяционного многочлена y=F(x) проходит через заданные точки, т.е. значения многочлена и данной функции y=f(x) совпадают в узлах $x=x_i$ (i=0, 1, …,n). Если функция f(x) сама является многочленом степени n, то имеет место тождественное совпадение: f(x)=F(x). В общем случае в точках, отличных от узлов интерполяции $R(x)= f(x)-F(x) \neq 0$. Эта разность есть погрешность интерполяции и называется остаточным членом интерполяционной формулы. Оценим его значение.

Предположим, что заданные числа y_i являются значениями некоторой функции y=f(x) в точках $x=x_i$.Пусть эта функция непрерывна и имеет непрерывные производные до n+1-го порядка включительно. Можно показать, что в этом случае остаточный член интерполяционного многочлена Лагранжа имеет вид

$$R_L(x) = \frac{(x - x_0)(x - x_1)...(x - x_n)}{(n+1)!} f^{(n+1)}(x_*) \tag{6.47}$$

Здесь $f^{(n+1)}(x_*)$ - производная n+1 го порядка функции f(x) в некоторой точке $x = x_* \in [x_0, x_n]$. Если максимальное значение этой производной равно

$$\max_{x_0 \le x \le x_n} \left| f^{(n+1)}(x) \right| = M_{n+1}$$

То можно записать формулу для оценки остаточного члена:

$$\left| R_L(x) \right| \le \frac{(x - x_0)(x - x_1)...(x - x_n)}{(n+1)!} M_{n+1}.$$

Остаточный член интерполяционного многочлена Ньютона можно записать в виде

$$R_N(x) = \frac{t(t-1)...(t-n)}{(n+1)!} f^{(n+1)}(x_*) h^{n+1}, \; t = \frac{x - x_0}{h}$$

Выбор способа интерполяции определяется различными соображениями: точностью, временем вычислении, погрешностями округлений и др. В некоторых случаях более предпочтительной может оказаться локальная

интерполяция, в то время как построение единого многочлена высокой степени (глобальная интерполяция) не приводит к успеху.

Такого рода ситуацию в 1901 г. обнаружил К. Рунге. Он строил на отрезке $-1 \le x \le 1$ интерполяционные многочлены с равномерным распределением узлов для функции $y = 1/(1+25x^2)$. Оказалось, что при увеличении степени интерполяционного многочлена последовательность его значении расходится для любой фиксированной точки x при $0.7 < |x| < 1$.

Положение в некоторых случаях может быть исправлено специальным распределением узлов интерполяции (если они не зафиксированы). Доказано, что если функция f(x) имеет непрерывную производную на отрезке [—1, 1], то при выборе значении x_i, совпадающих с корнями многочленов Чебышева степени n + 1, интерполяционные многочлены степени n сходятся к значениям функции в любой точке этого отрезка.

Таким образом, повышение точности интерполяции целесообразно производить за счет уменьшения шага и специального расположения точек x_i. Повышение степени интерполяционного многочлена при локальной интерполяции также уменьшает погрешность, однако здесь не всегда ясно поведение производной $f^{(n+1)}(x_*)$ при увеличении n. Поэтому на практике стараются использовать многочлены малой степени (линейную и квадратичную интерполяции, сплайны).

§4.Аппроксимация. Метод наименьших квадратов. Эмпирические формулы. Локальное сглаживание данных.

4.1Аппроксимация методом наименьших квадратов (МНК)

Метод наименьших квадратов (часто называемый МНК) обычно упоминается в двух контекстах. Во-первых, широко известно его применение в регрессионном анализе, как метода построения моделей на основе зашумленных экспериментальных данных. При этом помимо собственно построения модели обычно осуществляется оценка погрешности, с которой были вычислены её параметры, иногда решаются и некоторые другие задачи. Во-вторых, МНК часто применяется просто как метод аппроксимации, без какой-либо привязки к статистике. На этой странице МНК рассматривается как метод аппроксимации.

Общий линейный метод наименьших квадратов

При аппроксимации методом наименьших квадратов аппроксимируемая функция f задается набором N точек (xi , yi). Аппроксимирующая функция g строится, как линейная комбинация базисных функций Fj (количество функций M обычно меньше числа точек N)

$$g = \sum_{j=0}^{M-1} c_j F_j(x)$$

При этом коэффициенты c_j выбираются таким образом, чтобы минимизировать сумму квадратов отклонений аппроксимирующей функции от заданных значений

$$E = \sum_{i=0}^{N-1} \left(y_i - \sum_{j=0}^{M-1} c_j F_j(x_i) \right)^2$$

Иногда, если различным точкам назначен различный вес wi , каждое слагаемое в сумме квадратов умножается на квадрат соответствующего ему веса:

$$E = \sum_{i=0}^{N-1} w_i^2 \left(y_i - \sum_{j=0}^{M-1} c_j F_j(x_i) \right)^2$$

Такая задача называется построением взвешенной аппроксимации по МНК. Следует отметить, что выбор именно такого способа аппроксимации (линейной комбинации базисных функций) и именно такой оценочной функции (суммы квадратов отклонений) является далеко не единственно возможным. Базисные функции могут входить в функцию g нелинейно, а оценочная функция E может быть заменена максимумом отклонения или любой другой функцией оценки. Однако именно такой способ аппроксимации с такой оценочной функцией позволяет нам найти наилучшие cj за конечное число операций, сведя задачу к решению системы линейных уравнений.

Методы поиска коэффициентов

Перед тем, как рассматривать способы поиска коэффициентов cj , введем следующие обозначения:

$$A = \begin{pmatrix} a_{11} & a_{12} & .. \\ a_{21} & a_{22} & .. \\ .. & .. & .. \\ a_{n1} & .. & a_{nm} \end{pmatrix}, \quad a_{ij} = w_i F_j(x_i)$$

$$b = \begin{pmatrix} b_1 \\ .. \\ .. \\ b_n \end{pmatrix}, \quad b_i = w_i y_i$$

$$c = \begin{pmatrix} c_1 \\ .. \\ .. \\ c_n \end{pmatrix}$$

Первый способ поиска оптимальных cj состоит в решении системы уравнений $\frac{\partial E}{\partial c_j} = 0$(т.н. метод нормальных уравнений). С учетом особенностей задачи, равенство производных нолю является не только необходимым, но и достаточным условием минимума. Кроме того, эта система уравнений является линейной относительно c_j и записывается как !!!!(A TA)c - A Tb = 0. Основным недостатком этого метода является то, что полученная система уравнений часто бывает плохо обусловленной. Число обусловленности матрицы A само по себе может быть велико (например, в случае использования базиса из полиномов), а возведение матрицы A в квадрат увеличивает его ещё больше (также возводит в квадрат). По этой причине метод нормальных уравнений на практике обычно не применяется.

Второй способ основан на том, что задачу минимизации функции E можно записать в матричной форме, как поиск !!!!$\min_c \|Ac - b\|_2$!!!. Таким образом, задача сведена к поиску псевдорешения системы линейных уравнений, которое может быть найдено с использованием сингулярного разложения (SVD). Этот способ является наиболее естественным, хотя и требует более сложного программного кода для своей реализации.

Достоинством этого метода является то, что в этом случае не происходит возведения в квадрат числа обусловленности базиса. Мы решаем систему, обусловленность которой равна обусловленности базиса, что позволяет получать достаточно точные решения даже в тех случаях, когда использование метода нормальных уравнений приводит к возникновению системы с вырожденной матрицей коэффициентов. Следует отметить, что хотя для осуществления сингулярного разложения мог бы использоваться предназначенный для этого общий алгоритм, в этом нет необходимости. С учетом специфики задачи предпочтительнее использовать более компактную и быструю версию алгоритма, учитывающую тот факт, что требуется не само сингулярное разложение матрицы A, а произведение отдельных компонент этого разложения на вектор b.

Также существует третий способ, применяющий QR-разложение, и работающий несколько быстрее метода на основе SVD-разложения. Сначала матрица A представляется в виде произведения прямоугольной ортогональной матрицы Q и квадратной верхнетреугольной матрицы R. Затем решается система уравнений Rx = Q Tb. Достоинством этого метода является относительно высокая скорость работы, недостатком - применимость только к невырожденным задачам (т.е. задачам, имеющим одно и только одно решение). Для того, чтобы задача аппроксимации была невырожденной, требуется выполнение двух условий. Во-первых, система базисных функций должна быть линейно-независимой. Во-вторых, число точек должно быть не меньше числа базисных функций. Как правило, в практических задачах эти условия выполняются. Но все же очень редко встречаются задачи, в которых алгоритм на основе QR-разложения оказывается неприменим из-за нарушения одного из условий. Кроме того, обычно число строк в матрице A намного больше числа столбцов, а правильно реализованное SVD разложение таких матриц по трудоемкости сравнимо с QR разложением. Таким образом, более целесообразным представляется использование второго способа, т.е. SVD-разложения.

4.2Линейная аппроксимация по МНК

При линейной аппроксимации по МНК в качестве набора базисных функций используются f0 = 1 и f1 = x. Линейная комбинация функций из этого базиса позволяет получить произвольную прямую на плоскости.

Особенностью этого базиса является то, что в данном случае оказывается применим метод нормальных уравнений. Базис небольшой, хорошо обусловленный, и поэтому задача аппроксимации легко сводится к решению системы линейных уравнений 2x2. Для решения используется модификация метода вращений,

которая позволяет корректно обрабатывать вырожденные случаи (например, точки с совпадающими абсциссами).
Этот алгоритм реализован в подпрограмме BuildLinearLeastSquares.
Полиномиальная аппроксимация по МНК
Частой ошибкой является использование для полиномиальной аппроксимации следующего набора базисных функций: fi = x_i. Этот базис является наиболее естественным и, пожалуй, первым приходящим в голову вариантом. Но хотя с этим базисом удобно работать, он очень плохо обусловлен. Причем проблемы появляются даже при умеренном числе базисных функций (порядка десяти). Эта проблема была решена путем выбора базиса из полиномов Чебышева вместо базиса из степеней x. Полиномы Чебышева линейно выражаются через степени x и наоборот, так что эти базисы эквивалентны. Однако обусловленность базиса из полиномов Чебышева значительно лучше, чем у базиса из степеней x. Это позволяет без проблем осуществлять аппроксимацию полиномами практически любой степени.
Алгоритм работает следующим образом. Входной набор точек масштабируется и приводится к интервалу [−1,1], после чего на этом отрезке строится базис из полиномов Чебышева. После аппроксимации по МНК пользователь получает набор коэффициентов при полиномах Чебышева. Эта задача решается подпрограммой BuildChebyshevLeastSquares. Полученные коэффициенты могут быть переданы в подпрограмму CalculateChebyshevLeastSquares, которая вычисляет значение аппроксимирующей функции в требующей точке.
Если специфика решаемой задачи требует использования именно базиса из степеней x, то можно воспользоваться подпрограммой BuildPolynomialLeastSquares. Эта подпрограмма решает задачу с использованием полиномов Чебышева, после чего преобразовывает результат к степеням x. Это позволяет отчасти сохранить точность вычислений, т.к. все промежуточные расчеты ведутся в хорошо обусловленном базисе. Однако все же предпочтительным вариантом является базис из полиномов Чебышева.
В некоторых случаях требуется решение задачи с ограничениями. Например, мы можем знать, что аппроксимируемая функция равна 1 на границе, т.е. при x = 0. Если у нас имеется достаточно большое количество точек в окрестностях x = 0, то построенная функция пройдет в окрестностях точки (0,1). Однако точного совпадения скорее всего не будет. Во многих случаях точное соблюдение каких-то граничных условий не требуется. Однако иногда встречаются задачи, требующие именно точного прохождения аппроксимирующей функции через несколько заданных точек или равенства производной в некоторых точках заранее заданным значениям. При этом особые требования предъявляются только к нескольким точкам xk . В остальных же точках мы просто минимизируем сумму квадратов отклонений.
В таком случае имеет место задача аппроксимации с ограничениями вида

$$g(x_k) = g_k$$

или

$$\frac{dg(x_k)}{dx} = \widetilde{g}_k$$

Для решения таких задач предназначен специальный алгоритм, реализованный в подпрограмме BuildChebyshevLeastSquaresConstrained. В настоящее время подпрограмма способна обрабатывать произвольное количество ограничений (но строго меньшее, чем M) на значение функции и её первой производной. Ограничения на значения производных высших порядков не поддерживаются.

4.3Аппроксимация кубическими сплайнами

Ещё одним возможным набором базисных функций являются кубические сплайны. Множество сплайнов с общими узлами образует линейное пространство, что позволяет применить к ним линейный метод наименьших квадратов. В качестве базисных функций выбираются сплайны, удовлетворяющие следующим условиям:

$$S_j(i) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

$$i, j = 0 \ldots M-1$$

Для простоты мы предполагаем, что аппроксимация строится на отрезке [0,M-1], с узлами, равномерно распределенными по отрезку. Разумеется, в реальности отрезок, на котором осуществляется аппроксимация, может быть любым (однако равномерность распределения узлов сохраняется). На графике ниже приведен пример такой системы базисных функций:

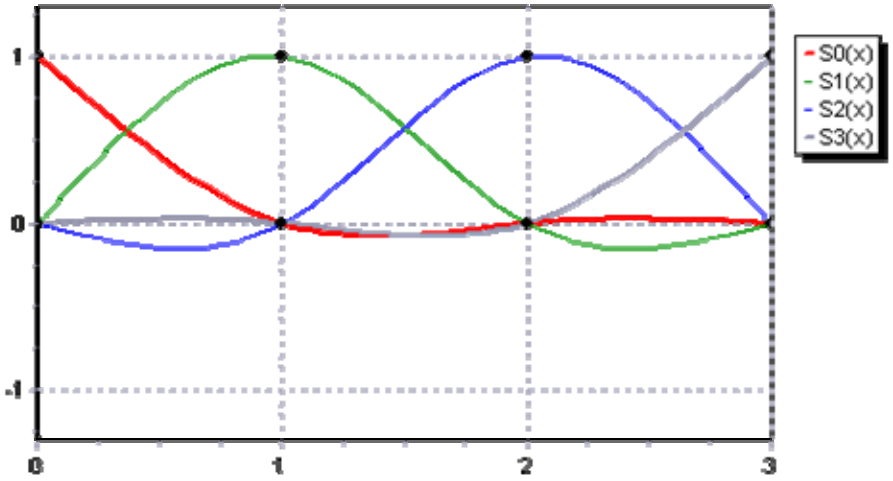


Рисунок 6.4

Построение аппроксимирующей функции осуществляется подпрограммой BuildSplineLeastSquares. Результатом её работы является массив коэффициентов, задающий результирующий кубический сплайн. Этот массив передается в подпрограмму SplineInterpolation (см. описание модуля для интерполяции кубическими сплайнами), которая рассчитывает значение аппроксимирующего сплайна в указанной точке.

4.4Аппроксимация произвольным набором базисных функций

Выше были приведены три набора базисных функций, которые могут использоваться для аппроксимации: прямые, полиномы и кубические сплайны. Однако это не единственные наборы функций, которые могут быть применены. И, разумеется, нельзя предусмотреть все возможные практические ситуации и то, какой набор функций окажется востребован. Поэтому любой программный пакет, решающий задачу аппроксимации, должен содержать в себе подпрограмму для аппроксимации произвольным набором базисных функций. Эту задачу решает подпрограмма BuildGeneralLeastSquares.

Поскольку заранее неизвестно, какой именно набор функций используется, в подпрограмму требуется передать информацию о значениях базисных функций в точках xi . Эта информация передается в виде матрицы FMatrix, содержащей значение i-ой базисной функции в j-ой точке. Также в подпрограмму передаются массивы y и w, содержащие ординаты исходных точек и их веса.

Обратите внимание, что абсциссы xi в подпрограмму не передаются. Дело в том, что при аппроксимации общим МНК абсциссы точек участвуют в процессе только как аргументы базисных функций, а эта информация уже содержится в матрице FMatrix. То есть процесс аппроксимации не зависит от размерности пространства, в котором она строится - достаточно пронумеровать точки, вычислить в них значения функции и передать информацию в алгоритм. Это позволяет использовать подпрограмму как для решения одномерных задач, так и для многомерных проблем.

4.5 Подбор эмпирических формул

Характер опытных данных.

При интерполировании функции мы использовали условие равенства значений интерполяционного многочлена и данной функции в известных точках — узлах интерполяции. Это предъявляет высокие требования к точности данных значений функции. В случае обработки опытных данных, полученных в результате наблюдении или измерении, нужно иметь в виду ошибки этих данных. Они могут быть вызваны несовершенством измерительного прибора, субъективными причинами, различными случайными факторами и т. д. Ошибки экспериментальных данных можно условно разбить на три категории по их происхождению и величине: систематические, случайные и грубые.

Систематические ошибки обычно дают отклонение в одну сторону от истинного значения измеряемой величины. Они могут быть постоянными пли закономерно изменяться при повторении опыта, и их причина и характер известны. Систематические ошибки могут быть вызваны условиями эксперимента (влажностью, температурой среды и др.), дефектом измерительного прибора, его плохой регулировкой (например, смещением указательной стрелки от нулевого положения) и т. д. Эти ошибки можно устранить наладкой аппаратуры или введением соответствующих поправок.

Случайные ошибки определяются большим числом факторов, которые не могут быть устранены либо достаточно точно учтены при измерениях или при обработке результатов. Они имеют случайный (несистематический) характер, дают отклонении от средней величины в ту и другую стороны при повторении измерении и не могут быть устранены в эксперименте, как бы тщательно он ни проводился. С вероятностной точки зрения математическое ожидание случайной ошибки равно нулю. Статистическая обработка экспериментальных данных позволяет определить величину случайной ошибки и довести ее до некоторого

приемлемого значения повторением измерений достаточное число раз.
Грубые ошибки явно искажают результат измерения; они чрезмерно большие и обычно пропадают при повторении опыта. Грубые ошибки существенно выходят за пределы случайной ошибки, полученные при статистической обработке. Измерения с такими ошибками отбрасываются и в расчет при окончательной обработке результатов измерений не принимаются.
Таким образом, в экспериментальных данных всегда имеются случайные ошибки. Они, вообще говоря, могут быть уменьшены до сколь угодно малой величины путем многократного повторения опыта. Однако это не всегда целесообразно, поскольку могут потребоваться большие материальные пли временные ресурсы. Значительно дешевле и быстрее можно в ряде случаев получить уточненные данные хорошей математической обработкой имеющихся результатов измерений.
В частности, с помощью статистической обработки результатов измерений можно найти закон распределения ошибок измерений, наиболее вероятный диапазон изменения искомой величины (доверительный интервал) и другие параметры. Рассмотрение этих вопросов выходит за рамки данного пособия; их изложение можно найти в некоторых книгах, приведенных в списке литературы. Здесь ограничимся лишь определением связи между исходным параметром x и искомой величиной y на основании результатов измерений.

4.6 Эмпирические формулы.

Пусть, изучая неизвестную функциональную зависимость между y и x, мы в результате серии экспериментов произвели ряд измерений этих величин и получили таблицу значении

x_0	x_1	$\cdots$	x_n
y_0	y_1	$\cdots$	y_n

Задача состоит в том, чтобы найти приближенную зависимость

$$y = f(x) \tag{6.48}$$

значения которой при $x = x_i \quad (i = 0,1,...,n)$ мало отличаются от опытных данных y_i. Приближенная функциональная зависимость (6.48), полученная на основании экспериментальных данных, называется *эмпирической формулой.*
Задача построения эмпирической формулы отличается от задачи интерполирования. График эмпирической зависимости, вообще говоря, не проходят через заданные точки (x_i, y_i), как в случае интерполяции. Это приводит к тому, что экспериментальные данные в некоторой степени сглаживаются, а интерполяционная формула повторила бы все ошибки, имеющиеся в экспериментальных данных.
Построение эмпирической формулы состоит из двух этапов: подбора общего вида этой формулы и определения наилучших значений содержащихся в ней параметров. Общий вид формулы иногда известен из физических соображений. Например, для упругой среды связь между напряжением σ и относительной деформацией ε определяется законом Гука: $\sigma = E\varepsilon$, где E — модуль упругости; задача сводится к определению одного неизвестного параметра E.
Если характер зависимости неизвестен, то вид эмпирической формулы может быть произвольным. Предпочтение обычно отдается наиболее простым формулам, обладающим достаточной точностью. Они первоначально выбираются из геометрических соображений: экспериментальные точки наносятся на график и примерно угадывается общий вид зависимости путем сравнения полученной кривой с графиками известных функций (многочлена, показательной пли логарифмической функций и т. п.). Успех здесь в значительной мере определяется опытом и интуицией исследователя.
Простейшей эмпирической формулой является линейная зависимость

$$y = ax + b \tag{6.49}$$

Близость экспериментального распределения точек к линейной зависимости легко просматривается после построения графика данной экспериментальной зависимости, Кроме того, эту зависимость можно проверить путем вычисления значений k_i:

$$k_i = \Delta y_i / \Delta x_i, \ \Delta y_i = y_{i+1} - y_i, \ \Delta x_i = x_{i+1} - x_i, \ i = 0,1,...,n-1.$$

Если при этом $k_i \approx const$, то точки (x_i, y_i) расположены приблизительно на одной прямой, и может быть поставлен вопрос о применимости эмпирической формулы (6.49). Точность такой аппроксимации определяется отклонением величин k_i от постоянного значения. В частном случае равноотстоящих точек x_i, (т. е. $\Delta x_i = const$) достаточно проверить постоянство разностей Δy_i.
Пример. Проверим возможность использования линейной зависимости для описания следующих данных:
Поскольку здесь x_i — равноотстоящие точки ($\Delta x_i = x_{i+1} - x_i = 0.5$), то достаточно вычислить разности Δy_i **:**
0.64, 0.69, 0.65, 0.64, 0.65. Так как эти значения близки друг к другу, то в качестве эмпирической формулы можно принять линейную зависимость.

В ряде случаев к линейной зависимости могут быть сведены и другие экспериментальные данные, когда их график в декартовой системе координат не является прямой линией. Это может быть достигнуто путем введения новых переменных ξ, η вместо x и y*:*

$$\xi = \varphi(x, y), \qquad \eta = \psi(x, y) \tag{6.50}$$

Функции $\varphi(x, y)$ и $\psi(x, y)$ выбираются такими, чтобы точки (ξ_i, η_i) лежали на некоторой прямой линии в плоскости (ξ, η). Такое преобразование называется *выравниванием данных.*

Для получения линейной зависимости

$$\eta = a\xi + b$$

с помощью преобразования (6.50) исходная формула должна быть записана в виде

$$\psi(x, y) = a\varphi(x, y) + b.$$

К такому виду легко сводится, например, степенная зависимость $y = ax^b$ (x>0, y>0). Логарифмируя эту формулу, получаем $\lg y = b \lg x + \lg a.$ Полагая $\xi = \lg x$, $\eta = \lg y$, находим линейную связь; $\eta = b\xi + c$ $(c = \lg a).$

4.7 Локальное сглаживание данных.

Как отмечалось ранее , опытные данные содержат случайные ошибки, что является причиной разброса этих данных. Во многих случаях бывает целесообразно провести их сглаживание для получения более плавного характера исследуемой зависимости. Существуют различные способы сглаживания. Рассмотрим один из них, основанный на методе наименьших квадратов.

Пусть в результате экспериментального исследования зависимости y = f{x) получена таблица значений искомой функции $y_0, y_1,\ldots,y_n$ в точках $x_1, x_0,\ldots, x_n$. Значения аргумента x (предполагаются равноотстоящими, а опытные данные y_i — имеющими одинаковую точность. Предполагается также, что функция y = f(x) на произвольной части отрезка $[x_0,x_n]$ может быть достаточно хорошо аппроксимирована многочленом некоторой степени n.

Рассматриваемый способ сглаживания состоит в следующем. Для нахождения сглаженного значения $\bar{y}_i$ в точке x_i выбираем по обе стороны от нее k значений аргумента. По опытным значениям рассматриваемой функции в этих точках $y_{i-h/2},\ldots,y_{i-1},y_i,y_{i+1},\ldots,y_{i+h/2}$ строим многочлен степени m с помощью метода наименьших квадратов (при этом $m \le k$). Значение полученного многочлена $\bar{y}_i$ в точке x_i и будет искомым (сглаженным) значением. Процесс повторяется для всех внутренних точек. Сглаживание значений, расположенных вблизи концов отрезка $[x_0, x_n]$, производится с помощью крайних точек.

Опыт показывает, что сглаженные значения $\bar{y}_i$, как правило, с достаточной степенью точности близки к истинным значениям. Иногда сглаживание повторяют. Однако это может привести к существенному искажению истинного характера рассматриваемой функциональной зависимости.

Приведем в заключение несколько формул для вычисления сглаженных опытных данных при различных m, k:

m=1:

$$\bar{y}_i = \frac{1}{3}(y_{i-1}+y_i + y_{i+1}), \qquad \text{k=2,}$$

$$\bar{y}_i = \frac{1}{5}(y_{i-2} + y_{i-1} + y_i + y_{i+1} + y_{i+2}), \quad \text{k=4,}$$

$$\bar{y}_i = \frac{1}{7}(y_{i-3} + y_{i-2} + y_{i-1} + y_i + y_{i+1} + y_{i+2} + y_{i+3}), \qquad \text{k=6,}$$

m=3:

$$\bar{y}_i = \frac{1}{35}(-3y_{i-2} + 12y_{i-1} + 17y_i + 12y_{i+1} - 3y_{i+2}), \qquad \text{k=4,}$$

$$\bar{y}_i = \frac{1}{21}(-2y_{i-3} + 3y_{i-2} + 6y_{i-1} + 7y_i + 6y_{i+1} + 3y_{i+2} - 2y_{i+3}), \quad \text{k=6,}$$

$$\bar{y}_i = \frac{1}{231}(-21y_{i-4} + 14y_{i-3} + 39y_{i-2} + 54y_{i-1} + 59y_i + 54y_{i+1} + 39y_{i+2} + 14y_{i+3} - 21y_{i+4}), \qquad \text{k=8;}$$

$$\bar{y}_i = \frac{1}{231}(-5y_{i-3} - 30y_{i-2} + 75y_{i-1} + 131y_i + 75y_{i+1} - 30y_{i+2} + 5y_{i+3}), \qquad \text{k=5,}$$

$$\bar{y}_i = \frac{1}{429}(15y_{i-4} - 55y_{i-3} + 30y_{i-2} + 135y_{i-1} + 179y_i + 135y_{i+1} + 30y_{i+2} - 55y_{i+3} + 15y_{i+4}), \quad \text{k=8.}$$

Глава 7
ДИФФЕРЕНЦИРОВАНИЕ И ИНТЕГРИРОВАНИЕ

§1.Численное дифференцирование. Аппроксимация производных. Погрешность численного дифференцирования. Использование интерполяционных формул. Метод неопределнных коэффициентов. Частные производные.

1.1 Численное дифференцирование.

Численное дифференцирование применяется в тех случаях, когда: функция f(x) задана таблично и, следовательно, методы дифференциального исчисления неприменимы; аналитическое выражение f(x) столь сложно, что вычисления производной представляют значительны трудности.

В основе численного дифференцирования лежит следующий прием: исходная функция f(x) заменяется на рассматриваемом отрезке [a,b] Интерполяционным полиномом $P_n(x)$ И считается, что f'(x) И $P'_n(x)$ Примерно равны, т.е. $f'(x)=P'_n(x)$, . $(a \le x \le b)$.

В работе рассматривается следующая задача: дана таблица функции, требуется построить таблицу ее производной с тем же шагом.

Всегда, когда это возможно, для численного дифференцирования используется интерполяционный многочлен с равноотстоящими узлами, так как это значительно упрощает формулы численного дифференцирования.

При равноотстоящих узлах в начале строится интерполяционный полином Ньютона, а затем он дифференцируется.

Если f(x) задана таблицей с неравноотстоящими узлами, то вначале строится интерполяционный полином Лагранжа, а затем он дифференцируется.

В работе изучается численное дифференцирование, основанное на первой интерполяционной формуле Ньютона. Из-за больших погрешностей численное дифференцирование практически используется для вычисления производных не выше второго порядка. Обычно при численном дифференцировании интерполяционный полином Ньютона строится не по всем узлам таблицы, а по трем-пяти узлам, близлежащим к точке…, в которой требуется вычислить производную. Если требуется вычислить производную во всех узлах, то вначале полином Ньютона строится по первым 3-5 узлам и в них вычисляется производная, потом полином Ньютона строится по следующим 3-5 узлам и в них вычисляется производная и т.д. до тех пор, пока не будет просчитана вся таблица.

Если по узлам x_0, x_1, x_2, x_3, x_4построить первый интерполяционный полином Ньютона и его продифференцировать с учетом

$$\frac{df(x)}{dx}=\frac{df(x)}{dq}\times\frac{dq}{dx}=\frac{1}{h}\times\frac{df(x)}{d(x)}$$

то получим приближенное выражение для *f'*(x) в виде:

$$f'(x)\approx\frac{1}{h}\left[\Delta y_0+\frac{2q-1}{2}\Delta^2 y_0+\frac{3q^2 6q+21}{6}\Delta^3 y_0+\frac{2q^3-9q^2+11q-3}{12}\Delta^4 y_0\right], \qquad (7.1)$$

где $q\,\frac{x-x0}{h}$ *и* $p=x_{j+1}-x_j \quad (j=0,1,2,...)$

Вторая производная *f''*(x)в результате дифференцирования формулы (7.1) с учетом равенства

$$f''(x)=\frac{d(f'(x)0}{dx}=\frac{d(f'(x))}{dq}\times\frac{dq}{dx}$$

имеет следующее приближенное выражение:

$$f''(x)\approx\frac{1}{h^2}\left[\Delta^2 y_0+(q-1)\Delta^3 y_0+\frac{6q^2-18q+11}{12}\Delta^4 y_0\right] \qquad (7.2)$$

Если требуется вычислить *f'*(x) в узлах x_0, x_1,…, x_n, то в формулу (7.1) необходимо поочередно поставить q=0, q=1, q=2, q=3, q=4 соответственно.

В случае выбора узлов x_j, x_{j+1}, ..., x_{j+4} вместо узлов x_0, x_1, ...,x_4 в формулах (7.1), (7.2) следует заменить конечные разности y_0 на конечные y_j, которые берутся из строчки с номером j таблицы конечных разностей.

Если требуется найти производные функции f(x) в основных табличных точка x_j то каждое табличное значение считается за начальное и тогда $x_j=x_0$, q=0, а формулы (7.1), (7.2) будут соответственно для всех *f'*(x) и *f''*(x) иметь вид:

$$f'(x_0) \approx \frac{1}{h}\left[\Delta y_0 - \frac{\Delta^2 y_0}{2} - \frac{\Delta^3 y_0}{3} - \frac{\Delta^4 y_0}{4}\right] \qquad (7.1')$$

$$f''(x_0) \approx \frac{1}{h^2}\left[\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0\right] \qquad (7.2')$$

Если интерполяционный полином строится не по пяти, а по четырем или трем узлам, то в формулах (7.1), (7.1'), (7.2), 7. 2' отбрасываются соответственно одно или два последних слагаемых.

Чтобы вычислить значение производной в точке, соответствующей концу таблицы, следут воспользоваться формулой, полученной дифференцированием второй интерполяционной формулы Ньютона, т.е. формулой

$$f'(x) = \frac{1}{h}\left(\Delta y_{n-1} + \frac{2q+1}{2}\Delta^2 y_{n-2} + \frac{3q^2+6q+2}{6}\Delta^3 y_{n-3} + \frac{2q^3+9q^2+11q+3}{12}\Delta^4 y_{n-4}\right]$$

Ошибка ограничения, получаемая при вычислении *k*-й производной по рассмотренным вые формулам, совпадает с *k*-й производной ошибки ограничения интерполяционного полинома Ньютона. Эту ошибку для первой производной можно приближенно оценить по формуле:

$$R'_k(x_0) = k^{k+1}\frac{q(q-1)(q-2)...(q-k)}{(k+1)!}y^{(k+1)},$$

где $\xi \in [a,b]$, но отлична от узла интерполяции.

для $x=x_0$ и $q=0$ эта формула упрощается:

$$R'_k(x_0) = (-1)^k \frac{h^k}{(k+1)} y^{(k+1)}(\xi)$$

А так как часто величину $y^{(k+1)}(\xi)$ получить трудно, то полагая при малом h, что

$$y_{\xi}^{(k+1)} \approx \frac{\Delta y_0}{h^{k+1}}$$

окончательно получаем

$$R'_k(x) \approx \frac{(-1)^k}{h} \times \frac{\Delta^{k+1} y_0}{k+1}$$

Таблица 7.1

i	x_i	y_i	Δy	Δ2y	Δ3y	Δ4y	Δ5y
0	X0	Y0	Δy0=y1-y0	Δ2y0=Δy1-Δy0	Δ3y0=Δ2y1-Δ2y0	Δ4y0=Δ3y1-Δ3y0	Δ5y0=Δ4y1-Δ4y0
1	X1	Y1	Δy1=y2-y1	Δ2y1=Δy2-Δy1	Δ3y1=Δ2y2-Δ2y1	Δ4y1=Δ3y2-Δ3y1	
2	X2	Y2	Δy2=y3-y2	Δ2y2=Δy3-Δy2	Δ3y2=Δ2y3-Δ2y2		
3	X3	Y3	Δy3=y4-y3	Δ2y3=Δy4-Δy3			
4	X4	Y4	Δy4=y5-y4				
5	X5	Y5					

1.2 Погрешность численного дифференцирования.

Аппроксимируем функцию f(x) некоторой функцией $\varphi(x)$, т. е. представим ее в виде

$$f(x) = \varphi(x) + R(x) \qquad (7.3)$$

В качестве аппроксимирующей функции $\varphi(x)$ можно принять частичную сумму ряда или интерполяционную функцию. Тогда погрешность аппроксимации R(x) определяется остаточным членом ряда или интерполяционной формулы.

Аппроксимирующая функция $\varphi(x)$ может быть использована также для приближенного вычисления производной функции f(x). Дифференцируя равенство (7.3) необходимое число раз, можно найти значения производных $f'(x), f'''(x),...$:

$$f'(x) = \varphi''(x) + R'(x)$$

В качестве приближенного значения производной порядка k функции f(x) можно принять соответствующее значение производной функции $\varphi(x)$, т.е. $f^{(k)}(x) \approx \varphi^{(k)}(x)$.

Величнна

$$R^{(k)}(x) = f^{(k)}(x) - \varphi^{(k)}(x),$$

характеризующая отклонение приближенного значения производной от ее истинного значения, называется погрешностью аппроксимации производной.

При численном дифференцировании функции, заданной в виде таблицы с шагом h, эта погрешность зависит от *h*, и ее записывают в виде O (h^k). Показатель степени k называется порядком погрешности аппроксимации производной (или просто порядком аппроксимации). При этом предполагается, что значение шага по модулю меньше единицы.

Оценку погрешности легко проиллюстрировать с помощью ряда Тейлора

$$f(x+\Delta x) = f(x) - f'(x)\Delta x + \frac{f''(x)}{2!}\Delta x^2 + \frac{f'''(x)}{3!}\Delta x^3 + \ldots$$

Пусть функция f(x) задана в виде таблицы $f(x_i)=y_i$, (i=0, 1, …,n) Запишем ряд Тейлора при $x=x_i$, $\Delta x = -h$ с точностью до членов порядка h;

$$y_0 = y_1 - y_1' h + O(h).$$

Отсюда найдем значение производной в точке $x = x_i$.

$$y_1' = \frac{y_1 - y_0}{h} + O(h)$$

Полагая $\Delta x = h$ и $\Delta x = -h$ соответственно получаем

$$y_0 = y_1 + y_1' h + \frac{y_1''}{2!}h^2 + \frac{y_1'''}{3!}h^3 + O(h^4) \qquad (7.4)$$

$$y_2 = y_1 - y_1' h + \frac{y_1''}{2!}h^2 - \frac{y_1'''}{3!} + O(h^4)$$

Вычитая эти равенства одно из другого, после очевидных преобразований получаем:

$$y_1' = \frac{y_2 - y_0}{2h} + O(h^2)$$

Это аппроксимация производной $\Delta y_1 = y_2 - y_0$, $\Delta x = 2h$, $y_1' \approx \frac{y_2 - y_0}{2h}$ с помощью центральных разностей. Она имеет второй порядок.

Складывая равенства (7.4), находим оценку погрешности аппроксимации производной второго порядка вида

$$y1'' = (y_1')' \approx \frac{y_2' - y_1'}{h} \approx \frac{(y_2 - y_1)/h - (y_1 - y_0)/h}{h} = \frac{y_2 - 2y_1 + y_0}{h^2}$$

Таким образом, эта аппроксимация имеет второй порядок. Аналогично можно получить аппроксимации производных более высоких порядков и оценку их погрешностей.

Мы рассмотрели лишь один из источников погрешности численного дифференцирования — погрешность аппроксимации (ее также называют погрешностью усечения). Она определяется величиной остаточного члена.

Анализ остаточного члена нетривиален, и сведения по этому вопросу можно найти в более полных курсах по численным методам и теории разностных схем. Отметим лишь, что погрешность аппроксимации при уменьшении шага h, как правило, уменьшается.

Погрешности, возникающие при численном дифференцировании, определяются также неточными значениями функции yi в узлах и погрешностями округлений при проведении расчетов на ЭВМ. В отличие от погрешности аппроксимации погрешность округления возрастает с уменьшением шага h. Поэтому суммарная погрешность численного дифференцирования может убывать при уменьшении шага лишь до некоторого предельного значения, после чего дальнейшее уменьшение шага не повысит точности результатов.

Оптимальная точность может быть достигнута за счет регуляризации процедуры численного дифференцирования. Простейшим способом регуляризации является такой выбор шага h, при котором справедливо неравенство $|f(x+h) - f(x)| > \varepsilon$, где $\varepsilon > 0$ — некоторое малое число. При вычислении производной это исключает вычитание близких по величине чисел, которое обычно приводит к увеличению погрешности. Это тем более опасно при последующем делении приращения функции на малое число h. Другой способ регуляризации — сглаживание табличных значений функции подбором некоторой гладкой аппроксимирующей функции, например многочлена.

1.3 Метод неопределенных коэффициентов

Аналогичные формулы можно получить и для случая произвольного расположения узлов. Использование многочлена Лагранжа в этом случае приводит к вычислению громоздких выражении, поэтому удобнее применять метод неопределенных коэффициентов. Он заключается в следующем. Искомое выражение для производной k-го порядка в некоторой точке $x=x_i$ представляется в виде линейной комбинации заданных

значений функции в узлах x_0, x_1, ..x_n:

$$y_i^{(k)} = c_0 y_0 + c_1 y_1 + \ldots + c_n y_n \qquad (7.5)$$

Предполагается, что эта формула имеет место для многочленов y=1, y=x-x_i, y=(x-x_i)^n. Подставляя последовательно эти выражения в (7.5), получаем систему n + 1 линейных алгебраических уравнений для определения неизвестных коэффициентов c_0, c_1 ..., c_n.

ПРИМЕР. Найти выражение для производной yx в случае четырех равноотстоящих узлов (n = 3).

Равенство ((7.5) запишется в виде

$$y_1' = c_0 y_0 + c_1 y_1 + c_2 y_2 + c_3 y_3 \qquad (7.6)$$

Используем следующие многочлены:

$$y = 1, y = x - x_0, y = (x - x_0)^2, y = (x - x_0)^3 \qquad (7.7)$$

Вычислим их производные:

$$y' = 0, y' = 1, y' = 2(x - x_0), y' = 3(x - x_0)^2 \qquad (7.8)$$

Подставляем последовательно соотношения (7.7) и (7.8) соответственно в правую и левую части (7.6) при x=x1:

$$0=c_0*1+c_1*1+c_2*1+c_3*1,$$
$$1=c_0(x_0-x_0)+c_1(x_1-x_0)+c_2(x_2-x_0)+c_3(x_3-x_0),$$
$$2(x_1-x_0)=c_0(x_0-x_0)^2+c_1(x_1-x_0)^2+c_2(x_2-x_0)^2+c_3(x_3-x_0)^2,$$
$$3(x_1-x_0)^2=c_0(x_0-x_0)^3+c_1(x_1-x_0)^3+c_2(x_2-x_0)^3+c_3(x_3-x_0)^3.$$

Получаем окончательно систему уравнений в виде

$$c_0+c_1+c_2+c_3=0$$
$$hc_1+2hc_2+3hc_3=1,$$
$$hc_1+4hc_2+9hc_3=2,$$
$$hc_1+8hc_2+27hc_3=3.$$

Решая эту систему получаем

$$c_0 = -\frac{1}{3h}, c_1 = -\frac{1}{2h}, c_2 = \frac{1}{h}, c_3 = -\frac{1}{6h}.$$

Подставляя эти значения в равенство((7.6), находим выражение для производной:

$$y_1' = \frac{1}{6h}(-2y_0 - 3y_1 + 6y_2 - y_3)$$

1.4 Частные производные

Рассмотрим функцию двух переменных u = f(x, y), заданную в табличном виде: u_{ij}=f(x,y),где $x_i=x_0+ih_i$ (i= 0, 1, …,I), $y_j=y_0+jh_2$ (j= 0, 1, J). В !!!Таблица 7.2 представлена часть данных, которые нам в дальнейшем понадобятся.

Используя понятие частной производной, можем приближенно записать для малых значений шагов h_1, h_2

$$\frac{\partial y}{\partial x} \approx \frac{f(x + h_1, y) - f(x, y)}{h1},$$
$$\frac{\partial u}{\partial y} \approx \frac{f(x, y + h_2) - f(x, y)}{h_2}$$

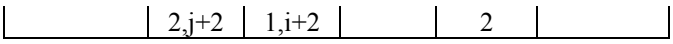
Воспользовавшись введенными выше обозначениями, получим следующие приближенные выражения (аппроксимации) для частных производных в узле (xi,yi)отношении конечных разностей:

$$\left(\frac{\partial u}{\partial x}\right) i_j \approx \frac{u_{i+1,j} - u_{ij}}{h_1}, \left(\frac{\partial u}{\partial x}\right)_{ij} \approx \frac{ui_{j+1} - u_{ij}}{h_2}.$$

Для численного дифференцирования функций многих переменных можно, как !!!а ранее, использовать интерполяционные многочлены. Однако рассмотрим здесь другой

!!!Таблица 7.2

x / y	xi-2	xi-1	Xi	xi+1	xi+2
Yj-2	Ui-2,j-2	Ui-1,i-2	Ui,j-2	Ui+1,j-2	Ui+2,j-2
Yj-1	Ui-2,j-1	Ui-1,i-1	Ui,j-1	Ui+1,j-1	Ui+2,j-1
Yj	Ui-2,j	Ui-1,i	Ui,j	Ui+1,j	Ui+2,j
Yj+1	Ui-2,j+1	Ui-1,i+1	Ui,j+2	Ui+1,j+1	Ui+2,j+1
Yj+2	Ui-	Ui-	Ui,j+2	Ui+1,j+	Ui+2,j+2



Способ разложение в ряд Тейлора функции двух переменных:

$$f(x+\Delta x, y+\Delta y) = f(x,y) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y + \\ + \frac{1}{2!}\left(\frac{\partial^2 f}{\partial x^2}\Delta x^2 + 2\frac{\partial^2 f}{\partial x \partial y}\Delta x \Delta y + \frac{\partial^2 f}{\partial y^2}\Delta y^2\right) + \\ + \frac{1}{3!}\left(\frac{\partial^3 f}{\partial x^3}\Delta x^3 + 3\frac{\partial^3 f}{\partial x^2 \partial y}\Delta x^2 \Delta y + 3\frac{\partial^3 f}{\partial x \partial y^2}\Delta x \Delta y^2 + \frac{\partial^3 f}{\partial y^3}\Delta y^3\right) + \ldots \quad (7.9)$$

Используем эту формулу дважды; 1) найдем $U_{i+1,j} = f(x_i+h_1, y_i)$ при $\Delta x = h_1$, $\Delta y = 0$; 2) найдем $U_{i-1,j} = f(x_i+h_1, y_i)$ при $\Delta x = -h_1$ $\Delta y = 0$. Получим

$$U_{i+1,j} = Ui_j + \left(\frac{\partial u}{\partial x}\right)_{ij} h_1 + \frac{1}{2!}\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij} h_1^2 + \frac{1}{3!}\left(\frac{\partial^3 u}{\partial x^3}\right)_{ij} h_1^3 + \ldots$$

$$U_{i-1,j} = Ui_j - \left(\frac{\partial u}{\partial x}\right)_{ij} h_1 + \frac{1}{2!}\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij} h_1^2 - \frac{1}{3!}\left(\frac{\partial^3 u}{\partial x^3}\right)_{ij} h_1^3 + \ldots$$

Вычитая почленно из первого равенства второе, получаем

$$U_{i+1,j} - U_{i-1,j} = 2h_1\left(\frac{\partial u}{\partial x}\right)_{ij} + O(h_1^3)$$

Отсюда найдем аппроксимацию производной с помощью центральных разностей:

$$\left(\frac{\partial u}{\partial x}\right)_{ij} \approx \frac{u_{i+1,j} - u_{i-1,j}}{2h_1} + O(h_1^2)$$

Она имеет второй порядок.

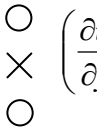
Аналогично могут быть получены аппроксимации производной $\frac{\partial u}{\partial y}$ а также старших производных. В частности, для второй производной можно, получить

$$\left(\frac{\partial^2 u}{\partial x^2}\right) = \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h_1^2} + O(h_1^2)$$

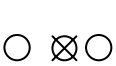
Записывая разложения в ряд (7.7) при разных значениях Δx и Δy, можно вывести формулы численного дифференцирования с необходимым порядком аппроксимации.

Приведем окончательные формулы для некоторых аппроксимаций частных производных. Слева указывается комбинация используемых узлов (шаблон), которые отмечены кружочками. Значения производных вычисляются в узле (xi,yi), отмеченном крестиком (напомним, что на шаблонах и в таблице 7.2 по горизонтали изменяются переменная x и индекс i, по вертикали — переменная y и индекс i):

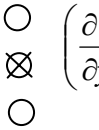
$$\left(\frac{\partial u}{\partial x}\right)_{ij} = \frac{u_{i+1.j} - u_{i-1,j}}{2h_1}$$



$$\left(\frac{\partial u}{\partial y}\right)_{ij} = \frac{u_{i+1,j} - u_{i-1,j}}{2h_2},$$



$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij} = \frac{u_{i+1.j} - 2u_{ij} + u_{i-1,j}}{2h_2},$$



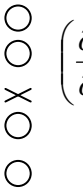
$$\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij} = \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{2h_2{}^2},$$

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{ij} = \frac{u_{i+1,j+1} - u_{i+1\,j-1} - u_{i-1\,j+2} + u_{i-1,j-1}}{4h_1 h_2},$$

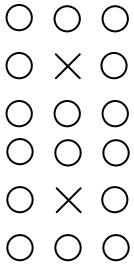
$$\left(\frac{\partial\, u}{\partial x}\right)_{ij} = \frac{u_{i+1,j+1} - u_{i+1\,j-1} - u_{i-1\,j+2} + u_{i-1,j-1}}{4h_1},$$

$$\left(\frac{\partial u}{\partial y}\right)_{ij} = \frac{u_{i+1,j+1} - u_{i+1\,j-1} - u_{i-1\,j+2} + u_{i-1,j-1}}{4h_2},$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij} = \frac{-u_{i+2,j} + 16u_{i+1,j} - 30u_{i\,j} + 16u_{i-1,j} - u_{i-2,j}}{12h_1^2},$$



$$\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij} = \frac{-u_{i+2,j} + 16u_{i+1,j} - 30u_{i\,j} + 16u_{i-1,j} - u_{i-2,j}}{12h_2^2}$$



$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij} = \frac{1}{3h_1^2}(u_{i+1,j+1} - 2u_{i,j+1} + u_{i+1\,j-1} + u_{i+1,j} - 2u_{ij} + u_{i-1\,j} + u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1})$$

$$\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij} = \frac{1}{3h_2^2}(u_{i+1,j+1} - 2u_{i,j+1} + u_{i+1\,j-1} + u_{i+1,j} - 2u_{ij} + u_{i-1\,j} + u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1})$$

Приведенные аппроксимации производных могут быть использованы при построении разностных схем для решения уравнений с частными производными.

§2. Интегрирование. Метод прямоугольников. Метод трапеций. Метод Симпсона. Метод сплайнов. Адаптивные алгоритмы. Кратные интегралы. Метод Монте-Карло

2.1 Интегрирование по частям

Для этого метода используется формула

$$\int u\ dv\ =\ u\,v\ -\int v\ du\,.$$

Где u и v – дифференцируемые функции от x.

Для применения этой формулы подынтегральное выражение следует представить в виде произведения одной функции на дифференциал другой функции. Если под интегралом стоит произведение логарифмической или обратной тригонометрической функции на алгебраическую, то за u обычно принимают не алгебраическую функцию. И наоборот.

ПРИМЕР:

$$\int x\cos x\,dx = \int x\,d(\sin x) = x\sin x - \int \sin x\,dx = x\sin x + \cos x + C$$

2.2 Метод разложения

Метод основан на разложении подынтегральной функции на сумму функций, каждая из которых является табличной.

2.3 Метод подстановки

Метод замены переменной заключается в том, что вводится новая переменная с помощью соотношения

$$x\ =\ \varphi(t)\,.$$

Тогда $dx\ =\ \varphi'(t)dt$ и исходный интеграл преобразуется следующим образом, где $\varphi(t)$ - дифференцируемая функция. Затем находится интеграл из правой части (если это возможно) и осуществляется возврат к исходной переменной x, используя соотношение $t\ =\ \phi(x)$, полученное из соотношения $x\ =\ \varphi(t)$, выражая t

через x. При интегрировании некоторых функций часто целесообразно осуществлять переход к новой переменной с помощью подстановки $t = \phi(x)$, а не $x = \varphi(t)$.

2.5 Методы прямоугольников

Итак, функция *y=f(x)* интегрируема на отрезке [a,b] и требуется вычислить ее интеграл $\int_a^b f(x)dx$. Составим интегральную сумму для *f(x)* на отрезке [a,b] . Для этого разобьем отрезок[a,b] на n равных между собой частей с помощью точек: x_1, x_2, ... , x_k, ... , x_n.

Если длину каждой части мы обозначим через Δx, так что $\Delta x = \frac{b-a}{n}$, то для каждой точки x_k будем иметь:

$$x_k = a + k\Delta x \text{ при } k = 0, 1, 2, \dots n.$$

Обозначим теперь через y_k значение подынтегральной функции $f(x)$ при $x = x_k = a + k\Delta x$, то есть положим $y_k = f(a + k\Delta x)$ (k = 0, 1, 2, …, n).

Тогда суммы $\sum_{k=1}^{n} y_{k-1}\Delta x$ и $\sum_{k=1}^{n} y_k \Delta x$ будут интегральными для функции *f(x)* на отрезке [a,b]. (При составлении первой суммы мы рассматриваем значения функции *y=f(x)* в точках, являющихся левыми концами частичных сегментов, а при составлении второй суммы – в точках, являющихся правыми концами этих сегментов.)

По определению интеграла имеем:

$$\int_a^b f(x)dx = \lim_{\Delta x \to 0} \sum_{k=1}^{n} y_{k-1}\Delta x$$

$$\int_a^b f(x)dx = \lim_{\Delta x \to 0} \sum_{k=1}^{n} y_k \Delta x$$

Поэтому в качестве приближенного значения $\int_a^b f(x)dx$ естественно взять интегральную сумму $\sum_{k=1}^{n} y_{k-1}\Delta x$ и $\sum_{k=1}^{n} y_k \Delta x$, то есть положить:

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} y_{k-1}\Delta x$$

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} y_k \Delta x$$

то есть:

$$\int_a^b f(x)dx \approx \frac{b-a}{n}(y_0 + y_1 + y_2 + \dots + y_{n-1}) \quad (7.10)$$

$$\int_a^b f(x)dx \approx \frac{b-a}{n}(y_1 + y_2 + y_3 + \dots + y_n) \quad (7.11)$$

Эти приближенные равенства называются формулами прямоугольников.

В том случае, когда *f(x)≥ 0*, формулы (7.10) и (7.11) с геометрической точки зрения означают, что площадь криволинейной трапеции *aABb*, ограниченной дугой кривой *y=f(x),* осью *Ox* и прямыми *x=a* и *x=b*, принимается приближенно равной площади ступенчатой фигуры, образованной из n прямоугольников с основаниями $\Delta x = \frac{b-a}{n}$ и высотами y_0, y_1, y_2, …, y_{n-1} – в случае формулы (7.10) (рисунок 7.1) и y_1, y_2, y_3, …, y_n - в случае формулы (7.11) рисунок 7.2.

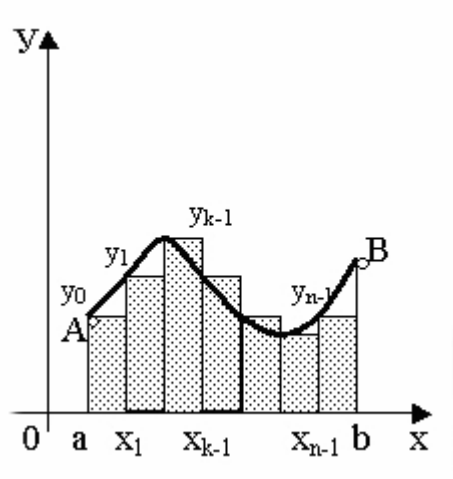


Рисунок 7.1

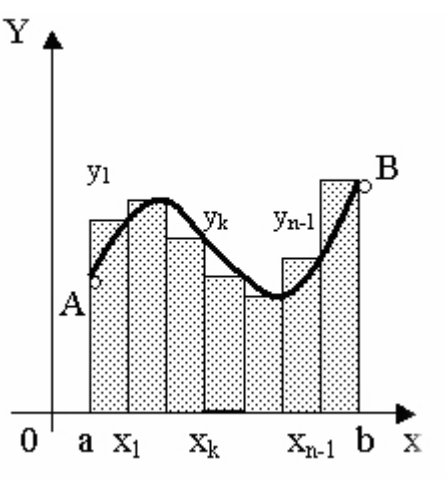


Рисунок 7.2

Исходя из приведенного выше геометрического смысла формул (7.10) и (7.11) способ приближенного вычисления определенного интеграла по этим формулам принято называть *методом прямоугольников*.

Всякое приближенное вычисление имеет определенную ценность лишь тогда, когда оно сопровождается оценкой допущенной при этом погрешности. Поэтому формулы прямоугольников будут практически пригодны для приближенного вычисления интегралов лишь в том случае, если будет существовать удобный способ оценки получающейся при этом погрешности (при заданном n), позволяющий к тому же находить и число частей n разбиения сегмента, гарантирующее требуемую степень точности приближенного вычисления.

Будем предполагать, что функция *f(x)* имеет ограниченную производную на сегменте [a, b], так что существует такое число *M>0*, что для всех значений x из [a, b] выполняется неравенство $|f'(x)| \le M$. Качественный смысл этого неравенства заключается в том, что скорость изменения значения функции ограничена. В реальных природных системах это требование практически всегда выполнено. В этих условиях абсолютная величина погрешности R_n будет стремиться к нулю, т.е. точность приближения будет тем больше, чем на большее число равных частей будет разделен сегмент [a, b]. Абсолютная погрешность результата будет заведомо меньше заданного числа $\varepsilon > 0$, если взять

$$n > \frac{M(b-a)^2}{2\varepsilon} \tag{7.12}$$

Следовательно, для вычисления интеграла $\int_a^b f(x)dx$ с указанной степенью точности достаточно сегмент [a,b] разбить на число частей, большее числа $n > \frac{M(b-a)^2}{2\varepsilon}$.

Метод прямоугольников – это наиболее простой и вместе с тем наиболее грубый метод приближенного интегрирования. Заметно меньшую погрешность дает другой метод – метод трапеций.

2.6 Метод трапеций

Очевидно, что чем больше будет число n отрезков разбиения, тем более точный результат дадут формулы (7.10) и (7.11). Однако увеличение числа отрезков разбиения промежутка интегрирования не всегда возможно. Поэтому большой интерес представляют формулы, дающие более точные результаты при том же числе точек разбиения.

Простейшая из таких формул получается как среднее арифметическое правых частей формул (7.10) и (7.11):

$$\int_a^b f(x)dx = \frac{b-a}{n} \cdot \frac{\sum_{k=0}^{n-1} y_k + \sum_{k=0}^{n-1} y_{k+1}}{2} = \frac{b-a}{n} \cdot \sum_{k=0}^{n-1} \frac{y_k + y_{k+1}}{2} \tag{7.13}$$

Легко усмотреть геометрический смысл этой формулы. Если на каждом отрезке разбиения дугу графика подинтегральной функции y=f(x) заменить стягивающей ее хордой (линейная интерполяция), то мы получим трапецию, площадь которой равна $\frac{b-a}{n} \cdot \frac{y_k + y_{k+1}}{2}$ и следовательно, формула (7.13) представляет собой площадь фигуры, состоящей из таких трапеций (рисунок 7.3) . Из геометрических соображений понятно, что площадь такой фигуры будет, вообще говоря, более точно выражать площадь криволинейной трапеции, нежели площадь ступенчатой фигуры, рассматриваемой в методе прямоугольников.

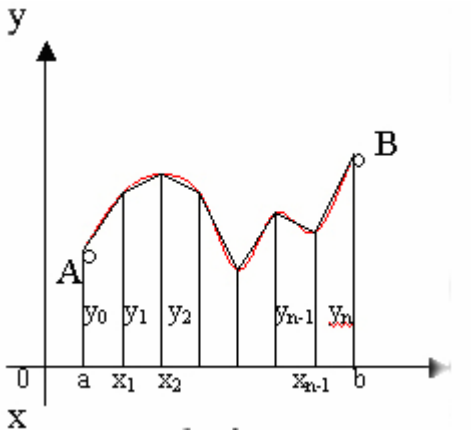


Рисунок 7.3

Приведя в формуле (7.1) подобные члены, окончательно получим:

$$\int_a^b f(x)dx \approx \frac{b-a}{n}\cdot\left(\frac{y_0}{2}+y_1+y_2+\ldots+y_{n-1}+\frac{y_n}{2}\right) \qquad (7.14)$$

Формулу (7.14) называют *формулой трапеций*.

Формулой трапеций часто пользуются для практических вычислений. Что касается оценки погрешности R_n, возникающей при замене левой части (7.14) правой, то доказывается, что абсолютная величина ее удовлетворяет неравенству:

$$|R_n| \le \frac{(b-a)^3}{12n^2}\cdot M_2 \qquad (7.15)$$

где M_2 - максимум модуля второй производной подинтегральной функции на отрезке [a,b], т.е.

$$M_2 = \max_{x\in[a,b]}|f''(x)|$$

Шаг численного интегрирования определяется из следующего неравенства:

$$h \le \sqrt{\frac{12\cdot\varepsilon}{(b-a)\cdot M_2}}$$

Следовательно, R_n убывает при $n\to\infty$. Абсолютная погрешность R_n будет меньше наперед заданного числа $\varepsilon > 0$ если взять $n > \sqrt{\frac{M(b-a)}{12\varepsilon}}$.

2.7 Метод парабол (метод Симпсона)

Значительное повышение точности приближенных формул может быть достигнуто за счет повышения порядка интерполяции. Одним из таких методов приближенного интегрирования является метод парабол. Идея метода исходит из того, что на частичном промежутке дуга некоторой параболы в общем случае теснее прилегает к кривой *y=f(x),* чем хорда, соединяющая концы дуги этой кривой, и поэтому значения площадей соответствующих элементарных трапеций,
дограниченных «сверху» угами парабол, являются более близкими к значениям площадей
соответствующих частичных криволинейных трапеций, ограниченных сверху дугой кривой *y=f(x),* чем значения площадей соответствующих прямолинейных трапеций. Сущность метода заключается в следующем. Отрезок [a,b] делится на *2n* равных частей. Пусть точки деления будут

$$x_0 = a, x_1, x_2, \ldots x_{2n-2}, x_{2n-1}, x_{2n} = b$$

а $y_0, y_1, \ldots y_{2n}$ – соответствующие значения подинтегральной функции на отрезке [a,b]. Произведем квадратичную интерполяцию данной

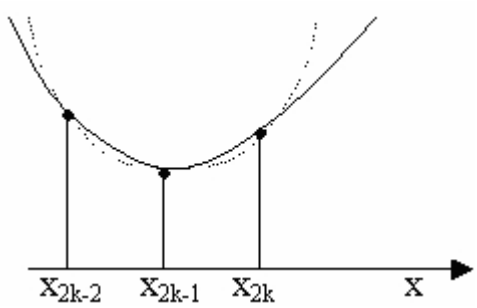


Рисунок 7.4

подинтегральной функции на каждом из отрезков разбиения (заменим дугу графика подинтегральной функции дугой параболы с вертикальной осью) (Рисунок 7.4)

Приведём без вывода формулу парабол в окончательном виде:

$$\int_a^b f(x)dx \approx \frac{h}{3}\cdot\left(y_0 + y_n + 4\sum_{i=1}^{m} y_{2i-1} + 2\sum_{i=1}^{m-1} y_{2i}\right) \tag{7.16}$$

Если подинтегральная функция *f(x)* имеет на отрезке [a,b] непрерывную четвертую производную, то для поправочного члена формулы (7.16) имеет место оценка:

$$|R_n| \le \frac{(b-a)^5}{180(2n)^4}\cdot M_4 \tag{7.17}$$

где M_4 - максимум модуля четвертой производной подинтегральной функции на отрезке [a,b].

Из формулы (7.17) при заданной точности ε получаем шаг:

$$h \le \sqrt[4]{\frac{180\cdot\varepsilon}{(b-a)\cdot M_4}} \tag{7.18}$$

где $M_4 = \max_{x\in[a;b]}\left|f^{IV}(x)\right|$

Сравнивая между собой оценки (7.15) и (7.17), замечаем, что с увеличением n поправочный член формулы трапеций уменьшается пропорционально величине $\frac{1}{n^2}$, а для формулы парабол – пропорционально величине $\frac{1}{n^4}$, т.е. метод парабол сходится значительно быстрее метода трапеций, тогда как с точки зрения техники вычислений оба метода одинаковы.

2.8 Адаптивные алгоритмы.

Из анализа погрешностей методов численного интегрирования следует, что точность получаемых результатов зависит как от характера изменения подынтегральной функции, так и от шага интегрирования. Будем считать, что величину шага мы задаем. При этом ясно, что для достижения сравнимой точности при интегрировании слабо меняющейся функции шаг можно выбирать большим, чем при интегрировании резко меняющихся функций.

На практике нередко встречаются случаи, когда подынтегральная функция меняется по-разному на отдельных участках отрезка интегрирования. Это обстоятельство требует такой организации экономичных численных алгоритмов, при которой они автоматически приспосабливались бы к характеру изменения функции. Такие алгоритмы называются *адаптивными (приспосабливающимися).* Они позволяют вводить разные значения шага интегрирования на отдельных участках отрезка интегрирования. Это дает возможность уменьшить машинное время без потери точности результатов расчета. Подчеркнем, что этот подход используется обычно при задании подынтегральной функция *y = f(x)* в виде формулы, а не в табличном виде.

Программа, реализующая адаптивный алгоритм численного интегрирования, входит обычно в виде стандартной подпрограммы в математическое обеспечение ЭВМ. Пользователь готовой программы задает границы отрезка интегрирования *a, Ь,* допустимую абсолютную погрешность ε и составляет блок программы для вычисления значения подынтегральной функции f(x). Программа вычисляет значение интеграла I с заданной погрешностью ε, т. е.

$$\left|I - \int_a^b f(x)dx\right| \le \varepsilon \tag{7.19}$$

Разумеется, не для всякой функции можно получить результат с заданной погрешностью. Поэтому в программе может быть предусмотрено сообщение пользователю о недостижимости заданной погрешности. Интеграл при этом вычисляется с максимально возможной точностью, л программа выдает эту реальную

точность.

Рассмотрим принцип работы адаптивного алгоритма. Первоначально отрезок [a, b] разбиваем на *n* частей. В дальнейшем каждый такой элементарный отрезок делим последовательно пополам. Окончательное число шагов, их расположение и размеры зависят от подынтегральной функции и допустимой погрешности ε.

К каждому элементарному отрезку $[x_{i-1}, x_i]$ применяем формулы численного интегрирования при двух различных его разбиениях. Получаем приближения $I_i^{(1)}, I_1^{(2)}$ для интеграла по этому отрезку:

$$I = \int_{xi-1}^{xi} f(x)dx. \tag{7.20}$$

Полученные значения сравниваем и проводим оценку их погрешности. Если погрешность находится в допустимых границах, то одно из этих приближений принимается за значение интеграла по этому элементарному отрезку. В противном случае происходит дальнейшее деление отрезка и вычисление новых приближений. С целью экономии машинного времени точки деления располагаются таким образом, чтобы использовались вычисленные значения функции в точках предыдущего разбиения.

Например, при вычислении интеграла (7.20) по формуле Симпсона отрезок $[x_{i-1}, x_i]$ сначала разбиваем на две части с шагом $h_i/2$ и вычисляем значение $I_i^{(1)}$.

Потом вычисляем $I_i^{(2)}$ с шагом $h_i/4$. Получим выражения:

$$I_i^{(1)} = \frac{h_i}{6}[f(x_{i-1}) + 4f(x_{i-1} + \frac{h_i}{2}) + f(x_i)] \tag{7.21}$$

$$I_i^{(2)} = \frac{h_i}{12}[f(x_{i-1}) + 4f(x_{i-1} + \frac{h_i}{4}) + 2f(x_{i-1} + \frac{h_i}{2}) + 4f(xi + \frac{3h_i}{4}) + f(x_i)] \tag{7.22}$$

Формулу (7.22) можно также получить двукратным применением формулы (7.21) для отрезков $[x_{i-1}, x_{i-1} + h_i/2] u [x_i + h_i/2, x_i]$

Процесс деления отрезка пополам о вычисления уточненных значений $I_i^{(1)}, I_1^{(2)}$ продолжается до тех пор, пока их разность станет не больше некоторой заданной величины δ_i, зависящей от ε и h:

$$\left| I_i^{(1)} - I_1^{(2)} \right| \le \delta_i \tag{7.23}$$

Аналогичная процедура проводится для всех *n* элементарных отрезков. Величина $I = \sum_{i=1}^{n} I_i$ принимается в качестве искомого значения интеграла. Условия (7.23) и соответствующий выбор величин δ_i обеспечивают выполнение условия (7.19).

2.9 Кратные интегралы.

Численные методы используются также для вычисления кратных интегралов. Ограничимся здесь рассмотрением *двойных интегралов* вида

$$I = \iint_G f(x, y)dxdy \tag{7.24}$$

Одним из простейших способов вычисления этого интеграла является *метод ячеек.* Рассмотрим сначала случай, когда областью интегрирования *G* является прямоугольник: $a \le x \le b, c \le y \le d$. По теореме о среднем найдем среднее значение функции f(x,y)*:*

$$\overline{f}(x, y) = \frac{1}{S}\iint_G f(x, y)dxdy, S = (b - a)(d - c). \tag{7.25}$$

Будем считать, что среднее значение приближенно равно значению функции в центре прямоугольника, т. Е $\overline{f}(x, y) = f(\overline{x}, \overline{y})$, Тогда из (7.25) получим выражение для приближенного вычисления двойного интеграла:

$$\iint_G f(x, y)dxdy \approx Sf(\overline{x}.\overline{y}) \tag{7.26}$$

$$\overline{x} = (a + b)/2, \overline{y} = (c + d)/2$$

Точность этой формулы можно повысить, если разбить область *G* на прямоугольные ячейки ΔG_{ij}:

$$x_{i-1} \le x \le x_i, (i = 1,2,..., M), y_{j-1} \le y \le y_j, (j = 1,2,..., N).$$

Применяя к каждой ячейке формулу7.26, получаем

$$\iint_{\Delta Gij} f(x,y)dxdy \approx f(\overline{x_i},\overline{y_j})\Delta x_i \Delta y_j$$

Суммируя эти выражения по всем ячейкам, находим значение двойного интеграла:

$$\iint_G f(x,y) = \sum_{i=1}^{M}\sum_{j=1}^{N} f(\overline{x_i},\overline{y_j})\Delta x_i \Delta y_j \qquad (7.27)$$

В правой части стоит интегральная сумма; поэтому при неограниченном уменьшении периметров ячеек (или стягивании их в точки) эта сумма стремится к значению интеграла для любой непрерывной функции *f(x, y)*.

Можно показать, что погрешность такого приближения интеграла для одной ячейки оценивается соотношением

$$R_{ij} \approx \frac{\Delta x_i \Delta y_j}{24}\left[(\frac{b-a}{M})^2 f''XX + (\frac{d-c}{N})^2 f''yy\right]$$

Суммируя эти выражения по всем ячейкам и считая все их площади одинаковыми, получаем оценку погрешности метода ячеек в виде

$$R \approx O(1/M^2 + 1/N^2) \approx O(\Delta x^2 + \Delta y^2)$$

Таким образом, формула 7.27 имеет второй порядок точности. Для повышения точности можно использовать обычные *методы сгущения узлов* сетки. При этом по каждой переменной шаги уменьшают в одинаковое число раз, т. е. отношение *M/N* остается постоянным.

Если область *G* непрямоугольная, то в ряде случаев ее целесообразно привести к прямоугольному виду путем соответствующей *замены переменных.* Например, пусть область задана в виде криволинейного четырехугольника: $a \le x \le b, \varphi_1(x) \le y \le \varphi_2(x)$. Данную область можно привести к прямоугольному виду с помощью замены

$$t = \frac{y-\varphi_1(x)}{\varphi_2(x)-\varphi_1(x)}, 0 \le t \le 1.$$

Кроме того, формула (7.27) может быть обобщена и на случай более сложных областей.

Другим довольно распространенным методом вычисления кратных интегралов является их *сведение к последовательному вычислению определенных интегралов.*

Интеграл (7.24) для прямоугольной области можно записать в виде

$$\iint_G f(x,y)dxdy = \int_a^b F(x)dx, F(x) = \int_a^b f(x,y)dy$$

Для вычисления обоих определенных интегралов могут быть использованы рассмотренные ранее численные методы.

Если область *G* имеет более сложную структуру, то она либо приводится к прямоугольному виду с помощью замены переменных, либо разбивается на простые элементы.

Для вычисления кратных интегралов используется также *метод замены подынтегральной функции многомерным интерполяционным многочленом.* Вычисление коэффициентов этих многочленов для простых областей обычно не вызывает затруднений.

Существует ряд других численных методов вычисления кратных интегралов. Среди них особое место занимает метод статистических испытаний, который мы вкратце изложим.

2.10 Метод Монте-Карло.

Во многих задачах исходные данные носят случайный характер, поэтому для их решения должен применяться статистико-вероятностный подход. На основе таких подходов построен ряд численных методов, которые учитывают случайный характер вычисляемых или измеряемых величин. К ним принадлежит и *метод статистических испытаний,* называемый также *методом Монте-Карло,* который применяется к решению некоторых задач вычислительной математики, в том числе и для вычисления интегралов.

Метод Монте-Карло состоит в том, что рассматривается некоторая случайная величина ξ, математическое ожидание которой равно искомой величине *x:*

$$M\xi = x.$$

Проводится серия *n* независимых испытаний, в результате которых получается (генерируется) последовательность *n* случайных чисел $\xi_1, \xi_2, \ldots, \xi_n$ и по совокупности этих значений приближенно определяется искомая величина

$$\xi = (\xi_1 + \xi_2 + \ldots + \xi_n)/n \approx x,$$

$$M\xi = M\left(\frac{1}{n}\sum_{i=1}^{n}\xi_i\right) = \frac{1}{n}M\sum_{i=1}^{n}\xi_i = \frac{nx}{n} = x,$$

Пусть η - равномерно распределенная на отрезке [0,1] случайная величина. Это означает, что ее плотность распределения задается соотношением

$$P\eta(x) = \begin{cases} 0, x < 0, \\ 1, 0 \le x \le 1, \\ 0, x > 1. \end{cases}$$

Тогда любая функция $\xi = f(\eta)$ также будет случайной величиной, и ее математическое ожидание равно

$$M\xi = \int_{-\infty}^{\infty} f(x) P\eta(x) dx = \int_0^1 f(x) dx.$$

Следовательно, читая это равенство в обратном порядке, приходим к выводу, что интеграл $\int_0^1 f(x)dx$ может быть вычислен как математическое ожидание некоторой случайной величины ξ, которая определяется независимыми реализациями η, случайной величины η с равномерным законом распределения:

$$\int_0^1 f(x,y)dxdy \approx \xi = \frac{1}{n}\sum_{i=1}^{n} f(\eta_i)$$

Аналогично могут быть вычислены и кратные интегралы. Для двойного интеграла получим

$$\iint_G f(x,y)dxdy \approx \frac{1}{n}\sum_{i=1}^{n} f(\eta_i, \xi_i)$$

где G: $0 \le x \le 1, 0 \le y \le 1$; η_i, ξ_i — независимые реализации случайных величин η, ξ равномерно распределенных на отрезке [0, 1].

Для использования метода Монте-Карло при вычислении определенных интегралов, как и в других его приложениях, необходимо вырабатывать последовательности случайных чисел с заданным законом распределения. Существуют различные способы генерирования -таких чисел.

Можно построить некоторый физический процесс (генератор) для -выработки случайных величин, однако при использовании ЭВМ этот способ непригоден, поскольку трудно дважды получить одинаковые совокупности случайных чисел, которые необходимы при отладке программ.

Известны многие таблицы случайных чисел, которые вычислялись независимо. Их можно вводить в ЭВМ, хранить в виде файла на магнитной ленте или магнитном диске коллективного пользования. А еще лучше заготовить собственный файл случайных чисел.

В настоящее время наиболее распространенный способ выработки случайных чисел на ЭВМ состоит в том, что в памяти хранится некоторый алгоритм выработка таких чисел по мере потребности в них (подобно тому как вычисляются значения элементарных функций, а не хранятся их таблицы). Поскольку эти числа генерируются по наперед заданному алгоритму, то они не совсем случайны *(псевдослучайны),* хотя и обладают свойственными случайным числам статистическими характеристиками.

ПРИМЕР:

Задача. Методом трапеций с точностью $\varepsilon = 10^{-2}$ и Симпсона с точностью $\varepsilon = 10^{-4}$ вычислить определённый интеграл:

$$\int_0^1 \frac{dx}{1+x}$$

Замечание. Данный в примере интеграл вычисляется точно:

$$\int_0^1 \frac{dx}{1+x} = \ln(1+x)\Big|_0^1 = \ln 2 \approx 0{,}69315$$

Примеры, предлагаемые для самостоятельного решения, не вычисляются точно.

Решение.

1) Метод трапеций.

Исходя из заданной точности $\varepsilon = 10^{-2}$, вычислим шаг численного интегрирования: $h \le \sqrt{\frac{12 \cdot \varepsilon}{(b-a) \cdot M_2}}$

$$M_2 = \max_{x \in [0;1]} |f''(x)| = \max_{x \in [0;1]} \left| \frac{2}{(1+x)^3} \right| = 2$$

$$h \le \sqrt{\frac{12 \cdot 10^{-2}}{(1-0) \cdot 2}} = 0{,}2449$$

Необходимо выбрать такой шаг, который удовлетворяет неравенству $h < 0{,}2449$ и чтобы на отрезке интегрирования $x \in [0;1]$ он укладывался целое число раз. Принимаем $h = 0{,}2$. Он удовлетворяет обоим этим требованиям.

Для подынтегральной функции $f(x) = (1+x)^{-1}$ с независимой переменной x_i изменяющейся в соответствии с равенством $x_i = x_0 + ih = 0 + i \cdot 0{,}2 \quad i = 0{,}5$ составляем сеточную функцию с точностью до второго знака после запятой:

i	0	1	2	3	4	5
x_i	0	0,2	0,4	0,6	0,8	1,0
y_i	1,0	0,83	0,71	0,63	0,56	0,5

Используется формула трапеций численного интегрирования (n=5), получим:

$$\int_0^1 \frac{dx}{1+x} \approx \frac{h}{2}\left(y_0 + y_n + 2\sum_{i=1}^{n-1} y_i \right) = \frac{0{,}2}{2}\left[1{,}0 + 0{,}5 + (0{,}83 + 0{,}71 + 0{,}63 + 0{,}56)\right] = 0{,}696$$

Сравнивая это значение с точным, видим, что абсолютная погрешность не превышает заданной точности ε:

$$\left|0{,}69315 - 0{,}696\right| < 0{,}01$$

2) Метод Симпсона

Исходя из заданной точности $\varepsilon = 10^{-4}$ вычисляется шаг численного интегрирования для метода Симпсона:

$$M_4 = \max_{x \in [0:1]} \left| \frac{24}{(1+x)^5} \right| = 24$$

$$h \le \sqrt[4]{\frac{180 \cdot 10^{-4}}{(1-0) \cdot 24}} = 0{,}165$$

Необходимо выбрать такой шаг, чтобы он удовлетворял неравенству $h \le 0{,}165$ и чтобы на отрезке интегрирования $x \in [0;1]$ он укладывался четное число раз. Принимаем $h = 0{,}1$. С этим шагом для подынтегральной функции $f(x) = (1+x)^{-1}$ формируется сеточная функция с независимой переменной x_i изменяющейся по закону $x_i = x_0 + ih = 0 + i \cdot 0{,}1$, $i = \overline{0,1}$, $n = 10$, $m = 5$, причём значения сеточной функции вычисляются с точностью до четвертого знака после запятой:

i	0	1	2	3	4	5	6	7	8	9	10
x_i	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
y_i	1,0	0,9091	0,833	0,7692	0,7143	0,6667	0,6250	0,5882	0,5556	0,5263	0,5000

Далее используем формулу Симпсона для численного интегрирования для данного интеграла. Результат получим следующий:

$$\int_0^1 \frac{dx}{1+x} = \frac{0{,}1}{3} \cdot \left[1{,}0 + 0{,}5 + 4(y_1 + y_3 + y_5 + y_7 + y_9) + 2(y_2 + y_4 + y_6 + y_8)\right] = 0{,}69314$$

Сравнение этого значения с точным значением интеграла показывает, что абсолютная погрешность не превышает заданной точности ε:

$$\left|0{,}69315 - 0{,}69314\right| < 0{,}0001 = 10^{-4}$$

Таким образом, за приближенное значение определенного интеграла по методу Симпсона с точностью $\varepsilon = 10^{-4}$ принимается значение:

$$\int_0^1 \frac{dx}{1+x} \approx 0{,}6931$$

Замечание. Ясно, что для большинства интегралов от непрерывных функций первообразная не вычисляется (однако она существует) и вычисленное приближенное значение сравнивать не с чем, однако шаг численного интегрирования, вычисленный по заданной точности, гарантирует эту точность вычисления.

Глава 8
ОБЫКНОВЕННЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ

§1 Методы решения обыкновенных дифференциальных уравнений. Разностные методы.

Обычно в теории разностных схем для компактности записи дифференциальные уравнения, начальные и граничные условия представляются в некотором символическом виде, называемом *операторным.* Например, любое из уравнений

$$Y' = f(x), \quad Y'' = f(x), \quad Y'' + k^2 Y = f(x)$$

можно записать в виде $LY = F(x)$. Здесь $L-$ дифференциальный оператор, содержащий операции дифференцирования; его значение различно для разных дифференциальных уравнений. Область изменения аргумента x можно обозначить через G, т. е. $x \in G$. В частности, областью G при решении обыкновенных дифференциальных уравнений может быть некоторый отрезок $[a,b]$, полуось $x>0$ (или $t>0$) и т. п.

Дополнительные условия на границе также представляются в операторном виде. Например, любое из условий

$$Y(0)=A, \quad Y(a)=0, \quad Y(b)=1, \quad Y'(0)=B, \quad Y'(a)=1$$

Можно записать в виде $lY = \Phi(x) \quad (x \in \Gamma)$. Здесь $l-$ оператор начальных или граничных условий, $\Phi(x)-$правая часть этих условий, $\Gamma-$ граница рассматриваемой области (т.е. точки $x=0, x=a, x=b$ и т.п.).

Таким образом, исходную задачу для дифференциального уравнения с заданными начальными и граничными условиями, называемую в дальнейшем *дифференциальной задачей,* можно в общем случае записать в виде

$$LY = F(x), \qquad x \in G \tag{8.1}$$

$$lY = \Phi(x), \qquad x \in \Gamma \tag{8.2}$$

В методе конечных разностей исходное дифференциальное уравнение (8.1) заменяется разностным уравнением путем аппроксимации производных соответствующими конечно-разностными соотношениями. При этом в области G введем сетку, *шаг* $h>0$ которой для простоты будем считать постоянным. Совокупность узлов $x_0, x_1, \ldots$ обозначим через g_h. Значения искомой функции Y в узлах сетки заменяются значениями сеточной функции y_h, которая является решением разностного уравнения.

Искомую функцию и сеточную функцию будем обозначать соответственно Y и y, чтобы подчеркнуть их различие: $Y-$функция непрерывно меняющегося аргумента x, а $y-$ дискретная сеточная функция, определенная на дискретном множестве $g_h = \{x_i\} (i=0,1\ldots)$. Сеточную функцию, принимающую значения y_i в узлах сетки, можно считать функцией целочисленного аргумента i. Итак, дифференциальное уравнение 8.1 заменяется разностным уравнением, которое также можно записать в операторном виде:

$$L_{hyh} = f_h, x \in g_h \tag{8.3}$$

Здесь L_h- разностный оператор, аппроксимирующий дифференциальный оператор L. Как известно, погрешность аппроксимации производных, а следовательно, и погрешность аппроксимации (8.3) в некоторой точке x может быть представлена в виде $\varepsilon(x) = O(h^k)$. При этом говорят, что в данной точке x имеет место *аппроксимация k-го порядка.* Индекс h в разностном уравнении (8.3) подчеркивает, что величина шага является параметром разностной задачи. Поэтому (8.3) можно рассматривать, как целое семейство разностных уравнений, которые зависит от параметра h.

При решении дифференциальных уравнений обычно требуется оценить погрешность аппроксимации не в одной точке, а на всей сетке g_h, т.е. в точках $x_0, x_1, \ldots$ В качестве погрешности аппроксимации ε_h на сетке можно принять некоторую величину, связанную с погрешностями аппроксимации в узлах например,

$$\varepsilon_h = \max \left| \varepsilon(x_i) \right|, \qquad \varepsilon_h = \left[\sum_i \varepsilon^2 \right]^{1/2}.$$

В этом случае L_h имеет k-й порядок аппроксимации на сетке, если $\varepsilon_h = O\left(h^k\right)$.

Наряду с аппроксимацией (8.3) дифференциального уравнения (8.1) необходимо также аппроксимировать дополнительные условия на границе (8.2). Эти условия запишутся в виде

$$l_h y_h = \varphi_h, \qquad x \in \gamma_h \tag{8.4}$$

Здесь γ_h – множество граничных узлов сетки, т.е. $\gamma_h \subset \Gamma$. Индекс h, как и в (8.3), означает зависимость разностных условий на границе от значения шага.

Совокупность разностных уравнений ((8.3),((8.4), аппроксимирующих исходное дифференциальное уравнение и дополнительные условия на границе, называется *разностной схемой.*

ПРИМЕР. Рассмотрим задачу Коши

$$LY = \frac{dY}{dx} = F\left(x\right), \quad x > x_0, \quad Y(x_0) = A.$$

Введем равномерную сетку с шагом h, приняв в качестве узлов значения аргумента $x_0, x_1, \ldots$ Значения сеточной функции, которая аппроксимирует искомое решение в данных узлах, обозначим через $y_0, y_1 \ldots$ Тогда разностную схему можно записать, например, в виде

$$L_h y_h = \frac{y_{i+1} - y_i}{h} = f_i, \quad i = 0,1,..., \quad y_0 = A.$$

Здесь f_i – значение правой части разностного уравнения в точке x_i. Можно, в частности, принять $f_i = F\left(x_i\right)$. Данная схема имеет первый порядок аппроксимации, т.е. $\varepsilon_h = O\left(h\right)$.

Решение разностной задачи, в результате которого находятся значения сеточной функции y_i в узлах x_i, приближенно заменяет решение $Y\left(x\right)$ исходной дифференциальной задачи. Однако не всякая разностная схема дает удовлетворительное решение, т.е. получаемые значения сеточной функции y_i не всегда с достаточной точностью аппроксимируют значения искомой функции $Y\left(x_i\right)$ в узлах сетки. Здесь важную роль играют такие понятия, как устойчивость, аппроксимация и сходимость разностной схемы.

Под *устойчивостью* схемы понимается непрерывная зависимость ее решения от входных данных (коэффициентов уравнений, правых частей, начальных и граничных условий). Или, другими словами, малому изменению входных данных соответствует малое изменение решения. В противном случае разностная схема называется *неустойчивой.* Естественно, что для практических расчетов используются устойчивые схемы, поскольку входные данные обычно содержат погрешности, которые в случае неустойчивых схем приводят к неверному решению. Кроме того, в расчетах на компьютере погрешности возникают в процессе счета из-за округлений, а использование неустойчивых разностных схем приводит к недопустимому накоплению этих погрешностей.

Разностная схема называется *корректной,* если ее решение существует и единственно при любых входных данных, а также, если эта схема устойчива.

При использовании метода конечных разностей необходимо знать, с какой точностью решение разностной задачи приближает решение исходной дифференциальной задачи. Рассмотрим погрешность δ_i, равную разности значений искомой функции в узлах сетки и сеточной функции, т.е. $\delta_h = Y_h - y_h$. Отсюда найдем $y_h = Y_h - \delta_h$. Подставляя это значение y_h в разностную схему (8.3), (8.4), получаем

$$L_h Y_h - L_h \delta_h = f_h, \quad x \in g_h,$$
$$l_h Y_h - l_h \delta_h = \varphi_h, \quad x \in \gamma_h.$$

Отсюда

$$L_h \delta_h = R_h, \qquad l_h \delta_h = r_h.$$

Здесь $\mathrm{R}_h = \mathrm{L}_h Y_h - f_h -$ погрешность аппроксимации (*невязка*) для разностного уравнения, а $\mathrm{r}_h = l_h Y_h - \varphi_h -$ погрешность аппроксимации для разностного граничного условия.

Если ввести характерные значения R и r невязок R_h и r_h (например, взять их максимальные по модулю значения на сетке), то при $R = O(h^k)$ и $r = O(h^k)$ говорят, что разностная схема 8.7!!!!, 8.8 !!! *имеет* ***k*****-ый** *порядок аппроксимации* на решении.

Введем аналогичным образом характерное значение δ погрешности решения δ_h. Тогда разностная схема сходится, если $\delta \to 0$ при $h \to 0$. Если при этом $\delta = O(h^k)$, то говорят, что разностная схема имеет точность k *-го порядка* при *сходятся со скоростью* $O(h^k)$.

В теории разностных схем доказывается, что если разностная схема устойчива и аппроксимирует исходную дифференциальную задачу, то она сходится. Иными словами*, из устойчивости и аппроксимации разностной схемы следует ее сходимость.* Это позволяет свести трудную задачу изучения сходимости и оценки порядка точности разностной схемы к изучению погрешности аппроксимации и устойчивости, что значительно легче.

§2. Задача Коши. Одношаговые методы – метод Эйлера, усовершенствованный метод Эйлера, метод Рунге-Кутта. Многошаговые методы - метод Адамса, метод Милна.

2.1 Метод ломаных Эйлера. Метод ломаных применяется для приближенного нахождения значений $x_i=\varphi(t_i)$ решения φ на некоторой сетке значений аргумента t: t_0, $t_1 = t_0 + \tau$, $t_2 = t_1 + \tau$, φ, $t_n = t_{n-1} + \tau$; τ — заданное положительное число, называемое шагом сетки. В применении к уравнению этот метод заключается в следующем. В точке $t = t_{i-1}$ уравнение принимает вид

$$x'_{i-1} = f(t_{i-1}, x_{i-1}).$$

Заменим здесь приближенно x'_{i-1} конечно-разностным отношением

$$\frac{x_i - x_{i-1}}{T} \approx f(t_{i-1}, x_{i-1})$$

и выразим x_i:

$$x_i \approx x_{i-1} + \tau f(t_{i-1}, x_{i-1}).$$

Если $x_0 = \varphi(t_0)$ задать произвольно, то полученная рекуррентная формула позволяет приближенно найти значения x_1, x_2, ...

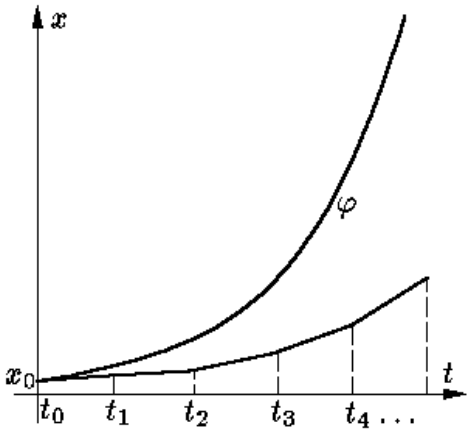


Рисунок 8.1

Возвращаясь к образу парка со стрелками-указателями, метод ломаных Эйлера можно представлять себе так (Рисунок 8.1): из точки (t_0, x_0) мы движемся, сообразуясь с указателем, помещенным в этой точке в течение "τ секунд". Придя (через время τ) в точку (t_1, x_1), мы меняем направление, пользуясь указателем в этой точке; через время τ мы приходим в точку (t_2, x_2), опять меняем направление, и т. д.

2.2 Модификации метода Эйлера.

При численном интегрировании дифференциального уравнения первого порядка

$$y' = F(x,y)$$

с начальным условием $y(x_0) = y_0$ (задача Коши) сначала выбираем фиксированное приращение аргумента $h=(x_f - x_0)/n$, где x_f - конечная точка интервала интегрирования, n - число шагов. Затем, применяя процедуру модифицированного метода Эйлера, вычисляем y_k по рекурентной формуле:

$$y_k = y_{k-1} + h[F_{k-1} + F(x_k, y_{k-1} + hF_{k-1})]/2$$

где $F_k = F(x_k, y_k)$.

Можно воспользоваться другой рекурентной формулой:

$y_k = y_{k-1} + F(x_{k-1} + h/2, y_{k-1} + F_{k-1} h/2)$

т.е. с начальным условием $y(x_0) = y_0$ составляют таблицу значений $y_k = y(x_k)$, где $x_k = x_0 + kh$ (k = 0, 1, 2…n), h = (b-a)/n, [a, b] – отрезок, на котором ищется решение. Значения y_{k+1} определяются по формуле

$$y_{k+1} = y_k + hf(x_k, y_k)\ (k = 0, 1, 2\ldots n-1)$$

Погрешность вычислений на каждом шаге составляет $Rk = 0{,}5h^2y''(\varepsilon)$, где $x_k <= \varepsilon <= x_{k+1}$

При помощи усовершенствованного метода сначала вычисляют промежуточные значения

$$x_{k+½} = x_k + h/2;\ y_{k+½} = y_k + h/2*f(x_k, y_k)$$

затем находят $y_{k+1} = y_k + hf(x_{k+½}, y_{k+½})$ Метод имеет большую точность, чем метод Эйлера.

Задание: Составить решение задачи Коши для обыкновенного дифференциального уравнения первого порядка усовершенствованным методом ломаных на отрезке [0,2; 1,2] с шагом = 0,1 при начальном условии y(0,2) = 0,25. Все вычисления выполнить с четырьмя десятичными знаками.

Образец

$$y' = 0{,}185(x^2 + \cos 0{,}7x) + 1{,}843y$$

Используем формулу $y_{i+1}=y_i+hy'_{i+1/2}$, где

$$y'_{i+1/2} = y'(x_i + h/2;\ y_{i+1/2}),\ y_{i+1/2} = y_i + h/2y_i$$

Вычисления представим в таблице (учитывая, что h/2=0,05)

Таблица 8.1

i	x_i	y_i	y'_i	$h/2 * y'_i$	$x_i+h/2$	$y_{i+1/2}$	$y'_{i+1/2}$	$h * y'_{i+1/2}$
0	0,2	0,25	0,6513	0,0326	0,25	0,2826	0,7145	0,0715
1	0,3	0,3215	0,7901	0,0395	0,35	0,3610	0,8675	0,0868
2	0,4	0,4083	0,9599	0,0480	0,45	0,4563	1,0543	0,1054
3	0,5	0,5137	1,1668	0,0583	0,55	0,5720	1,2816	0,1282
4	0,6	0,6419	1,4185	0,0709	0,65	0,7128	1,5581	0,1558
5	0,7	0,7977	1,7240	0,0862	0,75	0,8839	1,8932	0,1893
6	0,8	0,9870	2,0942	0,1047	0,85	1,0917	2,2989	0,2299
7	0,9	1,2169	2,5421	0,1271	0,95	1,3440	2,7895	0,2790
8	1,0	1,4959	3,0834	0,1542	1,05	1,6501	3,3823	0,3382
9	1,1	1,8341	3,7369	0,1868	1,15	2,0209	4,0974	0,4097
10	1,2	2,2438	-	-	-	-	-	-

Решение дает значения x_i, y_i (i=0, 1, 2, .., 10), полученные в процессе вычислений (первые два столбца таблицы).

Варианты:

1. $y' = 0{,}133(x^2 + \sin 2x) + 0{,}872y$
2. $y' = 0{,}215(x^2 + \cos 1{,}5x) + 1{,}283y$
3. $y' = 0{,}158(x^2 + \sin 0{,}8x) + 1{,}164y$
4. $y' = 0{,}173(x^2 + \cos 0{,}7x) + 0{,}754y$
5. $y' = 0{,}221(x^2 + \sin 1{,}2x) + 0{,}452y$
6. $y' = 0{,}163(x^2 + \cos 0{,}4x) + 0{,}635y$
7. $y' = 0{,}218(x^2 + \sin 1{,}6x) + 0{,}718y$
8. $y' = 0{,}145(x^2 + \cos 0{,}5x) + 0{,}842y$
9. $y' = 0{,}213(x^2 + \sin 1{,}8x) + 0{,}368y$
10. $y' = 0{,}127(x^2 + \cos 0{,}6x) + 0{,}573y$
11. $y' = 0{,}232(x^2 + \sin 1{,}4x) + 1{,}453y$
12. $y' = 0{,}417(x^2 + \cos 0{,}8x) + 0{,}972y$
13. $y' = 0{,}324(x^2 + \sin 1{,}5x) + 1{,}612y$
14. $y' = 0{,}263(x^2 + \cos 1{,}2x) + 0{,}453y$
15. $y' = 0{,}372(x^2 + \sin 0{,}7x) + 0{,}758y$
16. $y' = 0{,}343(x^2 + \cos 0{,}4x) + 1{,}315y$
17. $y' = 0{,}276(x^2 + \sin 1{,}6x) + 0{,}988y$
18. $y' = 0{,}173(x^2 + \cos 0{,}6x) + 1{,}534y$
19. $y' = 0{,}258(x^2 + \sin 0{,}4x) + 0{,}724y$
20. $y' = 0{,}317(x^2 + \cos 1{,}4x) + 1{,}344y$

2.3 **Метод Эйлера-Коши (Метод Хьюна)**

Пусть требуется найти приближенное решение дифференциального уравнения $y' = f(x,y)$, удовлетворяющее начальному условию $y(x_0) = y_0$. При решении задачи методом Эйлера-Коши, который иногда называют методом Хьюна, приближенное значение y_1 решения y(x) в точке $x_1 = x_0+h$, вычисляется по формуле $y_1=y_0+[f(x_0,y_0)+ f(x_1,y_0+hf(x_0,y_0))]h/2$. $f(x_1,y_0+hf(x_0,y_0)) = Y_{i+1}$ Приближенные значения y_i решения уравнения y(x) в точках $x_i = x_0 +ih$, i= 1, 2,..., n вычисляются по формулам $y_{i+1} = y_i+ (f(x_i,y_i)+f(x_{i+1},y_i+ hf(x_i,y_i)))h/2$. Метод Эйлера-Коши является одношаговым методом второго порядка. Метод Эйлера-Коши относится к классу методов прогноза и коррекции. Локальная погрешность метода на одном шаге равна $O(h^3)$.

Т.е. сначала вычисляют $y_{k+1}^{(0)} = y_k + hf(x_k, y_k)$,

а затем это значение уточняют по формуле

$$y_{k+1} = y_k + h/2 * [f(x_k, y_k) + f(x_{k+1}, y_{k+1}] \ (i = 1, 2, \ldots)$$

Погрешность имеет порядок h^3 на каждом шаге.

Задание: составить решение задачи Коши для обыкновенного дифференциального уравнения первого порядка методом Эйлера-Коши. Воспользоваться вариантами предыдущей работы. Вычисления выполнять с четырьмя десятичными знаками. В ответ включать цифры, совпавшие при решении в работах 1 и 2.

Образец

$$y' = 0{,}185(x^2 + \cos 0{,}7x) + 1{,}843y$$

используем формулу:

$y_{i+1} = y_i + h/2 * (y'_i + Y_{i+1})$,

где $Y_{i+1} = y'(x_{i+1}, Y_{i+1})$, $Y_{i+1} = y_i + hy_i$

Вычисления представим в таблице:

Таблица 8.2

i	x_i	y_i	y'_i	hy'_i	Y_{i+1}	Y'_{i+1}	$y'_i + Y'_{i+1}$	$(y'_i + Y'_{i+1})h/2$
0	0,2	0,25	0,6513	0,0651	0,3151	0,7784	1,4297	0,0715
1	0,3	0,3215	0,7901	0,0790	0,4005	0,9455	1,7356	0,0868
2	0,4	0,4083	0,9599	0,0960	0,5043	1,1495	2,1094	0,1055
3	0,5	0,5138	1,1670	0,1167	0,6305	1,3975	2,5645	0,1282
4	0,6	0,6420	1,4187	0,1419	0,7839	1,6986	2,1173	0,1559
5	0,7	0,7979	1,7244	0,1724	0,9703	2,0635	3,7879	0,1894
6	0,8	0,9873	2,0947	0,2095	1,1968	2,5050	4,5997	0,2300
7	0,9	1,2173	2,5428	0,2543	1,4716	3,0386	5,5814	0,2791
8	1,0	1,4964	3,0844	0,3084	1,8048	3,6830	6,7674	0,3384
9	1,1	1,8348	3,7382	0,3738	2,2086	4,6404	8,1986	0,4099
10	1,2	2,2447	-	-	-	-	-	-

Решение дают значения x_i, y_i (i=0, 1, 2, .., 10) (первые два столбца таблицы).

Сравнивая найденное решение с решением, полученным в работе 1, видим, что они расходятся в последних цифрах, поэтому в ответ исключим значения, округленные до тысячных.

Ответ:

Таблица 8.3

x_i	y_i	x_i	y_i
0,2	0,25	0,8	0,987
0,3	0,322	0,9	1,217
0,4	0,408	1,0	1,496
0,5	0,514	1,1	1,835
0,6	0,642	1,2	2,245
0,7	0,797		

2.4 Метод Эйлера с уточнением

Метод Эйлера с уточнением заключается в том, что каждое значение $y_{k+1} = y(x_{k+1})$, где y(x) – искомая функция, а $x_{k+1} = x_0 + h(k+1)$, k=0, 1, 2… определяется следующим образом:

За начальное приближение берется

$$y^{(0)}_{k+1} = y_k + hf(x_k, y_k), \text{ где } f(x, y) = y'(x, y);$$

найденное значение $y^{(0)}_{k+1}$ уточняется по формуле:

$$y^{(i)}_{k+1} = y_k + h/2 * [f(x_k, y_k) + f(x_{k+1}, y^{(i-1)}_{k+1}] \ (i = 1, 2, \ldots)$$

Уточнение продолжают до тех пор, пока в пределах требуемой точности два последовательных приближения не совпадут.

Все описанные вычисления удобно производить, составив следующие таблицы:

основную таблицу, в которой записывается ответ примера (Таблица 8.4);

таблицу, в которой выполняется процесс последовательных приближений (таблица 8.5); вспомогательную таблицу, в которой вычисляются значения функции $f(x_k, y_k)$ (таблица 8.6).

Образец

$$y' = x + \sin\frac{y}{2{,}25};\ y_0(1{,}4) = 2{,}2;\ x \in [1{,}4;2{,}4]$$

Таблица 8.4

K	x_k	y_k	$f_k=f(x_k, y_k)$	hf_k
0	1,4	2,2	2,2292	0,2229
1	1,5	2,4306	2,3821	0,2382
2	1,6	2,6761	2,5281	0,2528
3	1,7	2,9357	2,6648	0,2665
4	1,8	3,2084	2,7895	0,2790
5	1,9	3,4929	2,8998	0,2900
6	2,0	3,7876	2,9936	0,2994
7	2,1	4,0908	3,0696	0,3070
8	2,2	4,4006	3,1268	0,3127
9	2,3	4,7152	3,1654	0,3165
10	2,4	5,0328		

Таблица 8.5

k+1	x_{k+1}	y_k	i	$y^{(i)}_{k+1}$	f_k	$f^{(i)}_{k+1}$	$f_k + f^{(i)}_{k+1}$	$h/2*(f_k + f^{(i)}_{k+1})$
1	1,5	2,2	0	2,4229	2,2292	2,3805	4,6097	0,2305
			1	2,4305		2,3820	4,6112	0,2306
			2	2,4306		2,3821	4,6113	0,2306
2	1,6	2,4306	0	2,6688	2,3821	2,5268	4,9089	0,2454
			1	2,6760		2,5280	4,9101	0,2455
			2	2,6761		2,5281	4,9102	0,2455
3	1,7	2,6761	0	2,9289	2,5281	2,6641	5,1922	0,2596
			1	2,9357		2,6648	5,1929	0,2596
4	1,8	2,9357	0	3,2022	2,6648	2,7892	5,4540	0,2727
			1	3,2084		2,7895	5,4543	0,2727
5	1,9	3,2084	0	3,4874	2,7895	2,8998	5,6893	0,2845
			1	3,4929		2,8998	5,6893	0,2845
6	2,0	3,4929	0	3,7829	2,8998	2,9939	5,8937	0,2947
			1	3,7876		2,9936	5,8934	0,2947
7	2,1	3,7876	0	4,0870	2,9936	3,0700	6,0636	0,3032
			1	4,0908		3,0696	6,0632	0,3032
8	2,2	4,0908	0	4,3978	3,0696	3,1273	6,1969	0,3098
			1	4,4006		3,1268	6,1964	0,3098
9	2,3	4,4006	0	4,7133	3,1268	3,1658	6,2926	0,3146
			1	4,7152		3,1654	6,2922	0,3146
10	2,4	4,7152	0	5,0517	3,1654	3,1866	6,3520	0,3176
			1	5,0328		3,1863	6,3517	0,3176

Таблица 8.6

k	x	y	$\frac{y}{2,25}$	$\sin\frac{y}{2,25}$	$y' = x + \sin\frac{y}{2,25}$
0	1,4	2,2	0,9778	0,8292	2,2292
1	1,5	2,4229	1,0768	0,8805	2,3805
	1,5	2,4305	1,0802	0,8820	2,3820
	1,5	2,4306	1,0803	0,8821	2,3821
2	1,6	2,6688	1,1860	0,9268	2,5268
	1,6	2,6760	1,1893	0,9280	2,5280
	1,6	2,6761	1,1894	0,9281	2,5281
3	1,7	2,9289	1,3017	0,9641	2,6641
	1,7	2,9357	1,3048	0,9648	2,6648
4	1,8	2,3022	1,4232	0,9892	2,7822
	1,8	2,3084	1,4260	0,9895	2,7895
5	1,9	3,4874	1,5500	0,9998	2,8998
	1,9	3,4929	1,5524	0,9998	2,8998
6	2,0	3,7829	1,6813	0,9939	2,9939
	2,0	3,7876	1,6834	0,9936	2,9936
7	2,1	4,0870	1,8164	0,9700	3,0700
	2,1	4,0908	1,8181	0,9696	3,0696
8	2,2	4,3978	1,9546	0,9273	3,1273
	2,2	4,4006	1,9558	0,9268	3,1268
9	2,3	4,7133	2,0948	0,8658	3,1658
	2,3	4,7152	2,0956	0,8654	3,1654
10	2,4	5,0317	2,2363	0,7866	3,1866
	2,4	5,0328	2,2368	0,7863	3,1863

Ответом являются значения $y_k(x)$, полученные в Таблица 8.4

Варианты:

1. $y' = x + \cos\frac{y}{\sqrt{5}}$; $y_0(1,8) = 2,6$; $x \in [1,8; 2,8]$
2. $y' = x + \cos\frac{y}{3}$; $y_0(1,6) = 4,6$; $x \in [1,6; 2,6]$
3. $y' = x + \cos\frac{y}{\sqrt{10}}$; $y_0(0,6) = 0,8$; $x \in [0,6; 1,6]$
4. $y' = x + \cos\frac{y}{\sqrt{7}}$; $y_0(0,5) = 0,6$; $x \in [0,5; 1,5]$
5. $y' = x + \cos\frac{y}{\pi}$; $y_0(1,7) = 5,3$; $x \in [1,7; 2,7]$
6. $y' = x + \cos\frac{y}{2,25}$; $y_0(1,4) = 2,2$; $x \in [1,4; 2,4]$
7. $y' = x + \cos\frac{y}{e}$; $y_0(1,4) = 2,5$; $x \in [1,4; 2,4]$
8. $y' = x + \cos\frac{y}{\sqrt{2}}$; $y_0(0,8) = 1,4$; $x \in [0,8; 1,8]$
9. $y' = x + \cos\frac{y}{\sqrt{3}}$; $y_0(1,2) = 2,1$; $x \in [1,2; 2,2]$
10. $y' = x + \cos\frac{y}{\sqrt{11}}$; $y_0(2,1) = 2,5$; $x \in [2,1; 3,1]$
11. $y' = x + \sin\frac{y}{\sqrt{5}}$; $y_0(1,8) = 2,6$; $x \in [1,8; 2,8]$

12. $y' = x + \sin\frac{y}{3}$; $y_0(1,6) = 4,6$; $x \in [1,6; 2,6]$

13. $y' = x + \sin\frac{y}{\sqrt{10}}$; $y_0(0,6) = 0,8$; $x \in [0,6; 1,6]$

14. $y' = x + \sin\frac{y}{\sqrt{7}}$; $y_0(0,5) = 0,6$; $x \in [0,5; 1,5]$

15. $y' = x + \sin\frac{y}{\pi}$; $y_0(1,7) = 5,3$; $x \in [1,7; 2,7]$

16. $y' = x + \sin\frac{y}{2,8}$; $y_0(1,4) = 2,2$; $x \in [1,4; 2,4]$

17. $y' = x + \sin\frac{y}{e}$; $y_0(1,4) = 2,5$; $x \in [1,4; 2,4]$

18. $y' = x + \sin\frac{y}{\sqrt{2}}$; $y_0(0,8) = 1,3$; $x \in [0,8; 1,8]$

19. $y' = x + \sin\frac{y}{\sqrt{3}}$; $y_0(1,1) = 1,5$; $x \in [1,1; 2,1]$

20. $y' = x + \sin\frac{y}{\sqrt{11}}$; $y_0(0,6) = 1,2$; $x \in [0,6; 1,6]$

2.5Методы Рунге-Кутты

Пусть требуется найти приближенное решение дифференциального уравнения y'=f(x,y), удовлетворяющее начальному условию $y(x_0)=y_0$. Численное решение задачи состоит в построении приближенного значения y_1 решения уравнения y(x) в точке $x_1=x_0 + h$. Приближенное решение в точке x_0+h можно вычислить, используя разложение точного решения в окрестности точки x_0 по формуле Тейлора $y(x_0+h) = y(x_0)+y'(x_0)h+...+ y^{(n-1)}(x0)h^{n-1}+O(h^n) = y(x0)+h^n(x_0,y_0,h)$. Расчетные формулы для приближенного решения можно получить, ограничившись первыми членами разложения: $y(x_0+h) = y_0+hL(x_0,y_0,h)$, где $L(x_0,y_0,h) = f(x_0,y_0)+...+h^{n-1}f^{(n-1)}/n!$. Эти формулы содержат производные от правых частей уравнения. Методами Рунге-Кутты называют группу одношаговых методов, в которых формулы для вычисления L получены из Тейлоровского разложения, но не содержат производных от правой части f. Наиболее просты для реализации явные методы Рунге- Кутты: $y(x_0+h) = y_0+hL(x_0,y_0,h)$, где $L(x_0,y_0,h) = c_1k_1+c_2k_2+ ... +c_mk_m$, $k_1 = f(x_0,y_0)$, $k_r = f(x_0+a_rh)$, $y(x_0+h) = y_0+h(b_{r1}k_1+...+b_{rr-1}k_{r-1})$ r = 1, 2, ..., m. Коэффициенты расчетных формул подбирают так, чтобы $L(0) = L'(0) = ... = L^{(s)}(0) = 0$. Такие методы принято называть Метод Рунге-Кутты m-s. Локальная погрешность такого метода равна $O(h^s)$. Методы Рунге-Кутты не требуют вычисления дополнительных начальных точек и позволяют легко менять шаг. Наибольшее распространение получили явные методы Рунге-Кутты четвертого порядка.

2.6 Метод Рунге-Кутты 4-го порядка (4-4)

Пусть требуется найти приближенное решение дифференциального уравнения y'=f(x,y), удовлетворяющее начальному условию $y(x_0) = y_0$. Численное решение задачи состоит в построении приближенного значения y_1 решения уравнения y(x) в точке $x_1 = x_0 + h$. Методом Рунге-Кутты четвертого порядка называют метод Рунге-Кутты 4-4, в котором приближенное решение в точке x_0+h вычисляется по формулам: $y(x_0+h) = y_0+h(k_1+2k_2+2k_3+k_4)/6$, $k_1 = f(x_0,y_0)$, $k_2 = f(x_0+h/2,y_0+hk_1/2)$, $k_3 = f(x_0+h/2,y_0+hk_2/2)$, $k_4 = f(x_0+h/2,y_0+hk_3)$. Локальная погрешность метода равна $O(h^4)$.

2.7 Многошаговые методы.

Одним из путей построения разностных схем основан на том, что для вычисления значения y_{i+1} используются результаты не одного, а k предыдущих шагов, т.е. значения $y_{i-k+1}, y_{i-k+2},..., y_i$. В этом случае получается $k-$ шаговый метод.

Многошаговые методы могут быть построены следующим образом.

Запишем исходное уравнение $Y' = f(x,Y)$ в виде

$$dY(x) = f(x,Y)dx \tag{8.5}$$

Проинтегрируем обе части этого уравнения по x на отрезке $[x_i, x_{i+1}]$. Интеграл от левой части легко

вычисляется:

$$\int_{x_i}^{x_{i+1}} dY(x) = Y(x_{i+1}) - Y(x_i) \approx y_{i+1} - y_i \tag{8.6}$$

Для вычисления интеграла от правой части уравнения строится сначала интерполяционный многочлен P_{k-1} степени $k-1$ для аппроксимации функции $f(x,Y)$ на отрезке $[x_i, x_{i+1}]$ по значениям $f(x_{i-k+1}, y_{i-k+1}), f(x_{i-k+2}, y_{i-k+2}), ..., f(x_i, y_i)$. После этого можно написать

$$\int_{x_i}^{x_{i+1}} f(x,Y)\,dx \approx \int_{x_i}^{x_{i+1}} P_{k-1}(x)\,dx.$$

На основе этой формулы можно строить различные многошаговые методы любого порядка точности. Порядок точности зависит от степени интерполяционного многочлена $P_{k-1}(x)$, для построения которого используются значения сеточной функции $y_i, y_{i-1,}, ..., y_{i-k+1}$, вычисленные на k предыдущих шагах.

Широко распространенным семейством многошаговых методов являются *методы Адамса*. Простейший из них, получающийся при $k=1$, совпадает с рассмотренным ранее методом Эйлера первого порядка точности. В практических расчетах чаще всего используется вариант метода Адамса, имеющий четвертый порядок точности и использующий на каждом шаге результаты предыдущих четырех. Именно его и называют обычно методом Адамса. Рассмотрим этот метод.

Пусть найдены значения $y_{i-3,}, y_{i-2,}, y_{i-1,}, y_i$ в четырех последовательных узлах $(k=4)$. При этом имеются также вычисленные ранее значения правой части $f_{i-3,}, f_{i-2,}, f_{i-1,}, f_i$,где $f_l = f(x_l, y_l)$. В качестве интерполяционного многочлена $P_3(x)$ можно взять многочлен Ньютона . В случае постоянного шага конечные разности для правой части в узле имеют вид

$$\Delta f_i = f_i - f_{i-1,}$$
$$\Delta^2 f_i = f_i - 2f_{i-1} + f_{i-2,}$$
$$\Delta^3 f_i = f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3.}$$

Тогда разностную схему четвертого порядка метода Адамса можно записать после необходимых преобразований в виде

$$y_{i+1} = y_i + hf_i + \frac{h^2}{2}\Delta f_i + \frac{5h^3}{12}\Delta^2 f_i + \frac{3h^4}{8}\Delta^3 f$$

Сравнивая метод Адамса с методом Рунге-Кутта той же точности, отмечаем его экономичность, поскольку он требует вычисления лишь одного значения правой части на каждом шаге (в методе Рунге-Кутта - четырех). Но метод Адамса неудобен тем, что невозможно начать счет по одному лишь известному значению y_0. Расчет может быть начат только с узла x_3, а не x_0. Значения $y_{1,}y_{2,}y_{3,}$ необходимые для вычисления y_4, нужно получить каким-либо другим способом (например, методом Рунге-Кутта), что существенно усложняет алгоритм. Кроме того, метод Адамса не позволяет (без усложнения формул) изменить шаг h в процессе счета; этого недостатка лишены одношаговые методы.

Рассмотрим еще одно семейство многошаговых методов, которые используют неявные схемы, - *методы прогноза и коррекции* (они называются также *методами предиктор-корректор*). Суть этих методов состоит в следующем. На каждом шаге вводятся два этапа, использующих многошаговые методы: с помощью явного метода (*предиктора*) по известным значениям функции в предыдущих узлах находится начальное приближение $y_{i+1} = y_{i+1}^{(0)}$ в новом узле; используя неявный метод (*корректор*), в результате итераций находятся приближения $y_{i+1}^{(1)}, y_{i+1}^{(2)}, \ldots$

Одним из вариантов метода прогноза и коррекции может быть получен на основе метода Адамса четвертого порядка. Приведем окончательный вид разностных соотношений: на этапе предиктора

$$y_{i+1} = y_i + \frac{h}{24}\left(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}\right) \quad (8.7)$$

На этапе корректора

$$y_{i+1} = y_i + \frac{h}{24}\left(9f_{i+1} + 19f_i - 5f_{i-1} + f_{i+2}\right) \quad (8.8)$$

Явная схема (8.7) используется на каждом шаге один раз, а с помощью неявной схемы (8.8) строится итерационный процесс вычисления y_{i+1} ,поскольку это значение входит в правую часть выражения

$$f_{i+1} = f\left(x_{i+1}, y_{i+1}\right).$$

Заметим, что в этих формулах, как и в случае метода Адамса, при вычислении y_{i+1} необходимы значения сеточной функции в четырех предыдущих узлах: $y_i, y_{i-1}, y_{i-2}, y_{i-3}$. Следовательно, расчет по этому методу может быть начат только со значения . Необходимые при этом y_1, y_2, y_3 находятся по методу Рунге-Кутта, y_0 задается начальным условием. Это характерная особенность многошаговых методов.

2.8 Метод Адамса

Пусть требуется найти приближенное решение дифференциального уравнения y'=f(x,y), удовлетворяющее начальному условию $y(x_0) = y_0$. Численное решение задачи состоит в построении приближенного значения y_1 решения уравнения y(x) в точке $x_1 = x_0 + h$. Методами Адамса называют группу многошаговых методов, в которых приближенное решение $y_{n+1}=y(x_{n+1})$ в точке $x_{n+1}=x_0+h(n+1)$ вычисляется по формуле, использующей полином P(x) наименьшей степени, интерполирующий правую часть f(x,y) по значениям f_n, f_{n-1}, ...,f_{n-k+1}, $f_r = f(x_r,y_r)$. Методы, в которых $P(x) = P_{kn}(x)$ называют k-шаговыми явными методами Адамса-Башфорта, а методы, в которых $P(x) = P_{k+1n+1}$ - (k+1)- шаговыми неявными методами Адамса-Мултона. Методы Адамса k-гопорядка требуют предварительного вычисления решения в k начальных точках. Здесь для вычисления дополнительных начальных значений использован метод Рунге-Кутты 4-4. Локальная погрешность методов Адамса k-го порядка - $O(h^k)$. Методы Адамса обладают лучшей, по сравнению с методами Рунге-Кутты устойчивостью.

Формула со вторыми разностями:

$$y_{i+1} = y_i + q_i + \frac{1}{2}\Delta q_{i-1} + \frac{5}{12}\Delta^2 q_{i-2}, \text{ где } q_i = hf(x_i, y_i)$$

Задание: используя метод Адамса со вторыми разностями, составить таблицу приближенных значений интеграла дифференциального уравнения y′=f(x, y), удовлетворяющего начальным условиям $y(x_0)= y_0$ на отрезке [0, 1]; шаг h=0,1. Все вычисления вести с четырьмя десятичными знаками. Начальный отрезок определить методом Рунге-Кутта.

Образец: $y' = 1 +0,2y*\sin x – 1,5y^2 = f(x, y)$; y(0) = 0, $x \in [0, 1]$, h = 0,1.

1. Определим значения $y_1 = y(0,1)$, $y_2 = y(0,2)$ (начальный отрезок) методом Рунге – Кутта. При этом значения $y_{i+1} = y(x_{i+1)}$, где $x_{i+1} = x_i + h$, находятся по формулам:

$$y_{i+1} = y_i + \Delta y_i,$$

$$\Delta y_i = \frac{1}{6}(k_1^i + 2k_2^i + 2k_3^i + k_4^i),$$

где

$$k_1^i = hf(x_i, y_i),$$

$$k_2^i = hf(x_i + \frac{h}{2}, y_i + \frac{k_1^i}{2}),$$

$$k_3^i = hf(x_i + \frac{h}{2}, y_i + \frac{k_2^i}{2}),$$

$$k_4^i = hf(x_i + h, y_i + k_3^i)$$

Все вычисления будем располагать в таблице 8.7

Таблица 8.7

x	y(x)	sinx	0,2y*sinx	$-1,5y^2$	f(x, y)	hf(x,y)	Δy
0	0	0	0	0	1	0,1	0,1000
0,05 0,05 0,10	0,05 0,0498 0,0997	0,0500 0,0500 0,0998	0,0005 0,0005 0,0020	-0,0038 -0,0037 -0,0149	0,9967 0,9968 0,9871	0,0997 0,0997 0,0987	0,1994 0,1994 0,0987
							0,5979-(1/6) = 0,0996
0,10	0,0996	0,0998	0,0020	-0,0149	0,9871	0,0987	0,0987
0,15 0,15 0,20	0,1490 0,1482 0,1968	0,1494 0,1494 0,1987	0,0045 0,0044 0,0078	-0,0333 -0,0329 -0,0581	0,9712 0,9715 0,9497	0,0971 0,0972 0,0950	0,1942 0,1944 0,0950
							0,5823*(1/6) = 0,0970
0,20	0,1966	0,1987	0,0078	-0,0580	0,9498		

2. Вычисление последующих значений $y_i = y(x_i)$, где $x_i = x_0 + ih$ (i=3, 4, …), производим по формуле Адамса со вторыми разностями

$$y_{i+1} = y_i + q_i + \frac{1}{2}\Delta q_{i-1} + \frac{5}{12}\Delta^2 q_{i-2}, \text{ где } q_i = hf(x_i, y_i)$$

Вычисления производим в следующих таблицах 8.8, 8.9 и 8.10.
Таблица 8.8 содержит окончательные значения $y(x_i)$ и значения конечных разностей, имеющихся в вычислительной формуле.

Таблица 8.8

i	x_i	y_i	$f(x_i, y_i)$	$q_i = hf_i$	Δq_i	$\Delta^2 q_i$
0	0	0	0,1000	0,10000	-0,00129	-0,00244
1	0,1	0,0996	0,9871	0,09871	-0,00373	-0,00204
2	0,2	0,1966	0,9498	0,09498	-0,00577	-0,00154
3	0,3	0,2887	0,8921	0,08921	-0,00731	-0,00088
4	0,4	0,3742	0,8190	0,08190	-0,00819	-0,00035
5	0,5	0,4518	0,7371	0,07371	-0,00854	0,00008
6	0,6	0,5210	0,6517	0,06517	-0,00846	0,00049
7	0,7	0,5818	0,5671	0,05671	-0,00797	0,00067
8	0,8	0,6343	0,4874	0,04874	-0,00730	-
9	0,9	0,6792	0,4144	0,04144	-	-
10	1,0	0,7173	-	-	-	-

В таблице 8.9 представдлена результаты расчетов по формуле Адамса со вторыми разностями.

Таблица 8.9

i	2	3	4	5	6	7	8	9
y_i	0,1966	0,28870	0,37418	0,37418	0,54102	0,58177	0,63428	0,67924
q_i	0,09498	-0,08921	-0,0819	-0,08190	0,6517	0,05671	0,04874	0,04144
$\frac{1}{2}\Delta q_{i-1}$	-0,00186	-0,00288	-0,00366	-0,00366	-0,00427	-0,00423	-0,00398	-0,00365
$\frac{5}{12}$	-0,0102	-0,0085	-0,00064	-0,00064	-0,00015	0,00003	0,00020	0,00028
$\Delta^2 q_{i-2}$								
y_{i+1}	0,28870	0,37418	0,45178	0,45178	0,58177	0,63428	0,67924	0,71731

В таблице 8.10 представлены результаты вычисления значений функции

$$y' = f(x_i, y_i) = 1 + 0{,}2y_i * \sin x_i - 1{,}5y_i^2$$

Таблица 8.10

x_i	y_i	$0{,}2*\sin x_i$	$0{,}2y_i*\sin x_i$	$-1{,}5y_i^2$	$f(x_i, y_i)$
0,3	0,2887	0,0591	0,0171	-0,1250	0,8921
0,4	0,3742	0,0779	0,0292	0,2102	0,8190
0,5	0,4518	0,0959	0,0433	0,3062	0,7371
0,6	0,5210	0,1129	0,0588	0,4071	0,6517
0,7	0,5818	0,1288	0,0749	0,5078	0,5671
0,8	0,6343	0,1435	0,0910	0,6036	0,4874
0,9	0,6792	0,1567	0,1064	0,6920	0,4144

Ответом являются значения функции $y(x_i)$, полученные в таблице 8.8.

Варианты:

1. $y' = 1 + 0{,}2*y*\sin x - y^2$, $y(0) = 0$
2. $y' = \cos(x + y) + 0{,}5(x - y)$, $y(0) = 0$
3. $y' = \frac{\cos x}{x+1} - 0{,}5y^2$, $y(0) = 0$
4. $y' = (1 - y^2)\cos x + 0{,}6y$, $y(0) = 0$
5. $y' = 1 + 0{,}4*y*\sin x - 1{,}5y^2$, $y(0) = 0$
6. $y' = \frac{\cos y}{x+2} + 0{,}3y^2$, $y(0) = 0$
7. $y' = \cos(1{,}5x + y) + (x - y)$, $y(0) = 0$
8. $y' = 1 - \sin(x + y) + \frac{0{,}5y}{x+2}$, $y(0) = 0$
9. $y' = \frac{\cos y}{1{,}5+x} + 0{,}1y^2$, $y(0) = 0$
10. $y' = 0{,}6\sin x - 1{,}25y^2 + 1$, $y(0) = 0$
11. $y' = \cos(2x + y) + 1{,}5(x - y)$, $y(0) = 0$
12. $y' = 1 - \frac{0{,}1y}{x+2} - \sin(2x + y)$, $y(0) = 0$
13. $y' = \frac{\cos y}{1{,}25+x} - 0{,}1y^2$, $y(0) = 0$
14. $y' = 1 + 0{,}8*y*\sin x - 2y^2$, $y(0) = 0$
15. $y' = \cos(1{,}5x + y) + 1{,}5(x - y)$, $y(0) = 0$
16. $y' = 1 - \sin(2x + y) + \frac{0{,}3y}{x+2}$, $y(0) = 0$
17. $y' = \frac{\cos y}{1{,}75+x} - 0{,}5y^2$, $y(0) = 0$
18. $y' = 1 + (1 - x)\sin y - (2 + x)y$, $y(0) = 0$
19. $y' = (0{,}8 - y^2)\cos x + 0{,}3y$. $y(0) = 0$
20. $y' = 1 + 2{,}2\sin x + 1{,}5y^2$, $y(0) = 0$

2.9 Метод Милна.

Пусть для уравнения $y' = f(x, y)$, кроме начального условия $(x_0) = y_0$, найден «начальный отрезок», т.е. значения искомой функции $y(x_i) = y_i$ в точках $x_i = x_0 + ih$ $(i = 1, 2, 3)$

Последующие значения y_i при $i = 4, 5, \ldots$ определяют на каждом шаге следующим образом. Для предсказания используют первую формулу Милна

$$y_i^{пред} = y_{i-4} + \frac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1})$$

Используя $y_i^{пред}$, находят $f_i^{пред} = f(x_i, y_i^{пред})$ и производят уточнения (коррекцию) значения y_i по второй формуле Милна

$$y_i^{корр} = y_{i-2} + \frac{h}{3}(f_{i-2} + 4f_{i-1} + f_i^{пред})$$

Абсолютная погрешность ε_i значения $y_i^{корр}$ приближенно определятся по формуле

$$\varepsilon_i \approx \frac{1}{29}\left|y_i^{корр} - y_i^{пред}\right|$$

Если точность результата достаточна, то полагают $y_i \approx y_i^{корр}$

Задание: Используя метод Милна, составить таблицу приближенных значений интеграла дифференциального уравнения y = f(x, y), удовлетворяющего начальным условиям y(x0)=y0 на отрезке [0, 1]; шаг h = 0,1; все вычисления вести с четырьмя десятичными знаками. Начальный отрезок определить методом Рунге-Кутта.

Образец: $y' = 1{,}6x + 0{,}5y^2 = f(x, y)$; $y(0) = 0{,}3$

1. Определение начального отрезка производится по формуле Рунге-Кутта

$$y_{i+1} = y_i + \frac{1}{6}(k_1^i + 2k_2^i + 2k_3^i + k_4^i),$$

где

$$k_1^i = hf(x_i, y_i),$$

$$k_2^i = hf(x_i + \frac{h}{2}, y_i + \frac{k_1^i}{2}),$$

$$k_3^i = hf(x_i + \frac{h}{2}, y_i + \frac{k_2^i}{2}),$$

$$k_4^i = hf(x_i + h, y_i + k_3^i)$$

Все необходимы расчеты осуществляем с помощью таблице 8.11, в которой $\Delta y_i = \frac{1}{6}(k_1^i + 2k_2^i + 2k_3^i + k_4^i)$,

Таблица 8.11

I	x	y	1,6x	$0{,}5y^2$	f(x, y)	k	Δy_i
0	0	0,3	0	0,045	0,0450	0,00450	0,00450
	0,05	0,3022	0,08	0,0457	0,1257	0,01257	0,02514
	0,05	0,3063	0,08	0,0469	0,1269	0,01269	0,025318
	0,1	0,3127	0,16	0,0489	0,2089	0,02089	0,02089
							0,07591*1/6= =0,0127
1	0,1	0,3127	0,16	0,0489	0,2089	0,02089	0,02099
	0,15	0,3231	0,240	0,0522	0,2922	0,02922	0,05844
	0,15	0,3273	0,240	0,0536	0,2936	0,01936	0,05872
	0,20	0,3421	0,32	0,0585	0,3785	0,03785	0,03785
							0,17590*1/6= =0,0293
2	0,2	0,3420	0,32	0,0585	0,3785	0,03785	0,03785
	0,25	0,3609	0,40	0,0651	0,4651	0,04651	0,09302
	0,25	0,3653	0,40	0,0667	0,4667	0,04667	0,09334
	0,30	0,3887	0,48	0,0755	0,5555	0,05555	0,05555
							0,27976*1/6= =0,0466
3	0,30	0,3886	0,48	0,0755	0,5555		

Последующие значения функции $y_{i+1} = y(x_{i+1})$ (i = 3, 4, …, 9) будем определять методом Милна. Согласно этому методу, по ходу вычислений следует составить таблицу, содержащую значения y_i и $f(x_i, y_i)$ (таблица 8.12).

Таблица 8.12

i	x_i	y_i	$1{,}6x_i$	$0{,}5y^2_i$	f(x, y)
0	0	0,3	0	0,0450	0,0450
1	0,1	3,127	0,16	0,0489	0,2089
2	0,2	0,3420	0,32	0,0585	0,3785
3	0,3	0,3886	0,48	0,0755	0,5555
4	0,4	0,4534	0,64	0,1028	0,7428
5	0,5	0,5376	0,80	0,1445	0,9445
6	0,6	0,6430	0,96	0,2067	0,1667
7	0,7	0,7719	1,12	0,2979	0,4179
8	0,8	0,9280	1,28	0,4306	1,7105
9	0,9	0,1160	1,44	0,6227	2,0627
10	1,0	1,3434	-	-	-

На каждом шаге вычисление ведется в два этапа. Сначала по первой формуле Милна находим

$$y_i^{(1)} = y_{i-4} + \frac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1}),$$

а затем по второй формуле Милна находим окончательное значение

$$y_i = y_i^{(2)} = y_{i-2} + \frac{h}{3}(f_{i-2} + 4f_{i-1} + f_i^{(1)})$$

где $f_i^{(1)} = f(x_i, y_i^{(1)})$

1. $y_4^{(1)} = y_0 + \frac{0{,}4}{3}(2f_1 - f_2 + 2f_3) =$ 0,3 + 0,4/3*(2*0б2089 – 0б3785 + 2*0,5555) = 0,4534; $f_4^{(1)} = 0{,}64 +$ +0,1028 = 0,7428;

$$y_4^{(2)} = y_2 + \frac{h}{3}(f_2 + 4f_3 + f_4^{(1)}) = 0{,}3420 + 0{,}1/3*(0{,}3785 + 4*0{,}5555 + 0{,}7428) = 0{,}4534.$$

Из сравнения $y_4^{(1)}$ и $y_4^{(2)}$ имеем $y_4 = 0{,}4534$.

2. $y_5^{(1)} = y_1 + \frac{4h}{3}(2f_2 - f_3 + 2f_4) =$ 0,3127 + 0,4/3(2,03785 – 0,5555 + 2*0,7428) = 0,5376; $y_5^{(1)} = 0{,}80 +$ 0,1445 = 0.9445.

$$y_5^{(2)} = y_3 + \frac{h}{3}(f_3 + 4f_4 + f_5^{(1)}) = 0{,}3886 + 0{,}1/3*(0{,}5555 + 4*0{,}7428 + 0{,}9445) = 0{,}5376$$

Из сравнения $y_5^{(1)}$ и $y_5^{(2)}$ имеем $y_5 = 0{,}5376$.

3. $y_6^{(1)} = y_2 + \frac{4h}{3}(2f_3 - f_4 + 2f_5) = 0{,}6430$; $y_6^{(1)} = 0{,}96 + 0{,}2067 = 1{,}1667$

$$y_6^{(2)} = y_4 + \frac{h}{3}(f_4 + 4f_5 + f_6^{(1)}) = 0{,}6430.$$

4. $y_7^{(1)} = y_3 + \frac{4h}{3}(2f_4 - f_5 + 2f_6) = 0{,}7719$; $y_7^{(1)} = 1{,}12 + 0{,}2979 = 1{,}4179$;

$$y_7^{(2)} = y_5 + \frac{h}{3}(f_5 + 4f_6 + f_7^{(1)}) = 0{,}7719.$$

5. $y_8^{(1)} = y_4 + \frac{4h}{3}(2f_5 - f_6 + 2f_7) = 0{,}9278$; $y_8^{(1)} = 1{,}7104$;

$$y_8^{(2)} = y_6 + \frac{h}{3}(f_6 + 4f_7 + f_8^{(1)}) = 0{,}9280.$$

6. $y_9^{(1)} = y_5 + \frac{4h}{3}(2f_6 - f_7 + 2f_8) = 1{,}1158;$ $y_9^{(1)} = 1{,}44 + 0{,}6225 = 2{,}0625;$

$$y_9^{(2)} = y_7 + \frac{h}{3}(f_7 + 4f_8 + f_9^{(1)}) = 1{,}1160.$$

7. $y_{10}^{(1)} = y_6 + \frac{4h}{3}(2f_7 - f_8 + 2f_9) = 1{,}3431;$ $y_{10}^{(1)} = 1{,}6 + 0{,}9020 = 2{,}5020;$

$$y_{10}^{(2)} = y_8 + \frac{h}{3}(f_8 + 4f_9 + f_{10}^{(1)}) = 1{,}3434.$$

Варианты:

1. $y' = x + y^2$; $y(0) = 0{,}5$.
2. $y' = 2x + y^2$; $y(0) = 0{,}3$.
3. $y' = 0{,}2x + y^2$; $y(0) = 0{,}1$.
4. $y' = x^2 + 2y$; $y(0) = 0{,}1$.
5. $y' = x^2 + y^2$; $y(0) = 0{,}7$.
6. $y' = 0{,}3x + y^2$; $y(0) = 0{,}4$.
7. $y' = x + 0{,}3y^2$; $y(0) = 0{,}3$.
8. $y' = 0{,}1x^2 + 2xy$; $y(0) = 0{,}8$.
9. $y' = 3x^2 + 0{,}1xy$; $y(0) = 0{,}2$.
10. $y' = x^2 + 0{,}1y^2$; $y(0) = 0{,}7$.
11. $y' = 0{,}2x^2 + y^2$; $y(0) = 0{,}8$.
12. $y' = 2x + 0{,}1y^2$; $y(0) = 0{,}2$.
13. $y' = x^2 + xy$; $y(0) = 0{,}2$.
14. $y' = x^2 + y$; $y(0) = 0{,}4$.
15. $y' = xy + y^2$; $y(0) = 0{,}6$.

§3 Краевые задачи.Метод стрельбы. Метод конечных разностей.

3.1.1 Общая схема.

В идейном плане метод стрельбы для решения краевых задач основывается на методе стрельбы доказательства разрешимости задачи

$$x'' = f(t, x, x'), \quad t[0, T] \tag{8.9}$$

$$x'' = f(t, x, x') \quad t\in[0, T] \tag{8.10}$$

$$x(0) = a, \quad x(T) = b \tag{8.11}$$

Относительно параметра α Rm решается (в общем случае нелинейное) уравнение

$$\varphi(T, a) = b \tag{8.12}$$

где φ(T, α) — решение уравнения (8.10), удовлетворяющее начальному условию

$$x(0) = a, \quad x'(0) = a \tag{8.13}$$

Начальную задачу (8.10), (8.13) решают каким-либо приближенным методом решения задач Коши (например, Рунге — Кутты), а уравнение (8.12)– каким-либо приближенным методом решения нелинейных (или линейных, если оно линейно) уравнений (например, методом Ньютона).

3.1.2 Проблема выбора точки "склеивания".

Допустим, решения уравнения (8.10) "неустойчивы" (т. е. экспоненциально возрастают) вправо. Тогда решения φ будут найдены с большой погрешностью. Вследствие этого левая часть уравнения (8.12) будет

определена с большой ошибкой, что повлечет большую погрешность в определении α и увеличит ошибку в определении искомого решения φ(·, α). В описанной ситуации выход тривиален. Неустойчивость решений уравнения (8.10) $x'' = f(t, x, x')$ $t \in [0, T]$ (8.10)вправо означает их устойчивость влево. Поэтому можно "стрелять влево с большой точностью", т. е. задавать подлежащий определению параметр β Rm ("угол прицеливания") на правом конце промежутка [0, T]: решать уравнение

$$\psi(0, \beta) = a \tag{8.14}$$

где ψ(T, β) — решение на [0, T] задачи Коши для уравнения (8.10) с начальными условиями

$$x(T) = b, \quad x'(T) = \beta \tag{8.15}$$

Если уравнение (8.10) "неустойчиво в обе стороны" (например, оно автономно, линейно и имеет собственные значения как с положительной, так и с отрицательной вещественной частью), то такая модификация метода уже не помогает. В описанной ситуации можно уменьшить значимость ошибок, вызванных переносом ошибок вдоль неустойчивых решений уравнения (8.10), применяя следующее соображение. Выберем точку T1 (0, T). Очевидно, если параметры α, β Rm таковы, что

$$\varphi(T1, \alpha) - \psi(T1, \beta) = 0, \quad \varphi'(T1, \alpha) - \psi'(T1, \beta) = 0, \tag{8.16}$$

где φ(T, α) и ψ(T, β) — решения задач (8.10), (8.13) и (8.10), (8.15) на отрезках [0, T1] и [T1, T], соответственно, а штрих означает дифференцирование по первому аргументу, то функция

x(t) = {φ(t, α) при t [0, T1],

ψ(t, β) при t [T1, T]

будет искомым решением задачи (8.10)$x'' = f(t, x, x')$ $t \in [0, T]$ (8.10)– (8.11).

Задача 1. Докажите.

Если, например, в качестве T1 взять середину отрезка [0, T], то неустойчивость решений φ и ψ будут влиять вдвое меньшее время и, следовательно, левая часть уравнения (8.16) будет определена точнее, чем для уравнений (8.12) и (8.14). Разумеется эти преимущества не даются бесплатно: размерность уравнения (8.16) возрастает вдвое по сравнению с размерностью уравнений (8.12) и (8.14) (последние уравнения — это системы m уравнений с m неизвестными, уравнение же (8.16) - система 2m уравнений с 2m неизвестными).

Эта идея обобщается в следующем пункте.

3.1.3 Метод параллельной стрельбы.

Выберем точки T1 < ... < Tn–1, разбивающие отрезок [0, T] на n подотрезков. Для унификации обозначим 0 через T0, T через Tn, a через a0, a b через an. Для любого i = 0, ..., n – 1 через φi(·, ai, α i) обозначим решение задачи Коши для уравнения (8.10) на отрезке [Ti, Ti+1] с начальными условиями

$$x(Ti) = ai, \quad x'(Ti) = \alpha i \tag{8.17}$$

Тогда, если ai (i = 1, ..., n – 1) и αi (i = 0, ..., n – 1) удовлетворяют уравнениям φi(Ti+1, ai, αi) = ai+1, i = 0, ..., n–1,

$$(\varphi i)'(Ti + 1, ai, \alpha i) = \alpha i + 1, \tag{8.18}$$

i = 0, ..., n – 1

то функция x на [0, T], совпадающая с φi(·, ai, αi) на [Ti, Ti+1], является решением задачи (8.10)– (8.11).

Задача.2. Докажите.

Метод параллельной стрельбы заключается в нахождении x путем решения системы (8.18) m×(2n–1) уравнений с m×(2n–1) неизвестными a1, ..., an–1, α0, ..., αn–1, в которой функции φi — решения начальных задач (8.10), (8.17)— находятся тем или иным приближенным методом.

Задача.3. Предложите метод типа метода параллельной стрельбы для краевой задачи

x′ = f(t, x), Ax(0) + Bx(T) = a,

в которой f: [0, T]×Rm → Rm, a A и B — линейные операторы на Rm, такие, что A + B невырожден; a Rm.

Еще один вариант метода параллельной стрельбы выглядит так. В дополнение к точкам Ti выбираются точки Si (Ti, Ti+1) (i = 1, ..., n – 1). Как и выше, через φi(·, ai, αi) обозначается решение уравнения (8.10) на отрезке [Ti, Ti+1], удовлетворяющее начальным условиям

x(Si) = ai, x′(Si) = αi

(i = 1, ..., n – 1). Значения параметров ai и αi находятся из соотношений

φi(Ti+1, ai, αi) = φi+1(Ti+1, ai+1, αi+1), i = 0, ..., n – 2,

φ′i(Ti+1, ai, αi) = φ′i+1(Ti+1, ai+1, αi+1), i = 0, ..., n – 2,

φ0(T0, a0, α0) = a, φn–1(Tn, an–1, αn–1) = b.

3.1.4 Случай линейной краевой задачи.

В общем случае, как уже говорилось, уравнения (8.12), (8.14), (8.18) являются нелинейными и требуют каких-либо приближенных (как правило, итерационных) методов решения. Если же уравнение (8.10) линейное, то, как нетрудно видеть, линейными получаются и перечисленные уравнения.

Задача 4. Докажите это утверждение.

Эти уравнения можно решать более эффективно, используя их линейность. Рассмотрим для примера метод

для краевой задачи (8.11). Обозначим через φi (i = 0, 1, ..., m) решения на отрезке [0, T] уравнения, удовлетворяющие начальным условиям x(0) = a, x′(0) = ei,
где e_i при i = 1, ..., m — векторы канонического базиса в Rm, а $e_0 = 0$.

Задача 5. Докажите, что если $\sum m_i=0\alpha_i= 1$, то функция $\varphi(t) = \sum m_i=0\alpha i\varphi(t)$есть решение уравнения, удовлетворяющее начальному условию x(0) = a.
Если мы найдем решение (α0, ..., αm) системы

$$\sum_{i=0}^{m} \alpha i = 1,$$
$$\sum_{i=0}^{m} \alpha i \varphi i(T) = 1, \tag{8.19}$$

то функция $\varphi(t) = \sum mi=0\alpha i\varphi i(t)$ будет, очевидно, решением задачи (8.11). Таким образом, для нахождения решения этой краевой задачи потребуется решить всего m+1 начальную задачу , а затем линейную систему (8.19) размерности (m+1)×(m+1).

3.2Метод конечных разностей.

Основная идея конечно-разностных методов решения краевых задач полностью аналогична идее конечно-разностных методов решения задачи Коши. Краевая задача, скажем, для определенности, задача (8.10)– (8.11) или, в операторной форме, заменяется разностной схемой

$$F\tau(x) = 0 \tag{8.20}$$

в пространстве Sτ сеточных функций на сетке Gτ. Дословно повторяются понятия порядка аппроксимации, устойчивости, сходимости. Точно так же, как и в случае задачи Коши, имеет место теорема Лакса. Поэтому для доказательства сходимости решений схемы (8.20) достаточно показать, что схема является аппроксимирующей и устойчивой. И если с изучением свойств аппроксимации обычно проблем не возникает, то в случае краевых задач исследование устойчивости доставляет по сравнению со случаем задачи Коши ряд дополнительных трудностей.
Мы начнем с простейшей линейной краевой задачи и простейшей разностной схемы.

ПРИМЕР. Рассмотрим скалярную линейную краевую задачу

$$x'' + A(t)x = c(t) \quad t \in [0, T] \tag{8.21}$$
$$x(0) = a \quad x(t) = b \tag{8.22}$$

Предположим, функция A непрерывна и A(t) < 0 при всех t □ [0, T].
Задача 6. Докажите, что при этих условиях задача (8.12)– (8.13) имеет единственное решение для любой непрерывной функции c и любых a и b.
Пусть Gτ — равномерная сетка на [0, T], причем τn = T. Определим на Sτ разностный оператор [Fτ(x)]i = {xi+1– 2xi + xi–1,

τ2+ A(ti)xi – c(ti), если i = 1, ..., n – 1, !!!!

$$xi - a, если\ i = 0; \quad xi - b, если\ i = n \tag{8.23}$$

Здесь мы при i = 1, ..., n – 1 аппроксимировали вторую производную в (1) в точке t центральными разностями.

Задача 7. Покажите, что если A и c дважды непрерывно дифференцируемы, то схема (S) с таким разностным оператором аппроксимирует задачу (8.21)– (8.22) со вторым порядком, т. е. $||F\tau P\tau\varphi||\tau = O(\tau 2)$ для любого решения φ задачи (8.21)– (8.22) .
Устойчивость схемы (8.23).
Пусть z Sτ, Fτφτ = 0, Fτψτ = z. Нам надо доказать, что $||\varphi\tau - \psi\tau||\tau \le Cs||z||\tau$ при некотором Cs и всех достаточно малых τ. Легко видеть, что сеточная функция ξ = φτ – ψτ удовлетворяет уравнению
Lτ(ξ) = z,
где [Lτ(ξ)]i = [Fτ(ξ)]i + c(ti) при i = 1, ..., n – 1 и [Lτ(ξ)]i = ξi при i = 0 и i = n.

Задача 9. Проведите необходимые выкладки.
Разностный оператор Lτ представляет собой разностный оператор Fτ для однородной краевой задачи (8.21)– (8.22) т. е. задачи при c(t) ≡ 0, a = b = 0.
Докажем сначала вспомогательную лемму, являющуюся разностным аналогом известного принципа максимума (в данном контексте — минимума).

Лемма.
Пусть сеточная функция γ удовлетворяет условиям $[L\tau(\gamma)]i \le 0$ при i = 1, ..., n – 1, $\gamma 0 \ge 0$, $\gamma n \ge 0$. Тогда $\gamma i \ge 0$ при всех i = 0, ..., n.

Д о к а з а т е л ь с т в о. Пусть $\delta = \min_{0\le i\le n}\gamma i$, а j — наименьший номер, при котором $\gamma j = \delta$. Предположим, что утверждение леммы не верно, т. е. $\delta < 0$. Поскольку $\gamma 0 \ge 0$ и $\gamma n \ge 0$, выполнены неравенства $0 < j < n$. По определению j

$$\gamma j-1 > \gamma j, \quad \gamma j+1 \ge \gamma j.$$

Но тогда, так как $A(tj) < 0$ и $\gamma j = \delta < 0$,

$$[L\tau(\gamma)]j = (\gamma j+1 - \gamma j) - (\gamma j - \gamma j-1)$$
$$\tau 2 + A(tj)\gamma j = \gamma j+1 - \delta$$
$$\tau 2 - \delta - \gamma i-1$$
$$\tau 2 + A(tj)\,\delta > 0,$$

что противоречит условиям леммы.

Продолжение доказательства устойчивости системы (8.23).

Определим функцию η $S\tau$ равенством

$$\eta i = |z0|T - i\tau T + |zn| \qquad i\tau T + (T - i\tau)i\tau 2||z||\tau.$$

Глава 9
ОПТИМИЗАЦИЯ

§ 1. Задача оптимизации. Постановка задачи.

1.1 Определения. Под *оптимизацией* понимают процесс выбора наилучшего варианта из всех возможных. С точки зрения инженерных расчетов методы оптимизации позволяют выбрать наилучший вариант конструкции, наилучшее распределение ресурсов и т. п.

В процессе решения задачи оптимизации обычно необходимо найти оптимальные значения некоторых параметров, определяющих данную задачу. При решении инженерных задач их принято называть *проектными параметрами,* а в экономических задачах их обычно называют *параметрами плана.* В качестве проектных параметров могут быть, в частности, значения линейных размеров объекта, массы, температуры и т. п. Число n проектных параметров $x_1, x_2,..., x_n$ характеризует размерность (и степень сложности) задачи оптимизации.

Выбор оптимального решения или сравнение двух альтернативных решений проводится с помощью некоторой зависимой величины (функции), определяемой проектными параметрами. Эта величина называется *целевой функцией* (или *критерием качества).* В процессе решения задачи оптимизации должны быть найдены такие значения проектных параметров, при которых целевая функция имеет минимум (или максимум). Таким образом, целевая функция — это глобальный критерий оптимальности в математических моделях, с помощью которых описываются инженерные или экономические задачи.

Целевую функцию можно записать в виде

$$u=f(x_1,x_2,\ldots,x_n) \qquad (9.1)$$

Примерами целевой функции, встречающимися в инженерных и экономических расчетах, являются прочность или масса конструкции, мощность установки, объем выпуска продукции, стоимость перевозок грузов, прибыль и т. п.

В случае одного проектного параметра (n = 1) целевая функция (9.1) является функцией одной переменной, и ее график — некоторая кривая на плоскости. При n = 2 целевая функция является функцией двух переменных, и ее графиком является поверхность.

Следует отметить, что целевая функция не всегда может быть представлена в виде формулы. Иногда она может принимать только некоторые дискретные значения, задаваться в виде таблицы и т. п. Во всех случаях она должна быть однозначной функцией проектных параметров.

Целевых функций может быть несколько. Например, при проектировании изделий машиностроения одновременно требуется обеспечить максимальную надежность, минимальную материалоемкость, максимальный полезный объем (или грузоподъемность). Некоторые целевые функции могут оказаться несовместимыми. В таких случаях необходимо вводить приоритет той или иной целевой функции.

1.2 Задачи оптимизации. Можно выделить два типа задач оптимизации — безусловные и условные. *Безусловная задача* оптимизации состоит в отыскании максимума или минимума действительной функции (9.1 от n действительных переменных и определении соответствующих значений аргументов на некотором множестве σ n-мерного пространства. Обычно рассматриваются задачи минимизации; к ним легко сводятся и задачи на поиск максимума путем замены знака целевой функции на противоположный.

Условные задачи оптимизации, или *задачи с ограничениями,*— это такие, при формулировке которых задаются некоторые условия (ограничения) на множестве σ. Это ограничения задаются совокупностью некоторых функций, удовлетворяющих уравнениям или неравенствам.

Ограничения-равенства выражают зависимость между проектными параметрами, которая должна учитываться при нахождении решения. Эти ограничения отражают законы природы, наличие ресурсов, финансовые требования и т. п.

В результате ограничений область проектирования σ, определяемая всеми n проектными параметрами, может быть существенно уменьшена в соответствии с физической сущностью задачи. Число m ограничений-равенств может быть произвольным. Их можно записать в виде

$$
\begin{aligned}
&g_1(x_1, x_2, ..., x_n) = 0,\\
&g_2(x_1, x_2, ..., x_n) = 0,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&g_n(x_1, x_2, ..., x_n) = 0.
\end{aligned}
\tag{9.2}
$$

В ряде случаев из этих соотношений можно выразить одни проектные параметры через другие. Это позволяет исключить некоторые параметры из процесса оптимизации, что приводит к уменьшению размерности задачи и облегчает ее решение. Аналогично могут вводиться также *ограничения-неравенства,* имеющие вид

$$
\begin{aligned}
&a_1 \le \varphi_1(x_1, x_2, ..., x_n) \le b_1,\\
&a_2 \le \varphi_2(x_1, x_2, ..., x_n) \le b_2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_k \le \varphi_k(x_1, x_2, ..., x_n) \le b_k.
\end{aligned}
\tag{9.3}
$$

Следует отметить особенность в отыскании решения при наличии ограничений. Оптимальное решение здесь может соответствовать либо локальному экстремуму (максимуму или минимуму) внутри области проектирования, либо значению целевой функции на границе области. Если же ограничения отсутствуют, то ищется оптимальное решение на всей области проектирования, т. е. глобальный экстремум.

Теория и методы решения задач оптимизации при наличии ограничений составляют предмет исследования одного из важных разделов прикладной математики — *математического программирования.*

1.3 Пример постановки задачи. Пусть требуется спроектировать контейнер в форме прямоугольного параллелепипеда объемом $V = 1$ м^3, причем желательно израсходовать на его изготовление как можно меньше материала.

При постоянной толщине стенок последнее условие означает, что площадь полной поверхности контейнера S должна быть минимальной. Если обозначить через x_1 x_2, x_3 длины ребер контейнера, то задача сведется к минимизации функции

$$S = 2(x_1x_2 + x_2x_3 + x_1x_3). \tag{9.4}$$

Эта функция в данном случае является целевой, а условие V= 1 — ограничением-равенством, которое позволяет исключить один параметр:

$$
\begin{aligned}
&V = x_1x_2x_3 = 1, \quad x_3 = \frac{1}{x_1x_2},\\
&S = 2(x_1x_2 + \frac{1}{x_1} + \frac{1}{x_2}).
\end{aligned}
\tag{9.5}
$$

Задача свелась к минимизации функции двух переменных. В результате решения задачи будут найдены значения проектных параметров x_1, x_2, а затем и x_3. В приведенном примере фактически получилась задача безусловной оптимизации для целевой функции (9.5), поскольку ограничение-равенство было использовано для исключения параметра x_3.

Вместе с тем можно рассматриваемую задачу, усложнить и поставить дополнительные условия. Например, потребуем, чтобы данный контейнер имел длину не менее 2 м. Это условие запишется в виде ограничения-неравенства на один из параметров, например

$$x_1 \ge 2. \tag{9.6}$$

Таким образом, мы получили следующую условную задачу оптимизации: минимизируя функцию (9.5) и учитывая ограничение-неравенство (9.6), найти оптимальные значения параметров плана x_1, x_2 ($x_1 > 0$, $x_2 > 0$).

§ 2. Одномерная оптимизация. Задачи на экстремум. Методы поиска. Метод золотого сечения

2.1 Задачи на экстремум. *Одномерная задача оптимизации* в общем случае формулируется следующим образом. Найти наименьшее (или наибольшее) значение целевой функции $y = f(x)$, заданной на множестве σ, и определить значение проектного параметра $x \in \sigma$, при котором целевая функция принимает экстремальное значение. Существование решения поставленной задачи вытекает из следующей теоремы.

Теорема Вейерштрасса. *Всякая функция f(x), непрерывная на отрезке* [*a, b*], *принимает на этом отрезке наименьшее и наибольшее значения, т. е. на отрезке* [*a, b*] *существуют такие точки* x_1 *и* x_2, *что для любого* $x \in$ [*a, b*] *имеют место неравенства*

$$f(x_1) \le f(x) \le f(x_2).$$

Эта теорема не доказывает единственности решения. Не исключена возможность, когда равные экстремальные значения достигаются сразу в нескольких точках данного отрезка. В частности, такая ситуация

имеет место для периодической функции, рассматриваемой на отрезке, содержащем несколько периодов.
Будем рассматривать методы оптимизации для разных классов целевых функций. Простейшим из них является случай дифференцируемой функции *f(x)* на отрезке [*a,b*], причем функция задана в виде аналитической зависимости $y = f(x)$, и может быть найдено явное выражение для: ее производной $f'(x)$. Нахождение экстремумов таких функций можно проводить известными из курса высшей математики методами дифференциального исчисления. Напомним вкратце этот путь.
Функция *f(x)* может достигать своего наименьшего и наибольшего значений либо в граничных точках отрезка [*a, b*], либо в точках минимума и максимума. Последние точки обязательно должны быть критическими, т. е. производная *f(x)* в этих точках обращается в нуль — это необходимое условие экстремума. Следовательно, для определения наименьшего или наибольшего значений функции *f(x)* на отрезке [*a, b*] нужно вычислить ее значения во всех критических точках данного отрезка и его граничных точках и сравнить полученные значения; наименьшее или наибольшее из них и будет искомым значением.
Пример. Найти наименьшее и наибольшее значения функции $f(x) = x^3/3 - x^2$ на отрезке [1, 3].
Решение. Вычислим производную этой функции:

$$f'(x) = x^2 - 2x.$$

Приравнивая ее нулю, найдем критические точки:

$$x^2 - 2x = 0,\ x_1 = 0,\ x_2 = 2.$$

Точка $x = 0$ лежит вне рассматриваемого отрезка, поэтому для анализа оставляем три точки: $a = 1$, $x_2 = 2$, $b=3$.
Вычисляем значения функции в этих точках:

$$f(1) = -2/3,\quad f(2) = -4/3,\quad f(3) = 0.$$

Сравнивая полученные величины, находим, что наименьшего значения функция *f(x)* достигает в точке $x = 2$, наибольшего — в точке $x = 3$, т. е.

$$f_{min} = f(2) = -4/3,\quad f_{max} = f(3) = 0.$$

В рассмотренном примере уравнение $f'(x) = 0$ для отыскания критических точек удалось решить непосредственно. Для более сложных видов производной функции *f'(x)* необходимо использовать численные методы решения нелинейных уравнений.
Как уже отмечалось, используемый здесь метод, основанный на вычислении производной целевой функции, требует ее аналитического представления. В других случаях, когда целевая функция задана в табличном виде или может быть вычислена при некоторых дискретных значениях аргумента, используются различные *методы поиска.* Они основаны на вычислении целевой функции в отдельных точках и выборе среди них наибольшего или наименьшего значений. Существует ряд алгоритмов решения данной задачи. Рассмотрим некоторые из них.

2.2 Методы поиска. Численные методы поиска экстремальных значений функции рассмотрим на примере нахождения минимума функции *f(x)* на отрезке [*a, b*]. Будем предполагать, что целевая функция *унимодальна,* т. е. на данном отрезке она имеет только один минимум. Отметим, что в инженерной практике обычно встречаются именно такие целевые функции.
Процесс решения задачи методом поиска состоит в последовательном сужении интервала изменения проектного параметра, называемого *интервалом неопределенности.* В начале процесса оптимизации его длина равна $b - a$, а к концу она должна стать менее заданного допустимого значения ε, т. е. оптимальное значение проектного параметра должно находиться в интервале неопределённости - отрезке $[x_n, x_{n+1}]$, причем $x_{n+1} - x_n < \varepsilon$.
Наиболее простым способом сужения интервала неопределенности является деление его на некоторое число равных частей с последующим вычислением значений целевой функции, в точках разбиения. Пусть n — число элементарных отрезков, $h = (b — a)/n$ — шаг разбиения. Вычислим значения целевой функции $y_k — f_k(x)$ в узлах $x_k = a + kh$ ($k = 0, 1, ..., n$). Сравнивая полученные значения $f(x_k)$, найдем среди них наименьшее $y_i=f(x_i)$.
Число $m_n = y_i$ можно приближенно принять за наименьшее значение целевой функций *f(x)* на отрезке [*a, b*]. Очевидно, что близость m_n к минимуму m зависит от числа точек, и для непрерывной функции *f(x)*

$$\lim_{n\to\infty} m_n = m,$$

т. е. с увеличением числа точек разбиения погрешность в определении минимума стремится к нулю.

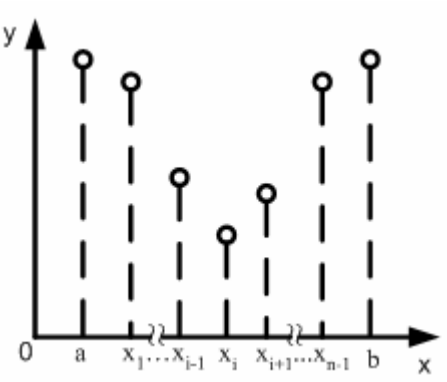


Рисунок 9.1

В данном методе, который можно назвать *методом перебора,* основная трудность состоит в выборе n и оценке погрешности. Можно, например, провести оптимизацию с разными шагами и исследовать сходимость такого итерационного процесса. Но это трудоемкий путь.

Более экономичным способом уточнения оптимального параметра является использование свойства унимодальности целевой функции, которое позволяет построить процесс сужения интервала неопределенности.

Пусть, как и ранее, среди всех значений унимодальной функции $y = f(x)$, вычисленных в узлах x_k ($k = 0, 1,...,n$), наименьшим оказалось y_i. Это означает, что оптимальное значение проектного параметра находится на отрезке $[x_{i-1}, x_{i+1}]$ (Рисунок 9.1), т. е. интервал неопределенности сузился до длины двух шагов. Если размер интервала недостаточен для удовлетворения заданной погрешности, т. е. $x_{i+1} - x_{i-1} > \varepsilon$, то его снова можно уменьшить путем нового разбиения. Получится интервал, равный двум длинам нового шага разбиения, и т. д. Процесс оптимизации продолжается до достижения заданного размера интервала неопределенности. В описанном методе общего поиска можно с помощью некоторой изобретательности, а также разумного выбора шага разбиения добиться эффективного поиска.

Например, пусть начальная длина интервала неопределенности равна $b - a = 1$. Нужно добиться его уменьшения в 100 раз. Этого легко достичь разбиением интервала на 200 частей. Вычислив значения целевой функции $f(x_k)$ ($k = 0, 1, ..., 200$), найдем ее минимальное значение $f(x_i)$. Тогда искомым интервалом неопределенности будет отрезок $[x_{i-1}, x_{i+1}]$.

Однако можно поступить и иначе. Сначала разобьем отрезок $[a, b]$ на 20 частей и найдем интервал неопределенности длиной 0.1, при этом мы вычислим значения целевой функции в точках $x_k = a + 0.05k$ (k=0,1,…,200). Теперь отрезок $[x_{i-1}, x_{i+1}]$ снова разобьём на 20 частей; получим искомый интервал длиной 0.01, причем значения целевой функции вычисляем в точках $x_k = x_{i-1} + 0.05k$ ($k = 0, 1, …, 19$) (в точках x_{i-1} и x_{i+1} значения $f(x)$ уже найдены). Таким образом, во втором случае в процессе оптимизации произведено 40 вычислений значений целевой функции против 201 в первом случае, т. е. способ разбиения позволяет получить существенную экономию вычислений.

Существует ряд специальных методов поиска оптимальных решений с разными способами выбора узлов и сужения интервала неопределенности: метод деления отрезка пополам, метод золотого сечения и др.

Рассмотрим один из них.

2.3 Метод золотого сечения. При построении процесса оптимизации стараются сократить объем вычислений и время поиска. Этого достигают обычно путем сокращения количества вычислений (или измерений — при проведении эксперимента) значений целевой функции $f(x)$. Одним из наиболее эффективных методов, в которых при ограниченном количестве вычислений $f(x)$ достигается наилучшая точность, является *метод золотого сечения.* Он состоит в построении последовательности отрезков $[a_0, b_0]$, $[a_1, b_1]$, ..., стягивающихся к точке минимума функции $f(x)$. На каждом шаге, за исключением первого, вычисление значения функции $f(x)$ проводится лишь один раз. Эта точка, называемая *золотым сечением,* выбирается специальным образом.

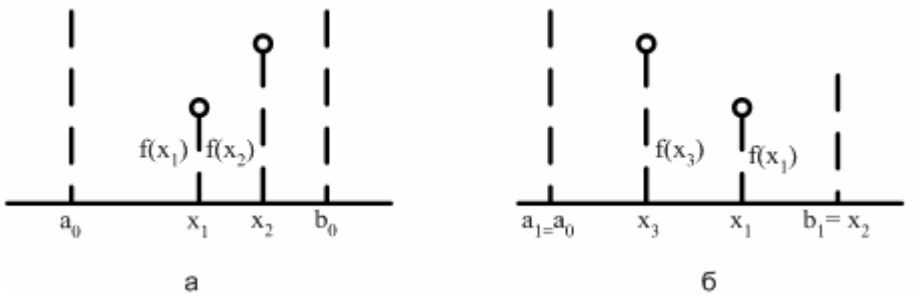


Рисунок 9.2

Поясним сначала идею метода геометрически, а затем выведем необходимые соотношения. На первом шаге процесса оптимизации внутри отрезка $[a_0, b_0]$ (Рисунок 9.2, а) выбираем две внутренние точки x_1 и x_2 и вычисляем значения целевой функции $f(x_1)$ и $f(x_2)$. Поскольку в данном случае $f(x_1)<f(x_2)$, очевидно, что минимум расположен на одном из прилегающих к x_1 отрезков $[a_0, x_1]$ или $[x_1, x_2]$. Поэтому отрезок $[x_2, b_0]$ можно отбросить, сузив тем самым первоначальный интервал неопределенности.

Второй шаг проводим на отрезке $[a_1, b_0]$ (Рисунок 9.2, б), где $a_1 = a_0$, $b_1 = x_2$. Нужно снова выбрать две внутренние точки, но одна из них (x_1) осталась из предыдущего шага, поэтому достаточно выбрать лишь одну точку x_3, вычислить значение $f(x_3)$ и провести сравнение. Поскольку здесь $f(x_3)>f(x_1)$, ясно, что минимум находится на отрезке $[x_3, b_1]$. Обозначим этот отрезок $[a_2, b_2]$, снова выберем одну внутреннюю точку и повторим процедуру сужения интервала неопределенности. Процесс оптимизации повторяется до тех пор, пока длина очередного отрезка $[a_n, b_n]$ не станет меньше заданной величины ε.

Теперь рассмотрим способ размещения внутренних точек на каждом отрезке $[a_k, b_k]$. Пусть длина интервала неопределенности равна l, а точка деления делит его на части l_1, l_2: $l_1>l_2$, $l = l_1+l_2$. *Золотое сечение* интервала неопределенности выбирается так, чтобы отношение длины большего отрезка к длине всего интервала равнялось отношению длины меньшего отрезка к длине большего отрезка:

$$\frac{l_1}{l} = \frac{l_2}{l_1}. \tag{9.7}$$

Из этого соотношения можно найти точку деления, определив отношение l_2/l_1. Преобразуем выражение (9.7) и найдем это значение:

$$l_1^2 = l_2 l, \quad l_1^2 = l_2(l_1 + l_2),$$
$$l_2^2 + l_1 l_2 - l_1^2 = 0,$$
$$\left(\frac{l_2}{l_1}\right)^2 + \frac{l_2}{l_1} - 1 = 0,$$
$$\frac{l_2}{l_1} = \frac{-1 \pm \sqrt{5}}{2}.$$

Поскольку нас интересует только положительное решение, то

$$\frac{l_2}{l_1} = \frac{l_1}{l} = \frac{-1+\sqrt{5}}{2} \approx 0.618.$$

Отсюда $l_1 \approx 0.618l, l_2 \approx 0.382l.$

Поскольку заранее неизвестно, в какой последовательности (l_1 и l_2 или l_2 и l_1) делить интервал неопределенности, то рассматривают внутренние точки, соответствующие двум этим способам деления. На Рисунок 9.2, *а* точки деления x_1 и x_2 выбираются с учетом полученных значений для частей отрезка. В данном случае имеем

$$x_1 - a_0 = b_0 - x_2 = 0.382d_0,$$
$$b_0 - x_1 = x_2 - a_0 = 0.618d_0,$$
$$d_0 = b_0 - a_0.$$

После первого шага оптимизации получается новый интервал неопределенности — отрезок $[a_1, b_2]$ (см. Рисунок 9.2, *б)*. Можно показать, что точка x_1 делит этот отрезок в требуемом отношении, при этом

$$b_1 - x_1 = 0.382d_1, \quad d_1 = b_1 - a_1.$$

Для этого проведем очевидные преобразования:

$$b_1 - x_1 = x_2 - x_1 = (b_0 - a_0) - (x_1 - a_0) - (b_0 - x_2) = d_0 - 0.382d_0 - 0.382d_0 = 0.236d_0,$$
$$d_1 = x_2 - a_0 = 0.618d_1,$$
$$b_1 - x_1 = 0.236(d_1 / 0.618) = 0.382d_1.$$

Вторая точка деления x_3 выбирается на таком же расстоянии от левой границы отрезка, т. е. $x_3 - a_1 = 0.382d_1$. И снова интервал неопределенности уменьшается до размера

$$d_2 = b_2 - a_2 = b_1 - x_3 = 0.618d_1 = 0.618^2 d_0.$$

Используя полученные соотношения, можно записать координаты точек деления y и z отрезка $[a_k, b_k]$ на $(k+1)$–м шаге оптимизации ($y < z$):

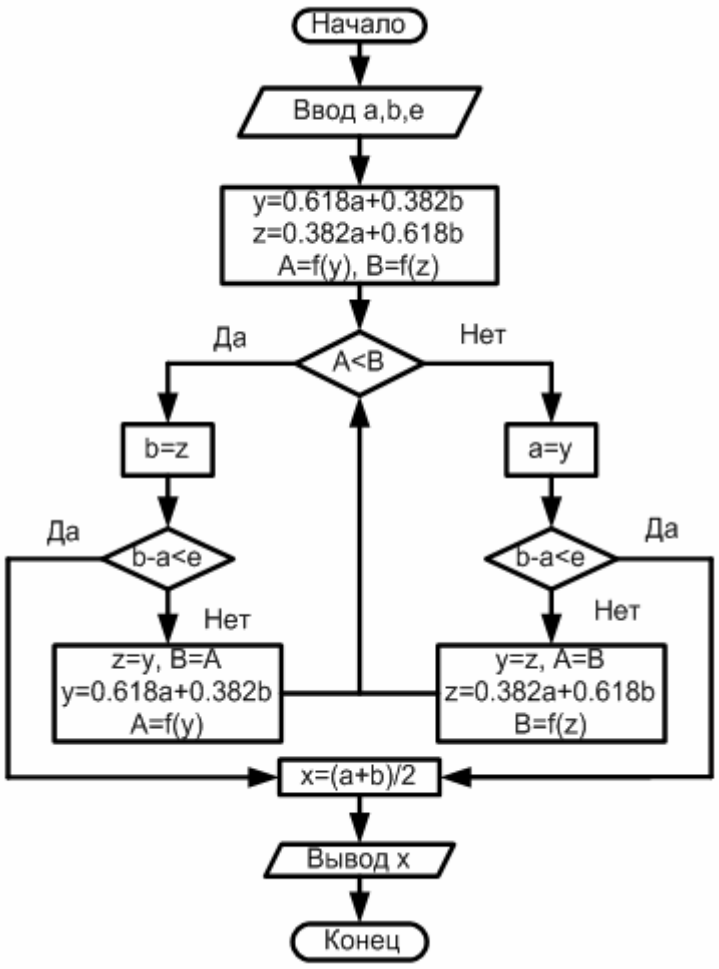
$$y = 0.618a_k + 0.382b_k,$$
$$z = 0.382a_k + 0.618b_k. \quad (9.8)$$

При этом длина интервала неопределенности равна

$$d_k = b_k - a_k = 0.618^k d_0. \quad (9.9)$$

Процесс оптимизации заканчивается при выполнении условия $d_k < \varepsilon$. При этом проектный параметр оптимизации составляет $a_k < x < b_k$. Можно в качестве оптимального значения принять $x = a_k$ (или $x = b_k$, или $x = (a_k + b_k)/2$ и т. п.).

На !!!Рисунок 9.3 Блок-схема метода золотого сечения представлена блок-схема процесса одномерной оптимизации методом золотого сечения. Здесь *y, z* — точки деления отрезка [*a, b*], причем $y < z$. В результате выполнения алгоритма выдается оптимальное значение



!!!Рисунок 9.3 Блок-схема метода золотого сечения

проектного параметра *x,* в качестве которого принимается середина последнего интервала неопределенности.

Пример. Для оценки сопротивления дороги движению автомобиля при скорости *v* км/ч можно использовать эмпирическую формулу $f(v) = 24 - \frac{2}{3}v + \frac{1}{30}v^2$ (для шоссе). Определить скорость, при которой сопротивление будет минимальным.

Решение. Это простейшая задача одномерной оптимизации. Здесь сопротивление *f(v)* — целевая функция, скорость *v* — проектный параметр. Данную задачу легко решить путём нахождения минимума с помощью вычисления производной, поскольку *f(v)* —функция дифференцируемая. Действительно,

$$f'(v) = -\frac{2}{3} + \frac{2v}{30} = 0, \quad v = 10 \text{ км/ч}.$$

Проиллюстрируем на этой простейшей задаче метод золотого сечения. Первоначально границы интервала неопределенности примем равными $a = 5$, $b = 20$. Результаты вычислений представим в виде таблицы (Таблица 9.1). Здесь обозначения аналогичны используемым в блок-схеме (см. !!!Рисунок 9.3 Блок-схема метода золотого сечения). Расчеты проводятся в соответствии с блок-схемой с погрешностью $\varepsilon = 1$ км/ч.

Таблица 9.1

Шаг	*a*	*y*	*z*	*b*	*A*	*B*	*b-a*
1	5	10.7	14.3	20	20.7	21.3	15
2	5	8.6	10.7	14.3	20.73	20.68	9.3
3	8.6	10.7	12.1	14.3	20.68	20.81	5.7
4	8.6	9.9	10.7	12.1	20.66	20.68	3.5
5	8.6	9.4	9.9	10.7	20.68	20.66	2.1
6	9.4			10.7			1.3

Приведем решение для первого этапа:

$$y = 0.618 * 5 + 0.382 * 20 \approx 10.7,$$

$$z = 0.382 * 5 + 0.618 * 20 \approx 14.3,$$

$$A = 24 - \frac{2}{3} * 10.7 + \frac{1}{30} * 10.7^2 \approx 20.7,$$

$$B = 24 - \frac{2}{3} * 14.3 + \frac{1}{30} * 14.3^2 \approx 21.3,$$

$$A < B.$$

При данной невысокой точности вычислений достаточно четырех шагов оптимизации. В этом случае искомое значение скорости равно y =(8.6 + 10.7)/2 = 9.65 км/ч. После пяти шагов этот результат получается с меньшей погрешностью:

$$v = (9.4 + 10.7)/2 = 10.05 \text{ км/ч}.$$

§ 3. Многомерная оптимизация. Минимум функции нескольких переменных. Метод покоординатного спуска. Метод градиентного спуска.

3.1 Минимум функции нескольких переменных. В § 2 мы рассмотрели одномерные задачи оптимизации, в которых целевая функция зависит лишь от одного аргумента. Однако в большинстве реальных задач оптимизации, представляющих практический интерес, целевая функция зависит от многих проектных параметров. В частности, рассмотренная выше задача об определении сопротивления дороги движению автомобиля на самом деле является многомерной, поскольку здесь наряду со скоростью имеются и другие проектные параметры (качество покрытия, уклон, температура и др.).

Минимум дифференцируемой функции многих переменных $u = f(x_1, x_2, ..., x_n)$ можно найти, исследуя ее значения в критических точках, которые определяются из решения системы дифференциальных уравнений

$$\frac{\partial f}{\partial x_1} = 0, \quad \frac{\partial f}{\partial x_2} = 0, ..., \frac{\partial f}{\partial x_n} = 0. \tag{9.10}$$

ПРИМЕР. Ранее была рассмотрена задача об определении оптимальных размеров контейнера объемом 1 м^3. Задача свелась к минимизации его полной поверхности, которая в данном случае является целевой функцией

$$S = 2\left(x_1 x_2 + \frac{1}{x_1} + \frac{1}{x_2} \right). \tag{9.11}$$

Решение. В соответствии с 9.10 получим систему

$$\frac{\partial S}{\partial x_1} = 2\left(x_2 - \frac{1}{x_1^2} \right) = 0,$$

$$\frac{\partial S}{\partial x_2} = 2\left(x_1 - \frac{1}{x_2^2} \right) = 0.$$

Отсюда находим $x_1 = x_2 = 1$ м, $x_3 = 1/(x_1 x_2) = 1$ м. Таким образом, оптимальной формой контейнера в данном случае является куб, длина ребра которого равна 1 м.

Рассмотренный метод можно использовать лишь для дифференцируемой целевой функции. Но и в этом случае могут возникнуть серьезные трудности при решении системы нелинейных уравнений (9.10).

Во многих случаях никакой формулы для целевой функции нет, а имеется лишь возможность определения ее значений в произвольных точках рассматриваемой области с помощью некоторого вычислительного алгоритма или путем физических измерений. Задача состоит в приближенном определении наименьшего значения функции во всей области при известных ее значениях в отдельных точках.

Для решения подобной задачи в области проектирования G, в которой ищется минимум целевой функции u =

$f(x_1, x_2, ..., x_n)$, можно ввести дискретное множество точек (узлов) путем разбиения интервалов изменения параметров $x_1, x_2, ..., x_n$ на части с шагами $h_1, h_2, ..., h_n$. В полученных узлах можно вычислить значения целевой функции и среди этих значений найти наименьшее.

Следует отметить, что такой метод может быть использован для функции одной переменной. В многомерных задачах оптимизации, где число проектных параметров достигает пяти и более, этот метод потребовал бы слишком большого объема вычислений.

Оценим, например, объем вычислений с помощью общего поиска при решении задачи оптимизации функции пяти неизвестных. Пусть вычисление ее значения в одной точке требует 100 арифметических операций (на практике это число может достигать нескольких тысяч и больше). Область проектирования разделим на 100 частей в каждом из пяти направлений, т. е. число расчетных точек равно 101^5, т. е. приблизительно 10^{10}. Число арифметических операций тогда равно 10^{12}, и для решения этой задачи на ЭВМ с быстродействием 1 млн. оп./с потребуется 10^6 с (более 10 сут) машинного времени.

Проведенная оценка показывает, что такие методы общего поиска, **с** использованием сплошного перебора для решения многомерных задач оптимизации не годятся. Необходимы специальные численные методы, основанные на целенаправленном поиске. Рассмотрим некоторые из них.

3.2 Метод покоординатного спуска. Пусть требуется найти наименьшее значение целевой функции $u = f(x_1, x_2, ..., x_n)$. В качестве начального приближения выберем в n-мерном пространстве некоторую точку M_0 с координатами $x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}$. Зафиксируем все координаты функции u, кроме первой. Тогда $u = f(x_1, x_2^{(0)}, ..., x_n^{(0)})$ — функция одной переменной x_1. Решая одномерную задачу оптимизации для этой функции, мы от точки M_0 переходим к точке $M_1 (x_1^{(1)}, x_2^{(0)}, ..., x_n^{(0)})$, в которой функция u принимает наименьшее значение по координате x_1 при фиксированных остальных координатах. В этом состоит первый шаг процесса оптимизации, состоящий в спуске по координате x_1.

Зафиксируем теперь все координаты, кроме x_2, и рассмотрим функцию этой переменной $u=f(x_1^{(1)}, x_2^{(0)}, x_3^{(0)}, ..., x_n^{(0)})$.

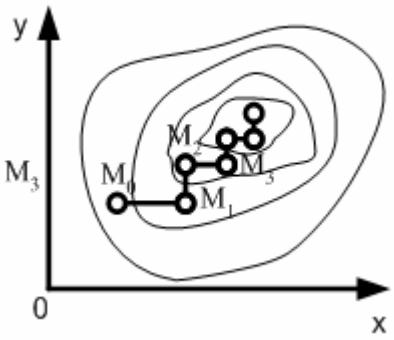


Рисунок 9.4 Спуск по координатам

Снова решая одномерную задачу оптимизации, находим ее наименьшее значение при $x_2 = x_2^{(1)}$, т. е. в точке $M_2 (x_1^{(1)}, x_2^{(1)}, x_3^{(0)}, ..., x_n^{(0)})$. Аналогично проводится спуск по координатам $x_3, x_4, ..., x_n$, а затем процедура снова повторяется от x_1 до x_n и т. д. В результате этого процесса получается последовательность точек $M_0, M_1, ...$, в которых значения целевой функции составляют монотонно убывающую последовательность $f(M_0) \geq f(M_1) \geq ...$

На любом k-м шаге этот процесс можно прервать, и значение $f(M_k)$ принимается в качестве наименьшего значения целевой функции в рассматриваемой области.

Таким образом, метод покоординатного спуска сводит задачу о нахождении наименьшего значения функции многих переменных к многократному решению одномерных задач оптимизации по каждому проектному параметру.

Данный метод легко проиллюстрировать геометрически для случая функции двух переменных $z = f(x, y)$, описывающей некоторую поверхность в трехмерном пространстве. На рисунке 9.4 нанесены линии уровня этой поверхности. Процесс оптимизации в этом случае проходит следующим образом. Точка $M_0(x_0, y_0)$ описывает начальное приближение. Проводя спуск по координате x, попадем в точку $M_1(x_1, y_0)$. Далее, двигаясь параллельно оси ординат, придем в точку $M_2(x_1, y_1)$ и т. д.

Важным здесь является вопрос о сходимости рассматриваемого процесса оптимизации. Другими словами, будет ли последовательность значений целевой функции $f(M_0), f(M_1), ...$ сходиться к наименьшему ее значению в данной области? Это зависит от вида самой функции и выбора начального приближения.

Для функции двух переменных, очевидно, что метод неприменим в случае наличия изломов в линиях уровня. Это соответствует так называемому *оврагу* на поверхности. Здесь возможен случай, когда спуск по одной координате приводит на «дно» оврага. Тогда любое движение вдоль другой координаты ведет к возрастанию функции, соответствующему подъему на «берег» оврага. Поскольку поверхности типа «оврага» встречаются в инженерной практике, то при использовании метода покоординатного спуска следует убедиться, что решаемая задача не имеет этого недостатка. Для гладких функций при удачно выбранном начальном приближении (в некоторой окрестности минимума) процесс сходится к минимуму. К достоинствам метода

покоординатного спуска следует также отнести возможность использования простых алгоритмов одномерной оптимизации. Блок-схема метода покоординатного спуска представлена на Рисунке 9.5

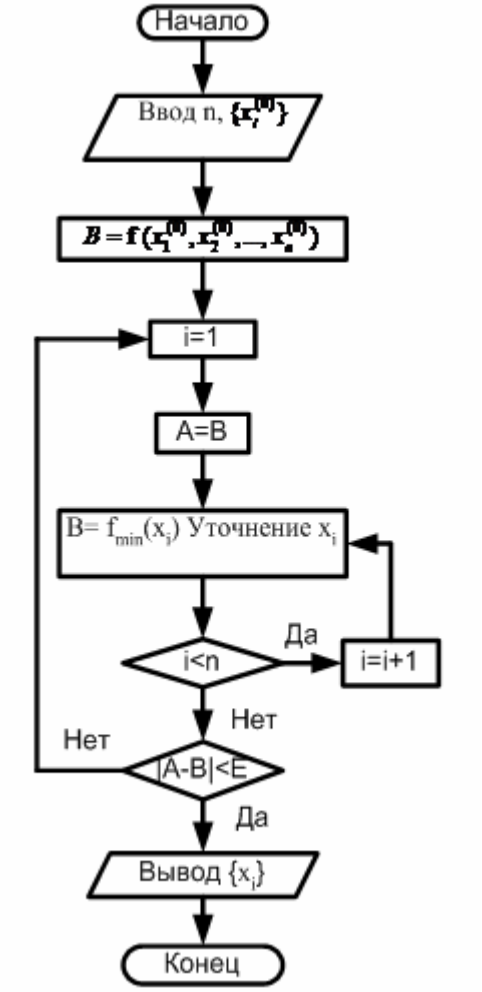


Рисунок 9.5 Блок-схема метода покоординатного спуска

3.3 Метод градиентного спуска. В природе мы нередко наблюдаем явления, сходные с решением задачи на нахождение минимума. К ним относится, в частности, стекание воды с берега котлована на дно. Упростим ситуацию, считая, что берега котлована «унимодальны», т. е. они гладкие и не содержат локальных углублений или выступов. Тогда вода устремится вниз в направлении наибольшей крутизны берега в каждой точке.

Переходя на математический язык, заключаем, что направление наискорейшего спуска соответствует направлению наибольшего убывания функции. Из курса математики известно, что направление наибольшего возрастания функции двух переменных *u = f(x, y)* характеризуется ее *градиентом*

$$grad\ u = \frac{\partial u}{\partial x} e_1 + \frac{\partial u}{\partial y} e_2,$$

где e_1, e_2 — единичные векторы (орты) в направлении координатных осей. Следовательно, направление, противоположное градиентному, укажет путь, ведущий вниз вдоль наиболее крутой линии. Методы, основанные на выборе пути оптимизации с помощью градиента, называются *градиентными.*

Идея метода градиентного спуска состоит в следующем. Выбираем некоторую начальную точку и вычисляем в ней градиент рассматриваемой функции. Делаем шаг в направлении, обратном градиентному. В результате приходим в точку, значение функции в которой обычно меньше первоначального. Если это условие не выполнено, т. е. значение функции не изменилось либо даже возросло, то нужно уменьшить шаг. В новой точке процедуру повторяем: вычисляем градиент и снова делаем шаг в обратном к нему направлении. Процесс продолжается до получения наименьшего значения целевой функции. Момент окончания поиска наступит тогда, когда движение из полученной точки с любым шагом приводит к возрастанию значения целевой функции. Строго говоря, если минимум функции достигается внутри рассматриваемой области, то в этой точке градиент равен нулю, что также может служить сигналом об окончании процесса оптимизации.

В описанном методе требуется вычислять на каждом шаге оптимизации градиент целевой функции $f(x_1, x_2, ..., x_n)$:

$$grad\ f = \left\{ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right\}.$$

Формулы для частных производных можно получить в явном виде лишь в том случае, когда целевая функция задана аналитически. В противном случае эти производные вычисляются с помощью численного дифференцирования:

$$\frac{\partial f}{\partial x_i} \approx \frac{1}{\Delta x_i}\left[f\left(x_1, ..., x_i + \Delta x_i, ..., x_n\right) - f\left(x_1, ..., x_i, ..., x_n\right) \right], \quad i = 1, 2, ..., \text{n}.$$

При использовании градиентного спуска в задачах оптимизации основной объем вычислений приходится обычно на вычисление градиента целевой функции в каждой точке траектории спуска. Поэтому целесообразно уменьшить количество таких точек без ущерба для самого решения. Это достигается в некоторых методах, являющихся модификациями градиентного спуска. Одним из них является *метод наискорейшего спуска.* Согласно этому методу, после определения в начальной точке направления, противоположного градиенту целевой функции, в этом направлении делают не один шаг, а двигаются до тех пор, пока целевая функция убывает, достигая таким образом минимума в некоторой точке. В этой точке снова определяют направление спуска (с помощью градиента) и ищут новую точку минимума целевой функции и т. д. В этом методе спуск происходит гораздо более крупными шагами и градиент функции вычисляется в меньшем числе точек.

Заметим, что метод наискорейшего спуска сводит многомерную задачу оптимизации к последовательности одномерных задач на каждом шаге оптимизации, как и в случае покоординатного спуска. Разница состоит в том, что здесь направление одномерной оптимизации определяется градиентом целевой функции, тогда как покоординатный спуск проводится на каждом шаге вдоль одного из координатных направлений.

§ 4. Задачи с ограничением. Метод штрафных функций. Линейное программирование. Геометрический метод. Симплекс метод.

4.1 Метод штрафных функций. Решение задач математического программирования значительно более трудоемко по сравнению с задачами безусловной оптимизации. Ограничения типа равенств или неравенств требуют их учета на каждом шаге оптимизации. Одним из направлений в методах решения задач математического программирования является сведение их к последовательности задач безусловной минимизации. К этому направлению относится, в частности, *метод штрафных функций.*

Сущность метода состоит в следующем. Пусть $j(x_1, x_2, ..., x_n)$ — целевая функция, для которой нужно найти минимум m в ограниченной области $D\ (x_1, x_2, ..., x_n \in D)$. Данную задачу заменяем задачей о безусловной минимизации однопараметрического семейства функций

$$F(x, \beta) = f(x) + \frac{1}{\beta}\varphi(x), \quad x = \{x_1, x_2, ..., x_n\}. \tag{9.12}$$

При этом дополнительную *(штрафную)* функцию $\varphi(x)$ выберем таким образом, чтобы при $\beta \to 0$ решение вспомогательной задачи стремилось к решению исходной или, по крайней мере, чтобы их минимумы совпадали: $\min F(x, \beta) \to m$ при $\beta \to 0$.

Штрафная функция $\varphi(x)$ должна учитывать ограничения, которые задаются при постановке задачи оптимизации. В частности, если имеются ограничения-неравенства вида $g_j(x_1, x_2, ..., x_n) > 0$ (j=1, 2, ...,J), то в качестве штрафной можно взять функцию, которая: 1) равна нулю во всех точках пространства проектирования, удовлетворяющих заданным ограничениям-неравенствам; 2) стремится к бесконечности в тех точках, в которых эти неравенства не выполняются. Таким образом, при выполнении ограничений-неравенств функции $f(x)$ и $F(x, \beta)$ имеют один и тот же минимум. Если хотя бы одно неравенство не выполнится, то вспомогательная целевая функция $F(x, \beta)$ получает бесконечно большие добавки, и ее значения далеки от минимума функции $f(x)$. Другими словами, при несоблюдении ограничений-неравенств налагается «штраф». Отсюда и термин «метод штрафных функций».

Теперь рассмотрим случай, когда в задаче оптимизации заданы ограничения двух типов — равенства и неравенства:

$$\begin{aligned} &g_i(x) = 0, \quad i = 1, 2, ..., I; \\ &h_j(x) \ge 0, \quad j = 1, 2, ..., J; \quad x = \{x_1, x_2, ..., x_n\}. \end{aligned} \tag{9.13}$$

В этом случае в качестве вспомогательной целевой функции, для которой формулируется задача безусловной оптимизации во всем n-мерном пространстве, принимают функцию

$$F(x, \beta) = f(x) + \frac{1}{\beta}\left\{ \sum_{i=1}^{I} g_i^2(x) + \sum_{j=1}^{J} h_i^2(x)\left[1 - sign\ h_j(x)\right], \right. \tag{9.14}$$

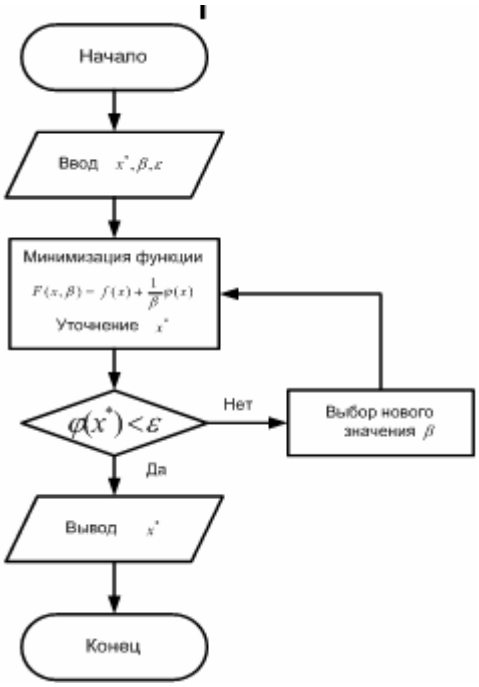
$$\beta > 0.$$

Здесь взята такая штрафная функция, что при выполнении условий 9.13 она обращается в нуль. Если же эти условия нарушены (т.е. $g_i(x) \neq 0, h_j(\mathrm{x}) < 0$ и $\operatorname{sign} \mathrm{h}_\mathrm{j}(x) = -1$), то штрафная функция положительна. Она увеличивает целевую функцию *f(x)* тем больше, чем больше нарушаются условия (9.13).

При малых значениях параметра β вне области *D* функция *F(x, β)* сильно возрастает. Поэтому ее минимум может быть либо внутри *D,* либо снаружи вблизи границ этой области. В первом случае минимумы функций *F(x, β)* и *f(x)* совпадают, поскольку дополнительные члены в (9.14) равны нулю. Если минимум функции *F(x, β)* находится вне *D,* то минимум целевой функции *f(x)* лежит на границе *D.* Можно при этом построить последовательность $\beta_k \to 0$ такую, что соответствующая последовательность минимумов функции *F (x, β)* будет стремиться к минимуму функции *f(x).*

Таким образом, задача оптимизации для целевой функции *f(x)* с ограничениями 9.13 свелась к последовательности задач безусловной оптимизации для вспомогательной функции 9.14, решение которых может быть проведено с помощью методов спуска. При этом строится итерационный процесс при $\beta \to 0$.

Укрупненная блок-схема решения задачи математического программирования с использованием метода штрафных функций представлена на !!!!Рисунок 9.6 Блок-схема метода штрафных функций. В качестве исходных данных вводятся начальное приближение искомого вектора $x^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$, начальное значение параметра β и некоторое малое число ε, характеризующее точность расчета. На каждом шаге итерационного процесса определяется оптимальное значение x^* вектора *x,* при этом в качестве начального приближения принимается результат предыдущей итерации. Значения параметра β каждый раз уменьшаются до тех пор, пока значение штрафной функции не станет заданной малой величиной. В этом случае точка x^* достаточно близка к границе области D и с необходимой точностью описывает оптимальные значения проектных параметров. Если точка минимума находится внутри области *D,* то искомый результат будет получен сразу после первого шага, поскольку в данном случае $\varphi(x^*) = 0$.



!!!!Рисунок 9.6 Блок-схема метода штрафных функций

4.2 Линейное программирование. До сих пор при рассмотрении задач оптимизации мы не делали никаких предположений о характере целевой функции и виде ограничений. Важным разделом математического программирования является *линейное программирование,* изучающее задачи оптимизации, в которых целевая функция является линейной функцией проектных параметров, а ограничения задаются в виде линейных уравнений и неравенств.

Стандартная (каноническая) постановка задачи линейного программирования формулируется следующим образом: найти значения переменных $x_1, x_2, \ldots, x_n$, которые:

1) удовлетворяют системе линейных уравнений

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2, \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n &= b_m; \end{aligned} \tag{9.15}$$

2) являются неотрицательными, т. е.

$$x_1 \geq 0, \quad x_2 \geq 0, \quad ..., \quad x_n \geq 0; \tag{9.16}$$

3) обеспечивают наименьшее значение линейной целевой функции

$$f(x_1, x_2, ..., x_n) = c_0 + c_1x_1 + c_2x_2 + ... + c_nx_n. \tag{9.17}$$

Всякое решение системы уравнений 9.15, удовлетворяющее системе неравенств (9.16), называется *допустимым решением*. Допустимое решение, которое минимизирует целевую функцию (9.17), называется *оптимальным решением*.

Рассмотрим пример задачи линейного программирования (транспортную задачу).

ПРИМЕР. Автобаза обслуживает три овощных магазина, причем товар доставляется в магазины из двух плодоовощных баз. Нужно спланировать перевозки так, чтобы их общая стоимость была минимальной.

Введем исходные данные. Ежедневно вывозится с первой базы 12 т товара, со второй 15 т. При этом завозится в первый магазин 8 т, во второй 9 т, в третий 10 т. Стоимость перевозки 1 т товара (в рублях) с баз в магазины дается следующей таблицей:

База	Магазин		
	первый	второй	третий
Первая	0.80	1.10	0.90
Вторая	1.00	0.70	1.20

Решение. Обозначим через x_1, x_2, x_3 количество товара, который нужно доставить с первой базы соответственно в первый, второй и третий магазины, а через x_4, x_5, x_6 количество товара, который нужно доставить со второй базы в те же магазины. Эти значения в соответствии с исходными данными должны удовлетворять следующим условиям:

$$\begin{aligned} x_1 + x_2 + x_3 &= 12, \\ x_4 + x_5 + x_6 &= 15, \\ x_1 + x_4 &= 8, \\ x_2 + x_5 &= 9, \\ x_3 + x_6 &= 10. \end{aligned} \tag{9.18}$$

Первые два уравнения этой системы описывают количество товара, которое необходимо вывезти с первой и второй баз, а три последних — сколько нужно завезти товара в каждый магазин.

К данной системе уравнений нужно добавить систему неравенств

$$x_i \geq 0, \quad i = 1, 2, ..., 6, \tag{9.19}$$

которая означает, что товар обратно с магазинов на базы не вывозится. Общая стоимость перевозок с учетом приведенных в таблице расценок выразится формулой

$$f = 0.8x_1 + 1.1x_2 + 0.9x_3 + x_4 + 0.7x_5 + 1.2x_6. \tag{9.20}$$

Таким образом, мы пришли к типичной задаче линейного программирования: найти оптимальные значения проектных параметров x_i (i = 1, 2, ..., 6), удовлетворяющих условиям (9.18), (9.19) и минимизирующих общую стоимость перевозок (9.20).

Из анализа системы уравнений (9.18) следует, что только первые четыре уравнения являются независимыми, а последнее можно получить из них (путем сложения первого и второго уравнений и вычитания из этой суммы третьего и четвертого уравнений). Поэтому фактически имеем систему

$$\begin{aligned} x_1 + x_2 + x_3 &= 12, \\ x_4 + x_5 + x_6 &= 15, \\ x_1 + x_4 &= 8, \\ x_2 + x_5 &= 9. \end{aligned} \tag{9.21}$$

Число неизвестных на два больше числа уравнений, поэтому выразим через x_1 и x_2 все остальные неизвестные. Получим

$$
\begin{aligned}
x_3 &= 12 - x_1 - x_2, \\
x_4 &= 8 - x_1, \\
x_5 &= 9 - x_2, \\
x_6 &= x_1 + x_2 - 2.
\end{aligned}
\tag{9.22}
$$

Поскольку в соответствии с (9.19) все проектные параметры должны быть неотрицательны, то с учетом (9.22) получим следующую систему неравенств:

$$
\begin{aligned}
x_1 \ge 0, \quad x_2 \ge 0, \\
12 - x_1 - x_2 \ge 0, \\
8 - x_1 \ge 0, \quad 9 - x_2 \ge 0, \\
x_1 + x_2 - 2 \ge 0.
\end{aligned}
\tag{9.23}
$$

Эти неравенства можно записать в более компактном виде:

$$
0 \le x_1 \le 8, \quad 0 \le x_2 \le 9, \quad 2 \le x_1 + x_2 \le 12. \tag{9.24}
$$

Данная система неравенств описывает все допустимые решения рассматриваемой задачи. Среди всех допустимых значений свободных параметров x_1 и x_2 нужно найти оптимальные, минимизирующие целевую функцию f. Формула (9.20) для неё с учетом соотношений (9.22) принимает вид

$$
f = 22.7 + 0.1x_1 + 0.7x_2. \tag{9.25}
$$

Отсюда следует, что стоимость перевозок растет с увеличением значений x_1, x_2; поэтому нужно взять их наименьшие допустимые значения. В соответствии с 9.24 $x_1 + x_2 \ge 2$;

Рисунок 9.7 Схема перевозок

примем $x_1 + x_2 = 2$. Исключая один из параметров, например x_2, получим $x_2 = 2 - x_1$. Тогда

$$
f = 24.1 - 0.6x_1.
$$

Очевидно, что стоимость перевозок f будет минимальной, если величина x_1 примет наибольшее значение в рамках сделанного ограничения $(x_1 + x_2 = 2)$. Таким оптимальным будет значение $x_1 = 2$. Тогда $x_2 = 0$, а оптимальные значения остальных проектных параметров можно найти по формулам (9.22: $x_3 = 10$, $x_4 = 6$, $x_5 = 9$, $x_6 = 0$. В этом случае минимальная общая стоимость перевозок f равна 22.9 р. На рисунке 9.7 показана схема доставки товаров, соответствующая полученному решению. Числа указывают количество товара (в тоннах).

4.3 Геометрический метод. Областью решения линейного неравенства с двумя переменными

$$
a_0 + a_1x_1 + a_2x_2 \ge 0 \tag{9.26}
$$

является полуплоскость. Для того чтобы определить, какая из двух полуплоскостей соответствует этому неравенству, нужно привести его к виду $x_2 \ge kx_1 + b$ или $x_2 \le kx_1 + b$. Тогда искомая полуплоскость в первом случае расположена выше прямой $a_0 + a_1x_1 + a_2x_2 = 0$, во втором — ниже нее. Если $a_2 = 0$, то неравенство 9.26 имеет вид $a_0 + a_1x_1 \ge 0$; в этом случае получим либо $x_1 \ge h$ — правую полуплоскость, либо $x_1 \le h$ — левую полуплоскость.

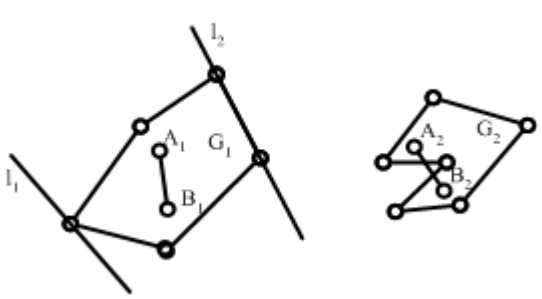


Рисунок 9.8 Выпуклая G_1 и не выпуклая G_2 области

Областью решений системы неравенств является пересечение конечного числа полуплоскостей, описываемых каждым отдельным неравенством. Это пересечение представляет собой многоугольную область *G.* Она может быть как ограниченной, так и неограниченной и даже пустой (если система неравенств противоречива).

Область решений *G* обладает важным свойством выпуклости. Область называется *выпуклой,* если произвольные две ее точки можно соединить отрезком, целиком принадлежащим данной области. На Рисунок 9.8 Выпуклая G1 и не выпуклая G2 области показаны выпуклая область G_1 и невыпуклая область G_2. В области G_1, две ее произвольные точки A_1 и B_1 можно соединить отрезком, все точки которого принадлежат области G_1. В области G_2 можно выбрать такие две ее точки A_2 и B_2, что не все точки отрезка A_2B_2 принадлежат области G_2.

Опорной прямой называется прямая, которая имеет с областью по крайней мере одну общую точку, при этом вся область расположена по одну сторону от этой прямой. На Рисунок 9.8 Выпуклая G1 и не выпуклая G2 области показаны две опорные прямые l_1 и l_2, т. е. в данном случае опорные прямые проходят соответственно через вершину многоугольника и через одну из его сторон.

Аналогично можно дать геометрическую интерпретацию системы неравенств с тремя переменными. В этом случае каждое неравенство описывает полупространство, а вся система — пересечение полупространств, т. е. многогранник, который также обладает свойством выпуклости. Здесь опорная плоскость проходит через вершину, ребро или грань многогранной области.

Основываясь на введенных понятиях, рассмотрим *геометрический метод* решения задачи линейного программирования. Пусть заданы линейная целевая функция $f = c_o + c_1x_1 + c_2x_2$ двух независимых переменных, а также некоторая совместная система линейных неравенств, описывающих область решений *G.* Требуется среди допустимых решений $(x_1, x_2) \in G$ найти такое, при котором линейная целевая функция *f* принимает наименьшее значение.

Положим функцию *f* равной некоторому постоянному значению C: $f = c_o + c_1x_1 + c_2x_2 = C$. Это значение достигается, в точках прямой, удовлетворяющих уравнению

$$c_0 + c_1x_1 + c_2x_2 = C. \qquad (9.27)$$

При параллельном переносе этой прямой в положительном направлении вектора нормали $\mathbf{n}(c_1, c_2)$ линейная функция *f* будет возрастать, а при переносе прямой в противоположном направлении — убывать.

Предположим, что прямая, записанная в виде (9.27), при параллельном переносе в положительном направлении вектора **n** первый раз встретится с областью допустимых решений *G* в некоторой ее вершине, при этом значение целевой функции равно C_1 и прямая становится опорной. Тогда значение c_1 !!!(9.28)!!! будет минимальным, поскольку дальнейшее движение прямой в том же направлении приведет к увеличению значения *f*.

Если в задаче оптимизации нас интересует максимальное значение целевой функции, то параллельный перенос прямой (9.27) осуществляется в направлении, противоположном **n**, пока она не станет опорной. Тогда вершина многоугольника *G,* через которую проходит опорная прямая, будет соответствовать максимуму функции *f*. При дальнейшем переносе прямой целевая функция будет убывать.

Таким образом, оптимизация линейной целевой функции на многоугольнике допустимых решений происходит в точках пересечения этого многоугольника с опорными прямыми, соответствующими данной целевой функции. При этом пересечение может быть в одной точке (в вершине многоугольника) либо в бесконечном множестве точек (на ребре многоугольника).

В заключение вернемся к рассмотренной ранее транспортной задаче (см. п. 2). На Рисунке 9.9 изображен многоугольник *ABCDEF* допустимых решений. Он получен как пересечение полуплоскостей, описываемых неравенствами (9.23). Опорная прямая *U* соответствует уравнению 9.25 при *f*=22.9. Точка *A* пересечения опорной прямой с многоугольником решений дает минимум целевой функции.

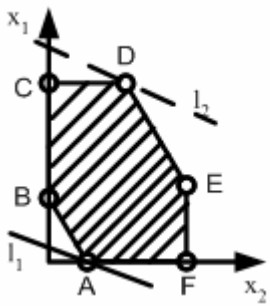


Рисунок 9.9 Область допустимых решений

При дальнейшем параллельном переносе этой прямой вверх можем попасть в точку *D* (опорная прямая l_2) и получить максимум целевой функции.

4.4 Симплекс-метод. Рассмотренный геометрический метод решения задач линейного программирования достаточно прост и нагляден для случая двух и даже трех переменных. Для большего числа переменных применение геометрического метода становится невозможным.

Правда, мы видели, что оптимальные значения целевой функции достигаются на границе области допустимых решений. Поэтому в случае *n* неизвестных ($n > 3$) можно построить n-мерный многогранник решений, найти его вершины и вычислить значения целевой функции в этих точках. Наименьшее среди полученных значений можно принять за искомое, а координаты соответствую щей вершины — за оптимальные значения проектных параметров.

Однако решение задачи линейного программирования не так просто, как может показаться на первый взгляд. Сложность состоит в том, что количество проектных параметров в реальных задачах (особенно в экономических) может достигать сотен и даже тысяч. При этом число вершин многогранника *G* может быть настолько большим, что перебор вершин и вычисление в них значений целевой функции приведет к такому объему вычислений, который практически невозможно осуществить в течение разумного времени даже с помощью ЭВМ.

Одним из методов, позволяющих эффективно решать подобные задачи, причем с гораздо меньшим числом операций, является симплекс-метод.

Симплексом называется простейший выпуклый многогранник при данном числе измерений. В частности, при $n = 2$ — произвольный треугольник, $n = 3$ — произвольный тетраэдр.

Идея *симплекс-метода* состоит в следующем. Примем в качестве начального приближения координаты некоторой вершины многогранника допустимых решений и найдем все ребра, выходящие из этой вершины. Двигаемся вдоль того ребра, по которому линейная целевая функция убывает. Приходим в новую вершину, находим все выходящие из нее ребра, двигаемся по одному из них и т. д. В конце концов мы придем в такую вершину, движение из которой вдоль любого ребра приведет к возрастанию целевой функции. Следовательно, минимум достигнут, и координаты этой последней вершины принимаются в качестве оптимальных значений рассматриваемых проектных параметров.

Отметим, что (поскольку *f* — линейная функция, а многогранник выпуклый) данный вычислительный процесс сходится к решению задачи, причем за конечное число шагов *k.* В данном случае их число порядка *n,* т. е. значительно меньше числа шагов в методе простого перебора вершин, где *k* может быть порядка 2^n.

Пусть задача линейного программирования состоит в том, что нужно найти такие неотрицательные значения проектных параметров $x_1, x_2, \ldots, x_n$, которые минимизируют линейную целевую функцию при заданных ограничениях в виде системы линейных уравнений

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m; \end{aligned} \qquad (9.29)$$

Если в качестве ограничений заданы неравенства, то их можно преобразовать в равенства путем введения дополнительных неотрицательных переменных, которые называются *балансовыми.* Например, имея некоторое неравенство $a_1x_1 + a_2x_2 + \ldots + a_nx_n \le b$, можно всегда ввести новую переменную $x_{n+1} > 0$ такую, что после ее прибавления к левой части данного неравенства получим равенство $a_1x_1 + a_2x_2 + \ldots + a_nx_n + x_{n+1} = b$.

Будем считать, что все уравнения, системы 9.29 *линейно независимы,* т. е. ни одно из них нельзя получить как линейную комбинацию других. В противном случае линейно зависимые уравнения можем отбросить. И кроме того, эта система *совместна,* т. е. среди уравнений системы нет противоречивых. Ясно, что при этих предположениях число уравнений *m* меньше числа переменных *n,* поскольку при $m = n$ система 9.29 имеет единственное решение, что исключает всякую оптимизацию; при $m > n$ не выполняются сделанные выше предположения.

Выразим первые m переменных $x_1, x_2, ..., x_m$ через остальные с помощью уравнений 9.29:

$$\begin{gathered} x_1 = p_1 + q_{1,\,m+1}x_{m+1} + q_{1,\,m+2}x_{m+1} + ... + q_{1n}x_n, \\ x_2 = p_2 + q_{2,\,m+1}x_{m+1} + q_{2,\,m+2}x_{m+1} + ... + q_{2n}x_n, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_m = p_m + q_{m,\,m+1}x_{m+1} + q_{m,\,m+2}x_{m+1} + ... + q_{mn}x_n, \\ p_i \geq 0, \quad i = 1, 2, ..., m. \end{gathered} \tag{9.30}$$

Переменные $x_1, x_2, ..., x_m$ называются *базисными,* а вектор $\{x_1, x_2, ..., x_m\}$ — *базисом;* $x_{m+1}, x_{m+2}, ..., x_n$ — свободные переменные.

Используя соотношения (9.30), можно выразить линейную целевую функцию через свободные переменные:

$$f = d_0 + d_{m+1}x_{m+1} + ... + d_n x_n. \tag{9.31}$$

Процесс оптимизации начнём с некоторого начального *(опорного)* решения, например при нулевых значениях свободных переменных. Тогда получим

$$x_1 = p_1, ..., x_m = p_m, \quad x_{m+1} = 0, ..., x_n = 0. \tag{9.32}$$

При этом целевая функция (9.31) принимает значение $f^{(0)} = d_0$.

Дальнейшее решение задачи симплекс-методом распадается на ряд этапов, заключающихся в том, что от одного решения нужно перейти к другому с таким условием, чтобы целевая функция не возрастала. Это достигается выбором нового базиса и значений свободных переменных.

Выясним, является ли опорное решение (9.32) оптимальным. Для этого проверим, можно ли уменьшить соответствующее этому решению значение целевой функции $f = d_0$ при изменении каждой свободной переменной. Поскольку $x_i > 0$, то мы можем лишь увеличивать их значения. Если коэффициенты $d_{m+1}, ..., d_n$ в формуле (9.31) неотрицательны, то при увеличении любой свободной переменной $x_{m+1}, ..., x_n$ целевая функция не может уменьшиться. В этом случае решение 9.32 окажется оптимальным.

Пусть теперь среди коэффициентов формулы (9.31) хотя бы один отрицательный, например $d_{m+1} < 0$. Это означает, что при увеличении переменной x_{m+1} целевая функция уменьшается по сравнению со значением d_0, соответствующим решению 9.32. Поэтому в качестве нового опорного выбирается решение при следующих значениях свободных параметров:

$$x_{m+1} = x_{m+1}^{(1)}, \quad x_{m+2} = 0, \quad ..., \quad x_n = 0. \tag{9.33}$$

При этом базисные переменные, вычисляемые по формулам ((9.30), равны

$$x_i = p_i + q_{i,\,m+1}x_{m+1}^{(1)}, \quad i = 1, 2, ..., m. \tag{9.34}$$

Если все коэффициенты $q_{i,m+1}$ неотрицательны, то x_{m+1} можно увеличивать неограниченно; в этом случае не существует оптимального, решения задачи. Однако на практике такие случаи, как правило, не встречаются. Обычно среди коэффициентов $g_{i,\,m+1}$ имеются отрицательные, а это влечет за собой угрозу сделать некоторые переменные x_i в 9.34 отрицательными из-за большого значения $x_{m+1}^{(1)}$. Следовательно, переменную x_{m+1} можно увеличивать лишь до тех пор, пока базисные переменные остаются неотрицательными. Это и является условием выбора значения $x_{m+1}^{(1)}$. Его можно записать в виде

$$p_i + q_{i,\,m+1}x_{m+1}^{(1)} \geq 0, \quad i = 1, 2, ..., m. \tag{9.35}$$

Среди всех отрицательных коэффициентов $q_{i,m+1}$ найдем наибольший по модулю. Пусть его значение равно Q, а соответствующее ему значение p_i равно P. Тогда из (9.35) получим максимально возможное значение переменой x_{m+1} на данном шаге оптимизации:

$$x_{m+1}^{(1)} = -P/Q \ \ (P < 0, Q < 0), \tag{9.36}$$

и новое опорное решение запишем в виде

$$\begin{gathered} x_1 = p_1 - \frac{P}{Q}q_{1,\,m+1}, \quad ..., \quad x_m = p_m - \frac{P}{Q}q_{m,\,m+1}, \\ x_{m+1} = -\frac{P}{Q}, \quad x_{m+2} = 0, \quad ..., \quad x_n = 0. \end{gathered} \tag{9.37}$$

Полученное значение целевой функции $f^{(1)}$ меньше предыдущего, поскольку в данной формуле второй член правой части больше нуля ($d_{m+1} < 0$, $Q < 0$, $P > 0$).

На этом заканчивается первый шаг оптимизации. Теперь нужно сделать второй шаг, используя аналогичную процедуру. Для этого необходимо выбрать новый базис, принимая в качестве базисных переменных параметры $x_1, ..., x_{m-1}, x_{m+1}$. После второго шага мы либо найдем новые оптимальные значения переменных и соответствующее им значение целевой функции $f^{(2)} < f^{(1)}$, либо покажем, что решение 9.36 является оптимальным. В любом случае после конечного числа шагов мы придем к оптимальному решению. Еще раз

подчеркнем, что в отличие от метода перебора симплекс-метод дает возможность вести поиск целенаправленно, уменьшая на каждом шаге значение целевой функции.
В качестве примера, иллюстрирующего симплекс-метод, рассмотрим задачу об использовании ресурсов.

4.5 Задача о ресурсах. В распоряжении бригады имеются следующие ресурсы: 300 кг металла, 100 м^2 стекла, 160 чел.-ч (человеко-часов) рабочего времени. Бригаде поручено изготовлять два наименования изделий — А и Б. Цена одного изделия А 10 р., для его изготовления необходимо 4 кг металла, 2 м^2 стекла и 2 чел.-ч рабочего времени. Цена одного изделия Б 12 р., для его изготовления необходимо 5 кг металла, 1 м^2 стекла и 3 чел.-ч рабочего времени. Требуется так спланировать объем выпуска продукции, чтобы ее стоимость была максимальной.
Сначала сформулируем задачу математически. Обозначим через x_1 и x_2 количество изделий А и Б, которое необходимо запланировать (т. е. это искомые величины). Имеющиеся ресурсы сырья и рабочего времени зададим в виде ограничений-неравенств:

$$\begin{aligned} 4x_1 + 5x_2 &\le 300, \\ 2x_1 + x_2 &\le 100, \\ 2x_1 + 3x_2 &\le 160. \end{aligned} \tag{9.38}$$

Полная стоимость запланированной к производству продукции выражается формулой

$$f = 10x_1 + 12x_2. \tag{9.39}$$

Таким образом, мы имеем задачу линейного программирования, которая состоит в определении оптимальных значений проектных параметров x_1, x_2, являющихся целыми неотрицательными числами, удовлетворяющих линейным неравенствам 9.38 и дающих максимальное значение линейной целевой функции (9.39).
Вид сформулированной задачи не является каноническим, поскольку условия(9.38) имеют вид неравенств, а не уравнений. Как уже отмечалось выше, такая задача может быть сведена к канонической путем введения дополнительных переменных x_3, x_4, x_5 по количеству ограничений-неравенств (9.38). При этом выбирают эти переменные такими, чтобы при их прибавлении к левым частям соотношений 9.38 неравенства превращались в равенства. Тогда ограничения примут вид

$$\begin{aligned} 4x_1 + 5x_2 + x_3 &= 300, \\ 2x_1 + x_2 + x_4 &= 100, \\ 2x_1 + 3x_2 + x_5 &= 160. \end{aligned} \tag{9.40}$$

При этом очевидно, что $x_3 \ge 0, x_4 \ge 0, x_4 \ge 0$. Заметим, что введение дополнительных неизвестных не повлияло на вид целевой функции (9.39), которая зависит только от параметров x_1, x_2. Фактически x_3, x_4, x_5 будут указывать остатки ресурсов, не использованные в производстве. Здесь мы имеем задачу максимизации, т.е. нахождения максимума целевой функции. Если функцию 9.39 взять со знаком минус, т. е. принять целевую функцию в виде

$$F = -10x_1 - 12x_2, \tag{9.41}$$

то получим задачу минимизации для этой целевой функции.
Примем переменные x_3, x_4, x_5 в качестве базисных и выразим их через свободные переменные x_1, x_2 из уравнений (9.40). Получим

$$\begin{aligned} x_3 &= 300 - 4x_1 - 5x_2, \\ x_4 &= 100 - 2x_1 - x_2, \\ x_5 &= 160 - 2x_1 - 3x_2. \end{aligned} \tag{9.42}$$

В качестве опорного решения возьмем такое, которое соответствует нулевым значениям свободных параметров:

$$x_1^{(0)} = 0, \quad x_2^{(0)} = 0, \quad x_3^{(0)} = 300, \quad x_4^{(0)} = 100, \quad x_5^{(0)} = 160. \tag{9.43}$$

Этому решению соответствует нулевое значение целевой функции 9.41:

$$F^{(0)} = 0. \tag{9.44}$$

Исследуя полученное решение, отмечаем, что оно не является оптимальным, поскольку значение целевой функции (9.41) может быть уменьшено по сравнению с (9.44) путем увеличения свободных параметров.
Положим $x_2 = 0$ и будем увеличивать переменную x_1 до тех пор, пока базисные переменные остаются положительными. Из (9.42) следует, что x_1 можно увеличить до значения $x_1 = 50$, поскольку при большем его значении переменная x_4 станет отрицательной.
Таким образом, полагая $x_1 = 50$, $x_2 = 0$, получаем новое опорное решение (значения переменных x_3, x_4, x_5 найдем по формуле (9.42):

$$x_1^{(1)} = 50, \quad x_2^{(1)} = 0, \quad x_3^{(1)} = 100, \quad x_4^{(1)} = 0, \quad x_5^{(1)} = 60. \tag{9.45}$$

Значение целевой функции (9.41) при этом, будет равно

$$F^{(1)} = -500. \tag{9.46}$$

Новое решение (9.45), следовательно, лучше, поскольку значение целевой функции уменьшилось по сравнению с (9.44).

Следующий шаг начнем с выбора нового базиса. Примем ненулевые переменные в ((9.45) x_1, x_3, x_5 в качестве базисных, а нулевые переменные x_2, x_4 в качестве свободных. Из системы (9.40) найдем

$$\begin{aligned} x_1 &= 50 - \frac{1}{2}x_2 - \frac{1}{2}x_4, \\ x_3 &= 100 - 3x_2 + 2x_4, \\ x_5 &= 60 - 2x_2 + x_4. \end{aligned} \tag{9.47}$$

Выражение для целевой функции (9.41) запишем через свободные параметры, заменив x_1 с помощью (9.47). Получим

$$F = -500 - 7x_2 + 5x_4. \tag{9.48}$$

Отсюда следует, что значение целевой, функции по сравнению с (9.46) можно уменьшить за счет увеличения x_2, поскольку коэффициент при этой переменной в (9.48) отрицательный. При этом увеличение x_4 недопустимо, поскольку это привело бы к возрастанию целевой функции; поэтому положим $x_4 = 0$.

Максимальное значение переменной x_2 определяется соотношениями (9.47). Быстрее всех нулевого значения достигнет переменная x_5 при $x_2 = 30$. Дальнейшее увеличение x_2 поэтому невозможно. Следовательно, получаем новое опорное решение, соответствующее значениям $x_2 = 30$, $x_4 = 0$ и определяемое соотношениями (9.47):

$$x_1^{(2)} = 35, \quad x_2^{(2)} = 30, \quad x_3^{(2)} = 10, \quad x_4^{(2)} = 0, \quad x_5^{(2)} = 0. \tag{9.49}$$

При этом значение целевой функции 9.48 равно

$$F^{(2)} = -710. \tag{9.50}$$

Покажем, что полученное решение является оптимальным. Для проведения следующего шага ненулевые переменные в (9.49), т. е. x_1, x_2, x_3, нужно принять в качестве базисных, а нулевые переменные x_4, x_5 — в качестве свободных переменных. В этом случае целевую функцию можно записать в виде

$$F = -710 + \frac{3}{2}x_4 + \frac{7}{2}x_5.$$

Поскольку коэффициенты при x_4, x_5 положительные, то при увеличении этих параметров целевая функция возрастает. Следовательно, минимальное значение целевой функции $F_{min} = -710$ соответствует нулевым значениям параметров x_4, x_5, и полученное решение является оптимальным.

Таким образом, ответ на поставленную задачу об использовании ресурсов следующий: для получения максимальной суммарной стоимости продукции при заданных ресурсах необходимо запланировать изготовление изделий А в количестве 35 штук и изделий Б в количестве 30 штук. Суммарная стоимость продукции равна 710 р. При этом все ресурсы стекла и рабочего времени будут использованы, а металла останется 10 кг.

Упражнения

1. Исследовать на экстремум функцию $y = (x - 5)e^x$.

2. Найти наибольшее и наименьшее значения функции $y = x\sqrt{1 - x^2}$ в области ее определения.

3. Удельный расход газа плотности ρ с показателем адиабаты k в газовой струе определяется формулой

$$Q = \rho v(1 - v^2 / v_{\max}^2)^{1/(k-1)}.$$

При какой скорости v расход газа будет максимальным?

4. Составить блок-схему определения наименьшего значения функции на отрезке с помощью метода общего поиска.

5. Усовершенствовать алгоритм предыдущей задачи путем повторного деления суженного интервала неопределенности.

6. Используя метод золотого сечения, найти на отрезке [0, 3] наименьшее значение функции

$$f(x) = \begin{cases} x^2 - 2x + 2, & 0 \le x \le 2, \\ x^2/(2x - 1), & x > 2. \end{cases}$$

7. Работа деформации рамы выражается формулой

$$A = \frac{l^3}{2EI}\left(\frac{4}{3}X^2 - XY + \frac{1}{3}Y^2 + \frac{1}{3}PX - \frac{1}{4}PY + \frac{1}{10}P^2\right),$$

где P — нагрузка, X и Y — горизонтальная и вертикальная реакции опоры, l — длина, E - модуль упругости, I — момент инерции. При каких значениях X, Y работа будет минимальной?

8. Спроектировать цилиндрический котел емкостью 200 л таким образом, чтобы на его изготовление было

израсходовано как можно меньше материала.

9. Начертить области, определенные системами неравенств:

а) $x \geq 0,\ \ y \geq 0,\ \ 2x + y \leq 4;$

б) $x - y \geq 0,\ \ x \leq 9,\ \ x + 3y \geq 6.$

10. Минимизировать функцию $f = 12x_1 + 4x_2$ при наличии ограничений

$x_1 + x_2 \geq 2,\ \ x_1 > 0.5,\ \ x_2 \leq 4,\ \ x_1 - x_2 \geq 0.$

11. Имеются два склада с сырьем. Ежедневно вывозится с первого склада 60 т сырья, со второго 80 т. Сырье используется двумя заводами, причем первый завод получает его 50 т, второй 90 т. Нужно организовать оптимальную (наиболее дешевую) схему перевозок, если известно, что доставка 1 т сырья с первого склада на первый завод стоит 70 к., с первого склада на второй завод — 90 к., со второго оклада на первый завод — 1 р., со второго склада на второй завод — 80 к.

Глава 10
БЫСТРОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

§1. Дискретное преобразование Фурье

1.1 Дискретное преобразование Фурье (ДПФ) — это одно из преобразований Фурье, широко применяемых в алгоритмах цифровой обработки сигналов (его гомоморфизмы применяются в сжатии звука в mp3, сжатие изображений в jpg и др.), а также в других областях, связанных с анализом частот в дискретном (к примеру, оцифрованном аналоговом) сигнале. Также дискретные преобразования Фурье помогают решать частные дифференциальные уравнения и выполнять такие операции, как свёртки. Преобразования бывают одномерные, двумерные и даже трехмерные.

Последовательность N действительных чисел x_0, ..., x_{N-1} преобразовывается в последовательность из N комплексных чисел X_0, ..., X_{N-1} с помощью дискретного преобразования Фурье по формуле:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn} \quad k = 0,\ldots,N-1$$

где i - это мнимая единица. Обратное дискретное преобразование Фурье задается формулой

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn} \quad n = 0,\ldots,N-1$$

Вывод преобразования

Рассмотрим некоторый периодический сигнал $x(t)$ с периодом равным T. Разложим его в ряд Фурье:

$$x(t) = \sum_{k=-\infty}^{+\infty} c_k e^{i\omega_k t}, \omega_k = \frac{2\pi k}{T}$$

Проведем дискретизацию сигнала так, чтобы на периоде было N отсчетов. Дискретный сигнал представим в виде отсчетов: $x_n = x(t_n)$, где $t_n = \frac{n}{N}T$, тогда ряд Фурье запишется следующим образом:

$$x_n = \sum_{k=-\infty}^{+\infty} c_k e^{i\omega_k t_n} = \sum_{k=-\infty}^{+\infty} c_k e^{\frac{2\pi i}{N}kn}$$

Используя соотношение: $e^{\frac{2\pi i}{N}(k+mN)n} = e^{\frac{2\pi i}{N}kn}$, получаем:

$$x_n = \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn}, X_k = \sum_{l=-\infty}^{+\infty} c_{k+lN}$$

Таким образом, мы получили обратное дискретное преобразование Фурье.

Умножим теперь скалярно выражение для x_n на $e^{\frac{2\pi i}{N}mn}$ и получим:

$$\sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}mn} = \sum_{k=0}^{N-1}\sum_{n=0}^{N-1} X_k e^{\frac{2\pi i}{N}(k-m)n}$$

Откуда следует, что:

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn}$$

Эта формула описывает прямое дискретное преобразование Фурье.

В литературе принято писать множитель $\frac{1}{N}$ в обратном преобразовании, и поэтому обычно пишут формулы преобразования в следующем виде:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}}, x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}k}$$

Матричное представление

Дискретное преобразование Фурье является линейным преобразованием, которое переводит вектор временных отсчетов $\vec{x}$ в вектор спектральных отсчетов той же длины. Таким образом, преобразование может быть реализовано как умножение квадратной матрицы на вектор:

$$\vec{X} = \hat{A}\vec{x}$$

матрица A имеет вид:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-\frac{2\pi i}{N}} & e^{-\frac{4\pi i}{N}} & e^{-\frac{6\pi i}{N}} & \dots & e^{-\frac{2\pi i}{N}(N-1)} \\ 1 & e^{-\frac{4\pi i}{N}} & e^{-\frac{8\pi i}{N}} & e^{-\frac{12\pi i}{N}} & \dots & e^{-\frac{2\pi i}{N}2(N-1)} \\ 1 & e^{-\frac{6\pi i}{N}} & e^{-\frac{12\pi i}{N}} & e^{-\frac{18\pi i}{N}} & \dots & e^{-\frac{2\pi i}{N}3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-\frac{2\pi i}{N}(N-1)} & e^{-\frac{2\pi i}{N}2(N-1)} & e^{-\frac{2\pi i}{N}3(N-1)} & \dots & e^{-\frac{2\pi i}{N}(N-1)^2} \end{pmatrix}$$

Элементы матрицы задаются следующей формулой:

$$A(m, n) = \exp\left(-2\pi i \frac{(m-1)(n-1)}{N}\right)$$

Свойства

- 1) линейность

$$ax(n) + by(n) \leftrightarrow aX(k) + bY(k)$$

- 2) сдвиг по времени

$$x(n-m) \leftrightarrow X(k)e^{-\frac{2\pi i}{N}knm}$$

- 3) периодичность

$$X(k + rN) = X(k), r \in Z$$

- 4) выполняется Теорема Парсеваля
- 5) симметрии

$X(k) = X(N - k)$

Таким образом информацию несут первые N/2 гармоник.

- 6) обладает спектральной плотностью

$S(k) = |x(k)|^2$

- 7) $X(0) \in \mathbb{R}$

$$N \bmod 2 = 0 \Rightarrow X(N/2) \in R$$

Стоит отметить, что нулевая гармоника является средним значением сигнала.

1.2 Быстрое Преобразование Фурье

Быстрое преобразование Фурье (БПФ) — это быстрый алгоритм вычисления дискретного преобразования Фурье. То есть алгоритм вычисления за количество действий, меньше чем N^2, требуемых для прямого (по формуле) вычисления ДПФ.
Наиболее популярным из алгоритмов ускоренного вычисления ДПФ является т.н. метод Cooley-Tukey, позволяющий вычислить ДПФ для числа отсчетов N = 2k за время порядка $N \cdot \log_2 N$ (отсюда и название - быстрое преобразование Фурье, БПФ). Этот способ чем-то неуловимо напоминает быструю сортировку. В ходе работы алгоритма также проводится рекурсивное разбиение массива чисел на два подмассива и сведение вычисления ДПФ от целого массива к вычислению ДПФ от подмассивов в отдельности.
Изобретение БПФ привело к потрясающему всплеску популярности преобразования Фурье. Целый ряд важных задач раньше решался за время порядка N^2 , но после проведения преобразования Фурье над исходными данными (за время порядка $N \cdot \log_2 N$) решается практически мгновенно. Преобразование Фурье лежит в основе цифровых корелляторов и методов свертки, активно используется при спектральном анализе (практически в чистом виде), применяется при работе с длинными числами.

§2. Алгоритм быстрого преобразования Фурье

2.1 Алгоритм

Покажем как выполнить дискретное преобразование Фурье за $O(N(p_1 + \ldots + p_n))$ действий при $N = p_1 p_2 \ldots p_n$. В частности, при N = 2n понадобится O(Nlog(N)) действий.

Дискретное преобразование Фурье преобразует набор чисел $a_0, \ldots, a_{n-1}$ в набор чисел $b_0, \ldots, b_{n-1}$, такой, что $b_i = \sum_{j=0}^{n-1} a_j \varepsilon^{ij}$, где $\varepsilon^n = 1$ и $\varepsilon^k \neq 1$ при 0 < k < n. Алгоритм быстрого преобразования Фурье применим к любым коммутативным ассоциативным кольцам с единицей. Чаще всего этот алгоритм применяют к полю комплексных чисел (с $\varepsilon = e^{2\pi i/n}$) и к кольцам вычетов.
Основной шаг алгоритма состоит в сведении задачи для N чисел к задаче для p = N / q числам, где q — делитель N. Пусть мы уже умеем решать задачу для N / q чисел. Применим преобразование Фурье к наборам $a_i, a_{q+i}, \ldots, a_{q(p-1)+i}$ для $i = 0, 1, \ldots, q-1$. Покажем теперь, как за O(Np) действий решить исходную задачу. Заметим, что $b_i = \sum_{j=0}^{q-1} a_j \varepsilon^{ij} (\sum_{k=0}^{p-1} a_{kq+j} \varepsilon^{kiq}$. Выражения в скобках нам уже известны — это i(mod p)-тое число после преобразования Фурье j-той группы. Таким образом, для вычисления каждого bi нужно O(q) действий, а для вычисления всех bi — O(Nq) действий, что и требовалось получить.

2.2 Обратное преобразование Фурье

Для обратного преобразования Фурье можно применять алгоритм прямого преобразования Фурье — нужно лишь использовать ε^{-1} вместо ε(или применить операцию комплексного сопряжения в начале к входным данным, а затем к результату полученному после прямого преобразования Фурье) и окончательный результат поделить на N.

2.3 Общий случай

Общий случай может быть сведён к предыдущему. Пусть $4N > 2^k \geq 2N$.
Заметим, что $b_i = \varepsilon^{-i^2/2} \sum_{j=0}^{N-1} \varepsilon^{(i+j)^2/2_\varepsilon - j^2/2_{a_j}}$. Обозначим

$$\overline{a_i} = \varepsilon^{-i^2/2} a_i, \overline{b_i} = \varepsilon^{i^2/2} b_i, c_i = \varepsilon^{(2N-2-i)^2/2}.$$

Тогда $\overline{b_i} = \sum_{j=0}^{2N-2-i} \overline{a_j} c_{2N-2-i-j}$, если положить $\overline{a}_i = 0$ при $i \geq N$.

Таким образом, задача сведена к вычислению свёртки, но это можно сделать с помощью трёх преобразований Фурье для 2k элементов. Выполняем прямое преобразование Фурье для $\{\overline{a}_i\}_{i=0}^{i=2^k-1}$ и $\{c_i\}_{i=0}^{i=2^k-1}$, перемножаем поэлементно результаты и выполняем обратное преобразование Фурье.

Вычисления всех $\bar{a}_i$ и ci требуют O(N) действий, три преобразования Фурье требуют O(Nlog(N)) действий, перемножение результатов преобразований Фурье требует O(N) действий, вычисление всех bi зная значения свертки требует O(N) действий. Итого для дискретного преобразования Фурье требуется O(Nlog(N)) действий для любого N.

Этот алгоритм быстрого преобразования Фурье может работать над кольцом только когда известны первообразные корни из единицы степеней 2N и 2k.

2.4 Принцип работы Быстрого преобразования Фурье

БПФ работает с разложением N-точечного временного промежутка сигнала на N временных сигнальных промежутков, каждый из которых состоял из одной точки. Второй шаг заключается в расчете N частотных спектров, соответствующего этим временным сигнальным промежуткам. После этого, на основе N спектров синтезируется единый частотный спектр.

Рисунок 10.1 Процедура разбиения сигналапоказывает пример разделения временной области с использованием БПФ. В этом примере 16-ти точечный сигнал разлагается на четыре отдельные компоненты. Первый этап заключается в разбиения 16-ти точечного сигнала на два сигнала по 8 точек. На втором этапе происходит разложение данных сигналов на четыре сигнала по 4 точки. Данная процедура продолжается до тех пор, пока не будет сформировано N сигналов состоящих из одной точки. Чересстрочное разложения используется при разделении выборки сигнала на две выборки. Сигнал разделяется как при четном, так и при нечетном количестве отсчетов в выборке. В системе проводится 2N этапов в разложении, т.е. на 16–ти точечный сигнал (24) предусматривает 4 этапа, 512 точки (27) требует 7 этапов, 4096 точки (212) предусматривает 12 этапов и т.д. Запомним, это значение - оно будет много раз встречаться в этой главе.

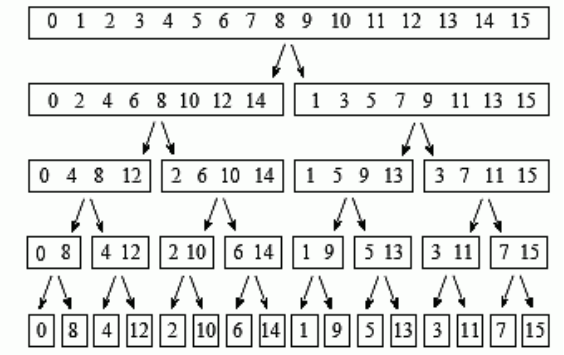


Рисунок 10.1 Процедура разбиения сигнала

Теперь, когда понятна структура разложения, ее можно значительно упростить. Разложения является не более чем перераспределение отсчетов сигнала. Это иллюстрирует Рисунок 10.2 Процедура перераспределения отсчетов сигнала Слева представлены номера отсчетов первоначального сигнала, вместе с их двоичными эквивалентами. Справа, измененные номера, а также их двоичные эквиваленты. Идея состоит в том, что двоичные числа являются инверсией друг друга. Например, отсчёт №3 (0011) можно получить из отсчёта №12 (1100). Отсчёт № 14 (1110) получается из отсчета № 7 (0111), и так далее. Разбиение временной области в БПФ обычно, проводится с использование алгоритма сортировки инвертированных бит. Это предполагает изменение N временных областей, с подсчетом в двоичном виде и переворачиванием бит слева на право.

Sample numbers in normal order			Sample numbers after bit reversal	
Decimal	*Binary*		*Decimal*	*Binary*
0	0000		0	0000
1	0001		8	1000
2	0010		4	0100
3	0011		12	1100
4	0100		2	0010
5	0101		10	1010
6	0110	⇨	6	0100
7	0111		14	1110
8	1000		1	0001
9	1001		9	1001
10	1010		5	0101
11	1011		13	1101
12	1100		3	0011
13	1101		11	1011
14	1110		7	0111
15	1111		15	1111

Рисунок 10.2 Процедура перераспределения отсчетов сигнала !!!

Следующим шагом алгоритма БПФ является нахождение частоты спектров одноточечного сигналов, определенного во временной области. Это делается элементарно: спектр одноточечного сигнала равен самому себе. Это означает, что на этом шаге ничего не надо. Хотя не стоит забывать, что каждый из одноточечных сигналов теперь представляет собой спектр частот, а не сигнал во временной области

Последний шаг в БПФ состоит в том, чтобы объединить N спектров частот в обратном по отношению к временному разделению порядке. В этом случае алгоритм работает некорректно. К сожалению, обратное быстрое преобразование не применяется, а мы должны вернуться на каком-то этапе ко времени. На первом этапе, 16 одноточечных частотных спектра объединяются в 8 частотных спектра по 2 точки каждый. На втором этапе 8 частотных спектров синтезируется в 4 спектра, и так далее. На последнем этапе получаем результат БПФ в виде 16 точечного частотного спектра.

Рисунок 10.3 Процедура синтеза частотного спектра сигналапоказывает, как два частотных спектра, каждый из которых состоял из 4 точек, объединены в единый частотный диапазон на 8 очков. Объединение должно исключить одинаковые разложения во временной области. Иными словами, операция в частотной области должна соответствовать операции во временной области с исключением пересечений. Рассмотрим два сигналы временной области. 8 точечный временной сигнал формируется в два этапа: разбавлением каждого 4-точечного сигнала нулями, для того, чтобы получить 8-ми точечный сигнал, а затем объединять сигналы вместе. Поэтому сигнал 1 превращается в a0b0c0d0, а 2 станет 0e0f0g0h. Суммирование этих двух сигналов дает 8-точечный сигнал aebfcgdh. Как показано на Рисунок 10.3 Процедура синтеза частотного спектра сигнала, дополнение временной реализации сигнала нулями соответствует дублированию частотного спектра. Таким образом, спектры частот в БПФ, дублируются, а потом прибавляет один к другому.

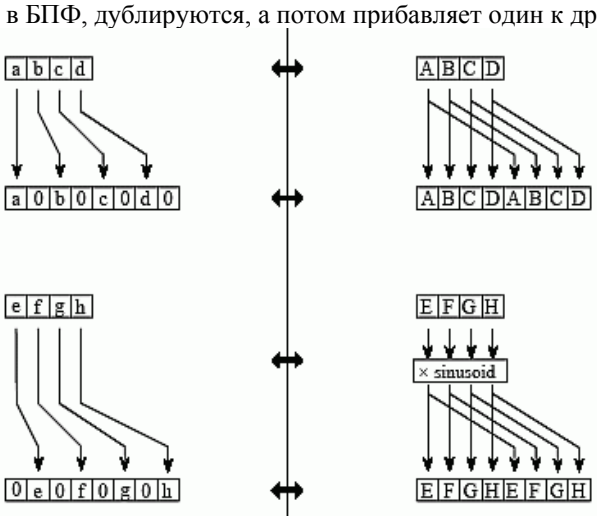


Рисунок 10.3 Процедура синтеза частотного спектра сигнала

Для достижения соответствия, двух сигналов во временной области разбавляются нулями в немного другом виде. В одном сигнала, нечетные отсчеты равны нулю, а в другом сигнале четные отсчеты равны нулю. Иными словами, один из сигналов временной области (0e0f0g0h на Рисунок 10.3 Процедура синтеза частотного спектра сигнала) сдвигается вправо на один отсчет. Этот сдвиг во временной области соответствует умножения на спектр синуса. Чтобы увидеть это, вспомним, что сдвиг во временной области эквивалентен свертыванию сигнала со смещенной дельта-функцией. Спектр смещенной дельта функция является синусоидой.

Рисунок 10.4 Схема совмещения двух 4-х точечных спектра в один 8-ми точечный показывает схема совмещения двух 4-х точечных спектра в один 8-ми точечный. Заметим, что Рисунок 10.4 Схема совмещения двух 4-х точечных спектра в один 8-ми точечныйформируется по базовой схеме Рисунок 10.5 Схема "Бабочки"с повторением вновь и вновь.

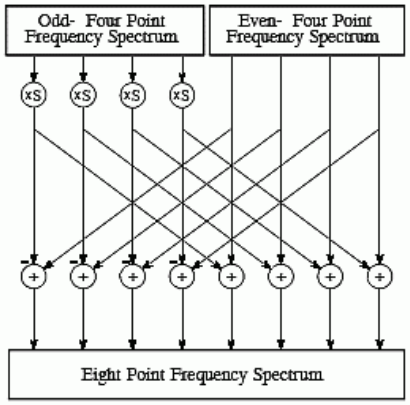


Рисунок 10.4 Схема совмещения двух 4-х точечных спектра в один 8-ми точечный !!!

Эта простая схема называется «бабочка» в силу своего крылатого вида. «Бабочка» является основным вычислительным элементом БПФ, преобразовывая две комплексные точки в две другие.

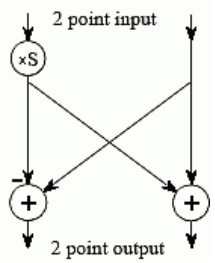


Рисунок 10.5 Схема "Бабочки" !!!

Рисунок 10.6 Блок-схема программы, реализующая БПФпоказывает структуру БПФ. Разбиение во временной области выполняется с помощью ранее рассмотренного алгоритма сортировки.

Синтез в частотной области требует трех циклов. Внешний цикл проводится Log_2N раз. Средний цикл движется по каждой отдельной частоте спектра. Внутренний цикл использует бабочку для расчета точек в каждой частоты спектра (т.е. циклы через образцы внутри одного окна на рис.1). Накладных клетки на рис.6 определяют начало и окончания индексов в циклах, а также расчета необходимых синусоиды бабочек. Блок-схема программы, реализующая БПФ приведена на Рисунок 10.6 Блок-схема программы, реализующая БПФ

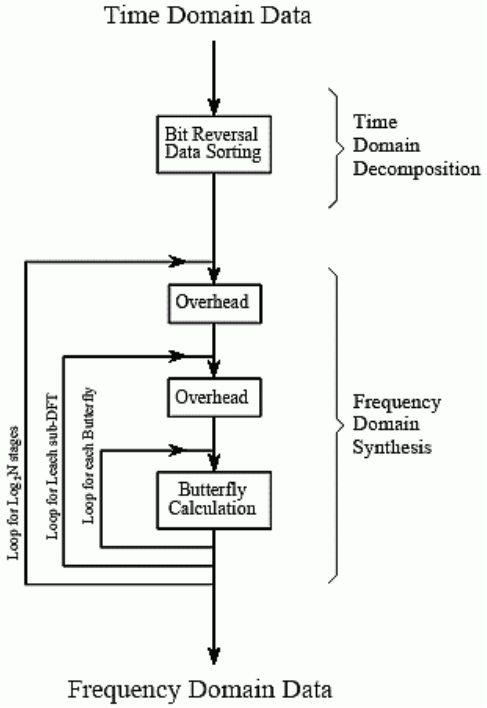


Рисунок 10.6 Блок-схема программы, реализующая БПФ !!!

2.5 *Программа, реализующая БПФ*

Реализация БПФ на языке программирования C

```
/*************************************************************************
Алгоритм проводит быстрое преобразование Фурье комплексной
функции, заданной nn отсчетами на действительной оси.

В зависимости от  переданных параметров, может выполняться
как прямое, так и обратное преобразование.

Входные параметры:
    nn  -   Число значений функции. Должно  быть  степенью
            двойки. Алгоритм   не  проверяет  правильность
            переданного значения.
    a   -   array [0 .. 2*nn-1] of Real
            Значения функции. I-ому значению соответствуют
            элементы   a[2*I]     (вещественная     часть)
            и a[2*I+1] (мнимая часть).
    InverseFFT
        -   направление преобразования.
            True, если обратное, False, если прямое.

Выходные параметры:
    a   -   результат   преобразования.
*************************************************************************/
void fastfouriertransform(ap::real_1d_array& a, int nn, bool inversefft)
{
    int ii;
    int jj;
    int n;
    int mmax;
    int m;
    int j;
    int istep;
    int i;
    int isign;
    double wtemp;
    double wr;
    double wpr;
    double wpi;
    double wi;
    double theta;
    double tempr;
    double tempi;

    if( inversefft )
    {
        isign = -1;
    }
    else
    {
        isign = 1;
    }
    n = 2*nn;
    j = 1;
    for(ii = 1; ii <= nn; ii++)
    {
        i = 2*ii-1;
        if( j>i )
        {
            tempr = a(j-1);
            tempi = a(j);
            a(j-1) = a(i-1);
            a(j) = a(i);
```

```
            a(i-1) = tempr;
            a(i) = tempi;
        }
        m = n/2;
        while(m>=2&&j>m)
        {
            j = j-m;
            m = m/2;
        }
        j = j+m;
    }
    mmax = 2;
    while(n>mmax)
    {
        istep = 2*mmax;
        theta = 2*ap::pi()/(isign*mmax);
        wpr = -2.0*ap::sqr(sin(0.5*theta));
        wpi = sin(theta);
        wr = 1.0;
        wi = 0.0;
        for(ii = 1; ii <= mmax/2; ii++)
        {
            m = 2*ii-1;
            for(jj = 0; jj <= (n-m)/istep; jj++)
            {
                i = m+jj*istep;
                j = i+mmax;
                tempr = wr*a(j-1)-wi*a(j);
                tempi = wr*a(j)+wi*a(j-1);
                a(j-1) = a(i-1)-tempr;
                a(j) = a(i)-tempi;
                a(i-1) = a(i-1)+tempr;
                a(i) = a(i)+tempi;
            }
            wtemp = wr;
            wr = wr*wpr-wi*wpi+wr;
            wi = wi*wpr+wtemp*wpi+wi;
        }
        mmax = istep;
    }
    if( inversefft )
    {
        for(i = 1; i <= 2*nn; i++)
        {
            a(i-1) = a(i-1)/nn;
        }
    }
}
```

Реализация БПФ на языке программирования Pascal

```
procedure LinearFDFT (Re, Im: PReal; Count: Word; Direct: ShortInt);
  var
    I: Word;
    K: Real;
    PIm, PRe: PReal;
  procedure LFDFT (SrcRe, SrcIm: PReal; Cnt: Word);
    var
      EvenRe, OddRe, PEvenRe, POddRe, PRe, PSrcRe: PReal; …; //то же – для
мнимых
      Factor: Real;
      HalfCnt, I, Size: Word;
      HEvenIm, HEvenRe, HOddIm, HOddRe: THandle;
      X, Y, WIm, WRe: Real;
```

```
  begin
    PSrcRe := SrcRe; …;
    if Cnt = 2 then
      begin //тривиальная операция «бабочка»
        Inc (PSrcRe); …;
        X := SrcRe^; Y := PSrcRe^;
        SrcRe^ := X + Y; PSrcRe^ := X - Y;
        X := SrcIm^; Y := PSrcIm^;
        SrcIm^ := X + Y; PSrcIm^ := X - Y;
      end      else
      begin //переупорядочение и рекурсивный вызов для полпоследовательностей
        Factor := K / Cnt;
        HalfCnt := Cnt div 2;
        Size := HalfCnt * SizeOfReal + 1;
        HEvenRe := GlobalAlloc (GMem_Fixed, Size); …;
        HOddRe := GlobalAlloc (GMem_Fixed, Size); …;
        EvenRe := GlobalLock (HEvenRe); …;
        OddRe := GlobalLock (HOddRe); …;
        PEvenRe := EvenRe; …;
        POddRe := OddRe; …;
        //Разбивка на (не)чётные подпоследовательности
        for I := 0 to Cnt - 1 do
          begin
            if Odd (I) then
              begin
                POddRe^ := PSrcRe^; …;
                Inc (POddRe); …;
              end      else
              begin
                PEvenRe^ := PSrcRe^; …;
                Inc (PEvenRe); …;
              end;
            Inc (PSrcRe); …;
          end;
        //Рекурсивная обработка (не)чётных подпоследовательностей
        LFDFT (EvenRe, EvenIm, HalfCnt);
        LFDFT (OddRe, OddIm, HalfCnt);
        //Сборка обработанных подпоследовательностей
        PSrcRe := SrcRe; …;
        PRe := SrcRe; …;
        Inc (pRe, HalfCnt); …;
        PEvenRe := EvenRe; …;
        POddRe := OddRe; …;
        for I := 0 to HalfCnt - 1 do
          begin
            WRe := Cos (Factor * I); WIm := - Sin (Factor * I);
            PSrcRe^ := POddRe^ * WRe - POddIm^ * WIm;
            PSrcIm^ := POddRe^ * WIm + POddIm^ * WRe;
            PRe^ := pSrcRe^; …;
            PSrcRe^ := pEvenRe^ + PSrcRe^; …;
            PRe^ := PEvenRe^ - PRe^; …;
            Inc (PSrcRe); …;
            Inc (PRe); …;
            Inc (PEvenRe); …;
            Inc (POddRe); …;
          end;
        GlobalUnlock (HEvenRe); …;
        GlobalUnlock (HOddRe); …;
        GlobalFree (HEvenRe); …;
        GlobalFree (HOddRe); …;
      end;
  end;
begin
  K := 2 * Pi * Direct;
```

```
    LFDFT (Re, Im, Count);
    if Direct < 0 then
      begin //Дополнительное деление на n для обратного преобразования
        PRe := Re; …;
        for I := 1 to Count do
          begin
            PRe^ := PRe^ / Count; …;
            Inc (PRe); …;
          end;
      end;
end;
end.
```

Глава 11
АЛГОРИТМЫ ГЕНЕРАЦИИ СЛУЧАЙНЫХ ЧИСЕЛ С РАЗЛИЧНЫМИ ЗАКОНАМИ РАСПЕРЕДЕЛЕНИЯ

§1. Генерация случайных чисел с нормальным законом распределения.

1.1 Алгоритмы моделирования случайных чисел

Дискретные случайные величины.

Слова "случайная величина" в обыденном смысле употребляют тогда, когда хотят подчеркнуть, что неизвестно, каким будет конкретное значение этой величины. Причем иногда за этими словами скрывается просто незнание, какова эта величина.

Итак, чтобы задать случайную величину, надо указать, какие значения она может принимать, и каковы вероятности этих значений.

Определение

Случайная величина X называется дискретной, если она может принимать дискретное множество значений x1, x2, …, xn.

Формально случайная дискретная величина X определяется таблицей

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ p_1 & p_2 & \cdots & p_n \end{pmatrix} \qquad (11.1)$$

где x1, x2, …, xn - возможные значения величины X;

p1, p2, …, pn - соответствующие вероятности.

Точнее говоря, вероятность P{X=xi } того, что случайная величина X примет значение xi, равна:

$$P\{X = x_i\} = p_i \qquad (11.2)$$

!!!!Таблица (1.1) называется распределением случайной дискретной величины. Числа x1, x2, …, xn могут быть вообще говоря, любыми. Однако вероятности p1, p2, …, pn должны удовлетворять двум условиям:

$$p_i > 0 \qquad (11.3)$$

и

$$p_1 + p_2 + \ldots + p_n = 1 \qquad (11.4)$$

Последнее условие означает, что X обязана в каждом случае принять одно из значений x1, x2, …, xn.

Кроме распределения случайной величины, которая является исчерпывающей характеристикой, вводятся числовые характеристики, основными среди которых являются математическое ожидание и дисперсия.

Получение случайных величин на ЭВМ.

Сама постановка вопроса "получение случайных чисел на ЭВМ" иногда вызывает недоумение: ведь все, что делает компьютер, должно быть заранее запрограммировано; откуда же может появиться случайность?

Вопрос о том, можно ли с их помощью описать какое-либо явление природы, решается опытным путем. Такое описание всегда является приближенным. Более того, случайная величина, которая вполне удовлетворительно описывает какую-то физическую величину в одном классе явлений, может оказаться плохой характеристикой этой же величины при исследовании других явлений.

Обычно различают три способа получения случайных величин:

- из заранее составленных таблиц случайных чисел;
- физические генераторы случайных чисел;
- с помощью формул (генераторов или датчиков) псевдослучайных чисел.

Поскольку "качество" используемых в имитационном моделировании случайных чисел проверяется с помощью специальных тестов, можно не интересоваться тем, как эти числа получены: лишь бы они удовлетворяли принятой системе тестов.

Числа, получаемые по какой-либо формуле и имитирующие значения случайной величины X, называются псевдослучайными числами. Под словом "имитирующие" подразумевается, что эти числа удовлетворяют ряду тестов так, как если бы они были значениями этой случайной величины.

Основой или «сырьем» для моделирования случайных величин с заданным законом распределения являются так называемые базовые случайные числа.

Определение. Совокупность {Ri}, i=1,2, … независимых равномерно распределенных на отрезке [0,1] случайных величин называется последовательностью базовых случайных чисел.

Мы называем эти числа псевдослучайными потому, что фактически они остаются полностью детерминированными в том смысле, что если каждое обращение к соответствующей формуле (точнее, к алгоритму) начинается с одними и теми же исходными данными (константами и начальными значениями), то на выходе получаются одинаковые последовательности чисел R.

В настоящее время почти все стандартные библиотечные программы вычисления равномерных случайных

чисел основаны на конгруэнтных1 методах, разработанных Лемером.
Основная формула мультипликативного конгруэнтного метода Лемера имеет вид:

$$R_{i+1} = aR_i (\text{mod } m) \quad (11.5)$$

где a и m - неотрицательные целые числа.
Согласно этому выражению, нужно взять случайное число Ri, умножить его на постоянный коэффициент a и взять модуль полученного числа по m (т.е. разделить на aRi и остаток считать как Ri+1). Поэтому для вычисления (или генерирования) последовательности Ri нам необходимы начальные значения R0, множитель a и модуль m. Выбираются a, R0 и m так, чтобы обеспечить максимальную длину (или, как говорят период) неповторяющейся последовательности Ri и минимальную корреляцию между генерируемыми числами. Базовые случайные числа позволяют генерировать новые случайные последовательности, подчиняющиеся любому закону распределения.
Существует два основных пути преобразования базовых случайных чисел {Ri}, в случайные числа {yi}, распределенные по заданному закону распределения.
Один из них, который называется методом инверсии, состоит в реализации определенных арифметических операций над базовым числом Ri, чтобы получить yi.
Второй метод основывается на моделировании условий соответствующей предельной теоремы теории вероятностей. Кроме указанных двух основных подходов можно также выделить эвристические способы генерирования случайных чисел.
Моделирование нормальной случайной величины на основе центральной предельной теоремы.
Нормальное (или гауссово) распределение (рисунок 11.1)—это, несомненно, один из наиболее важных и часто используемых в имитационном моделировании видов непрерывных распределений.

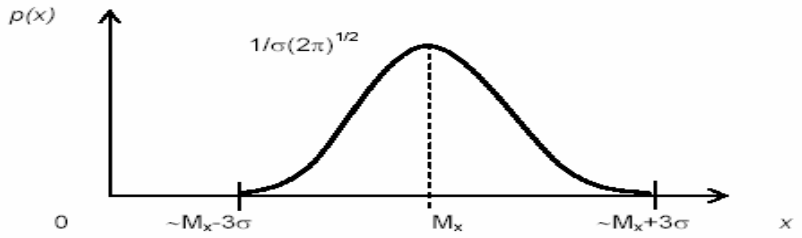


Рис. 2. Нормальная (гауссовская) плотность вероятностей

Рисунок 11.1 Нормальная плотность распределения !!!

Плотность вероятности нормальной плотности записывается так:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-M_i)^2}{2\delta^2}},$$

где Mx – математическое ожидание;
σ - среднеквадратическое отклонение.
Функция распределения нормальной случайной величины равна:

$$P(x) = \frac{l}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(x-M_x)^2}{2\delta^2}} dx$$

Поэтому алгоритмы моделирования нормальных случайных чисел базируются на предельных теоремах теории вероятностей.
Центральная предельная теорема говорит о том, что сумма n одинаково распределенных независимых случайных величин x со средним Mx и дисперсией Dx стремится к нормально распределенной величине с параметрами nMx и nDx при бесконечном увеличении n.
Следствием теоремы является, в частности, и то, что для получения нормальной выборки, можно воспользоваться базовыми случайными числами R.
Идея алгоритма состоит в следующем. Определим новую случайную величину s в виде суммы базовых чисел Ri, (i=1,2,3, …,n):

$$s = R_1 + R_2 + \ldots + R_n \tag{11.6}$$

Тогда, согласно утверждению центральной предельной теоремы, случайная величина s является асимптотически нормальной величиной с математическим ожиданием Ms и дисперсией Ds равными соответственно:

$$M_s = n/2 \tag{11.7}$$

и

$$D_s = n/12 \tag{11.8}$$

Для практического использования формула (11.6) неудобна, поэтому введем вспомогательную случайную величину z равную

$$Z = \frac{(s - n/2)}{\sqrt{n/12}} \tag{11.9}$$

Из (11.9) следует, что z - случайная величина, распределенная по нормальному закону с нулевым средним и единичной дисперсией. Тогда для любого нормального распределения со средним μ и дисперсией σ2 случайное отклонение y, соответствующее указанным выше n случайным числам, получается из формулы

$$(y - \mu)/\sigma = z = \frac{(s - n/2)}{\sqrt{n/12}} \tag{11.10}$$

Следовательно

$$y = \mu + \frac{\sigma(s - n/2)}{\sqrt{n/12}} = \mu + (\frac{\sigma}{\sqrt{n/12}})(\sum_{i=1}^{n} R_i - n/2) \tag{11.11}$$

Согласно той же предельной теореме, нормальность достигается быстро даже при сравнительно небольших значениях n. В практических задачах n обычно принимается равным 12. При этом последняя формула упрощается и принимает вид

$$y = \mu + \sigma(\sum_{i=1}^{12} R_i - 6) \tag{11.12}$$

Формула (11.12) и дает алгоритм моделирования нормальных случайных чисел с требуемыми параметрами μ и σ2.

Описанный метод считается малоэффективным, так как требует генерации нескольких случайных базовых чисел R для получения одного нормального выборочного значения y. Можно оценить относительную ошибку εм математического ожидания и дисперсии εD последовательности случайных нормальных чисел {yi} в зависимости от числа используемых базовых чисел n в формуле (11.11).

§2. Генерация случайных чисел с экспоненциальным законом распределения

Как известно, случайная величина x, распределенная по экспоненциальному закону описывается следующей плотностью распределения:

$$p(x) = \lambda \cdot e^{-\lambda \cdot x} \tag{11.13}$$

На рисунке 11.2 построены графики экспоненциальных плотностей распределения при различных значениях параметра λ.

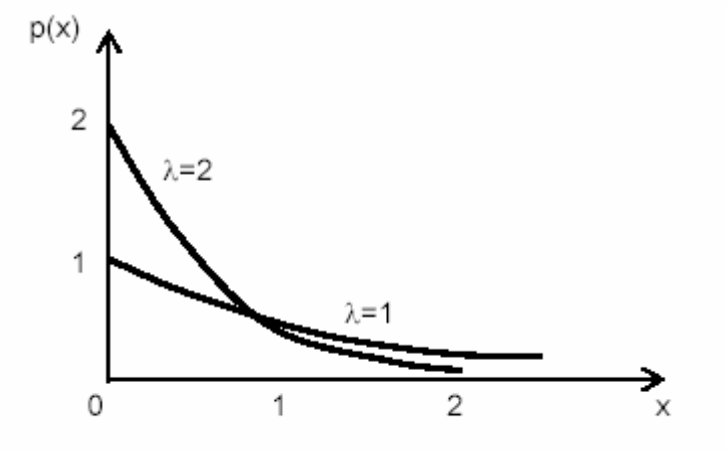


Рисунок 11.2 Экспоненциальная плотность вероятностей с разными значениями параметра

Экспоненциальному распределению, как правило, подчиняется случайный интервал времени τ между поступлениями заявок в систему массового обслуживания. Поэтому весьма важно уметь моделировать потоки заявок разной интенсивности.

Напомним, что математическое M[τ] ожидание экспоненциально распределенной случайной величины τ равно

$$M[\tau] = 1/\lambda$$

а дисперсия

$$D[\tau] = 1/\lambda^2$$

Чтобы найти алгоритм имитации экспоненциально распределенных чисел τ, применим метод инверсии:

$$\int_0^{\tau} \lambda e^{-\lambda x} = R \tag{11.14}$$

$$1 - e^{-\lambda\tau} = R \tag{11.15}$$

откуда

$$\tau = -\frac{1}{\lambda}\ln(1-R) \tag{11.16}$$

но, поскольку случайная величина (1 - R) распределена точно так же, как R, и находится в том же интервале (0,1), то (11.16) можно заменить на более удобную формулу:

$$\tau = -\frac{1}{\lambda}\ln R \tag{11.17}$$

что дает искомый ответ.

§3. Генерация случайных чисел с равномерным законом распределения на отрезке (a,b).

Предположим, что нам необходимо составить программу для моделирования входного потока заявок распределенного по равномерному закону в интервале (a,b). Уравнение метода инверсии для рассматриваемого случая выглядит так

$$\int_a^y \frac{dy}{b-a} = R$$

где R – равномерно распределенное случайное число на (0,1), т.е. базовое число.

Это интегральное уравнение решается легко и ответ ясен:

$$\frac{y-a}{b-a} = R$$

Отсюда мы имеем явное выражение для y:

$$y = a + R(b-a)$$

где R – базовое случайное число.

§4. Генерация случайных чисел с распределением Пуассона.

Распределение Пуассона моделирует случайную величину, равную числу событий, произошедших за фиксированное время, при условии, что данные события происходят с некоторой фиксированной средней интенсивностью и независимо друг от друга.

Распределение Пуассона - это дискретное распределение, являющееся одним из важных предельных случаев биномиального распределения. При росте n и зафиксированном значении произведения np=λ > 0 биномиальное распределение B(n,p) сходится к распределению Пуассона. Таким образом, случайная величина, имеющая распределение Пуассона с параметром λ, принимает неотрицательные целые значения с вероятностью P.

Выберем фиксированное число λ > 0 и определим дискретное распределение, задаваемое следующей функцией вероятности:

$$p_n = P(\xi = n) = \frac{\lambda^n}{n!} e^{-\lambda}$$

Функция распределения равна

$$F(x) = e^{-\lambda} \sum_{n=0}^{\lfloor x \rfloor} \frac{\lambda^n}{n!}$$

Параметр λ является одновременно и математическим ожиданием, и дисперсией случайной величины, имеющей распределение Пуассона. Часто параметр называется интенсивностью.

Производящая функция моментов распределения Пуассона имеет вид:

$$M_y(t) = e^{\lambda(e^t - 1)},$$

откуда

$$\mathbb{E}[Y] = \lambda,$$
$$\mathrm{D}[Y] = \lambda.$$

Классическим примером случайной величины, распределенной по Пуассону, является количество машин, проезжающих через какой-либо участок дороги за заданный период времен. Также можно отметить такие примеры, как количество звезд на участке неба заданной величины, количество ошибок в тексте заданной длины, количество телефонных звонков в call-центре или количество обращений к веб-серверу за заданный период времени.

Биномиальное распределение B(n,p) - это распределение числа успехов в серии из n экспериментов, каждый из которых завершается успехом с вероятностью p. Важными предельными случаями биномиального распределения являются распределение Пуассона и нормальное распределение.

Если X - биномиальная случайная величина, то

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$
$$P(X \le k) = I_{1-p}(n-k, k+1)$$

§5. Генерация случайных чисел с показательным законом распределения.

Определение. Показательным (экспоненциальным) называется распределение вероятностей непрерывной случайной величины X, которое описывается плотностью

$$f(x) = \begin{cases} 0, при x < 0 \\ \lambda e^{-\lambda x}, при x \ge 0 \end{cases}$$

где λ - положительное число.

Найдем закон распределения.

$$F(x) = \int_{-\infty}^{x} f(x)dx = \int_{-\infty}^{0} 0\,dx + \lambda \int_{0}^{x} e^{-\lambda x} dx = 1 - e^{-\lambda x}$$

$$F(x) = \begin{cases} 0, при x < 0 \\ 1 - e^{-\lambda x}, при x \ge 0 \end{cases}$$

Графики функции распределения(рисунок 11.3) и плотности распределения(рисунок 11.4):

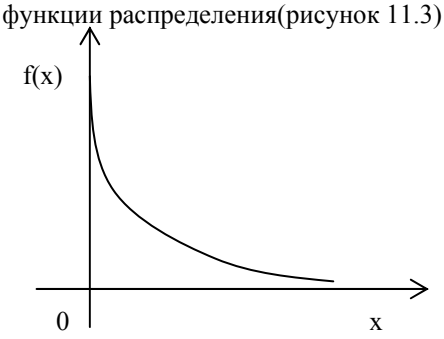


Рисунок 11.3

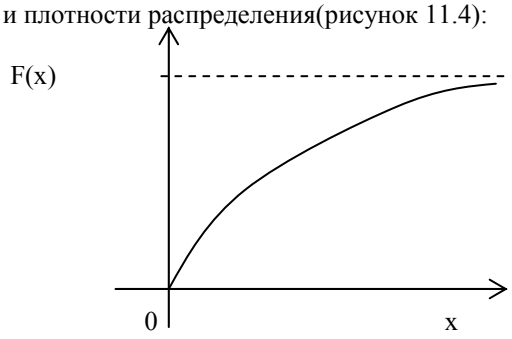


Рисунок 11.4

Найдем математическое ожидание случайной величины, подчиненной показательному распределению.

$$m_x = \int_{-\infty}^{\infty} xf(x)dx = \int_0^{\infty} x\lambda e^{-\lambda x}dx = \left\{\begin{matrix} u = x, e^{-\lambda x}dx = dv, \\ du = dx, -\frac{e^{-\lambda x}}{\lambda} = v, \end{matrix}\right\} = \lambda\left(-\frac{xe^{-\lambda x}}{\lambda}\Big|_0^{\infty} + \int_0^{\infty}\frac{e^{-\lambda x}}{\lambda}dx\right) =$$

$$= \int_0^{\infty} e^{-\lambda x}dx = -\frac{e^{-\lambda x}}{\lambda}\Big|_0^{\infty} = \frac{1}{\lambda}$$

Результат получен с использованием того факта, что

$$xe^{-\lambda x}\Big|_0^{\infty} = \lim_{x\to\infty}\frac{x}{e^{\lambda x}} - 0 = \left\{\begin{matrix} По & правилу \\ Лопиталя & \end{matrix}\right\} = \lim_{ч\to\infty}\frac{1}{-\lambda e^{\lambda x}} = 0$$

Для нахождения дисперсии найдем величину M(X2).

$$M(X^2) = \int_{-\infty}^{\infty} x^2 f(x)dx = \int_0^{\infty} \lambda x^2 e^{-\lambda x}dx$$

Дважды интегрируя по частям, аналогично рассмотренному случаю, получим:

$$M(X^2) = \frac{2}{\lambda^2}$$

$$D(X) = M(X^2) - \left[M(X)\right]^2 = \frac{1}{\lambda^2}$$

Тогда

$$M(X) = \frac{1}{\lambda}; D(X) = \frac{1}{\lambda^2}; \sigma_x = \frac{1}{\lambda}$$

Итого:

Видно, что в случае показательного распределения математическое ожидание и среднее квадратическое отклонение равны.

Также легко определить и вероятность попадания случайной величины, подчиненной показательному закону распределения, в заданный интервал.

$$P(a < x < b) = F(b) - F(a) = e^{-\lambda a} - e^{-\lambda b}$$

Показательное распределение широко используется в теории надежности.

Допустим, некоторое устройство начинает работать в момент времени t0=0, а через какое– то время t происходит отказ устройства.

Обозначим T непрерывную случайную величину – длительность безотказной работы устройства.

Таким образом, функция распределения F(t) = P(T<t) определяет вероятность отказа за время длительностью t.

Вероятность противоположного события (безотказная работа в течение времени t) равна R(t)=P(T>t) = 1–F(t).

§6. Примеры программ генераторов случайных чисел.

AP.cs
Алгоритмы
Библиотека AP для C# содержит базовый набор математических функций и классов, которые требуются для работы программ.

HQRND.cs
Алгоритмы
В состав модуля входят два типа подпрограмм: подпрограммы для работы с низкоуровневым генератором и для его настройки, и высокоуровневые подпрограммы, реализованные на базе низкоуровневого генератора.
К первой категории относятся подпрограммы RndIntegerBase, RndIntegerMax и RndInitialize. Первая подпрограмма возвращает случайное число в диапазоне от 0 (не включительно) до границы, возвращаемой подпрограммой RndIntegerMax. Эта подпрограмма является основой для всех генераторов случайных чисел в этом модуле. Подпрограмма RndInitialize служит для инициализации генератора и является аналогом подпрограммы randomize, с тем отличием, что она принимает не одно значение seed-переменной, а два.
Ко второй категории относится целый ряд подпрограмм. Для получения случайных целых чисел, расположенных в указанном диапазоне, может использоваться подпрограмма RndUniformI. Подпрограмма RndUniformR возвращает вещественные числа в диапазоне (0, 1). Подпрограммы RndNormal и RndNormal2 служат для генерации нормально распределенных случайных чисел. Первая из них генерирует одно нормально распределенное случайное число, вторая - пару независимых чисел (при этом по трудоемкости обе подпрограммы одинаковы, поэтому для генерации больших количеств случайных чисел стоит пользоваться второй подпрограммой). Подпрограмма RndExponential служит для генерации экспоненциально распределенных случайных чисел.

POISSONDISTR.cs
Алгоритмы
Подпрограммы PoissonDistribution и PoissonCDistribution предназначены для вычисления площади под левым и под правым хвостами графика (т.е. для вычисления интегральной функции вероятности и её дополнения). Подпрограмма InvPoissonDistribution вычисляет функцию, обратную к интегральной функции вероятности (подбирает такое значение параметра λ, что $F_{\lambda}(x)=y$).

NORMALDISTR.cs
Алгоритмы
Нормальное распределение, функции erf и erfc. Нормальное распределение, также известное, как распределение Гаусса, является одним из наиболее известных непрерывных распределений. Строго говоря, существует целое семейство нормальных распределений, различающихся лишь масштабом и смещением. Здесь и далее под нормальным распределением подразумевается т.н. стандартное нормальное распределение - нормальное распределение с нулевым математическим ожиданием и единичным стандартным отклонением.
Подпрограммы Erf и ErfC служат для вычисления значения специальной функции erf(x) и её дополнения до единицы. Функция, обратная к erf(x) вычисляется подпрограммой InvErf.
Интегральная функция вероятности нормального распределения вычисляется подпрограммой NormalDistribution. Функция, обратная к интегральной функции распределения, вычисляется подпрограммой InvNormalDistribution.