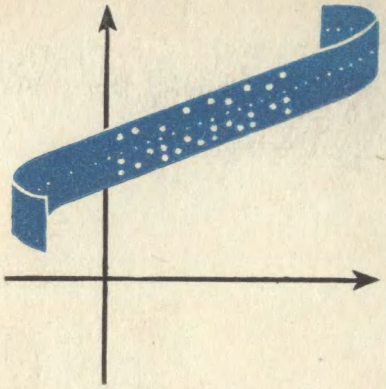


Р.С.ГУТЕР
П.Т.РЕЗНИКОВСКИЙ

ПРОГРАММИРОВАНИЕ И ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

ВЫПУСК 2

Р. С. ГУТЕР, П. Т. РЕЗНИКОВСКИЙ

ПРОГРАММИРОВАНИЕ И ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

ВЫПУСК ВТОРОЙ

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ МЕТОДОВ

Допущено Министерством приборостроения, средств автоматизации и систем управления в качестве учебника для средних специальных учебных заведений по специальности «Прикладная математика»

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
МОСКВА 1971

518
Г 97
УДК 518.5

Программирование и вычислительная математика, вып. 2. Вычислительная математика. Программная реализация вычислительных методов. Гутер Р. С., Резниковский П. Т.

Книга является учебником для специальности «Прикладная математика» в средних специальных учебных заведениях (техникумах) и соответствует утвержденной программе. Выпуск 2 рассчитан на второй курс. Книга может быть также использована студентами вузов, инженерами и научными работниками нематематических специальностей для изучения методов вычислительной математики и программирования этих методов.

В учебнике рассматриваются: вычисление элементарных функций, решение уравнений и систем уравнений, интерполяция, численное интегрирование и численное решение дифференциальных уравнений. Кроме основных вычислительных схем и примеров ручных расчетов, приводится также программирование численных методов в содержательных обозначениях для трехадресных машин типа М-20 и на алголе.

Рисунков 20, таблиц 62.

2-2-4
24-71

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Эта книга является вторым выпуском учебника по курсу «Программирование и вычислительная математика» для математических техникумов. Она написана по инициативе лаборатории прикладной математики Института содержания и методов обучения Академии педагогических наук СССР и соответствует программе упомянутого курса для второго года обучения. Как указывалось в предисловии к первому выпуску, книга может быть использована в качестве пособия и для других специальностей техникумов, а также во втузах.

Книга содержит методы вычисления элементарных функций, численного решения уравнений и систем, интерполирования, приближенного интегрирования и численного решения дифференциальных уравнений. Излагаемый в ней материал требует знакомства с основными понятиями математического анализа: производной, определенным интегралом, дифференциальным уравнением в объеме книги Н. Я. Виленкина и С. И. Шварцбурда «Математический анализ».

В книге детально разобраны основные методы вычислительной математики. Кроме подробного разбора алгоритмов и вычислительных схем, рассматривается программирование численных методов на языке содержательных обозначений (для вычислительных машин типа М-20) и на алголе.

Материал, не относящийся к программной реализации численных методов, может изучаться независимо от первого выпуска учебника.

Весь материал книги был проверен в практике преподавания в Московском математическом техникуме. При написании книги мы использовали третью часть пособия

для средних школ с математической специализацией «Программирование и вычислительная математика» Р. С. Гутера, Б. В. Овчинского и П. Т. Резниковского, изд-во «Наука», Главная редакция физико-математической литературы, 1965 г.

В книге введена сплошная нумерация глав и параграфов. Формулы, примеры и таблицы нумеруются в каждом параграфе заново, причем после номера формулы (примера, таблицы) указывается соответствующий параграф.

Большую помощь в работе над книгой нам оказала Т. А. Муратова, выполнившая или проверившая все вычисления. С. И. Шварцбурд и В. Л. Арлазаров внимательно прочли рукопись и сделали ряд существенных замечаний. Очень большое число различных исправлений и улучшений внес в рукопись ее редактор И. А. Румянцев. Всем названным лицам авторы выражают свою глубокую благодарность.

Авторы

ГЛАВА I

ВЫЧИСЛЕНИЕ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ

§ 1. Общие замечания. Вычисление многочленов. Схема Горнера

Почти во всех случаях при выполнении вычислений используются значения тех или иных элементарных функций. При этом, как уже говорилось в первом выпуске (см. § 5), к элементарным функциям относят степенные функции, многочлены и алгебраические функции, логарифмическую, показательную, тригонометрические и обратные им, а также различные комбинации перечисленных функций. При ручных расчетах всегда можно воспользоваться для этой цели готовыми таблицами, которые достаточно разнообразны, и, быть может, еще интерполяцией по ним. Исключение составляют здесь лишь самые простые функции — многочлены, которые приходится вычислять, так как все многочлены затабулировать невозможно.

Иначе обстоит дело при вычислениях на электронных вычислительных машинах, так как запись в память машины больших таблиц элементарных функций оказывается совершенно невозможной из-за ограниченного объема памяти. Поэтому для работы на машине необходимо иметь стандартные программы вычисления элементарных функций. Принципам, на которых основаны такие программы, и посвящена настоящая глава.

Начнем с вычисления многочленов, необходимого и при ручном счете. При однократном вычислении значения многочлена невысокой степени последовательность выполнения операций не имеет особого значения. Однако для вычисления многочленов достаточно высокой степени

или же для вычисления многих значений многочлена в различных точках последовательность выполнения операций уже существенна.

Предварительное вычисление всех нужных степеней аргумента x^2, x^3,... обычно является невыгодным, так как требует довольно большого числа операций. При вычислении значений многочлена n-й степени для получения степеней до x^n включительно требуется $n-1$ умножений. Кроме этого, нужно еще n умножений на коэффициенты, т. е. всего $2n-1$ умножений, и n сложений.

Меньшего количества действий — n умножений и n сложений — требует вычисление многочлена по так называемой *схеме Горнера*, с которой мы сейчас познакомимся.

Из курса алгебры известна *теорема Безу*, которая состоит в том, что *остаток от деления многочлена $f(x)$ на двучлен $x-\alpha$ равен значению многочлена при $x=\alpha$, т. е. $f(\alpha)$*. Обозначив частное от деления многочлена n-й степени $f(x)$ на $x-\alpha$ через $\varphi(x)$, а остаток — через b_n, можем записать

$$f(x)=(x-\alpha)\,\varphi(x)+b_n, \tag{1.1}$$

где, как мы уже знаем, $b_n=f(\alpha)$.

Если

$$f(x)=a_0x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n$$

и

$$\varphi(x)=b_0x^{n-1}+\ldots+b_{n-2}x+b_{n-1},$$

то мы имеем тождество

$$\begin{aligned}a_0x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n=\\ =(x-\alpha)\,(b_0x^{n-1}+\ldots+b_{n-2}x+b_{n-1})+b_n.\end{aligned} \tag{2.1}$$

Раскрывая скобки в правой части тождества (2.1) и сравнивая коэффициенты в левой и правой частях тождества, находим

$$\left.\begin{aligned}a_0&=b_0,\\ a_1&=b_1-\alpha b_0,\\ a_2&=b_2-\alpha b_1,\\ &\ldots\ldots\ldots\\ a_{n-1}&=b_{n-1}-\alpha b_{n-2},\\ a_n&=b_n-\alpha b_{n-1}.\end{aligned}\right\} \tag{3.1}$$

Равенства (3.1) можно переписать в виде

$$\left.\begin{aligned} b_0 &= a_0, \\ b_1 &= a_1 + \alpha b_0, \\ b_2 &= a_2 + \alpha b_1, \\ &\cdots\cdots\cdots \\ b_{n-1} &= a_{n-1} + \alpha b_{n-2}, \\ b_n &= a_n + \alpha b_{n-1}. \end{aligned}\right\} \qquad (4.1)$$

С помощью полученных выражений можно вычислять последовательно все коэффициенты частного и остаток b_n, равный $f(\alpha)$. Эта схема вычисления $f(\alpha)$ и называется *схемой Горнера*.

При ручном счете (если речь идет о нахождении одного значения многочлена) вычисления записывают с помощью схемы

$$\begin{array}{c|cccccc} & a_0 & a_1 & a_2 & \dots & a_{n-1} & a_n \\ \hline \alpha & b_0 & b_1 & b_2 & \dots & b_{n-1} & b_n \end{array}.$$

Числа в нижней строке вычисляются последовательно слева направо, причем $b_0 = a_0$, а каждое следующее число b_k равно сумме коэффициента a_k, стоящего над ним, и произведения предыдущего коэффициента b_{k-1} на α.

Пример 1.1. Вычислим частное и остаток от деления многочлена $x^4 - 3x^3 + 2x^2 + x + 3$ на двучлен $x - 1{,}3$. Пользуясь приведенной схемой, находим

| 1	— 3	2	1	3
1,3 | 1	$-3+1\times 1{,}3 = -1{,}7$	$2+(-1{,}7)\times 1{,}3 = -0{,}21$	$1+(-0{,}21)\times 1{,}3 = 0{,}727$	$3+0{,}727\times 1{,}3 = 3{,}9451$

Итак, частное есть многочлен $x^3 - 1{,}7x^2 - 0{,}21x + 0{,}727$, а остаток, равный значению многочлена в точке $x = 1{,}3$, есть 3,9451.

Если в формулах (4.1) подставить каждую предыдущую формулу в последующую, то получится такая цепочка равенств:

$$\begin{aligned} b_1 &= a_1 + \alpha a_0, \\ b_2 &= a_2 + \alpha(a_1 + \alpha a_0), \\ b_3 &= a_3 + \alpha(a_2 + \alpha(a_1 + \alpha a_0)), \\ &\cdots\cdots\cdots\cdots \end{aligned}$$

Дойдя до последней формулы и переписав правую часть в обратном порядке, получим такое равенство:

$$b_n = f(\alpha) = (\ldots((a_0\alpha + a_1)\alpha + a_2)\alpha + \ldots a_{n-1})\alpha + a_n. \qquad (5.1)$$

Таким образом, для вычисления значения многочлена

$$f(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n$$

его следует представлять в виде

$$f(x) = (\ldots((a_0x + a_1)\cdot x + a_2)\cdot x + \ldots + a_{n-1})\cdot x + a_n.$$

Пример 2.1. Составим таблицу значений многочлена

$$x^4 - 3x^3 + 2x^2 + x + 3$$

для значений x от 1 до 2 с шагом 0,1.

Расписка для выполнения вычислений вручную и все вычисления приведены в табл. 1.1. В ней выписаны результаты всех промежуточных вычислений. При работе на клавишных вычислительных машинах ряд промежуточных колонок можно опускать не выписывая. На

Таблица 1.1

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
x	(1) — «3»	(2) · (1)	(3)+«2»	(4) · (1)	(5)+«1»	(6) · (1)	(7)+«3»
1	—2,0	—2,00	0	0	1,000	1,0000	4,0000
1,1	—1,9	—2,09	—0,09	—0,099	0,901	0,9911	3,9911
1,2	—1,8	—2,16	—0,16	—0,192	0,808	0,9696	3,9696
1,3	—1,7	—2,21	—0,21	—0,273	0,727	0,9451	3,9451
1,4	—1,6	—2,24	—0,24	—0,336	0,664	0,9296	3,9296
1,5	—1,5	—2,25	—0,25	—0,375	0,625	0,9375	3,9375
1,6	—1,4	—2,24	—0,24	—0,384	0,616	0,9856	3,9856
1,7	—1,3	—2,21	—0,21	—0,357	0,643	1,0931	4,0931
1,8	—1,2	—2,16	—0,16	—0,288	0,712	1,2816	4,2816
1,9	—1,1	—2,09	—0,09	—0,171	0,829	1,5751	4,5751
2	—1,0	—2,00	0	0	1,000	2,0000	5,0000

клавишной машине и даже на ручном арифмометре к произведению легко прибавить любое слагаемое, поэ-

тому можно пользоваться распиской, приведенной в таблице 2.1. После получения суммы ее приходится сбрасывать с регистра результата и набирать вновь на регистре набора множителя. Поэтому сумму следует записать, а значит, в этом месте можно перейти к вычислению значений в следующей строке.

Таблица 2.1

(1)	(2)	(3)	(4)	(5)
x	(1)—«3»	(2) · (1)+«2»	(3) · (1)+«1»	(4) · (1)+«3»

Если на машине можно переносить число с регистра результата на регистр установки множителя, то и эта расписка не требуется, так как в этом случае все вычисления для нахождения значения многочлена в одной точке можно выполнять, не сбрасывая результата, а значит, нет никакой надобности в записи промежуточных результатов.

Программирование схемы Горнера для вычисления значений многочлена в одной точке не представляет никакого труда. Для многочлена невысокой степени проще всего писать обычную бесцикловую расписку формулы. Для многочленов достаточно высокой степени (например, $n \geqslant 6$) удобнее программировать цикл с переадресацией, предполагая, что коэффициенты многочлена расположены в ячейках памяти подряд. Если использовать регистр адреса, то программу можно представить, например, так:

$$\begin{array}{llll} PA & 0^* & 0 & \Omega-1 \\ & & a_0=\gamma & \\ & \alpha \quad \cdot & \gamma=\gamma & \\ & \gamma \quad + & a_1^*=\gamma & \uparrow \\ PA<n-1 & & \overline{1}^* & \\ Б & & \Omega-1 & \end{array}$$

§ 2. Вычисление элементарных функций с помощью степенных рядов

Степенные ряды являются очень важным аппаратом для табулирования элементарных функций, так как их применение позволяет свести задачу вычисления значений функции к задаче вычисления многочлена, т. е. к выполнению арифметических операций.

Разложения основных элементарных функций в степенные ряды известны из курса математического анализа. Для других функций эти разложения могут быть получены путем комбинации известных разложений или с помощью общей формулы Тейлора.

Формула Тейлора имеет вид

$$f(x)=f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\ldots+ \\ +\frac{f^{(n)}(a)}{n!}(x-a)^n+\ldots \qquad (1.2)$$

Обычно в ней полагают $a=0$*), благодаря чему получается обычный степенной ряд по степеням x, обрывая который в нужном месте, мы получаем многочлен, приближающий данную функцию.

Наиболее употребительными являются степенные ряды для функций e^x, $\cos x$ и $\sin x$, которые сходятся при любом значении x:

$$\left.\begin{aligned} e^x &= 1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}+\ldots, \\ \cos x &= 1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots+(-1)^n\frac{x^{2n}}{(2n)!}+\ldots, \\ \sin x &= x-\frac{x^3}{3!}+\frac{x^5}{5!}-\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}+\ldots \end{aligned}\right\} \qquad (2.2)$$

Для логарифмической функции непосредственно из формулы Тейлора (1.2) можно получить ряд

$$\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\ldots+(-1)^{n-1}\frac{x^n}{n}+\ldots,$$

*) Распространенным, хотя и исторически необоснованным названием для этого частного случая является название *формула Маклорена*.

который, однако, мало пригоден для вычисления логарифмов, так как сходится только для значений x, удовлетворяющих условию $-1<x\leqslant 1$. Практически для вычисления значений натурального логарифма используют получающийся из приведенной формулы ряд

$$\ln\frac{1+x}{1-x}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots+\frac{x^{2n+1}}{2n+1}+\ldots\right). \qquad (3.2)$$

Этот ряд сходится лишь при $|x|<1$, но дробь $\frac{1+x}{1-x}$ может при этом принимать любые положительные значения. Например, для нахождения величины $\ln 5$ достаточно положить $\frac{1+x}{1-x}=5$, откуда $x=\frac{2}{3}$. Таким образом,

$$\ln 5=2\left[\frac{2}{3}+\frac{1}{3}\left(\frac{2}{3}\right)^3+\frac{1}{5}\left(\frac{2}{3}\right)^5+\ldots\right].$$

Часто используется также и *биномиальный ряд*, частным случаем которого при натуральном m является известная формула бинома Ньютона:

$$(1+x)^m=1+mx+\frac{m(m-1)}{2!}x^2+\frac{m(m-1)(m-2)}{3!}x^3+$$
$$+\ldots+\frac{m(m-1)\ldots(m-n+1)}{n!}x^n+\ldots \qquad (4.2)$$

Действительно, при натуральном m коэффициенты ряда, начиная с некоторого места, обратятся в нуль, т. е. ряд оборвется.

Биномиальный ряд удобен при возведении в дробную степень и при извлечении корней.

Просто выглядит и легко получается также ряд для функции $\operatorname{arctg} x$:

$$\operatorname{arctg} x=x-\frac{x^3}{3}+\frac{x^5}{5}-\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}+\ldots, \qquad (5.2)$$

который, как и ряд для логарифмической функции, сходится лишь в области $-1\leqslant x\leqslant 1$. Тем не менее, для вычисления $\operatorname{arctg} x$ ряда (5.2) вполне достаточно, потому что для $|x|>1$ можно воспользоваться тождеством

$$\operatorname{arctg} x=\frac{\pi}{2}-\operatorname{arctg}\frac{1}{x}.$$

При вычислении элементарных функций с помощью степенных рядов часто бывает удобно пользоваться *рекуррентными соотношениями*, которые позволяют вычислять очередной член ряда не непосредственно, а через уже вычисленные предыдущие члены.

Рекуррентным (или *возвратным*) соотношением называют равенство, связывающее между собою два или несколько соседних членов последовательности или ряда. С помощью такого равенства можно определить последующий член ряда через предыдущие. В некоторых случаях последовательность задается не выражением общего члена, а заданием ее первых членов и определяющего остальные члены рекуррентного соотношения. Так задается, например, известная последовательность *чисел Фибоначчи*, 1, 1, 2, 3, 5, 8, 13, 21, 34, ..., в которой каждое последующее число равно сумме двух предыдущих:

$$F_0=1, \quad F_1=1, \quad F_{n+1}=F_n+F_{n-1}.$$

Это соотношение много проще, чем выражение для общего члена

$$F_n=\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}-\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right].$$

Последовательности, определяемые одним или несколькими рекуррентными соотношениями, обычно называют *рекуррентными* (или *возвратными*) *последовательностями*.

Для приведенных выше рядов рекуррентные соотношения могут быть легко выведены непосредственно. Проще всего взять отношение двух соседних членов. Рассмотрим, например, степенной ряд (2.2) для функции e^x. Его общий член имеет вид $a_n=\frac{x^n}{n!}$. Взяв отношение последующего члена к предыдущему, получим

$$a_{n+1}:a_n=\frac{x^{n+1}}{(n+1)!}:\frac{x^n}{n!}=\frac{x}{n+1}.$$

Таким образом, для двух соседних членов ряда получаем рекуррентное соотношение

$$a_n=a_{n-1}\cdot(x/n), \tag{6.2}$$

которое очень удобно для вычисления членов ряда последовательно.

При вычислениях с рядами мы заменяем сумму ряда его частичной суммой, т. е. ограничиваемся конечным числом членов. Естественно возникает вопрос об оценке погрешности сделанного приближения в том случае, когда

мы сохраняем данное число членов ряда. Решение вопроса в такой форме дает возможность решить и основной практический вопрос, который ставится таким образом: сколько членов ряда нужно сохранить, чтобы получающаяся погрешность не превышала заданной?

Если члены ряда убывают достаточно быстро и притом с самого начала, то выгодно иметь дело со знакопеременным рядом *), погрешность которого легко оценивать. Действительно, из свойств рядов известно, что *сумма знакопеременного ряда меньше его первого члена* (по абсолютной величине). Отсюда следует, что, заменяя сумму ряда его частичной суммой, мы допускаем погрешность, не превосходящую по модулю *первого из отброшенных членов*. Однако нужно помнить, что эта оценка полезна лишь при указанных выше условиях. Если же члены ряда убывают медленно или убывают хотя и быстро, но не сначала, а первые члены ряда достаточно велики, то хотя общая теорема о сумме ряда остается в силе, но фактическая погрешность будет заметно большей из-за погрешностей вычитания первых больших членов. В таких случаях гораздо выгоднее иметь дело с рядами, все члены которых положительны.

Например, члены рядов (2.2) убывают достаточно быстро и ряды сходятся при любом x. Тем не менее при больших x (скажем, $x > 10$) первые члены этих рядов довольно быстро возрастают, и поэтому вычисление $\cos x$ и $\sin x$ с помощью этих рядов чрезвычайно затруднительно, потому что при вычитании больших первых членов происходит такая потеря точности, которую нельзя возместить вычислением большого числа слагаемых. Для рядов с положительными членами оценка погрешности более сложна, и никаких общих методов, пригодных для всех рядов, предложить нельзя. Рассмотрим такую оценку на конкретном примере.

Пример 1.2. Вычислим значение $\ln 5$ с помощью ряда (3.2), взяв пять членов ряда, и оценим полученную погрешность.

---

*) Ряд называют *знакопеременным* (или *знакочередующимся*), если его члены попеременно меняют знаки. Примерами знакопеременных рядов являются степенные ряды для синуса и косинуса.

Как уже указывалось выше, в этом случае надо положить $x=2/3$. Тогда

$$\ln 5 \approx 2\left[\frac{2}{3}+\frac{1}{3}\left(\frac{2}{3}\right)^3+\frac{1}{5}\left(\frac{2}{3}\right)^5+\frac{1}{7}\left(\frac{2}{3}\right)^7+\frac{1}{9}\left(\frac{2}{3}\right)^9\right].$$

Абсолютная погрешность этого равенства равна сумме ряда

$$R=2\left[\frac{1}{11}\left(\frac{2}{3}\right)^{11}+\frac{1}{13}\left(\frac{2}{3}\right)^{13}+\ldots\right].$$

Оценку величины R можно получить следующим образом. Если в квадратной скобке все множители перед степенями 2/3 заменить на 1/11, то величина суммы может только возрасти. Следовательно,

$$R<\frac{2}{11}\left[\left(\frac{2}{3}\right)^{11}+\left(\frac{2}{3}\right)^{13}+\ldots\right].$$

Но сумма справа теперь представляет собой сумму геометрической прогрессии с первым членом $(2/3)^{11}$ и знаменателем $(2/3)^2=4/9$. Поэтому

$$R<\frac{2}{11}\frac{(2/3)^{11}}{1-4/9}=\frac{18}{55}\left(\frac{2}{3}\right)^{11}<0{,}004.$$

Приведенный метод позволяет решить и обратную задачу — определить число членов ряда, которые нужно сохранить, чтобы получить значение $\ln 5$ с наперед заданной точностью.

Скорость сходимости приведенных выше рядов, т. е., в конечном счете, число членов, которое нужно взять, чтобы получить сумму с нужной точностью, весьма различны. Более того, даже для одного и того же ряда требуемое число членов меняется в зависимости от x. Вследствие этого быстродействие программ, вычисляющих значения элементарных функций с помощью рядов, оказывается различным для разных x.

Рассмотренные степенные ряды являются удобным средством для программирования вычисления элементарных функций. Каждый из рядов легко программируется как обычный цикл, который можно писать и как арифметический, и (чаще) как итерационный.

Начнем с показательной функции $y=e^x$. Заметим, что члены ряда

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

связаны соотношением $u_n = u_{n-1} \cdot x/n$. Если не ограничивать область изменения аргумента, то цикл следует писать как итерационный. Для вычислений с машинной точностью можно проверять общий член ряда на совпадение с нулем, т. е. считать до тех пор, пока члены ряда не станут машинным нулем. Впрочем, в этом нет никакой необходимости. Гораздо проще и быстрее сравнивать две соседние суммы и прекращать вычисления тогда, когда они совпадут, т. е. когда прибавление очередного слагаемого не будет изменять суммы. Обычно этот момент наступает раньше, чем члены ряда обращаются в машинный нуль.

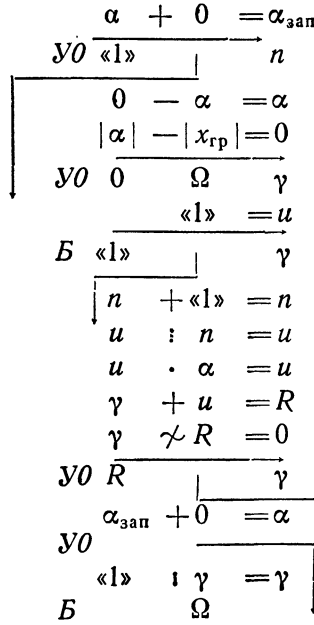
Программу будем писать как стандартную с входной ячейкой α и выходной γ.

$$«1» = \gamma$$
$$«1» = u$$
$$Б \; \overrightarrow{«1» \quad | \quad n}$$
$$n + «1» = n$$
$$u : n = u$$
$$u \cdot \alpha = u$$
$$\gamma + u = R$$
$$\gamma \not\sim R = 0$$
$$У0\; \overrightarrow{R \quad | \quad \gamma}$$
$$Б \quad \Omega$$

Написанная программа неудобна для отрицательных аргументов, больших по абсолютной величине, так как в этом случае ряд получается знакопеременным и его первые члены велики по абсолютной величине, тогда как значение функции мало́. В этом случае может получиться большая потеря точности. Выгоднее иметь дело с другой программой, которая для отрицательного аргумента пользуется тождеством $e^x = 1/e^{-x}$, вычисляя значения функции все-таки для положительного показателя степени, т. е. работая с знакопостоянным рядом.

У такой программы был бы другой недостаток. Для очень больших положительных показателей степени (в рассматриваемой машине — для $x > 42$) значение функции не помещается в разрядной сетке машины и при попытке вычисления этого значения программа остановится по переполнению разрядной сетки. Это явление вполне естественно. Но при вычислении показательной функции для очень большого отрицательного показателя степени машина тоже будет выходить на аварийный останов, а это уже неестественно. Программа должна в этом случае не останавливаться и выдавать нуль в качестве значения функции. Для этого мы вставим в нашу программу дополнительную проверку: отрицательное значение аргумента сравнивается с некоторым граничным значением $x_{гр}$; если $|\alpha| < |x_{гр}|$, то значение функции вычисляется так, как сказано ранее; если же $|\alpha| \geqslant |x_{гр}|$, то в ячейку γ засылается нуль.

Программу, удовлетворяющую этим условиям, можно написать, например, так:



Сначала аргумент α пересылается в запасную ячейку $\alpha_{\text{зап}}$. Одновременно мы анализируем знак аргумента. При $\alpha > 0$ первая команда выработает сигнал $\omega = 0$ и управление будет передано на начало рабочей части, вычисляющей значение показательной функции, как и в предыдущей программе. При $\alpha < 0$ первая команда вырабатывает сигнал $\omega = 1$ и управление передается следующей команде, которая меняет знак у аргумента. Затем идет сравнение α с $x_{\text{гр}}$. Если $|\alpha| > |x_{\text{гр}}|$, то следующая команда засылает 0 в γ и передает управление ячейке Ω, т. е. выходит из программы. В противном случае программа вычисляет значение $e^{-\alpha}$.

После того как заканчивается работа цикла по вычислению функции, из ячейки $\alpha_{\text{зап}}$ в ячейку α засылается первоначальное значение аргумента и снова проверяется его знак. При $\alpha > 0$ программа уходит на Ω, а при $\alpha < 0$ сначала находится обратная величина, т. е. значение e^{α}.

Кроме показательной функции e^x, часто приходится иметь дело с комбинациями показательных функций, которые называются *гиперболическими функциями*. Наиболее важными из них являются *гиперболический косинус* $\operatorname{ch} x$ и *гиперболический синус* $\operatorname{sh} x$, которые определяются формулами

$$\operatorname{ch} x = (e^x + e^{-x})/2, \quad \operatorname{sh} x = (e^x - e^{-x})/2. \tag{7.2}$$

Разложение этих функций в степенные ряды имеет вид

$$\left.\begin{aligned} \operatorname{ch} x &= 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots + \frac{x^{2n}}{(2n)!} + \ldots, \\ \operatorname{sh} x &= x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + \frac{x^{2n+1}}{(2n+1)!} + \ldots \end{aligned}\right\} \tag{8.2}$$

Ряды (8.2) легко получить, взяв полусумму и полуразность рядов для e^x и e^{-x}. Последний сразу получается из ряда для e^x, если в нем заменить x на $-x$. Читатель без труда самостоятельно справится с этими выкладками.

Члены этих двух рядов удобно вычислять поочередно, как это и делалось в программе для вычисления e^x.

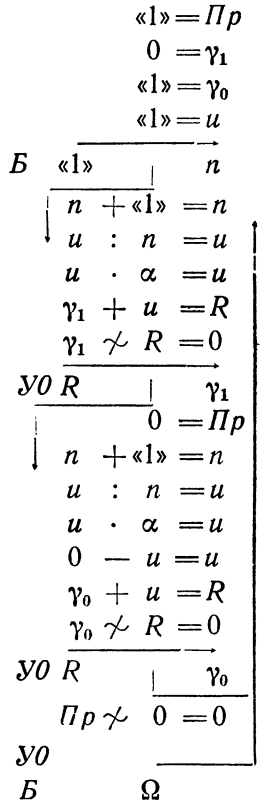
Поэтому выгодно иметь одну программу, одновременно вычисляющую функции $\gamma_0 = \operatorname{ch}\alpha$ и $\gamma_1 = \operatorname{sh}\alpha$. Так как члены обоих рядов, начиная с некоторого места, убывают, достижение машинной относительной точности достаточно проверить только для $\operatorname{ch} x$. Тогда программу можно будет написать следующим образом:

		$0 = \gamma_1$
		«1» $= \gamma_0$
		«1» $= u$
	Б	«1» | n
А		$n +$ «1» $= n$
		$u : n = u$
		$u \cdot \alpha = u$
		$\gamma_1 + u = \gamma_1$
		$n +$ «1» $= n$
		$u : n = u$
		$u \cdot \alpha = u$
		$\gamma_0 + u = R$
		$\gamma_0 \not\sim R = 0$
	УО	R А γ_0
	Б	Ω

Сравнив ряды (8.2) с рядами (2.2) для тригонометрических функций $\cos x$ и $\sin x$, заметим, что они состоят из тех же членов. Поэтому написанная программа может быть использована для вычисления $\gamma_0 = \cos\alpha$ и $\gamma_1 = \sin\alpha$, если вставить еще одну команду перемены знака. Впрочем, кроме этого, нужно еще позаботиться о том, чтобы проверять точность вычислений обеих функций, так как неизвестно, для какой из них требуется большее число членов.

Последнего можно добиться, например, таким путем. Возьмем специальную ячейку — признак (*Пр*), куда вначале запишем единицу. После того как будет вычислен синус с машинной точностью, в эту ячейку запишем нуль. Проверка окончания цикла после вычисления очередного члена для косинуса будет передавать управление на начало цикла до тех пор, пока не будет вычислен с машинной точностью косинус, независимо от признака. После того как косинус будет сосчитан, мы будем проверять содержимое ячейки *Пр* и если в ней записана пока еще единица, то управление снова будет передаваться началу цикла. Вычисления будут закончены лишь тогда, когда обе функции будут сосчитаны с машинной точностью. В написанной ниже программе легко

теперь разобраться самостоятельно



Как уже было отмечено, программы для вычисления тригонометрических и гиперболических функций с помощью рядов отличаются одной командой. Поэтому экономнее объединить обе эти программы в одну программу с двумя разными входами, которые либо зашлют на нужное место команду перемены знака, если нужно считать тригонометрические функции, либо на место этой команды зашлют какую-нибудь «пустую» команду, вроде $0 \vee 0 = 0$, если требуется вычисление гиперболических. По сравнению с написанной программой нужно будет

потратить еще только четыре ячейки — две для заголовков и две для засылаемых команд, вместо того чтобы размещать в памяти еще одну такую же программу. Единая программа будет выглядеть так:

cos, sin	*Б*	Триг			T
ch, sh				Гип	$=T$
				«1»	$=Пр$
				0	$=\gamma_1$
				«1»	$=\gamma_0$
				«1»	$=u$
	Б	«1»			n
A		n	$+$	«1»	$=n$
		u	:	n	$=u$
		u	$\cdot$	α	$=u$
		γ_1	$+$	u	$=R$
		γ_1	$\not\sim$	R	$=0$
	УО	R			γ_1
				0	$=Пр$
		n	$+$	«1»	$=n$
		u	:	n	$=u$
		u	$\cdot$	α	$=u$
T		н. п.			
		γ_0	$+$	u	$=R$
		γ_0	$\not\sim$	R	$=0$
	УО	R		A	γ_0
		$Пр$	$\not\sim$	0	$=0$
	УО			A	
	Б			Ω	
Триг		0	—	u	$=u$
Гип		0	$\vee$	0	$=0$

Для вычисления натуральных логарифмов с помощью ряда можно воспользоваться рядом (3.2) для $\ln\alpha = \ln\frac{1+x}{1-x}$. Отсюда следует, что $x=\frac{\alpha-1}{\alpha+1}$. Работу нужно начать с вычисления значения x, которое мы запишем в ячейку R_1. Кроме того, в ячейке R_2 вычислим x^2; в ячейке n будем вычислять последовательные нечетные

числа. Общий член ряда вычисляется делением нечетной степени x, образуемой в ячейке R_1, на соответствующее нечетное целое число. Последняя команда программы удваивает полученную сумму ряда.

При $\alpha < 0$ мы получим $|x| > 1$, и ряд окажется расходящимся. В этом случае частичные суммы ряда неограниченно возрастают и машина остановится по переполнению разрядной сетки.

Программу можно написать так:

```
α  —«1»=R1
α  +«1»=R2
R1 : R2 =R1
R1 · R1 =R2
     R1 =γ0
     «1»=n
 n +«2»=n   ↑
R1 · R2 =R1 |
R1 : n  =u  |
γ0 + u  =u  |
γ0 ↛ u  =0  |
У0 u    |  γ0
γ0 + γ0 =γ0
Б      Ω
```

Нетрудно написать и программу для вычисления $\operatorname{arctg} x$ с помощью ряда (5.2). В самом деле, этот ряд содержит те же члены, что и ряд для логарифма, использованный нами в предыдущей программе, но является знакопеременным. Так как этот ряд сходится лишь при $|x| < 1$, а для $|x| \geqslant 1$ нужно пользоваться тождеством $\operatorname{arctg} x = \pi/2 - \operatorname{arctg}(1/x)$, то программу можно написать так, как мы это делали для показательной функции.

Прежде всего, перенесем значение α в $\alpha_{\text{зап}}$, выяснив при этом, верно ли неравенство $|\alpha| < 1$. Если оно верно, перейдем к вычислению суммы ряда, а если нет, заменим α на $1/\alpha$. После вычисления суммы ряда тем же способом проверим, восстанавливая значение α, получен

ли уже окончательный результат или полученное значение надо вычесть из $\pi/2$. Программу можно написать так:

$$
\begin{array}{llll}
 & \alpha & \cdot\ «1» & = \alpha_{зап} \\
У0 & «1» & & n \\
 & «1» & :\ \alpha & = \alpha \\
 & & \alpha & = R_1 \\
 & \alpha & \cdot\ \alpha & = R_2 \\
 & & \alpha & = \gamma_1 \\
 & n & +\ «2» & = n \\
 & R_2 & \cdot\ R_1 & = R_1 \\
 & 0 & -\ R_1 & = R_1 \\
 & R_1 & :\ n & = u \\
 & \gamma_1 & +\ u & = u \\
 & \gamma_1 & \not\sim\ u & = 0 \\
У0 & u & & \gamma_1 \\
 & \alpha_{зап} & \cdot\ «1» & = \alpha \\
У0 & & & \\
 & «\pi/2» & -\ \gamma_1 & = \gamma_1 \\
Б & & \Omega &
\end{array}
$$

Для получения, кроме значения $\gamma_1 = \operatorname{arctg} \alpha$, еще и $\gamma_0 = \operatorname{arcctg} \alpha$, вместо последних трех команд написанной только что программы можно написать следующие пять:

$$
\begin{array}{llll}
У0 & \gamma_1 & & \gamma_0 \\
 & «\pi/2» & -\ \gamma_0 & = \gamma_1 \\
Б & & \Omega & \\
 & «\pi/2» & -\ \gamma_1 & = \gamma_0 \\
Б & & &
\end{array}
$$

Все написанные нами программы рассчитаны на вычисление элементарных функций с машинной точностью. Не составляет труда внести в них изменения таким образом, чтобы значения функции вычислялись с заданной

абсолютной или относительной точностью. Для этого нужно только изменить команды проверки окончания цикла. Быстродействие программы при этом, естественно, может повыситься. Однако скорость сходимости рядов изменяется в зависимости от значения аргумента. Поэтому стандартные программы, написанные с помощью рядов, выгодно писать всегда как итерационные циклы, а не как арифметические. Заранее оценивать требуемое число членов ряда имеет смысл только для таких программ, в которых ряды используются для точно фиксированных областей изменения аргумента.

§ 3. Вычисление элементарных функций с помощью цепных дробей

Кроме степенных рядов, при вычислении элементарных функций на электронных счетных машинах используются также цепные дроби, которые рассматривались в курсе алгебры.

Цепной или *непрерывной дробью* называют выражение вида

$$\cfrac{a_0}{b_0+\cfrac{a_1}{b_1+\cfrac{a_2}{b_2+\dots}}} \tag{1.3}$$

Эта дробь может быть как конечной, так и бесконечной. Числа a_0, a_1, a_2, ... в выражении (1.3) называют *частными числителями* цепной дроби, b_0, b_1, b_2, ...—*частными знаменателями*.

Если оборвать цепную дробь в любом месте, то обыкновенную дробь, которая получится после преобразования оставшейся, называют *подходящей дробью*.

Запись (1.3) слишком громоздка и неудобна. Вместо нее обычно используют более короткую, записывая звенья цепной дроби, соединенные знаками плюс, поставленными на уровне знаменателей:

$$\frac{a_0}{b_0}+\frac{a_1}{b_1}+\frac{a_2}{b_2}+\dots \tag{2.3}$$

Цепную дробь можно привести к виду

$$\cfrac{1}{\alpha_0+\cfrac{1}{\alpha_1+\cfrac{1}{\alpha_2+\ldots}}} \tag{3.3}$$

или, в сокращенной записи,

$$\frac{1}{\alpha_0}+\frac{1}{\alpha_1}+\frac{1}{\alpha_2}+\ldots$$

Такие цепные дроби называют *обыкновенными*, а ее частные знаменатели α_i — *неполными частными* обыкновенной цепной дроби. Для обыкновенной цепной дроби используется также сокращенное обозначение

$$[\alpha_0,\ \alpha_1,\ \ldots,\ \alpha_n,\ \ldots]. \tag{4.3}$$

Ограничиваясь рассмотрением только обыкновенных цепных дробей, можно сформулировать простую и удобную для использования теорему об условиях сходимости.

Теорема. *Если все неполные частные обыкновенной цепной дроби*

$$\frac{1}{\alpha_0}+\frac{1}{\alpha_1}+\ldots+\frac{1}{\alpha_n}+\ldots$$

положительны, то для сходимости этой цепной дроби необходимо и достаточно, чтобы ряд из ее неполных частных

$$\sum_{n=1}^{\infty}\alpha_n=\alpha_1+\alpha_2+\ldots+\alpha_n+\ldots$$

расходился.

На доказательстве этой теоремы мы останавливаться не будем.

Л. Эйлер еще в 1739 г. предложил общую формулу для преобразования степенного ряда в цепную дробь. Этой формулой можно воспользоваться для разложения элементарных функций в цепные дроби. Подходящие дроби будут давать при этом *рациональные приближения* соответствующих функций. Цепную дробь можно строить таким образом, чтобы ее n-я подходящая дробь тожде-

ственно равнялась соответствующей частичной сумме степенно́го ряда (такие дроби называют *равноценными*).

Формула Эйлера для преобразования степенно́го ряда в цепную дробь имеет вид

$$\sum_{n=0}^{\infty} c_n x^n = \cfrac{c_0}{1} \cfrac{c_1 x}{-c_0 + c_1 x} \cfrac{c_0 c_2 x}{-c_1 + c_2 x} \cfrac{c_1 c_3 x}{-c_2 + c_3 x} - \ldots - \cfrac{c_{n-2} c_n x}{-c_{n-1} + c_n x} - \ldots \tag{5.3}$$

Нужно только иметь в виду, что вопрос о сходимости цепной дроби, полученной по формуле (5.3), приходится решать отдельно, независимо от сходимости или расходимости исходного степенно́го ряда. Оказывается, что сходимость или расходимость преобразуемого степенно́го ряда и получающейся цепной дроби никак не связаны между собою и фактически могут встретиться все четыре возможные комбинации.

Пользуясь формулой Эйлера (5.3), можно получать различные разложения элементарных функций в цепные дроби. Приведем некоторые из формул, наиболее употребительные при вычислении элементарных функций.

Для показательной функции e^x справедливы следующие разложения:

$$e^x = \cfrac{1}{1} \cfrac{x}{-1} \cfrac{x}{+2} \cfrac{x}{-3} \cfrac{x}{+2} \cfrac{x}{-5} + \ldots + \cfrac{x}{2} \cfrac{x}{-2k+1} + \ldots \tag{6.3}$$

Приведенная цепная дробь сходится для всех значений x, хотя скорость ее сходимости невелика. Чаще используется другое разложение:

$$e^x = 1 + \cfrac{2x}{2-x} \cfrac{x^2}{+6} \cfrac{x^2}{+10} \cfrac{x^2}{+14} + \ldots + \cfrac{x^2}{2(2k+1)} + \ldots \tag{7.3}$$

Логарифмическую функцию можно представить в виде цепной дроби

$$\ln(1+x) = \cfrac{x}{1} \cfrac{x}{+2} \cfrac{x}{+3} \cfrac{2x}{+2} \cfrac{2x}{+5} + \ldots + \cfrac{kx}{2} \cfrac{kx}{+2k+1} + \ldots \tag{8.3}$$

Тригонометрические функции также могут быть представлены в виде цепных дробей, сходящихся для всех

значений x. Наиболее употребительны следующие разложения:

$$\left.\begin{aligned}\sin x &= x - \frac{x^3}{6} + \frac{3x^2}{10} - \frac{11x^2}{42} + \frac{25x^2}{66} - \ldots \\ \cos x &= \frac{1}{1} + \frac{x^2}{2} - \frac{5x^2}{6} + \frac{3x^2}{50} - \frac{313x^2}{126} + \ldots\end{aligned}\right\} \tag{9.3}$$

Более удобно следующее представление для $\operatorname{tg} x$, в котором, в отличие от приведенных разложений (9.3), легко выписать общий член разложения

$$\operatorname{tg} x = \frac{x}{1} - \frac{x^2}{3} - \frac{x^2}{5} - \frac{x^2}{7} - \ldots - \frac{x^2}{2n+1} - \ldots \tag{10.3}$$

Можно привести также представление в виде цепной дроби обратной тригонометрической функции $y = \operatorname{arctg} x$

$$\operatorname{arctg} x = \frac{x}{1} + \frac{x^2}{3} + \frac{4x^2}{5} + \frac{9x^2}{7} + \ldots + \frac{n^2x^2}{2n+1} + \ldots \tag{11.3}$$

При вычислении значений $\arcsin x$ пользуются формулой

$$\arcsin x = \operatorname{arctg} \frac{x}{\sqrt{1-x^2}}.$$

Представление элементарных функций в виде цепных дробей дает еще один важный аппарат для вычисления элементарных функций. Его преимуществом является возможность оценки погрешности и нужного порядка подходящей дроби, что позволяет писать стандартные программы в виде арифметических, а не итерационных циклов.

В качестве первого примера рассмотрим вычисление тангенса по формуле (10.3). Чтобы лучше представлять себе требуемые действия, запишем эту дробь в развернутом виде

$$\operatorname{tg} x = \cfrac{x}{1 - \cfrac{x^2}{3 - \cfrac{x^2}{5 - \cfrac{x^2}{7} - \ldots}}} \tag{12.3}$$

Из написанного выражения видно, что вычисления удобно программировать в виде цикла, идущего снизу вверх. Вследствие этого цикл должен быть необходимо арифметическим, так как мы должны начать с определенного места.

Пусть N — некоторое заранее выбранное нечетное число. Напишем цикл для вычисления по приведенной формуле, полагая, что аргумент находится в ячейке x, а ответ нужно поместить в ячейку tg. Цикл можно запрограммировать, например, так:

$$
\begin{array}{ll}
 & N = n \\
 & x \cdot x = R_1 \\
 & \longrightarrow \\
Б & n \quad | \quad R_0 \\
 & \left|\begin{array}{l} n - R_0 = R_0 \\ R_1 : R_0 = R_0 \\ n - \text{«2»} = n \end{array}\right. \uparrow \\
У0 & \\
 & R_0 : x = \mathrm{tg} \\
 & стоп
\end{array}
$$

Читатель легко разберется в приведенной программе. Заметим только, что после окончания работы цикла мы получим значение дроби, в числителе которой стоит x^2. Поэтому для нахождения тангенса найденное значение нужно еще разделить на x *).

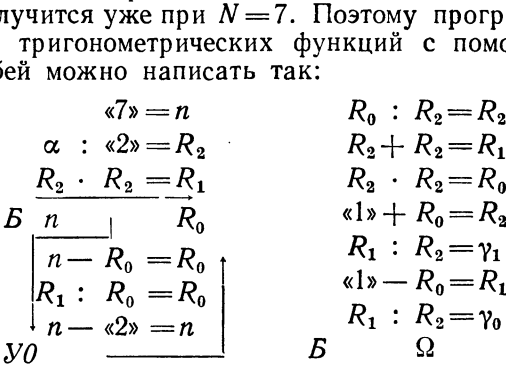
Для нахождения значений тригонометрических функций $\sin x$ и $\cos x$ удобно воспользоваться формулами, выражающими эти функции через тангенс половинного угла:

$$\cos x = \frac{1 - \mathrm{tg}^2 (x/2)}{1 + \mathrm{tg}^2 (x/2)}, \quad \sin x = \frac{2 \mathrm{tg} (x/2)}{1 + \mathrm{tg}^2 (x/2)}. \qquad (13.3)$$

Вычислив при данном значении аргумента x величину $\mathrm{tg}\,(x/2)$ по формуле (12.3) с помощью написанного выше арифметического цикла, можно после этого получить значения $\cos x$ и $\sin x$ по формулам (13.3).

*) По этой причине случай $x = 0$ следует рассматривать отдельно.

Напишем соответствующую программу, оформив ее как стандартную программу без информации. Аргумент будем считать записанным в ячейке α, а ответы должны быть получены в ячейках $\gamma_0=\cos x$ и $\gamma_1=\sin x$. Оценка величины цепной дроби показывает, что достаточная точность получится уже при $N=7$. Поэтому программу вычисления тригонометрических функций с помощью цепных дробей можно написать так:



Аналогичную программу можно написать также и для гиперболических функций. Для них верны и формулы, аналогичные (13.3), и разложение (12.3), в котором только следует заменить все знаки минус на плюсы. Поэтому, как и при вычислении с помощью рядов (см. § 2), программа для вычисления гиперболических функций будет отличаться от написанной лишь одной командой. Не составит труда объединить эти две программы в одну, как это было сделано нами в § 2.

Рассмотрим теперь вычисление показательной функции e^x. Прежде всего, воспользуемся тождеством

$$e^x=2^{\frac{x}{\ln 2}}.$$

Далее, можно написать

$$e^x=2^A\cdot 2^B,$$

где $A=\left[\frac{x}{\ln 2}\right]$ — целая часть числа $\frac{x}{\ln 2}$, а $B=\left\{\frac{x}{\ln 2}\right\}=\frac{x}{\ln 2}-\left[\frac{x}{\ln 2}\right]$ — его дробная часть *). Выделение целой и

*) Фактически удобнее брать в качестве A ближайшее к $\frac{x}{\ln 2}$ целое число, но это не меняет существа дела.

дробной части числа уже рассматривалось нами в первом выпуске (см. § 21). Окончательно напишем

$$e^x = 2^A \cdot e^{B \ln 2}. \qquad (14.3)$$

Два множителя, входящие в правую часть равенства (14.3), вычисляются отдельно. Заметим, что мантисса числа 2^A, где A — целое, состоит из одной единицы в 36-м разряде, т. е. имеет вид (4000,0,0). Истинная величина порядка числа 2^A равна самому числу A, записанному в виде количества единиц порядковой (кодовой) части ячейки с соответствующим знаком. Поэтому для получения условного порядка 2^A надо число A прибавить к 100_8 с помощью операции сложения порядков или вычесть A из 100_8, если A отрицательно *).

Таким образом, можно взять ячейку, имеющую вид (100;4000,0,0), которая представляет собой число 1/2, записанное в плавающей форме, и прибавлять к ее порядку (или вычитать из него) число A, записанное в виде числа единиц порядковой части ячейки. Это можно осуществить с помощью программы

```
1)                          T⁰      =T
2)      (200, 0, 0, 0) Λ    A      =0
3)  У1                      5)
4)          T     ,+ (20; 0, 0, 0) =T
5)          A     Λ (F, F, F)     =R1
6)          A     [→]   R1        =R1
T 7)    (   «1/2»   ,+    R1       =γ0) н. п.
```

Команда 1) служит здесь для восстановления переменной команды T, первоначальное состояние которой T^0 указано в скобках. Далее, команда 2) проверяет знак числа A, записанного в плавающей форме. Если в знаковом разряде ячейки A записан нуль, т. е. $A \geqslant 0$, то в результате команды 2) в пересечении получится нулевое слово и выработается сигнал $\omega = 1$. Тогда команда 3) передаст управление команде 5), которая является началом рабочей части.

*) Можно воспользоваться операциями изменения порядка.

При $A<0$ в знаковом разряде ячейки A будет записана единица, которая останется и в пересечении. Поэтому после команды 2) выработается сигнал $\omega=0$ и команда 3) передаст управление команде 4), которая к кодовой части ячейки прибавит 20. При этом команда T из команды сложения кодовых частей (с кодом 53) превратится в команду вычитания (с кодом 73). После этого управление перейдет команде 5).

Рабочая часть программы состоит из трех команд. Команда 5) высекает мантиссу числа A. Затем команда 6) сдвигает эту мантиссу в младшие разряды кодовой части, так что после нее число A оказывается записанным в фиксированной форме, в виде числа единиц кодовой части ячейки R_1. После этого команда 7) прибавит полученное число к заготовке (100; 4000,0,0) или вычтет его из нее, если команда T переформировывалась командой 4); в результате в ячейке γ_0 будет записано 2^A.

Остается вычислить второй множитель формулы (14.3), имеющий вид e^t, где $t=B\ln 2$. Для этого мы воспользуемся разложением показательной функции в цепную дробь по формуле (7.3). Поскольку при любых x мы будем получать $t<1$, то достаточно взять небольшое число членов. Оценка показывает, что можно ограничиться тремя членами дроби, т. е. разложением вида

$$e^t=1+\cfrac{2t}{(2-t)+\cfrac{t^2}{6+\cfrac{t^2}{10}}}. \qquad (15.3)$$

Для последней дроби выгодно писать бесцикловую программу. Поэтому дробь (15.3) удобно переписать в виде подходящей дроби

$$e^t=\frac{120+60t+12t^2+t^3}{120-60t+12t^2-t^3},$$

которую можно записать так:

$$e^t=\frac{12\,(10+t^2)+t\,(60+t^2)}{12\,(10+t^2)-t\,(60+t^2)}. \qquad (16.3)$$

Программирование формулы (16.3) не вызывает никаких затруднений.

Требуемую программу можно, например, написать так:

$$\begin{array}{llll} t \cdot t & = t^2 & R_1 \cdot t & = R_1 \\ t^2 + \text{«10»} & = R_0 & R_0 + R_1 & = R_2 \\ R_0 \cdot \text{«12»} & = R_0 & R_0 - R_1 & = R_1 \\ t^2 + \text{«60»} & = R_1 & R_2 : R_1 & = R_0, \end{array}$$

после чего команда

$$\gamma_0 \cdot R_0 = \gamma_0$$

дает нам в ячейке γ_0 нужное значение e^{α}.

Для получения окончательной программы вычисления $\gamma_0 = e^{\alpha}$ достаточно объединить написанные нами программы, присоединив к ним команды получения величин $\alpha/\ln 2$, $A = [\alpha/\ln 2]$ и $t = \alpha - A \ln 2$, а также все требуемые константы. С этой работой читатель легко справится самостоятельно.

Необходимо иметь в виду, что фактические библиотечные программы для вычисления элементарных функций как с помощью рядов, так и с помощью цепных дробей, хотя и построены на описанных нами принципах, но не совпадают в точности с теми, которые приводятся в нашей книге.

Это объясняется прежде всего тем, что использование специальных операций, вроде модификаций арифметических действий, изменения порядка и т. п., позволяет программировать отдельные места более коротко и просто. Кроме того, так как при составлении библиотечных программ экономия ячеек имеет очень большое значение, в ряде случаев удается сократить программу при помощи различных искусственных приемов.

ГЛАВА II

ЧИСЛЕННОЕ РЕШЕНИЕ АЛГЕБРАИЧЕСКИХ И ТРАНСЦЕНДЕНТНЫХ УРАВНЕНИЙ

§ 4. Подбор корней

Как известно, далеко не всякое уравнение может быть решено точно. В первую очередь это относится к большинству трансцендентных уравнений, т. е. уравнений, в которых неизвестная x находится под знаком трансцендентной функции. Доказано также, что нельзя построить формулу, по которой можно было бы решать произвольное алгебраическое уравнение степени выше четвертой *).

Однако точное решение уравнения не всегда является необходимым. Задачу отыскания корней уравнения можно считать практически решенной, если мы сумеем определить корни с нужной степенью точности.

Большинство употребляющихся приближенных способов решения уравнений являются, по существу, способами *уточнения корней*, т. е. для их применения необходимо знание примерных значений корня. Для этой последней цели могут служить графические способы, о которых и будет сейчас идти речь.

Пусть рассматриваемое уравнение имеет вид

$$f(x)=0. \tag{1.4}$$

Построим в декартовой системе координат схематический график функции $y=f(x)$. Абсциссы точек пересе-

*) Доказательство этого факта связано с именами замечательных математиков Абеля (1802—1829) и Галуа (1811—1832).

чения построенной кривой с осью Ox дадут нам значения действительных корней уравнения (1.4).

После того как схематический график построен и примерно выделены участки оси абсцисс, в которых будут лежать корни функции, мы приступим к уточнению значений корней. Для этого можно построить на выделенных участках график функции в более крупном масштабе, производя при этом более точное вычисление значений функции.

Разумеется, при этом мы значительно точнее найдем точки пересечения графика с осью абсцисс. Для построения такого графика полезно воспользоваться миллиметровой бумагой.

Графическое отыскание корня можно производить иначе. Допустим, что уравнение (1.4) можно представить в виде

$$f_1(x)=f_2(x). \tag{2.4}$$

В этом случае строим графики функций $y=f_1(x)$ и $y=f_2(x)$; абсциссы точек пересечения кривых будут действительными корнями уравнения.

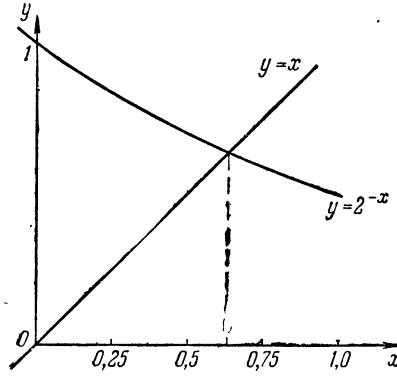


Рис. 1.

Пример 1.4. Найдем приближенно корни уравнения

$$x\cdot 2^x=1.$$

Построение графика функции $y=x\cdot 2^x$ может представлять затруднения. Поэтому для графического подбора корней выгоднее записать уравнение в виде (2.4). Это можно сделать двумя различными способами:

$$2^x=1/x$$

или

$$x=2^{-x}.$$

Ясно, что второй из них предпочтительнее, потому что

в первом случае нужно строить две кривые, а во втором — кривую и прямую.

Построив графики кривой $y=2^{-x}$ и прямой $y=x$ (рис. 1), найдем, что точка пересечения этих линий имеет абсциссу $x\approx 0{,}64$. Это значение можно считать грубым приближением корня.

Перейдем теперь к аналитическим способам уточнения значений корней. Сразу подчеркнем: все эти способы предполагают, что нам известен некоторый интервал $[a, b]$, в котором лежит уточняемый корень уравнения. Выбор этого интервала производится на основании известного свойства непрерывных функций: если функция $f(x)$ непрерывна на замкнутом интервале $[a, b]$ и на его концах имеет различные знаки, т. е. $f(a)\cdot f(b)<0$, то между точками a и b имеется хотя бы один корень уравнения $f(x)=0$.

Пусть интервал $[a, b]$ настолько мал, что на нем лежит в точности о д и н корень нашего уравнения. Тогда говорят, что интервал $[a, b]$ является *интервалом изоляции корня* *).

Сужение интервала изоляции можно производить самым простым образом. Выбираем какую-либо точку c, лежащую внутри интервала (обычно за точку c принимают середину рассматриваемого интервала или близкую к ней точку, в которой удобнее вычислять значение функции), и вычисляем значение $f(c)$.

В качестве нового интервала изоляции мы примем ту из двух половинок интервала $[a, b]$, на концах которой функция имеет разные знаки.

Таким путем можно как угодно сузить участок, на котором находится корень, т. е. получить приближенное значение корня с любой степенью точности. При этом мы получаем оценку точности приближенного решения (корень заключен между концами очередного участка). Однако, несмотря на принципиальную простоту, такое последовательное сужение участка на практике не всегда проводится, ибо часто требует слишком большого количества вычислений, особенно если решение производится

*) Условия, при соблюдении которых интервал $[a, b]$ будет интервалом изоляции, сформулированы ниже.

вручную или с помощью клавишных вычислительных машин, а не на электронной вычислительной машине.

При применении других способов уточнения корня будем требовать, чтобы на рассматриваемом отрезке $[a, b]$ функция $f(x)$ удовлетворяла следующим условиям:

1. Функция $f(x)$ непрерывна на отрезке $[a, b]$ вместе со своими производными первого и второго порядков.

2. Значения $f(x)$ на концах участка $[a, b]$ имеют разные знаки:

$$f(a) \cdot f(b) < 0.$$

3. Первая и вторая производная $f'(x)$ и $f''(x)$ сохраняют определенный знак и не обращаются в нуль на всем участке.

Эти условия гарантируют, что корень уравнения (1.4) содержится в интервале $[a, b]$ и других корней на этом участке не имеется. В самом деле, условия 1 и 2 гарантируют, что между точками a и b находится хотя бы один корень уравнения; из условия 3 следует, что функция $f(x)$ на данном интервале монотонна и поэтому корень будет единственным *).

В дальнейшем во всей этой главе мы будем предполагать, что функция $f(x)$ и интервал $[a, b]$ удовлетворяют перечисленным выше условиям 1—3.

Заметим, кстати, что мы можем ограничиваться только тем случаем, когда корни уравнения $f(x)=0$ положительны. Случай отрицательных корней может быть сведен к рассмотрению положительных, для чего в уравнении достаточно заменить x на $-x$.

§ 5. Способ хорд и проведение параболы

Идея способа хорд состоит в том, что можно, с известным приближением, допустить, что функция на достаточно малом участке $[a, b]$ изменяется линейно. Тогда кривую $y=f(x)$ на участке $[a, b]$ можно заменить хордой и в качестве приближенного значения корня принять точку пересечения хорды с осью абсцисс.

Для лучшего выяснения сути способа обратимся к чертежу (рис. 2). Построим график функции $y=f(x)$ на участке $[a, b]$. Истинный корень уравнения $f(x)=0$ есть абсцисса точки A, являющейся точкой пересечения кривой MM' с осью абсцисс. Заменив кривую MM' хордой MM', мы примем в качестве приближенного значения

*) Значение постоянства знака $f''(x)$ выяснится несколько позже.

корня абсциссу точки B, в которой хорда пересекается с осью.

Найдем аналитическое выражение для приближенного значения корня. Как видно из рис. 2, угловой коэффициент хорды MM' равен

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

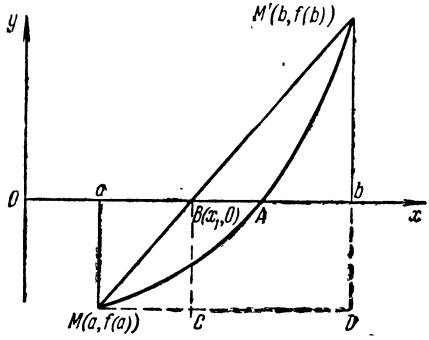


Рис. 2.

Легко заметить, что это выражение не изменится и в том случае, если $f(a)>0$, а $f(b)<0$. Таким образом, уравнение хорды MM' можно записать в виде

$$y=\frac{f(b)-f(a)}{b-a}x+m. \quad (1.5)$$

Коэффициент m можно определить, например, из условия, что при $x=a$ хорда должна проходить через точку M кривой, т. е. мы должны иметь $y=f(a)$. Тогда

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

откуда

$$m=f(a)-\frac{f(b)-f(a)}{b-a}a,$$

и уравнение хорды получает вид

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

Чтобы получить абсциссу точки B пересечения хорды с осью Ox, положим $y=0$ и решим полученное уравнение относительно x. Будем иметь

$$\frac{f(b)-f(a)}{b-a}(x_1-a)=-f(a),$$

откуда

$$x_1=a-\frac{(b-a)f(a)}{f(b)-f(a)}. \quad (2.5)$$

Это и есть формула для приближенного значения корня, полученного по способу хорд.

В ряде случаев бывает удобнее отправляться от точки b, вместо точки a, как мы сделали это выше. Тогда коэффициент m в уравнении (1.5) нужно искать, исходя из того, что при $x=b$ должно быть $y=f(b)$, т. е.

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

откуда

$$m=f(b)-\frac{f(b)-f(a)}{b-a}b,$$

и уравнение хорды получает вид

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

Отсюда для абсциссы точки B, т. е. для приближенного значения корня уравнения, находим, как и выше,

$$x_1=b-\frac{(b-a)f(b)}{f(b)-f(a)}. \tag{3.5}$$

Очевидно, что формулы (2.5) и (3.5) тождественны. В этом легко убедиться и непосредственными преобразованиями. Мы будем пользоваться той из них, которая окажется более удобной.

Полученное значение x_1 можно снова использовать для дальнейшего уточнения корня по способу хорд, рассматривая интервал $[a, x_1]$ или же $[x_1, b]$, смотря по тому, в каком из них лежит истинный корень. Чтобы определить это, находят знак $f(x_1)$.

Пример 1.5. Найдем по способу хорд положительный корень уравнения $x^3-2x^2+x-3=0$.

Прежде всего определяем знаки функции в различных точках. Результаты вычислений приведены в таблице (табл. 1.5), причем значения аргумента помещены в том порядке, в каком вычисления проводились.

Таблица 1.5

x	0	1	2	3	2,5	2,2	2,1
$f(x)$	−	−	−	+	+	+	−

Из таблицы видно, что функция меняет знак на отрезке [2, 3]. Однако этот участок слишком велик; дальнейшее сужение дает отрезок [2,1; 2,2], к которому мы и применим способ хорд. Вычисление значений функции дает

$$f(2,1) = -0,459,$$
$$f(2,2) = +0,168.$$

По формуле (2.5)

$$x_1 = 2,1 - \frac{(2,2-2,1)(-0,459)}{0,168+0,459} = 2,173.$$

Вычислив значение функции при $x_1 = 2,173$, находим, что $f(2,173) = -0,01011$. Отсюда видно, что истинный корень расположен в интервале [2,173; 2,2]. Снова применив к этому участку способ хорд, получим

$$x_1^{(2)} = 2,173 - \frac{(2,2-2,173)(-0,01011)}{0,168+0,01011} = 2,1745.$$

Вычисления значений функции дают

$$f(2,1745) = -0,38543 \cdot 10^{-3}, \quad f(2,1746) = +0,26335 \cdot 10^{-3}.$$

Полагая значение корня равным $x = 2,17455$, видим, что погрешность полученного приближения меньше 0,00005.

Более точно корень уравнения $f(x) = 0$ можно получить, если вычислить значения функции $f(x)$ в трех точках и провести через них параболу, которая лучше приближает функцию, чем хорда. За приближенное значение корня можно взять тогда точку пересечения этой параболы с осью Ox. Рассмотрим соответствующий пример.

Пример 2.5. Найдем корень уравнения, рассмотренного в примере 1.4:

$$x \cdot 2^x = 1.$$

Так как здесь не требуется строить графика функции, то можно записать уравнение в виде

$$x \cdot 2^x - 1 = 0,$$

т. е. положить

$$f(x) = x \cdot 2^x - 1.$$

Как уже было замечено, корень этого уравнения лежит на участке (0,1). Выбрав еще точку $x = 1/2$ и вычис-

лив значения функции для этих трех значений аргумента, найдем точки

$$x=0, \quad y=-1,$$
$$x=1/2, \quad y=\sqrt{2}/2-1\approx-0{,}293,$$
$$x=1, \quad y=1,$$

через которые и надо проводить параболу. Запишем уравнение параболы в виде

$$y=ax^2+bx+c.$$

Для того чтобы эта парабола проходила через полученные нами точки, коэффициенты a, b, c должны удовлетворять уравнениям

$$a\cdot 0^2+b\cdot 0+c=-1,$$
$$a\cdot 1/4+b\cdot 1/2+c=-0{,}293,$$
$$a\cdot 1^2+b\cdot 1+c=1,$$

откуда сразу находим $c=-1$; для остальных двух коэффициентов получаем систему

$$a+b=2,$$
$$a+2b=2{,}828,$$

так что $a=1{,}172$ и $b=0{,}828$. Таким образом, уравнение искомой параболы имеет вид

$$y=1{,}172x^2+0{,}828x-1.$$

Чтобы получить приближенно корень интересующего нас уравнения, надо найти точку пересечения этой параболы с осью Ox, т. е. решить квадратное уравнение

$$1{,}172x^2+0{,}828x-1=0,$$

что дает корень на участке (0, 1)

$$x=0{,}636.$$

Это и есть приближенное значение корня нашего уравнения со значительно большей точностью, чем оно было получено графически.

Внимательный читатель сможет заметить связь, существующую между способом хорд и линейной интерполяцией, рассматривавшейся в § 5 первого выпуска. Аналогичная связь существует также между проведением

параболы и квадратичной интерполяцией, которая будет рассматриваться нами ниже, в четвертой главе. Без этого вывод общей формулы для приближенного значения корня способом проведения параболы оказывается затруднительным. Поэтому здесь придется ограничиться рассмотренным примером. Мы еще раз вернемся к этому способу в § 26, где будет рассмотрено программирование приближенного нахождения корня с помощью проведения параболы.

§ 6. Способ касательных. Комбинированный способ

Наряду со способами хорд и проведения параболы часто используется другой способ, основанный на замене графика функции участком прямой линии.

Обратимся снова к уравнению $f(x)=0$.

Возьмем некоторую точку c участка $[a, b]$ и проведем в точке $[c, f(c)]$ графика функции касательную к этому графику. Уравнение касательной имеет вид

$$y-f(c)=f'(c)\cdot(x-c).$$

В качестве приближенного корня уравнения $f(x)=0$ примем абсциссу точки пересечения касательной с осью Ox.

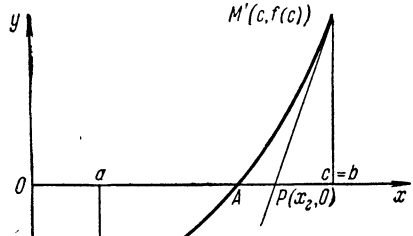


Рис. 3.

Полагая в уравнении касательной $y=0$, находим для абсциссы точки пересечения

$$x_2=c-f(c)/f'(c) \quad (1.6)$$

(см. рис. 3, где принято $c=b$).

Остается решить вопрос о выборе точки c. На рис. 3 мы приняли $c=b$. Нетрудно видеть, что в этом случае $f(c)>0$ и $f''(c)>0$, ибо кривая вогнута. Обычно принимают $c=a$ или $c=b$, смотря по тому, в какой из этих точек знак функции совпадает со знаком второй производной, т. е. c выбирают так, чтобы произведение $f(c)\cdot f''(c)$ было положительно.

Как будет показано ниже, в этом случае можно гарантировать, что приближенное значение корня, полученное по способу касательных, лежит на участке $[a, b]$, т. е. что $a < x_2 < b$.

Как и в способе хорд, значение x_2 можно использовать для дальнейшего уточнения корня, беря участок $[a, x_2]$ или $[x_2, b]$.

Пример 1.6. Рассмотрим то же уравнение, что и в примере 1.5:

$$x^3 - 2x^2 + x - 3 = 0.$$

Здесь $f'(x) = 3x^2 - 4x + 1 > 0$ и $f''(x) = 6x - 4 > 0$, ибо мы выбрали интервал $[2{,}1;\ 2{,}2]$. Если принять $c = a = 2{,}1$, то $f(c) \cdot f''(c) < 0$, ибо $f(2{,}1) < 0$. Наоборот, при $c = b = 2{,}2$ имеем $f(c) \cdot f''(c) > 0$, так что касательную следует проводить в точке $c = b$. Формула (1.6) дает

$$x_2 = 2{,}2 - 0{,}168/6{,}72 = 2{,}175.$$

Так как $f(2{,}175) > 0$, то на участке $[2{,}1;\ 2{,}175]$ можно вновь применить способ касательных, полагая $c = 2{,}175$. Воспользовавшись снова формулой (1.6), получим

$$x_2^{(2)} = 2{,}175 - 0{,}28594 \cdot 10^{-2}/6{,}491875 = 2{,}17456.$$

Рассмотренный пример показывает, что способ хорд и способ касательных дают приближение корня с разных сторон (больше и меньше истинного корня). Поэтому обычно бывает выгодно применять оба эти способа одновременно, благодаря чему уточнение корня может быть получено быстрее.

Ограничения, наложенные в § 4 на функцию и отрезок $[a, b]$, показывают, что возможны четыре различных случая поведения функции на этом отрезке, в зависимости от знаков производных. На рис. 4—7 изображены все возможные случаи поведения функции. Точка A изображает истинный корень, точка B — приближенное значение корня по способу хорд, точка P — приближенное значение корня, полученное по способу касательных.

Обозначим абсциссу точки A через $\bar{x}$, точки B — через x_1 и точки P — через x_2. Из рис. 4 видно, что для случая, когда $y' > 0$ и $y'' > 0$, имеют место неравенства

$$a < x_1 < \bar{x} < x_2 < b. \qquad (2.6)$$

Это же неравенство имеет место для случая $y' < 0$, $y'' < 0$

(рис. 5). Если $y'>0$, но $y''<0$ (рис. 6) или, наоборот, $y'<0$, $y''>0$ (рис. 7), то

$$a<x_2<\bar{x}<x_1<b. \tag{3.6}$$

При этом во всех случаях касательная проводится в соответствии со сделанными указаниями.

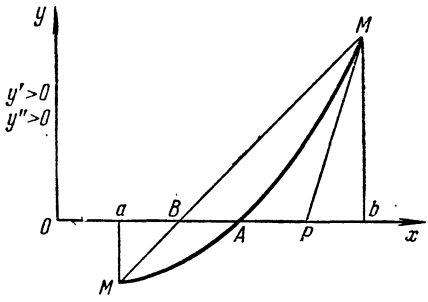


Рис. 4.

Таким образом, во всех четырех случаях истинный корень уравнения заключен между приближенными корнями, получающимися по способу хорд и по способу касательных. Расположение корней можно охарактеризовать таблицей 1.6.

Сделаем теперь несколько замечаний, касающихся практического применения комбинации способов хорд и касательных. Прежде всего необходимо, как было указано выше, выделить интервал $[a, b]$, т.е. изолировать корень. Далее, определив знаки производных, мы можем судить о расположении корней по приведенной здесь таблице.

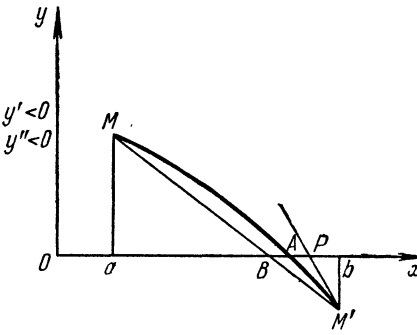


Рис. 5.

Пусть, например, $y'y''>0$. Тогда способ хорд дает значение корня с недостатком. Для нахождения этого корня мы воспользуемся формулой (2.5), записав ее в виде

$$a_1=a+\Delta a, \quad \text{где} \quad \Delta a=-\frac{(b-a)f(a)}{f(b)-f(a)}. \tag{4.6}$$

Для способа касательных следует выбрать $c=b$, так что формулу (1.6) можно записать в виде

$$b_1=b+\Delta b, \quad \Delta b=-f(b)/f'(b). \tag{5.6}$$

Так как способ касательных дает значение с избытком, то истинный корень находится на интервале $[a_1, b_1]$. Поэтому можно продолжать дальнейшее уточнение корня, получая новый участок по формулам

$$a_2 = a_1 + \Delta a_1, \quad \Delta a_1 = -\frac{(b_1 - a_1) f(a_1)}{f(b_1) - f(a_1)}, \tag{6.6}$$

$$b_2 = b_1 + \Delta b_1, \quad \Delta b_1 = -f(b_1)/f'(b_1). \tag{7.6}$$

Подобное уточнение следует продолжать до тех пор, пока не получится интервал, длина которого не превосходит допустимой погрешности.

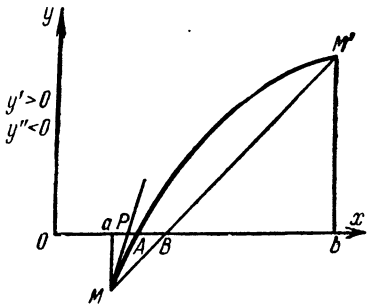


Рис. 6.

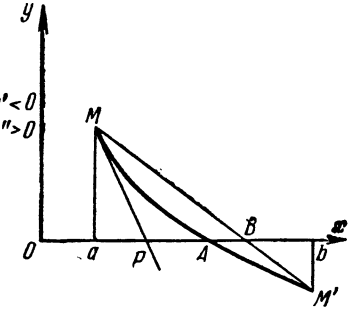


Рис. 7.

Отметим, что обычно среднее арифметическое приближенных корней, полученных по способу хорд и способу

Таблица 1.6

	Способ хорд	Способ касательных
$y'y'' > 0$	с недостатком	с избытком
$y'y'' < 0$	с избытком	с недостатком

касательных, дает лучшее приближение, нежели каждый из этих корней в отдельности.

Если $y'y'' < 0$, то ход решения остается тем же, но формулы меняются местами. Способ хорд дает в этом

случае корень с избытком, а способ касательных — с недостатком.

Для способа касательных следует выбирать $c=a$, поэтому формулу (1.6) мы запишем в виде

$$a_1=a+\Delta a, \quad \Delta a=-f(a)/f'(a). \tag{8.6}$$

В способе хорд мы воспользуемся формулой (3.5), записав ее в виде

$$b_1=b+\Delta b, \quad \Delta b=-\frac{f(b)(b-a)}{f(b)-f(a)}. \tag{9.6}$$

Истинный корень находится в интервале $[a_1, b_1]$. Дальнейшее уточнение производится по формулам

$$a_2=a_1+\Delta a_1, \quad \Delta a_1=-f(a_1)/f'(a_1), \tag{10.6}$$

$$b_2=b_1+\Delta b_1, \quad \Delta b_1=-\frac{f(b_1)(b_1-a_1)}{f(b_1)-f(a_1)} \tag{11.6}$$

и т. д.

Пример 2.6. Решим способом хорд и касательных уравнение

$$(4+x^2)(e^x-e^{-x})=18.$$

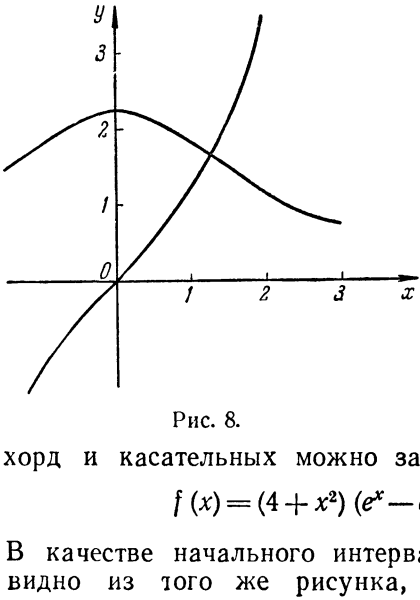


Рис. 8.

Представив это уравнение в виде

$$(e^x-e^{-x})/2=9/(4+x^2),$$

построим графики функций $y=9/(4+x^2)$ и $y=(e^x-e^{-x})/2$ ($=\operatorname{sh} x$). Как видно по рис. 8, приближенное значение корня можно считать равным $x=1{,}25$. Для уточнения способом хорд и касательных можно записать уравнение в виде

$$f(x)=(4+x^2)(e^x-e^{-x})-18=0.$$

В качестве начального интервала изоляции корня, как видно из того же рисунка, можно взять интервал

(1,2; 1,3). Производные на этом участке равны

$$f'(x) = 2x(e^x - e^{-x}) + (4 + x^2)(e^x + e^{-x}) > 0,$$

$$f''(x) = 2(e^x - e^{-x}) + 4x(e^x + e^{-x}) + (4 + x^2)(e^x - e^{-x}) > 0.$$

Из табл. 1.6 следует, что способ хорд дает в этом случае приближение с недостатком, а способ касательных — с избытком. Дальнейшие вычисления приведены в табл. 2.6.

Таблица 2.6

(1)	(2)	(3)	(4)	(5)	(6)
x	(1)^3 + «4»	$e^{(1)}$	$e^{-(1)}$	(3) — (4)	(2) · (5) — «18»
1,2	5,44	3,320117	0,301194	3,018923	—1,577059
1,3	5,69	3,669297	0,272532	3,396765	1,327593
1,254	5,572516	3,504332	0,285361	3,218971	—0,062233
1,257	5,580049	3,514861	0,284506	3,230365	0,025539
1,256127					
1,256129					

(7)	(8)	(9)	(10)	(11)	(12)
(1) · (5)	(3) + (4)	(2) · (8)	$f'(x)$ «2» · (7) + (9)	Δa $-\frac{(b-a)f(a)}{f(b)-f(a)}$	Δb — (6) : (10)
4,415794	3,941829	22,429007	31,260595	0,054	—0,043
4,060556	3,799367	21,200654	29,321766	0,002127	—0,000871

Неравенства (2.6) и (3.6), характеризующие расположение корней, мы получили, исходя из геометрических соображений. Нетрудно доказать их и аналитически.

Рассмотрим, например, случай $y' > 0$, $y'' > 0$. Возьмем определенную в (a, b) вспомогательную функцию $\varphi(x) = \frac{f(x) - f(a)}{x - a}$. Ее производная равна

$$\varphi'(x) = \frac{f'(x)(x-a) - f(x) + f(a)}{(x-a)^2}$$

Представив $f(a)$ по формуле Тейлора в окрестности точки x

$$f(a)=f(x)+(a-x)f'(x)+\frac{(a-x)^2}{2!}f''(\xi) \qquad (a<\xi<b),$$

и подставив это выражение для $f(a)$ в числитель выражения для $\varphi'(x)$, найдем

$$\varphi'(x)=\frac{1}{2}f''(\xi),$$

так что $\varphi'(x)>0$ и $\varphi(x)$ на (a, b) возрастает. Следовательно, $\varphi(x)<\varphi(b)$, т. е.

$$\frac{f(x)-f(a)}{x-a}<\frac{f(b)-f(a)}{b-a},$$

откуда

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

Но правая часть последнего неравенства равна ординате хорды в точке x и обращается в нуль при $x=x_1$, если x_1 означает корень, полученный по способу хорд. Отсюда следует, что $f(x_1)<0$ и $x_1<\bar{x}$, где $\bar{x}$ — истинный корень, поскольку $f(\bar{x})=0$, а функция $f(x)$ монотонно возрастает. Неравенство же $x_1>a$ следует из формулы (4.6), так как $f(a)<0$ и, следовательно, $\Delta a>0$.

Перейдем теперь к способу касательных. В предположении $y'>0$, $y''>0$ выберем $c=b$. Пусть снова $\bar{x}$ — истинный корень и $x_2=b-f(b)/f'(b)$ корень, полученный по способу касательных. Тогда $x_2<b$, поскольку $f(b)>0$ и $f'(b)>0$. Далее,

$$\bar{x}-x_2=\bar{x}-b+f(b)/f'(b).$$

С другой стороны, формула Тейлора дает

$$0=f(\bar{x})=f(b)+(\bar{x}-b)f'(b)+\frac{1}{2!}(\bar{x}-b)^2f''(\xi),$$

откуда

$$f(b)=-(\bar{x}-b)f'(b)-\frac{1}{2}(\bar{x}-b)^2f''(\xi).$$

Подставив полученное выражение для $f(b)$ в предыдущее равенство, найдем

$$\bar{x}-x_2=-\frac{1}{2}(\bar{x}-b)^2\frac{f''(\xi)}{f'(b)}<0,$$

откуда вытекает $\bar{x}<x_2$. Объединяя все полученные неравенства, получаем

$$a<x_1<\bar{x}<x_2<b,$$

что совпадает с неравенством (2.6). Аналогично доказываются неравенства и для всех остальных случаев.

Остается только показать, что, пользуясь этими двумя способами в отдельности (или их комбинацией) несколько раз, можно получить корень с любой степенью точности, т. е. доказать сходимость рассматриваемых процессов.

Пусть $x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(n)}, \ldots$ — последовательность приближенных корней уравнения, полученных по способу хорд, а $\bar{x}$ — точный корень того же уравнения. Как было показано выше, в зависимости от знака производных, имеют место неравенства

$$a < x_1^{(1)} < x_1^{(2)} < \ldots < x_1^{(n)} < \ldots < \bar{x}$$

или

$$b > x_1^{(1)} > x_1^{(2)} > \ldots > x_1^{(n)} > \ldots > \bar{x}.$$

При этом два соседних значения связаны соотношением

$$x_1^{(n+1)} = x_1^{(n)} - \frac{(b - x_1^{(n)}) f(x_1^{(n)})}{f(b) - f(x_1^{(n)})}. \tag{12.6}$$

Так как последовательность $\{x_1^{(n)}\}$ монотонна и ограничена, то она имеет предел. Обозначив его через α,

$$\lim_{n \to \infty} x_1^{(n)} = \alpha,$$

перейдем к пределу в равенстве (12.6), что дает

$$\alpha = \alpha - \frac{(b - \alpha) f(\alpha)}{f(b) - f(\alpha)},$$

откуда вытекает $f(\alpha) = 0$, т. е. $\alpha = \bar{x}$ *), ибо других корней в интервале $[a, b]$, по предположению, нет.

Аналогично доказывается сходимость для способа касательных. Обозначим через $\{x_2^{(n)}\}$ последовательность корней, получающихся по способу касательных. Как и выше, эта последовательность монотонна и ограничена, а потому имеет предел. Кроме того,

$$x_2^{(n+1)} = x_2^{(n)} - f(x_2^{(n)})/f'(x_2^{(n)}). \tag{13.6}$$

Пусть $\lim_{n \to \infty} x_2^{(n)} = \beta$. Переходя к пределу в равенстве (13.6), получим

$$\beta = \beta - f(\beta)/f'(\beta),$$

что дает $f(\beta) = 0$, т. е. $\beta = \bar{x}$.

§ 7. Способ итераций

В ряде случаев весьма удобным приемом решения уравнений является *метод итераций* (повторений). Для применения этого метода исходное уравнение должно

*) Здесь всегда выполнены условия $\alpha \neq b$ и $f(\alpha) \neq f(b)$.

быть записано в форме

$$x=\varphi(x). \tag{1.7}$$

Пусть каким-либо способом выделен интервал $[a, b]$ изоляции корня этого уравнения, и x_0 — любая точка этого интервала (*нулевое приближение*). Для получения следующего приближения x_1 в правую часть уравнения (1.7) вместо x подставляем x_0, так что

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

Следующие приближения получаются по схеме

$$\begin{aligned} x_2&=\varphi(x_1),\\ x_3&=\varphi(x_2),\\ &\cdots\cdots\cdot\\ x_n&=\varphi(x_{n-1}),\\ &\cdots\cdots\cdot \end{aligned}$$

Если последовательность $x_1, x_2, \ldots, x_n, \ldots$ имеет предел $\lim\limits_{n\to\infty} x_n=\bar{x}$, то $\bar{x}$ является корнем уравнения (1.7). В самом деле, предполагая, что $\varphi(x)$ — непрерывная функция, получим

$$\bar{x}=\lim_{n\to\infty} x_n=\lim_{n\to\infty}\varphi(x_{n-1})=\varphi\left(\lim_{n\to\infty} x_n\right)=\varphi(\bar{x}),$$

т. е.

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

Следовательно, $\bar{x}$ является корнем нашего уравнения. Поэтому одно из значений x_n с достаточно большим номером можно принять за приближенное значение корня. Однако может случиться, что последовательность $x_1, x_2, \ldots, x_n, \ldots$ не имеет предела, и тогда метод итераций не приводит к цели.

Представляет большой интерес выяснить условия, при которых итерационный процесс сходится. Имеет место следующая теорема.

Теорема. *Пусть интервал* $[a, b]$ *является интервалом изоляции корня уравнения* $x=\varphi(x)$ *и во всех точках этого интервала производная* $\varphi'(x)$ *удовлетворяет неравенству*

$$|\varphi'(x)|\leqslant M<1. \tag{2.7}$$

Если при этом выполняется условие $a \leqslant \varphi(x) \leqslant b$, *то итерационный процесс сходится, причем за нулевое приближение* x_0 *можно брать любую точку интервала* $[a, b]$ *).

Доказательство. Пусть дано уравнение $x = \varphi(x)$ и интервал $[a, b]$ изоляции корня этого уравнения. Будем считать, что функция $\varphi(x)$ дифференцируема и ее производная удовлетворяет условию (2.7).

Пусть x_0 — любая точка интервала $[a, b]$ и $x_1 = \varphi(x_0)$ — первое приближение. Если $\bar{x}$ — точный корень уравнения, то, применяя теорему Лагранжа, получим

$$\bar{x} - x_1 = \varphi(\bar{x}) - \varphi(x_0) = (\bar{x} - x_0)\,\varphi'(\xi_0),$$

где точка ξ_0 лежит между точками $\bar{x}$ и x_0, т. е. заведомо на интервале $[a, b]$. Согласно неравенству (2.7) будем иметь

$$|\bar{x} - x_1| = |\bar{x} - x_0| \cdot |\varphi'(\xi_0)| \leqslant M\,|\bar{x} - x_0|.$$

Для второго приближения $x_2 = \varphi(x_1)$ аналогично получим (по условию теоремы точка x_1 принадлежит интервалу $[a, b]$)

$$\bar{x} - x_2 = \varphi(\bar{x}) - \varphi(x_1) = (\bar{x} - x_1)\,\varphi'(\xi_1),$$

где ξ_1 заключено опять-таки между a и b. Применяя предыдущее неравенство, установим оценку:

$$|\bar{x} - x_2| \leqslant |\bar{x} - x_0|\,M^2.$$

Повторяя указанный процесс, найдем, что

$$|\bar{x} - x_n| \leqslant |\bar{x} - x_0| \cdot M^n. \tag{3.7}$$

Так как $M < 1$, то $M^n \to 0$ и, следовательно,

$$\lim_{n\to\infty} (\bar{x} - x_n) = 0, \text{ т. е. } \lim_{n\to\infty} x_n = \bar{x}.$$

---

*) Из неравенства (2.7) уже следует, что уравнение (1.7) не может иметь в интервале $[a, b]$ двух корней. В самом деле, допустив противное: $x_1 = \varphi(x_1)$ и $x_{11} = \varphi(x_{11})$, придем к равенству $x_1 - x_{11} = \varphi(x_1) - \varphi(x_{11}) = (x_1 - x_{11}) \cdot \varphi'(\xi)$, которое невозможно, так как $|\varphi'(\xi)| < 1$. При соблюдении условия $a \leqslant x \leqslant b$ и сохранении знака производной все последующие приближения будут лежать в интервале $[a, b]$. Если $0 \leqslant \varphi'(x) < 1$, то это условие следует само собой. Если же $-1 < \varphi'(x) \leqslant 0$, то нужно только проверить, что первое приближение x_1 не выходит за границы интервала $[a, b]$; тогда остальные приближения будут обязательно лежать в $[a, b]$.

Таким образом, для сходимости итерационного процесса достаточно, чтобы неравенство $|\varphi'(x)|<1$ соблюдалось на рассматриваемом интервале. При этом неравенство (3.7) позволяет дать оценку точности приближения. Так как $|\bar{x}-x_0|<b-a$, то

$$|\bar{x}-x_n|<(b-a)\,M^n. \qquad (4.7)$$

Пример 1.7. Решим методом итераций уравнение

$$5x-8\ln x=8.$$

Представив это уравнение в виде

$$5x/8-1=\ln x,$$

графически найдем, что оно имеет два корня, приближенно равные $x_0\approx 0{,}45$ и $x_0\approx 3{,}7$. (Рекомендуем читателю проделать построение самостоятельно.) Эти значения можно принять за нулевые приближения.

Для более точного нахождения правого корня запишем уравнение в виде

$$x=1{,}6\,(1+\ln x).$$

Здесь $\varphi(x)=1{,}6\,(1+\ln x)$. Итерационный процесс сходится, так как производная $\varphi'(x)=1{,}6/x$ в окрестности правого корня положительна и меньше единицы. Вычисления приведены в табл. 1.7.

Таблица 1.7

(1)	(2)	(3)
x	«1» + ln (1)	$\varphi(x)$ = «1,6» · (2)
3,7	2,3	3,68
3,68	2,303	3,685
3,685	2,3043	3,6869
3,6869	2,30479	3,68766
3,68766	2,304992	3,687987
3,687987	2,3050808	3,6881293
3,6881293	2,30511936	3,68819098
3,68819098	2,30513608	3,68821773
3,68821773	2,30514334	3,68822934
3,68822934	2,30514649	3,68823656
3,68823656	2,30514845	3,68823752
3,68823752	2,30514871	3,68823794
3,68823794	2,30514882	3,68823811
3,68823811	2,30514887	3,68823819
3,68823819		

При отыскании левого корня итерационный процесс окажется расходящимся, потому что $\varphi'(x) = 1{,}6/x$ в окрестности точки $x = 0{,}45$ имеет значение около 3,5. Поэтому первоначальное уравнение следует переписать иначе. Записав его в виде

$$x = e^{0{,}625x - 1},$$

найдем, что $\varphi(x) = e^{0{,}625x-1}$ и $\varphi'(x) = 0{,}625e^{0{,}625x-1}$. Здесь производная тоже положительна и меньше единицы (около 0,3). Вычисления, приведенные в табл. 2.7, показывают, что для вычисления корня с точностью до шести знаков требуется одиннадцать шагов.

Таблица 2.7

(1)	(2)	(3)
x	«0,625» · (1) — «1»	$\varphi(x)$ $e^{(2)}$
0,45	—0,719	0,487
0,487	—0,6956	0,4988
0,4988	—0,68825	0,50245
0,50245	—0,68597	0,503601
0,503601	—0,685249	0,5039647
0,5039647	—0,6850221	0,50407909
0,50407909	—0,68495057	0,504126306
0,504126306	—0,684921059	0,504130022
0,504130022	—0,684918736	0,504131193
0,504131193	—0,684918005	0,504131561
0,504131561		

Для приведения уравнения $f(x) = 0$ к виду $x = \varphi(x)$ таким образом, чтобы получить сходящийся итерационный процесс, часто поступают так: уравнение $f(x) = 0$ равносильно уравнению $\lambda f(x) = 0$; прибавив x к левой и правой частям этого равенства, приходим к уравнению

$$\lambda f(x) + x = x, \tag{5.7}$$

в котором уже можно положить $\lambda f(x) + x = \varphi(x)$, так что уравнение (5.7) приведется к виду (1.7):

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

Параметр λ остался свободным, и его можно подобрать таким образом, чтобы в окрестности корня $\varphi'(x) = \lambda f'(x) + 1$ было меньше единицы по абсолютной величине, что будет гарантировать сходимость итерационного процесса.

Иногда прибегают и к другим искусственным приемам для преобразования уравнения к виду, обеспечивающему сходимость итерационного процесса.

Пример 2.7. Выведем формулу для приближений к кубическому корню. Речь идет, следовательно, о решении уравнения $x^3 = a$. Преобразуем наше уравнение так. Сначала прибавим к обеим его частям $2x^3$. Тогда

$$3x^3 = a + 2x^3.$$

Разделив обе части равенства на $3x^2$ (очевидно, $x \neq 0$), найдем

$$x = a/3x^2 + 2x/3,$$

т. е. $\varphi(x) = a/3x^2 + 2x/3$. Производная этой функции равна

$$\varphi'(x) = \frac{2}{3}\frac{x^3 - a}{x^3}.$$

Отсюда видно, что если выбрать начальное приближение достаточно близким к истинному значению корня, то выполнение неравенства $|\varphi'(x)| < 1$, гарантирующего сходимость итерационного процесса, будет обеспечено.

Если x_{n-1} означает некоторое приближение для кубического корня, то следующее приближение получается по формуле

$$x_n = \frac{1}{3}\frac{a}{x_{n-1}^2} + \frac{2}{3}x_{n-1}.$$

Этой формулой мы уже пользовались в первом выпуске (см. § 14).

Интересно рассмотреть геометрический смысл итерационного процесса. Построим графики функций $y = \varphi(x)$ и $y = x$ (рис. 9). Корнем уравнения является абсцисса точки пересечения кривой $y = \varphi(x)$ с биссектрисой координатного угла. Если x_0 — абсцисса нулевого приближения, то $x_1 = \varphi(x_0)$ равно ординате соответствующей точки M кривой или же абсциссе точки M_1. Аналогично

находятся следующие приближения (рис.9). Здесь можно также установить роль условия $|\varphi'(x)|<1$.

Так, рис. 9 изображает случай, когда $0<\varphi'(x)<1$, так что кривая пересекает биссектрису слева направо и

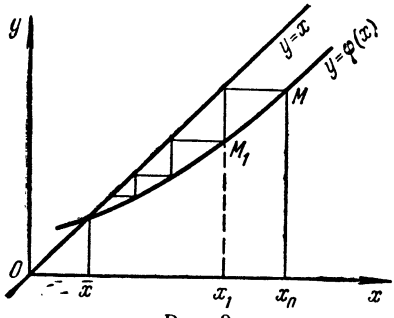


Рис. 9.

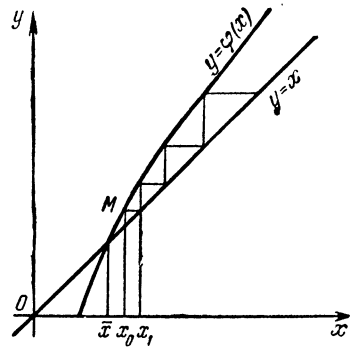


Рис. 10.

справа лежит под биссектрисой. Итерационный процесс в этом случае сходится, причем приближения монотонно убывают, если $x_0>\bar{x}$, или монотонно возрастают при

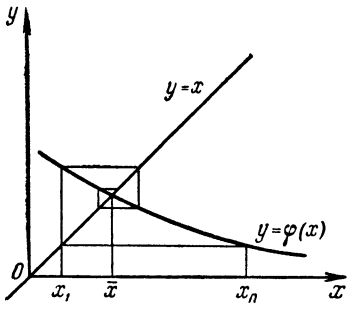


Рис. 11.

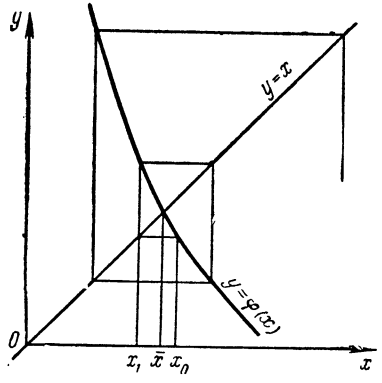


Рис. 12.

$x_0<\bar{x}$. На рис. 10 приведен случай, когда $\varphi'(x)>1$. Здесь кривая пересекает биссектрису снизу вверх, процесс оказывается расходящимся. На рис. 11 и 12

соответственно изображены случаи, когда производная $\varphi'(x)$ отрицательна. Если при этом $|\varphi'(x)|<1$ (рис. 11), то итерационный процесс сходится, но приближения колеблются около истинного значения корня. При $|\varphi'(x)|>1$ (рис. 12) процесс расходится.

§ 8. Случай алгебраического уравнения. Комплексные корни

Рассматривавшиеся до сих пор методы одинаково применимы как к алгебраическим уравнениям, так и к трансцендентным. Дополнительное предположение о том, что данное уравнение является алгебраическим, облегчает отделение корней.

Пусть уравнение $f(x)=0$ является алгебраическим уравнением n-й степени и записано в виде

$$f(x)\equiv a_0x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n=0. \qquad (1.8)$$

Как известно из курса алгебры, такое уравнение в поле комплексных чисел имеет ровно n корней, если каждый корень считать столько раз, какова его кратность. Для уравнений (1.8) с действительными коэффициентами, которыми мы ограничимся, обычно ставится задача отыскания всех его корней, как действительных, так и комплексных.

Из уже упоминавшейся в § 1 теоремы Безу вытекает, что если α является корнем уравнения (1.8), то многочлен $f(x)$ делится на $x-\alpha$. Поэтому, разделив многочлен $f(x)$ на двучлен $x-\alpha$, придем к уравнению вида $(x-\alpha)\varphi(x)=0$, где $\varphi(x)$ — частное от деления. Отсюда видно, что для отыскания следующего действительного корня уравнения (1.8) надо решать уравнение $\varphi(x)=0$ степени $n-1$.

Таким образом можно найти все действительные корни уравнения (1.8). Деление многочлена на двучлен $x-\alpha$, как и вычисление значений многочлена, требуемых для методов приближенного отыскания корней, лучше производить по схеме Горнера, разобранной в § 1.

Следующая теорема позволяет определить границы для корней уравнения (1.8).

Теорема. *Если*

$$|x|\geqslant 1+A/|a_0|, \qquad (2.8)$$

где A — *наибольшее из чисел* $|a_1|, |a_2|, \ldots, |a_n|$, *то*

$$|a_0x^n| > |a_1x^{n-1}+a_2x^{n-2}+\ldots+a_{n-1}x+a_n|. \tag{3.8}$$

Доказательство. Из (2.8) выводим

$$|x|-1 \geqslant A/|a_0|, \text{ или } |a_0| \geqslant A/(|x|-1),$$

откуда

$$|a_0x^n| \geqslant A|x|^n/(|x|-1). \tag{4.8}$$

С другой стороны,

$$|a_1x^{n-1}+a_2x^{n-2}+\ldots+a_{n-1}x+a_n| \leqslant$$
$$\leqslant |a_1|\cdot|x|^{n-1}+|a_2|\cdot|x|^{n-2}+\ldots+|a_{n-1}|\cdot|x|+|a_n| \leqslant$$
$$\leqslant A(|x^{n-1}|+|x^{n-2}|+\ldots+|x|+1) = A\frac{|x|^n-1}{|x|-1}.$$

Из (2.8) следует $|x|>1$. Поэтому

$$\frac{|x|^n-1}{|x|-1} < \frac{|x|^n}{|x|-1}$$

и, следовательно,

$$|a_1x^{n-1}+a_2x^{n-2}+\ldots+a_{n-1}x+a_n| < A\frac{|x|^n}{|x|-1}. \tag{5.8}$$

Сравнение неравенств (4.8) и (5.8) приводит к неравенству (3.8), которое и требовалось доказать.

Если x является корнем уравнения (1.8), то должно иметь место равенство

$$|a_0x^n| = |a_1x^{n-1}+\ldots+a_{n-1}x+a_n|.$$

Поэтому значения x, удовлетворяющие неравенству (2.8), вследствие неравенства (3.8) корнями уравнения (1.8) служить не могут. Итак, корни алгебраического уравнения (1.8) удовлетворяют неравенству

$$|x| < 1+A/|a_0|, \tag{6.8}$$

или иначе, *число* $N = 1+A/|a_0|$ *служит верхней границей модулей корней алгебраического уравнения.*

Оценке (6.8) удовлетворяют модули всех корней уравнения, как действительных, так и комплексных. Если речь идет только о действительных корнях, то можно получить значительно более точную оценку.

Прежде всего, можно ограничиться лишь положительными корнями уравнения. В самом деле, если число t является положительным корнем уравнения $f(-x)=0$, то число $-t$ будет отрицательным корнем уравнения $f(x)=0$. Поэтому вместо непосредственного нахождения отрицательных корней уравнения $f(x)=0$ достаточно найти положительные корни уравнения $f(-x)=0$.

Пусть $a_0>0$. Если все последующие коэффициенты многочлена положительны, то при любом $x>0$ будем иметь $f(x)>0$, так что уравнение не имеет положительных корней; если положительные корни есть, то среди коэффициентов должны быть отрицательные. Пусть a_k $(k \geqslant 1)$ является старшим (первым слева) отрицательным

коэффициентом. Обозначим через B наибольшую из абсолютных величин отрицательных коэффициентов. Тогда *верхней границей положительных корней уравнения* (1.8) *служит число*

$$1+\sqrt[k]{B/a_0}. \tag{7.8}$$

Доказательство этой оценки вполне аналогично доказательству предыдущей.

Нижнюю границу корней можно получить так. Заменив в уравнении (1.8) x на $1/x$, получим новое уравнение

$$\frac{a_0}{x^n}+\frac{a_1}{x^{n-1}}+\ldots+\frac{a_{n-1}}{x}+a_n=\frac{1}{x^n}\left(a_0+a_1x+\ldots+a_{n-1}x^{n-1}+a_nx^n\right)=0,$$

корни которого обратны по величине корням исходного (можно считать, что $a_n\neq 0$).

Если K — верхняя граница положительных корней уравнения

$$a_0+a_1x+\ldots+a_{n-1}x^{n-1}+a_nx^n=0, \tag{8.8}$$

то число $1/K$ будет служить нижней гранью корней уравнения (1.8).

Необходимо только помнить, что указание границ положительных корней уравнения не означает, что такие корни на самом деле имеются.

Пример 1.8. Рассмотрим алгебраическое уравнение

$$x^5+5x^4-2x^3-4x^2+7x-3=0.$$

Здесь $a_0=1$, $A=7$. Правая часть неравенства (6.8) равна $1+7/1=8$ так что все корни этого уравнения по модулю не превосходят 8.

Первым отрицательным коэффициентом является $a_2=-2$, так что $k=2$, $B=4$, поэтому согласно оценке (7.8) верхняя граница положительных корней равна $1+\sqrt{4/2}=2{,}41$. Для нахождения нижней границы составим уравнение (8.8):

$$-3x^5+7x^4-4x^3-2x^2+5x+1=0$$

или, после перемены знаков,

$$3x^5-7x^4+4x^3+2x^2-5x-1=0.$$

Здесь $a_0=3$, $k=1$, $B=7$. Верхняя граница положительных корней нового уравнения равна $1+7/3=3{,}33$, откуда следует, что нижняя граница положительных корней первоначального уравнения $1/3{,}33=0{,}30$. Таким образом, положительные корни уравнения, если они существуют, удовлетворяют неравенствам $0{,}30<x<2{,}41$.

Для отыскания границ отрицательных корней заменим в данном уравнении x на $-x$. Получим

$$-x^5+5x^4+2x^3-4x^2-7x-3=0$$

или

$$x^5-5x^4-2x^3+4x^2+7x+3=0.$$

Здесь $a_0=1$, $k=1$, $B=5$, поэтому верхняя граница положительных корней этого уравнения $1+5/1=6$. Составив снова уравнение (8.8),

найдем

$$3x^5+7x^4+4x^3-2x^2-5x+1=0,$$

где $a_0=3$, $k=3$ и $B=5$, так что верхняя граница положительных корней для этого уравнения $1+\sqrt[3]{5/3}=2{,}18$, что дает нижнюю границу корней уравнения с измененным знаком 0,46. Итак, окончательно, отрицательные корни исходного уравнения (если они существуют) находятся в границах $-6<x<-0{,}46$.

Рассмотрим теперь способ приближенного нахождения комплексных корней алгебраического уравнения с действительными коэффициентами. Можно ограничиться рассмотрением уравнения, имеющего лишь комплексные корни, предполагая действительные уже выделенными. Так как такое выделение происходит приближенно, то корни нового уравнения будут несколько отличаться от корней исходного; однако, заниматься оценками допущенной погрешности мы не будем.

Комплексные корни алгебраического уравнения с действительными коэффициентами являются попарно сопряженными. Если $\alpha\pm\beta i$ — два таких корня, то произведение соответствующих линейных множителей дает

$$[x-(\alpha+\beta i)]\,[x-(\alpha-\beta i)]=x^2-2\alpha x+(\alpha^2+\beta^2),$$

т. е. квадратный трехчлен с действительными коэффициентами. Следовательно, для отыскания пары сопряженных комплексных корней достаточно выделить множитель вида x^2+px+q.

Для выделения такого множителя можно воспользоваться схемой деления многочлена на квадратный трехчлен, аналогичной схеме Горнера. Ограничимся многочленом четвертой степени. Результат деления его на квадратный трехчлен можно записать в виде

$$a_0x^4+a_1x^3+a_2x^2+a_3x+a_4=$$
$$=(x^2+px+q)\,(b_0x^2+b_1x+b_2)+b_3\,(x+p)+b_4. \qquad (9.8)$$

Раскрывая скобки и приравнивая коэффициенты при одинаковых степенях x, приходим к системе

$$\begin{aligned}
a_0&=b_0,\\
a_1&=b_0p+b_1,\\
a_2&=b_0q+b_1p+b_2,\\
a_3&=b_1q+b_2p+b_3,\\
a_4&=b_2q+b_3p+b_4.
\end{aligned}$$

Отсюда находим

$$\left.\begin{aligned}
b_0&=a_0,\\
b_1&=a_1-pb_0,\\
b_2&=a_2-pb_1-qb_0,\\
b_3&=a_3-pb_2-qb_1,\\
b_4&=a_4-pb_3-qb_2.
\end{aligned}\right\} \qquad (10.8)$$

Нашей задачей является подбор таких значений p и q, при которых $b_3=b_4=0$, тогда как при произвольно выбранных p и q это

условие не выполняется. Пусть выбраны некоторые значения p и q. Покажем, как выбрать поправки к этим значениям, чтобы уменьшить полученные значения b_3 и b_4.

Пусть новые значения коэффициентов будут $p+\Delta p$, $q+\Delta q$. Не изменяя первых вычисленных коэффициентов, подставим эти значения в последние два уравнения (10.8). Тогда

$$a_3-(p+\Delta p)\,b_2-(q+\Delta q)\,b_1=\bar{b}_3,$$
$$a_4-(p+\Delta p)\,b_3-(q+\Delta q)\,b_2=\bar{b}_4.$$

Полагая $\bar{b}_3=\bar{b}_4=0$ и замечая, что $a_3-pb_2-qb_1=b_3$, $a_4-pb_3-qb_2=b_4$, находим систему уравнений для отыскания поправок Δp, Δq:

$$\left.\begin{aligned} b_2\Delta p+b_1\Delta q&=b_3,\\ b_3\Delta p+b_2\Delta q&=b_4,\end{aligned}\right\} \tag{11.8}$$

с помощью которых находятся новые значения $p_1=p+\Delta p$, $q_1=q+\Delta q$.

Теперь можно пересчитать значения коэффициентов $b_0, \ldots, b_4$ по формулам (10.8) и определить новые поправки из системы (11.8). Это делается до тех пор, пока коэффициенты b_3, b_4 не станут пренебрежимо малыми по сравнению с остальными.

Остается указать способ выбора первоначальных значений p и q. Если x по модулю невелико (меньше единицы), то старшие степени малы по сравнению с младшими. Поэтому меньшие по модулю корни могут приближаться корнями квадратного уравнения

$$a_2x^2+a_3x+a_4=0,$$

т. е. в качестве начальных значений можно брать $p=a_3/a_2$, $q=a_4/a_2$.

Наоборот, для больших $|x|$ младшие члены будут малы по сравнению со старшими и корни уравнения можно пытаться приближать парой корней квадратного уравнения

$$a_0x^2+a_1x+a_2=0,$$

т. е. принимать в качестве исходных значений $p=a_1/a_0$, $q=a_2/a_0$.

Вычисления располагают обычно в виде схемы, показанной в табл. 1.8. Верхняя ее часть предназначена для нахождения коэффициентов, нижняя — для вычисления поправки.

Пример 2.8. Решим уравнение $x^4-2x^3+7x^2+2x+15=0$.

Начальные значения выберем, исходя из квадратного трехчлена $7x^2+2x+15$, или, после округления, $p=0{,}3$, $q=2{,}1$. Вычисления приведены в табл. 2.8.

Проделав три шага, остановимся на значениях $p=0{,}961$, $q=1{,}8701$. Следующий шаг проделан лишь для проверки величины остатка и нахождения коэффициентов частного. Пренебрегая остатком, найдем, что заданное уравнение распадается на два

$$(x^2+0{,}961x+1{,}8701)\,(x^2-2{,}961x+7{,}9754)=0,$$

откуда получаем четыре комплексных корня

$$x_{1,2}=-0{,}480\pm 1{,}281i,\quad x_{3,4}=1{,}480\pm 2{,}405i.$$

Таблица 1.8

(1)	(2)	(3)	(4)
a	p	q	b
a_0			$b_0=a_0$
a_1	$-b_0p$		$b_1=a_1-b_0p$
a_2	$-b_1p$	$-b_0q$	$b_2=a_2-b_1p-b_0q$
a_3	$-b_2p$	$-b_1q$	$b_3=a_3-b_2p-b_1q$
a_4	$-b_3p$	$-b_2q$	$b_4=a_4-b_3p-b_2q$
b_3	b_2	b_1	
b_4	b_3	b_2	
$D=b_2^2-b_1b_3$	$\Delta p=$ $=(b_3b_2-b_1b_4)/D$	$\Delta q=$ $=(b_2b_4-b_3^2)/D$	

В качестве заключительного контроля вычислим сумму и произведение всех корней, которые должны равняться $-a_1/a_0$ и a_4/a_0 (для нашего уравнения соответственно 2 и 15). Подсчеты дают $\sum x_i=2{,}000$ и $\prod x_i=14{,}923$.

Рассмотренный процесс нахождения поправок может оказаться расходящимся. Более устойчивым и быстрым является нахождение поправок из аналогичной системы

$$\left.\begin{aligned} c_2\Delta p+c_1\Delta q&=b_3,\\ c_3\Delta p+c_2\Delta q&=b_4, \end{aligned}\right\} \tag{12.8}$$

где коэффициенты c_i получаются по формулам

$$\left.\begin{aligned} c_0&=b_0,\\ c_1&=b_1-pc_0,\\ c_2&=b_2-pc_1-qc_0,\\ c_3&=\quad -pc_2-qc_1. \end{aligned}\right\} \tag{13.8}$$

На выводе этих формул мы останавливаться не будем. Отметим только, что вычисления можно производить по такой же схеме, как и табл. 1.8, добавив несколько столбцов для нахождения коэффициентов c_i.

Таблица 2.8

(1)	(2)	(3)	(4)
шаг I	$p=0,3$	$q=2,1$	
1			1
—2	—0,3		—2,3
7	0,69	—2,1	5,59
2	—1,68	4,83	5,15
15	—1,54	—11,74	1,72
5,15	5,59	—2,3	
1,72	5,15	5,59	
$D=43,09$	$\Delta p=0,76$	$\Delta q=-0,39$	
шаг II	$p=1,06$	$q=1,71$	
1			1
—2	—1,060		—3,060
7	3,244	—1,710	8,534
2	—9,046	5,233	—1,813
15	1,922	—14,593	2,329
—1,813	8,534	—3,060	
2,329	—1,813	8,534	
$D=67,281$	$\Delta p=-0,124$	$\Delta q=0,247$	
шаг III	$p=0,936$	$q=1,954$	
1			1
—2	—0,9360		—2,9360
7	2,7481	—1,9540	7,7941
2	—7,2953	5,7369	0,4416
15	—0,4133	—15,2297	—0,6430
0,4416	7,7941	—2,9360	
—0,6430	0,4416	7,7941	
$D=62,0445$	$\Delta p=0,0250$	$\Delta q=-0,0839$	

Продолжение

(1)	(2)	(3)	(4)
шаг 1V	$p = 0{,}9610$	$q = 1{,}8701$	
1			1
—2	—0,96100		—2,96100
7	2,84552	—1,87010	7,97542
2	—7,66438	5,53737	—0,12701
15	0,12206	—14,91483	0,20723

§ 9. Программирование подбора корней

Пусть ищется корень уравнения

$$f(x) = 0$$

на участке $[a, b]$. Предположим, что вычисление функции $f(x)$ осуществляется с помощью отдельной подпрограммы *Счет f*, аргументом которой служит ячейка x, а ответом — ячейка γ.

Подбор корней можно осуществить следующим образом.

Выберем некоторый начальный шаг Δ и будем вычислять значения $f(x)$, начиная с точки a, с шагом Δ до тех пор, пока функция не изменит знак или пока мы не дойдем до точки b. Во втором случае уравнение не имеет корней и в ответную ячейку мы будем засылать *Щ*, как признак отсутствия корня. В первом случае уменьшим шаг Δ (например, в восемь раз) и сменим у него знак, т. е. пойдем от полученной точки в обратную сторону. Такой процесс можно продолжать до тех пор, пока мы не получим корня с достаточной степенью точности, т. е. пока шаг Δ не сделается достаточно малым.

Описанный процесс можно осуществить с помощью следующей программы, в которой ответной ячейкой также

служит ячейка γ:

1)			Δ_0	$=\Delta$	9)		$R_0 \cdot$	γ $=R_1$
2)			a	$=x$	10)		R_1+	0 $=0$
3)		*Счет f*			11)	*У0*	γ	5) R_0
4)			γ	$=R_0$	12)		$\Delta \cdot «\left(-\frac{1}{8}\right)» = \Delta$	
5)		$x+$	Δ	$=x$	13)		$\lvert\varepsilon\rvert -$	$\lvert\Delta\rvert = 0$
6)		$b-$	x	$=0$	14)	*У1*	x	5) γ
7)	*У1*	*Щ*	15)	γ	15)	*стоп*		
8)		*Счет f*						

Здесь команда 1) засылает начальное значение шага в рабочую ячейку, команды 2)—4) вычисляют $f(a)$ и засылают это значение в ячейку R_0. Далее идет увеличение аргумента на Δ (5)), проверка достижения правого конца (6)—7)) и вычисление функции в новой точке (8)). Команды 9)—11) проверяют совпадение знаков у двух значений функции. Если произведение этих значений R_1 положительно, то мы возвращаемся к команде 5) и делаем еще один шаг вправо. Если же $R_1 < 0$, то функция изменила знак. Тогда мы переходим к команде 12), которая уменьшит шаг и изменит его знак. Далее, команды 13)—15) проверят, не достигнута ли нужная точность и либо возвратят на команду 5), либо остановят машину.

Написанная нами программа не является стандартной, так как здесь предполагаются известными ячейки a, b, Δ_0, x, ε, *Счет f*. Для стандартной программы все эти сведения должны быть заданы в информации к программе, а команды, содержащие адреса этих ячеек, должны быть сформированы. В нашем случае необходимо формировать команды 1), 2), 3), 5), 6), 8), 13), 14), т. е. восемь команд.

Как мы видели в § 37 первого выпуска, формирование каждой команды занимает довольно много места. Поэтому выгоднее несколько изменить программу так, чтобы уменьшить в ней число формируемых команд. Кроме того, если мы хотим сделать программу блоком, а не самостоятельной программой, то следует заменить команду *стоп* неперфорируемой ячейкой *конец* и начать блок командой Ω = конец.

Приведенную программу можно изменить таким образом:

```
1)                Ω  =конец      10)   R0 ·  γ   =R1
2)                Δ0 =Δ          11)   R1+  0   =0
3)                a  =z                ————————→
4) БВ  5)        17)  Ω          12) У0  γ   6)   R0
5)                γ  =R0         13)   Δ ·«(— 1/8)»=Δ
6)           z+   Δ  =z          14)   |ε|—|Δ|   =0
7)           b —  z  =0                  ————————→
             ————————→           15) У0  z   6)   γ
8) У1  Щ        16)  γ           16)   конец
9) БВ 10)       17)  Ω           17) Б   z  Счет f   x
```

Вместо ячейки x «для внутреннего употребления» здесь использована рабочая ячейка z. Пересылка значения z в аргумент x производится одновременно с обращением к функции в команде 17). Обращение к программе счета функции происходит только из одного места, причем команда возврата засылается в ячейку Ω раньше — командой 9) (в цикле) или 4) (в начале работы программы). В этом варианте программы нужно сформировать только пять команд: 2), 3), 7), 14) и 17), которые мы обозначим соответственно A, B, C, D, E.

Пусть обращение к нашей программе *Корень* имеет вид *)

$Я$:	*БВ*	$Я+3$	*Корень*	Ω
$Я+1$		a	b	x
$Я+2$		f	Δ_0	ε

Напишем формирование команд $A-E$ для стандартной программы по заданной информации.

Прежде всего необходимо перенести информацию из ячеек $Я+1$ и $Я+2$ в рабочие ячейки. Этими рабочими ячейками могут быть, например, ячейки R_0 и R_1. Для переноса ячейки $Я+2$ в ячейку R_1 нужно сформировать команду

$$Q\colon\ 0 \vee Я+2 = R_1.$$

Так как в среднем адресе ячейки Ω после выполнения команды $Я$ обращения к корню записан адрес $Я+3$, то команду Q можно получить из константы $F \vee F = R_1$, где F означает число 7777, путем фиксированного сложения

$$(F \vee F = R_1)+,\quad \Omega = Q.$$

Действительно, при этом средний адрес команды сделается равным $Я+2$, а левый — нулю. Для переноса ячейки $Я+1$ в R_0 надо иметь команду P

$$P\colon\ 0 \vee Я+1 = R_0,$$

*) Для удобства формирования здесь и в дальнейшем при обращении к стандартным подпрограммам мы засылаем в Ω возврат на ячейку, следующую за информацией.

которую легко получить переадресацией команды Q:

$$Q —, \quad (0,\ 1,\ 1) = P.$$

Сформировав и выполнив команды P и Q, мы перенесем информацию в ячейки R_0 и R_1, так что их содержимое будет

$$R_0\colon\ a\quad b\quad x,$$
$$R_1\colon\ f\quad \Delta_0\quad \varepsilon.$$

Команду

$$A\colon\ \Delta_0 = \Delta$$

можно теперь сформировать из заготовки $0 \vee 0 = \Delta$, прибавив к среднему адресу этой заготовки средний адрес ячейки R_1. Это можно выполнить двумя командами:

$$\begin{array}{rcl} R_1 & \wedge & (0,\ F,\ 0) = R_2 \\ (0 \vee 0 = \Delta) & \vee & R_2 \quad = A \end{array}$$

Аналогично формируется и команда B, которую можно записывать в виде $a \vee 0 = z$.

Для формирования команды

$$C\colon\ b - z = 0$$

надо взять заготовку $0 - z = 0$ и прибавить адрес b к ее левому адресу. Адрес b записан в среднем адресе ячейки информации. Поэтому его надо после высечения сдвинуть. На формирование команды C придется потратить уже три команды

$$\begin{array}{rcl} \underline{R_0} & \wedge & (0,\ F,\ 0) = R_2 \\ \overline{114} & \rightarrow, & R_2 \quad = R_2 \\ (0 - z = 0) & \vee & R_2 \quad = C \end{array}$$

Аналогично формируются и все остальные команды.

Приведем теперь стандартную программу *Корень* в окончательном виде, со всем формированием и всеми константами:

	1)			Ω =конец
	2)	$(F \vee F = R_1)$	+ ,	Ω $= Q$
Q	3)	(0	$\vee$	$Я + 2$ $= R_1$) н. п.
	4)	Q	— ,	$(0,\ 1,\ 1) = P$
P	5)	(0	$\vee$	$Я + 1$ $= R_0$) н. п.
	6)	R_1	$\wedge$	$(0,\ F,\ 0) = R_2$
	7)	$(0 \vee 0 = \Delta)$	$\vee$	$R_2 \quad = A$
	8)	R_0	$\wedge$	$(F,\ 0,\ 0) = R_2$
	9)	$(0 \vee 0 = z)$	$\vee$	$R_2 \quad = B$
	10)	R_0	$\wedge$	$(0,\ F,\ 0) = R_2$
	11)	$\overline{114}$	$\rightarrow$,	$R_2 \quad = R_2$

	12)	$(0-z=0)$	$\vee$	R_2	$=C$
	13)	$\overline{130}$	$\rightarrow$,	R_1	$=R_2$
	14)	$(\lvert 0\rvert-\lvert\Delta\rvert=0)$	$\vee$	R_2	$=D$
	15)	R_0	$\wedge$	$(0,\ 0,\ F)$	$=R_2$
	16)	R_1	$\wedge$	$(F,\ 0,\ 0)$	$=R_3$
	17)	$\overline{64}$	$\rightarrow$,	R_3	$=R_3$
	18)	R_2	$\vee$	R_3	$=R_2$
	19)	$(Б\ z\ \overrightarrow{0\ 0})$	$\vee$	R_2	$=E$
A	20)	(		Δ_0	$=\Delta)$ н. п.
B	21)	(a	$\vee$	0	$=z)$ н. п.
	22) *БВ*	$Я+1$		E	Ω
	23)			γ	$=R_0$
	24)	z	$+$	Δ	$=z$
C	25)	(b	$-$	z	$=0)$ н. п.
		$\longrightarrow$			
	26) *У1*	*Щ*		*конец*	γ
	27) *БВ*	$Я+1$		E	Ω
	28)	R_0	$\cdot$	γ	$=R_0$
	29)	R_0	$+$	0	$=0$
		$\longrightarrow$			
	30) *У0*	γ		24)	R_0
	31)	Δ	$\cdot$	$\ll\left(-\frac{1}{8}\right)\gg$	$=\Delta$
D	32)	$(\lvert\varepsilon\rvert$	$-$	$\lvert\Delta\rvert$	$=0)$ н. п.
		$\longrightarrow$			
	33) *У1*	z		24)	γ
	34)	*конец*			н. п.
		$\longrightarrow$			
E	35) (*Б*	z	*Счет*	f	$x)$ н. п.
	36)	F	$\vee$	F	$=R_1$
	37)	0	$\vee$	0	$=\Delta$
	38)	0	$\vee$	0	$=z$
	39)	0	$-$	z	$=0$
	40)	$\lvert 0\rvert$	$-$	$\lvert\Delta\rvert$	$=0$
		$\longrightarrow$			
	41) *Б*	z		0	0

§ 10. Программы для способа хорд и касательных

Составим программу для решения уравнения $f(x)=0$ по способу хорд и касательных. Так как при этом потребуется вычислять значение функции $f(x)$ и ее первой и второй производных, то предположим, что уже написаны три соответствующих блока, которые мы будем называть *счет* $f(\alpha)$, *счет* $f'(\alpha)$ и *счет* $f''(\alpha)$. Они предполагаются оформленными как стандартные подпрограммы с входной ячейкой (аргументом) α и выходной (ответной) γ.

Применение способа хорд и касательных требует задания первоначального интервала изоляции корня. Пусть такой интервал $[a_0, b_0]$ известен, т. е. числа a_0 и b_0 уже лежат в соответствующих ячейках памяти. Кроме того, пусть задана точность ε, с которой мы хотим знать значение корня.

Требуемую программу можно написать как цикл, на каждом шаге которого вычисляются очередные значения (a_i, b_i) границ интервала, содержащего внутри себя искомый корень. Проверку окончания естественно организовать по длине этого интервала $\Delta=b_i-a_i$ и заканчивать вычисления при $\Delta<\varepsilon$. При этом, так как величина Δ участвует в вычислениях, то удобно поставить эту проверку не в конце, а в начале цикла. За значение корня можно принять тогда середину интервала $(a_i+b_i)/2$.

В начале программы мы выполним те действия, которые нужно выполнять во всех случаях, как, например, вычисление значений функции и ее первой и второй производных в какой-либо точке. Затем идет проверка выполнения условия $y'y''>0$ и, в зависимости от результатов этой проверки, разветвление программы. При $y'y''>0$ мы будем пользоваться формулами (4.6)—(5.6), а при $y'y''<0$—формулами (6.6)—(7.6). Так как во всех этих формулах поправки содержат знак минус, то вместо перемены знака после вычисления дроби мы будем для вычисления новых значений a и b пользоваться формулами

$$a_1=a-\Delta a,$$

$$b_1=b-\Delta b.$$

Написанную программу можно использовать как библиотечную. Поэтому мы напишем ее как блок, начиная с команды Ω = конец. После сделанных пояснений работа этой программы должна быть достаточно понятной.

```
                Ω      =конец
                a0     =a
                b0     =b
          b—    a      =Δ
          Δ—    ε      =0
У1              W
                a      =α
                f(α)
                γ      =f(a)
                f'(α)
                γ      =f'
                f''(α)
                γ      =R
                b      =α
                f(α)
                γ      =f(b)
          f(b)—f(a)    =Δf
          f' ·  R      =R
```

```
                R + 0  =0
                ------------>
У0              b   M  α
                f(a) : f'  =Δa
                Δ  : Δf    =R
                R  · f(b)  =Δb
Б                   N
M                   f'(α)
                    γ      =f'
                f(b) : f'  =Δb
                Δ  : Δf    =R
                R  · f(a)  =Δa
N               a  — Δa    =a
                b  — Δb    =b
Б                   □
W               a  + b     =R
                R  · «1/2» =γ
                конец
```

Эта программа, очевидно, не является стандартной. Чтобы сделать ее стандартной, необходимо, как мы это делали в предыдущем параграфе, сформировать команды, зависящие от информации. Легко видеть, что здесь таких команд довольно много.

Алгоритм способа хорд и касательных легко записать на алголе. При этом мы будем предполагать, что имеются три процедуры *F* (alpha, gamma), *F*1 (alpha, gamma), *F*2 (alpha, gamma), вычисляющие соответственно значения функции и ее первой и второй производных. Под *a*0 и *b*0 мы будем понимать определенные числа. Нужную программу можно написать, например, следующим

образом:

```
begin real a, b, fa, fb, f1, f2, delta, epsilon, gamma;
      a: =a0; b: =b0;
  M1: delta: =b—a; if delta<epsilon then go to K;
      F(a, fa); F(b, fb); F1(a, f1); F2(a, f2);
      if f1×f2<0 then go to M2;
      F1(b, f1); a: =a—(delta×fa)/(fb—fa); b:=b—fb/f1;
      go to M1;
  M2: a: =a—fa/f1; b: =b—(delta×fb)/(fb—fa);
                                  go to M1;
   K: gamma: =(a+b)/2
end
```

§ 11. Программирование итерационного процесса

Программирование итерационного процесса, собственно, сводится к написанию обычного итерационного цикла, который уже рассматривался в § 14 первого выпуска. Там же было приведено и некоторое число примеров. Мы можем теперь написать в самом общем виде программу для решения уравнения $x=\varphi(x)$ методом итераций, в предположении, что итерационный процесс сходится. Если блок счета $\varphi(x)$ написан как стандартная подпрограмма, считающая $\gamma=\varphi(\alpha)$, то программу нахождения корня можно записать в таком виде:

1)	$x_0 \quad =\alpha$	4)	$\lvert\delta\rvert-\lvert\varepsilon\rvert \quad =0$
2)	*БВ Я+1* $\varphi(\alpha)$ Ω	5)	*УО* γ 2) α
3)	$\gamma-\alpha \quad =\delta$	6)	*стоп*

Здесь x_0 — нулевое приближение, которое считается известным, а ε — требуемая точность.

Не составляет труда превратить эту программу в стандартную. Пусть, например, обращение к программе *Итерация* имеет вид

Я) *БВ Я+2 Итерация* Ω
Я+1) x_0 $\varphi(\alpha)$ ε

где x_0 — адрес ячейки, содержащей начальное приближение, $\varphi(\alpha)$ — адрес начала блока счета функции $\varphi(x)$ и ε — ячейка, содержащая требуемую точность. Как видно из приведенной программы, в ней

требуется формировать команды 1), 2) и 4), которые мы назовем соответственно A, B, C.

Перенос информации из ячейки $Я+1$ в рабочую ячейку R_0 достигается с помощью команд

$$\begin{array}{l} (F \vee F=R_0)+, \quad \Omega \quad =Q \\ Q:(\quad 0 \quad \vee \quad Я+1=R_0) \text{ н. п.} \end{array}$$

Команду 1) удобно формировать в виде $x_0 \vee 0=\alpha$, для чего достаточно написать

$$\begin{array}{l} R_0 \quad \wedge (F,\ 0,\ 0)=R_1 \\ (0 \vee 0=\alpha) \vee \quad R_1 \quad =A \end{array}$$

Так же легко формируются и остальные команды. Только для команды C высеченный адрес ε необходимо будет сдвинуть из правого адреса в средний.

Стандартная программа для метода итераций будет иметь следующий вид:

	1)			Ω	$=$*конец*	
	2)	$(F \vee F=R_0)$	$+,$	Ω	$=Q$	
Q	3)	$($ 0	$\vee$	$Я+1$	$=R_0)$	н. п.
	4)	R_0	$\wedge$	$(F,\ 0,\ 0)$	$=R_1$	
	5)	$(0 \vee 0=\alpha)$	$\vee$	R_1	$=A$	
	6)	R_0	$\wedge$	$(0,\ F,\ 0)$	$=R_1$	
	7)	$(БВ\ 13)\ 0\ \Omega)$	$\vee$	R_1	$=B$	
	8)	$\underline{R_0}$	$\wedge$	$(0,\ 0,\ F)$	$=R_1$	
	9)	114	$\rightarrow,$	R_1	$=R_1$	
	10)	$(\lvert\delta\rvert-\lvert 0\rvert=0)$	$\vee$	R_1	$=C$	
A	11)	$($ x_0	$\vee$	0	$=\alpha)$	н. п.
B	12)	$(БВ$ 13)		$\varphi(\alpha)$	$\Omega)$	н. п.
	13)	γ	$-$	α	$=\delta$	
C	14)	$($ $\lvert\delta\rvert$	$-$	$\lvert\varepsilon\rvert$	$=0)$	н. п.
	15)	*УО* γ		B	α	
	16)	*конец*				н. п.
Q_0	17)	F	$\vee$	F	$=R_0$	
	18)	0	$\vee$	0	$=\alpha$	
	19)	$БВ$ 13)		0	Ω	
	20)	$\lvert\delta\rvert$	$-$	$\lvert 0\rvert$	$=0$	

Используя регистр адреса, можно организовать перенос информации без формирования команды Q. Это потребует тех же двух ячеек памяти и двух операций, но окажется излишней заготовка Q_0 (ячейка 17)). Для переноса информации в ячейку R_0 можно взамен команд

2)—3) написать:

2)′ [PA] 0* Ω *предконец*
3)′ $F^* = R_0$

Однако теперь нам потребовалась еще неперфорируемая ячейка *предконец* для восстановления регистра. Ее можно поместить на место 16), сдвинув *конец* в освободившуюся ячейку 17). Таким образом, объем программы окажется прежним.

Программу решения уравнения методом итераций запишем теперь на алголе. Уравнение будем писать в виде $x = f(x)$ и предположим, что написана процедура F(alpha, gamma), вычисляющая $\gamma = f(\alpha)$. Кроме того, под $x0$ будем понимать определенное ч и с л о — начальное значение x. Программу можно написать, например, так:

```
begin real x, y, epsilon, gamma;
        y: =x0;
    M: x: =y, F (x, y);
        if (y—x > epsilon) ∨ (x—y > epsilon) then
                                        go to M;
        gamma: =y
end
```

ГЛАВА III

СИСТЕМЫ УРАВНЕНИЙ

§ 12. Решение системы линейных уравнений по способу Гаусса

Способ Гаусса является одним из наиболее распространенных способов решения систем линейных уравнений. Он является *точным*, т. е. если точно выполнить все требуемые в нем действия, то мы получим точное решение системы. Практически, впрочем, точного решения получить не удается, поскольку арифметические действия далеко не всегда могут быть выполнены вполне точно.

Способ Гаусса может быть реализован в виде различных вычислительных схем, в основе которых лежит одна и та же идея последовательного исключения неизвестных. Мы рассмотрим схему, которая называется *схемой единственного деления.*

Вычислительную схему удобно иллюстрировать на конкретном примере. Поэтому ограничимся рассмотрением системы четвертого порядка. Те же приемы могут быть применены и в любом другом случае.

Рассмотрим систему линейных уравнений четвертого порядка:

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + a_{15} = 0,\\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + a_{25} = 0,\\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 + a_{35} = 0,\\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 + a_{45} = 0, \end{aligned}\right\} \qquad (1.12)$$

в которой все члены для удобства дальнейших рассуждений перенесены в одну часть равенства. Примем,

что $a_{11} \neq 0$, либо, в противном случае, переставим уравнения так, чтобы это условие было выполнено. Находя x_1 из первого уравнения, получим

$$x_1 = \alpha_{12}x_2 + \alpha_{13}x_3 + \alpha_{14}x_4 + \alpha_{15}, \tag{2.12}$$

где

$$\alpha_{1i} = -a_{1i}/a_{11} \qquad (i = 2,\ 3,\ 4,\ 5).$$

С помощью уравнения (2.12) можно исключить из оставшихся уравнений x_1, для чего достаточно подставить значение (2.12) для x_1 во второе, третье в четвертое уравнения системы. Тогда мы придем к системе трех уравнений, не содержащих x_1:

$$\left.\begin{aligned} a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + a_{24}^{(1)}x_4 + a_{25}^{(1)} &= 0, \\ a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + a_{34}^{(1)}x_4 + a_{35}^{(1)} &= 0, \\ a_{42}^{(1)}x_2 + a_{43}^{(1)}x_3 + a_{44}^{(1)}x_4 + a_{45}^{(1)} &= 0. \end{aligned}\right\} \tag{3.12}$$

Из способа получения этой системы видно, что

$$a_{22}^{(1)} = a_{21} \cdot \alpha_{12} + a_{22}$$

и вообще

$$a_{ik}^{(1)} = a_{i1}\alpha_{1k} + a_{ik} \qquad (i = 2,\ 3,\ 4; \quad k = 2,\ 3,\ 4,\ 5).$$

Полученную систему (3.12) можно подвергнуть тому же преобразованию, что и первоначальную. Как и ранее, можно предположить, что $a_{22}^{(1)} \neq 0$, либо переставить уравнения (или неизвестные). Находя теперь x_2 из первого уравнения, получим

$$x_2 = \alpha_{23}x_3 + \alpha_{24}x_4 + \alpha_{25}, \tag{4.12}$$

где положено

$$\alpha_{2i} = -a_{2i}^{(1)}/a_{22}^{(1)} \qquad (i = 3,\ 4,\ 5).$$

Подставляя выражение (4.12) для x_2 во второе и третье уравнения системы (3.12), получим систему

$$\left.\begin{aligned} a_{33}^{(2)}x_3 + a_{34}^{(2)}x_4 + a_{35}^{(2)} &= 0, \\ a_{43}^{(2)}x_3 + a_{44}^{(2)}x_4 + a_{45}^{(2)} &= 0, \end{aligned}\right\} \tag{5.12}$$

коэффициенты которой находятся по формулам

$$a_{ik}^{(2)} = a_{i2}^{(1)} \cdot \alpha_{2k} + a_{ik}^{(1)} \qquad (i = 3,\ 4; \quad k = 3,\ 4,\ 5).$$

Наконец, из системы (5.12) легко тем же путем перейти к уравнению с одним неизвестным. Сначала мы

приходим к уравнению

$$x_3 = \alpha_{34} x_4 + \alpha_{35}, \tag{6.12}$$

с коэффициентами

$$\alpha_{3i} = -a_{3i}^{(2)} / a_{33}^{(2)} \qquad (i = 4,\ 5).$$

Затем, подставляя найденное значение x_3 во второе уравнение (5.12), получаем

$$a_{44}^{(3)} x_4 + a_{43}^{(3)} = 0,$$

где

$$a_{4k}^{(3)} = a_{43}^{(2)} \cdot \alpha_{3k} + a_{4k}^{(2)} \qquad (k = 4,\ 5).$$

Последнее уравнение можно переписать в виде

$$x_4 = \alpha_{45}, \tag{7.12}$$

положив

$$\alpha_{45} = -a_{43}^{(3)} / a_{44}^{(3)}.$$

Итак, мы получили четыре уравнения (2.12), (4.12), (6.12) и (7.12), которые можно объединить в систему

$$\left.\begin{aligned} x_1 &= \alpha_{12} x_2 + \alpha_{13} x_3 + \alpha_{14} x_4 + \alpha_{15}, \\ x_2 &= \alpha_{23} x_3 + \alpha_{24} x_4 + \alpha_{25}, \\ x_3 &= \alpha_{34} x_4 + \alpha_{35}, \\ x_4 &= \alpha_{45}. \end{aligned}\right\} \tag{8.12}$$

Из этих уравнений последовательно находятся значения всех четырех неизвестных x_4, x_3, x_2, x_1.

Процесс нахождения значений неизвестных по способу Гаусса распадается, таким образом, на два этапа. Первый этап состоит в приведении системы к треугольному виду (8.12). Его принято называть *прямым ходом*. Определение неизвестных по полученным формулам составляет второй этап вычислительного процесса, который называют *обратным ходом*.

Чтобы помочь лучше уяснить суть способа, прежде чем перейти к рассмотрению вычислительной схемы, разберем числовой пример. При этом берется система с целыми коэффициентами, и все вычисления производятся в таком же порядке, как и при теоретических выкладках.

Пример 1.12. Решим систему уравнений

$$\begin{aligned} 2x_1+3x_2+11x_3+5x_4-2&=0,\\ x_1+x_2+5x_3+2x_4-1&=0,\\ 2x_1+x_2+3x_3+2x_4+3&=0,\\ x_1+x_2+3x_3+4x_4+3&=0. \end{aligned}$$

Эта система имеет вид (1.12). Чтобы привести первое уравнение к виду (2.12), найдем из него x_1, разделив его на $-a_{11}=-2$. Уравнение примет вид

$$x_1=-1{,}5x_2-5{,}5x_3-2{,}5x_4+1.$$

С помощью этого уравнения надо исключить x_1 из оставшихся трех уравнений. Подставив значение x_1 во второе уравнение, получим

$$(-1{,}5\cdot 1+1)x_2+(-5{,}5\cdot 1+5)x_3+ \\ +(-2{,}5\cdot 1+2)x_4+(1\cdot 1-1)=0,$$

т. е.

$$-0{,}5x_2-0{,}5x_3-0{,}5x_4=0.$$

Подставляя x_1 в третье и четвертое уравнения, находим

$$(-1{,}5\cdot 2+1)x_2+(-5{,}5\cdot 2+3)x_3+ \\ +(-2{,}5\cdot 2+2)x_4+(1\cdot 2+3)=0,$$

$$(-1{,}5\cdot 1+1)x_2+(-5{,}5\cdot 1+3)x_3+ \\ +(-2{,}5\cdot 1+4)x_4+(1\cdot 1+3)=0,$$

т. е. мы приходим к системе трех уравнений вида (3.12)

$$\begin{aligned} -0{,}5x_2-0{,}5x_3-0{,}5x_4&=0,\\ -2x_2-8x_3-3x_4+5&=0,\\ -0{,}5x_2-2{,}5x_3+1{,}5x_4+4&=0. \end{aligned}$$

С этой системой нужно проделать ту же операцию. Разделим первое уравнение на $-a_{22}^{(1)}=0{,}5$. Тогда

$$x_2=-x_3-x_4.$$

Это и есть уравнение (4.12). Умножаем полученное значение x_2 на $a_{32}^{(1)}=-2$ и $a_{42}^{(1)}=-0{,}5$ и подставляем во

второе и третье уравнения:

$$((-1)\cdot(-2)+(-8))\,x_3+((-1)(-2)+ \\ +(-3))\,x_4+(0\cdot(-2)+5)=0,$$
$$((-1)\cdot(-0{,}5)+(-2{,}5))\,x_3+((-1)\cdot(-0{,}5)+1{,}5)\,x_4+ \\ +(0\cdot(-0{,}5)+4)=0$$

или

$$-6x_3-x_4+5=0,$$
$$-2x_3+2x_4+4=0.$$

Мы получили систему (5.12). Разделив первое уравнение на $-a_{33}^{(2)}=6$, получаем уравнение (6.12):

$$x_3=-(1/6)\,x_4+5/6.$$

Умножая его на $a_{43}^{(2)}=-2$ и подставляя во второе, приходим к уравнению

$$((-1/6)\cdot(-2)+2)\,x_4+((5/6)\cdot(-2)+4)=0,$$
$$(7/3)\,x_4+7/3=0,$$

откуда делением на $-a_{44}^{(3)}=-7/3$ приходим к уравнению (7.12)

$$x_4=-1.$$

Итак, в результате вычислений, составляющих *прямой ход*, мы пришли к системе (8.12), имеющей в данном случае вид

$$\begin{aligned} x_1&=-1{,}5x_2-5{,}5x_3-2{,}5x_4+1,\\ x_2&=-x_3-x_4,\\ x_3&=-(1/6)\,x_4+5/6,\\ x_4&=-1, \end{aligned}$$

откуда легко находим (вычисления, составляющие обратный ход, мы опускаем ввиду их простоты) $x_4=-1$, $x_3=1$, $x_2=0$, $x_1=-2$. Подстановка полученных значений в первоначальную систему показывает, что неизвестные найдены правильно.

В рассмотренном примере вычисления проводились так, чтобы проиллюстрировать теоретические выкладки. При решении систем следует располагать вычисления

в виде определенной вычислительной схемы. Прямой ход делится на несколько этапов, которые мы условно обозначим через A_1, A_2, A_3, A_4 (для системы четвертого порядка), обратный ход обозначим через B.

На первом этапе, A_1, в таблицу записываются коэффициенты при неизвестных и свободные члены данной системы уравнений. Кроме того, под коэффициентами последнего уравнения записывается строка коэффициентов уравнения (2.12), т. е. величины α_{1k}. Числа в этой строке получаются делением соответствующих элементов первой строки на ее крайний левый элемент, взятый с противоположным знаком. На этом вычисления первого этапа A_1 заканчиваются.

На втором этапе производятся вычисления коэффициенгов $a^{(1)}$ системы (3.12). При этом *к каждому элементу матрицы этапа* A_1 (кроме первой и последней строк) *прибавляется произведение крайнего левого элемента той же строки на крайний нижний элемент того же столбца* (см. табл. 1.12). Вычисления этапа A_2 завершаются получением строки α_{2k} путем деления элементов первой полученной строки на ее крайний левый элемент, взятый с противоположным знаком. Аналогично выполняются и вычисления этапов A_3, A_4.

Этап A_4 завершает прямой ход вычислений, давая в то же время и значение неизвестного x_4. После этого вычисляются значения остальных неизвестных, что составляет этап B вычислений. Значения неизвестных x_1, x_2, x_3, x_4 заносятся в первую строку таблицы, отведенную для этапа B. В столбец свободных членов в этой строке ставится единица.

Неизвестное x_4 получается умножением этой единицы на число α_{45}, стоящее над ним, и записывается в строку x и столбец x_4. После этого x_3 можно получить, составляя сумму попарных произведений уже найденных чисел строки x и соответствующей части последней строки схемы A_3

$$x_3 = x_4\alpha_{34} + 1 \cdot \alpha_{35}.$$

Так же вычисляются и x_2, и x_1. Вообще, можно воспользоваться следующим правилом: *каждое неизвестное* x_k *равно скалярному произведению уже вычисленной строки*

Таблица 1.12

	x_1	x_2	x_3	x_4	Своб. члены	Σ
A_1	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	c_1
	$\boxed{a_{21}}$	$\boxed{a_{22}}$	a_{23}	a_{24}	a_{25}	c_2
	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	c_3
	a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	c_4
	-1	$\boxed{\alpha_{12}}$	α_{13}	α_{14}	α_{15}	β_1
A_2		$a'^{(1)}_{22}$	$a'^{(1)}_{23}$	$a'^{(1)}_{24}$	$a'^{(1)}_{25}$	$c'^{(1)}_2$
		$a'^{(1)}_{32}$	$a'^{(1)}_{33}$	$a'^{(1)}_{34}$	$a'^{(1)}_{35}$	$c'^{(1)}_3$
		$a'^{(1)}_{42}$	$a'^{(1)}_{43}$	$a'^{(1)}_{44}$	$a'^{(1)}_{45}$	$c'^{(1)}_4$
		-1	α_{23}	α_{24}	α_{25}	β_2
A_3			$a'^{(2)}_{33}$	$a'^{(2)}_{34}$	$a'^{(2)}_{35}$	$c'^{(2)}_3$
			$a'^{(2)}_{43}$	$a'^{(2)}_{44}$	$a'^{(2)}_{45}$	$c'^{(2)}_4$
			-1	α_{34}	α_{35}	β_3
A_4				$a'^{(3)}_{44}$	$a'^{(3)}_{45}$	$c'^{(3)}_4$
				-1	α_{45}	β_4
B	x_1	x_2	x_3	x_4	1	
	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_3$	$\bar{x}_4$		1

неизвестных на соответствующую часть нижней строки схемы A_k.

Для способа Гаусса можно получить простые *контрольные соотношения*, которые служат хорошим контролем вычислений. Рассмотрим новую линейную систему с той же матрицей коэффициентов и со свободными членами, равными суммам всех элементов строки

$$c_i = a_{i1} + a_{i2} + a_{i3} + a_{i4} + a_{i5} \qquad (i = 1, 2, 3, 4).$$

Пусть числа x_1, x_2, x_3, x_4 образуют решение системы (1.12). Легко проверить, что *системе со свободными членами* c_i *удовлетворяют числа* $x_1 - 1$, $x_2 - 1$, $x_3 - 1$, $x_4 - 1$.

Действительно, подставим эти значения в левую часть уравнения

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + c_1 = 0.$$

Тогда получим

$$a_{11}(x_1 - 1) + a_{12}(x_2 - 1) + a_{13}(x_3 - 1) + a_{14}(x_4 - 1) + c_1 =$$
$$= (a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4) - (a_{11} + a_{12} + a_{13} + a_{14}) + c_1 =$$
$$= (a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + a_{15}) -$$
$$- (a_{11} + a_{12} + a_{13} + a_{14} + a_{15}) + c_1.$$

Здесь мы прибавили a_{15} в первой и второй скобках. Но теперь ясно, что первая скобка равна нулю, а вторая равна c_1, так что вся сумма обращается в нуль. Произведя аналогичную проверку для каждого из уравнений, убеждаемся в справедливости высказанного утверждения.

Присоединим теперь к схеме вычислений по способу Гаусса еще один столбец, элементы которого равны суммам элементов соответствующих строк. Выполняя с ним те же вычисления, что и с предыдущим, мы решаем одновременно две системы уравнений со свободными членами a_{i5} и c_i. Тем самым мы получаем хороший заключительный контроль, так как решения второй системы (вторая строка этапа B) должны быть на единицу меньше решений первой. Но, кроме этого, мы получаем возможность текущего контроля: *в процессе всех вычислений сумма элементов строки (без последнего элемента) всегда должна быть равна последнему элементу.*

Общая вычислительная схема способа Гаусса изображена в табл. 1.12.

Пример 2.12. Решим по способу Гаусса систему уравнений

$$\begin{aligned} 4{,}11x_1 - 1{,}26x_2 - 5{,}99x_3 + 1{,}29x_4 + 0{,}75 &= 0, \\ -1{,}26x_1 + 2{,}00x_2 + 4{,}00x_3 \qquad\quad - 1{,}08 &= 0, \\ 3{,}18x_1 - 1{,}97x_2 + 0{,}49x_3 - 1{,}00x_4 - 3{,}38 &= 0, \\ 1{,}29x_1 + 3{,}81x_2 - 1{,}56x_3 \qquad\quad - 0{,}87 &= 0. \end{aligned}$$

Вычисления будем выполнять с двумя запасными знаками, т. е. с пятью значащими цифрами, по схеме, описанной выше. Все вычисления приведены в табл. 2.12.

§ 13. Применение схемы Гаусса для вычисления определителя и нахождения обратной матрицы

Рассмотренный в предыдущем параграфе способ Гаусса для решения системы линейных уравнений основан, в сущности, на том, что матрица коэффициентов приводится к диагональному виду. Благодаря этому его можно использовать и для вычисления определителей.

Пусть требуется вычислить определитель

$$D = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix},$$

причем $a_{11} \neq 0$. Вынесем из первой строки за знак определителя элемент a_{11} (*ведущий элемент* первой строки), разделив на него все элементы этой строки. Вводя обозначение

$$\alpha_{1j} = a_{1j}/a_{11} \qquad (j = 2, 3, \dots, n),$$

запишем определитель в виде

$$D = a_{11} \begin{vmatrix} 1 & \alpha_{12} & \alpha_{13} & \dots & \alpha_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{vmatrix}.$$

Таблица 2.12

x_1	x_2	x_3	x_4	Своб. члены	Σ
4,11	—1,26	—5,99	1,29	0,75	—1,10
—1,26	2,00	4,00	0	—1,08	3,66
3,18	—1,97	0,49	—1,00	—3,38	—2,68
1,29	3,81	—1,56	0	—0,87	2,67
—1	0,3066	1,4574	—0,3139	—0,1825	0,2676
	1,6137	2,1637	0,3955	—0,8500	3,3229
	—0,9950	5,1245	—1,9982	—3,9604	—1,8291
	4,2055	0,3200	—0,4049	—1,1054	3,0152
	—1	—1,3408	—0,2451	0,5267	—2,0592
		6,4586	—1,7543	—4,4845	0,2198
		—5,3187	—1,4357	1,1096	—5,6448
		—1	0,2716	0,6944	—0,0340
			—2,8803	—2,5837	—5,4640
			—1	—0,8970	—1,8970
0,7996	0,1421	0,4508	—0,8970	1	
—0,2004	—0,8579	—0,5492	—1,8970		1

Вычитая первую строку, умноженную на надлежащие множители, из следующих строк, можно добиться того, чтобы в первом столбце все элементы, кроме первого, стали равными нулю. Величина определителя при этом остается неизменной. Для этого надо вычитать из второй строки первую, умноженную на a_{21}, из третьей — первую, умноженную на a_{31}, ..., из n-й — первую, умноженную на a_{n1}. Получившиеся элементы мы обозначим через $a_{ij}^{(1)}$. Они будут равны:

$$\begin{aligned}
a_{22}^{(1)} &= a_{22} - a_{21}\cdot\alpha_{12},\\
a_{23}^{(1)} &= a_{23} - a_{21}\cdot\alpha_{13},\\
&\cdots\cdots\cdots\\
a_{2n}^{(1)} &= a_{2n} - a_{21}\alpha_{1n},\\
a_{32}^{(1)} &= a_{32} - a_{31}\alpha_{12},\\
&\cdots\cdots\cdots
\end{aligned}$$

и вообще

$$a_{ij}^{(1)} = a_{ij} - a_{i1}\alpha_{1j}, \tag{1.13}$$

а определитель примет вид

$$D = a_{11}\begin{vmatrix} 1 & \alpha_{12} & \dots & \alpha_{1n} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{vmatrix}.$$

Разлагая этот определитель по элементам первого столбца, получим

$$D = a_{11}\begin{vmatrix} a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \cdots & \cdots & \cdots \\ a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{vmatrix},$$

причем определитель будет уже $(n-1)$-го порядка. С этим определителем проделаем ту же операцию, что и с исходным, вынося множитель $a_{22}^{(1)}$, который также следует предполагать отличным от нуля. Тогда

$$D = a_{11}a_{22}^{(1)}\begin{vmatrix} 1 & \alpha_{23} & \dots & \alpha_{2n} \\ a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{vmatrix},$$

где

$$\alpha_{2j}=a_{2j}^{(1)}/a_{22}^{(1)} \quad (j=3,\ 4,\ \ldots,\ n).$$

Умножая первую строку последовательно на $a_{32}^{(1)}, \ldots, a_{n2}^{(1)}$ и вычитая получающиеся строки соответственно из второй, третьей, ..., строк, получим

$$D=a_{11}a_{22}^{(1)}\begin{vmatrix} 1 & \alpha_{23} & \ldots & \alpha_{2n} \\ 0 & a_{33}^{(2)} & \ldots & a_{3n}^{(2)} \\ \cdot & \cdot & \cdot & \cdot \\ 0 & a_{n3}^{(2)} & \ldots & a_{nn}^{(2)} \end{vmatrix}=a_{11}\cdot a_{22}^{(1)}\begin{vmatrix} a_{33}^{(2)} & \ldots & a_{3n}^{(2)} \\ \cdot & \cdot & \cdot \\ a_{n3}^{(2)} & \ldots & a_{nn}^{(2)} \end{vmatrix},$$

куда входит уже определитель $(n-2)$-го порядка.

Продолжая такие преобразования, придем к формуле

$$D=a_{11}a_{22}^{(1)}a_{33}^{(2)}\ \ldots\ a_{n-2,\,n-2}^{(n-3)}\begin{vmatrix} a_{n-1,\,n-1}^{(n-2)} & a_{n-1,\,n}^{(n-2)} \\ a_{n,\,n-1}^{(n-2)} & a_{n,\,n}^{(n-2)} \end{vmatrix}.$$

Вынесем из первой строки элемент $a_{n-1,\,n-1}^{(n-2)}$. Вычитая из второй строки первую, умноженную на $a_{n,\,n-1}^{(n-2)}$, получим окончательно

$$D=a_{11}a_{22}^{(1)}\ \ldots\ a_{nn}^{(n-1)}. \tag{2.13}$$

Таким образом, *определитель равен произведению ведущих элементов схемы Гаусса.*

При этом все ведущие элементы должны быть отличны от нуля, так как в противном случае деление на них будет невозможно. Этого легко добиться путем перестановки строк или столбцов в матрице, что можно сделать на любом шаге, соблюдая только правило знаков. Если же на некотором шаге такая перестановка окажется невозможной, то в этом случае определитель будет равен нулю.

Вычислительная схема для нахождения определителя имеет тот же вид, что и для решения системы линейных уравнений, с той только разницей, что в ней отсутствует столбец свободных членов. Контрольные соотношения также остаются прежними. Поэтому мы не будем приводить схему в общем виде, ограничившись рассмотрением примера.

Пример 1.13. Вычислим по схеме Гаусса определитель

$$\begin{vmatrix} 8,2 & 1,4 & -2,3 & 0,2 \\ -1,6 & 5,4 & -7,7 & 3,1 \\ 0,7 & 1,9 & -8,5 & 4,8 \\ 5,3 & -5,9 & 2,7 & -7,9 \end{vmatrix}.$$

Вычисления приведены в табл. 1.13. Как и в предыдущем параграфе, вычисления разбиты на этапы. На каждом этапе приводится соответствующая матрица, а затем строка, получающаяся делением первой строки матрицы

Таблица 1.13

8,2	1,4	—2,3	0,2	7,5
—1,6	5,4	—7,7	3,1	—0,8
0,7	1,9	—8,5	4,8	—1,1
5,3	—5,9	2,7	—7,9	—5,8
1	0,1707	—0,2805	0,0244	0,9146
	5,6731	—8,1488	3,1390	0,6633
	1,7805	—8,3036	4,7829	—1,7402
	—6,8047	4,1866	—8,0293	—10,6474
	1	—1,4364	0,5533	0,1169
		—5,7461	3,7977	—1,9484
		—5,5877	—4,2643	—9,8520
		1	—0,6609	0,3391
			—7,9572	—7,9572

на ее ведущий элемент. Переход к матрице следующего этапа происходит по формулам (1.13). Определитель равен произведению ведущих элементов, которые в таблице обведены рамкой. Как показывают вычисления, $D=0{,}213\cdot 10^4$.

Рассмотрим теперь применение этой же схемы Гаусса к вычислению обратной матрицы. Пусть дана квадратная матрица

$$A=\begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{in} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{Vmatrix}. \tag{3.13}$$

Как известно, обратной к матрице A называется матрица A^{-1}, удовлетворяющая условию $A\cdot A^{-1}=E$, где E — единичная матрица. Если определитель матрицы A отличен от нуля (такие матрицы называют *невырожденными* или *неособенными*), то обратная матрица всегда существует. Обозначим элементы искомой матрицы A^{-1} буквой x с соответствующими индексами

$$A^{-1}=\begin{Vmatrix} x_{11} & x_{12} & \dots & x_{1k} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2k} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nk} & \dots & x_{nn} \end{Vmatrix}. \tag{4.13}$$

Равенство $A\cdot A^{-1}=E$, служащее определением обратной матрицы, позволяет написать систему линейных уравнений для неизвестных элементов обратной матрицы. Так как матрица имеет n^2 элементов, то у нас есть n^2 неизвестных, и мы получаем систему из n^2 уравнений, которую можно решить по способу Гаусса.

Для получения первого столбца матрицы $E=A\cdot A^{-1}$ нужно взять скалярное произведение первой, второй, ..., n-й строк матрицы A на первый столбец матрицы A^{-1}, причем в качестве первого элемента столбца должна получиться единица, а все остальные элементы равны

нулю. Таким образом, приходим к системе уравнений

$$\left.\begin{aligned} a_{11}x_{11}+a_{12}x_{21}+a_{13}x_{31}+\ldots+a_{1n}x_{n1}&=1,\\ a_{21}x_{11}+a_{22}x_{21}+a_{23}x_{31}+\ldots+a_{2n}x_{n1}&=0,\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots&\\ a_{n1}x_{11}+a_{n2}x_{21}+a_{n3}x_{31}+\ldots+a_{nn}x_{n1}&=0. \end{aligned}\right\} \quad (5.13)$$

Матрицей коэффициентов в системе (5.13) служит заданная матрица A, неизвестными являются элементы первого столбца искомой матрицы A^{-1}. Единицу в первом уравнении можно перенести влево, и тогда система будет иметь вид системы (1.12).

Находя элементы второго столбца матрицы $E=AA^{-1}$, мы придем к новой системе уравнений, аналогичной системе (5.13), в которой неизвестными будут являться элементы второго столбца матрицы A^{-1}. Правые части (или свободные члены после переноса единицы влево) этой системы будут иными (единица переместится во второе уравнение), но матрица коэффициентов останется прежней; ею будет служить матрица A.

Итак, система из n^2 уравнений с n^2 неизвестными для нахождения элементов обратной матрицы распадается на n групп уравнений с n неизвестными в каждой. *Все они имеют ту же матрицу коэффициентов, отличаясь лишь свободными членами.* Поскольку при решении системы по способу Гаусса основные вычисления приходится проводить над матрицей коэффициентов, то решение этих n систем можно *объединить в одной схеме, рассматривая одновременно n столбцов свободных членов.* Решив все системы, мы найдем элементы x_{ik} обратной матрицы A^{-1}. Схема расположения вычислений для случая матрицы четвертого порядка приведена в табл. 2.13.

Пример 2.13. Найдем матрицу, обратную для матрицы

$$\begin{Vmatrix} 7{,}13 & 8{,}21 & 4{,}47 & -2{,}11 \\ 3{,}25 & 15{,}4 & 2{,}91 & 5{,}43 \\ -6{,}34 & -8{,}17 & -10{,}2 & 3{,}93 \\ 4{,}52 & 6{,}73 & 1{,}37 & -9{,}89 \end{Vmatrix}.$$

Вычисления приведены в табл. 3.13 и никаких пояснений не требуют. Как видно из таблицы, обратная

матрица имеет вид

$$A^{-1} = \begin{Vmatrix} 0{,}3155 & -0{,}0815 & 0{,}1058 & -0{,}0702 \\ -0{,}0743 & 0{,}0779 & -0{,}0026 & 0{,}0577 \\ -0{,}1062 & -0{,}0060 & -0{,}1519 & -0{,}0410 \\ 0{,}0789 & 0{,}0150 & 0{,}0255 & -0{,}0996 \end{Vmatrix}.$$

Таблица 2.13

a_{11}	a_{12}	a_{13}	a_{14}	-1	0	0	0
a_{21}	a_{22}	a_{23}	a_{24}	0	-1	0	0
a_{31}	a_{32}	a_{33}	a_{34}	0	0	-1	0
a_{41}	a_{42}	a_{43}	a_{44}	0	0	0	-1
-1	α_{12}	α_{13}	α_{14}	α_{15}	0	0	0
	$a_{22}^{(1)}$	$a_{23}^{(1)}$	$a_{24}^{(1)}$	$a_{25,\,\mathrm{I}}^{(1)}$	$a_{25,\,\mathrm{II}}^{(1)}$	0	0
	$a_{32}^{(1)}$	$a_{33}^{(1)}$	$a_{34}^{(1)}$	$a_{35,\,\mathrm{I}}^{(1)}$	0	$a_{35,\,\mathrm{III}}^{(1)}$	0
	$a_{42}^{(1)}$	$a_{43}^{(1)}$	$a_{44}^{(1)}$	$a_{45,\,\mathrm{I}}^{(1)}$	0	0	$a_{45,\,\mathrm{IV}}^{(1)}$
	-1	α_{23}	α_{24}	$\alpha_{25,\,\mathrm{I}}$	$\alpha_{25,\,\mathrm{II}}$	0	0
		$a_{33}^{(2)}$	$a_{34}^{(2)}$	$a_{35,\,\mathrm{I}}^{(2)}$	$a_{35,\,\mathrm{II}}^{(2)}$	$a_{35,\,\mathrm{III}}^{(2)}$	0
		$a_{43}^{(2)}$	$a_{44}^{(2)}$	$a_{45,\,\mathrm{I}}^{(2)}$	$a_{45,\,\mathrm{II}}^{(2)}$	0	$a_{45,\,\mathrm{IV}}^{(2)}$
		-1	α_{34}	$\alpha_{35,\,\mathrm{I}}$	$\alpha_{35,\,\mathrm{II}}$	$\alpha_{35,\,\mathrm{III}}$	0
			$a_{44}^{(3)}$	$a_{45,\,\mathrm{I}}^{(3)}$	$a_{45,\,\mathrm{II}}^{(3)}$	$a_{45,\,\mathrm{III}}^{(3)}$	$a_{45,\,\mathrm{IV}}^{(3)}$
			-1	$\alpha_{45,\,\mathrm{I}}$	$\alpha_{45,\,\mathrm{II}}$	$\alpha_{45,\,\mathrm{III}}$	$\alpha_{45,\,\mathrm{IV}}$
x_{11}	x_{21}	x_{31}	x_{41}	1			
x_{12}	x_{22}	x_{32}	x_{42}		1		
x_{13}	x_{23}	x_{33}	x_{43}			1	
x_{14}	x_{24}	x_{34}	x_{44}				1

Таблица 3.13

7,13	8,21	4,47	—2,11	—1	0	0	0	16,70
3,25	15,4	2,91	5,43	0	—1	0	0	25,99
—6,34	—8,17	—10,2	3,93	0	0	—1	0	—21,78
4,52	6,73	1,37	—9,89	0	0	0	—1	1,73
—1	—1,151	—0,627	0,296	+0,140	0	0	0	—2,342
	11,659	0,872	6,392	0,455	—1	0	0	18,378
	—0,873	—6,225	2,054	—0,888	0	—1	0	—6,932
	1,527	—1,464	—8,552	0,633	0	0	—1	—8,856
	—1	—0,0748	—0,5482	—0,0390	0,0857	0	0	—1,5763
		—6,1597	2,5326	—0,8540	—0,0748	—1	0	—5,5559
		—1,5782	—9,3891	0,5734	0,1309	0	—1	—11,2630
		—1	0,4112	—0,1386	—0,0122	—0,1624	0	—0,9020
			—10,0381	0,7921	0,1502	0,2563	—1	—9,8395
			—1	0,0789	0,0150	0,0255	—0,0996	—0,9802
0,3155	—0,0743	—0,1062	0,0789	1				
—0,0815	0,0779	—0,0060	0,0150		1			
0,1058	—0,0026	—0,1519	0,0255			1		
—0,0702	0,0577	—0,0410	—0,0996				1	

§ 14. Итерации для линейных систем

В противоположность способу Гаусса, являющемуся, как уже было сказано, точным, способ итераций дает возможность получить последовательность приближенных значений, сходящуюся к точному решению системы. С вычислительной стороны применение способа итераций часто бывает более выгодным, так как здесь требуется гораздо меньшая точность промежуточных вычислений. Кроме того, способ итераций устойчив относительно просчетов в промежуточных вычислениях: отдельные просчеты на предыдущих шагах итерации не влекут за собой необходимость браковки последующих расчетов.

Рассмотрим систему n линейных уравнений с n неизвестными:

$$\left.\begin{array}{l} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n+a_{1,n+1}=0, \\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n+a_{2,n+1}=0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n+a_{n,n+1}=0. \end{array}\right\} \quad (1.14)$$

Разрешим первое уравнение относительно x_1, второе — относительно x_2, ..., последнее — относительно x_n. Тогда систему (1.14) можно будет переписать в виде

$$\left.\begin{array}{l} x_1=\alpha_{12}x_2+\alpha_{13}x_3+\ldots+\alpha_{1n}x_n+\alpha_{1,n+1}, \\ x_2=\alpha_{21}x_1+\alpha_{23}x_3+\ldots+\alpha_{2n}x_n+\alpha_{2,n+1}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n=\alpha_{n1}x_1+\alpha_{n2}x_2+\alpha_{n3}x_3+\ldots+\alpha_{n,n-1}x_{n-1}+\alpha_{n,n+1}, \end{array}\right\} \quad (2.14)$$

где

$$\alpha_{ik}=-\frac{a_{ik}}{a_{ii}} \qquad (i=1,\ 2,\ \ldots,\ n;\ k=1,\ 2,\ \ldots,\ n,\ n+1). \quad (3.14)$$

Правые части системы (2.14) являются функциями переменных $x_1, x_2, \ldots, x_n$. Обозначим правую часть первого из уравнений через $L_1(x_1, x_2, \ldots, x_n)$. При этом мы не обращаем внимания на то обстоятельство, что фактически правая часть первого уравнения не зависит от x_1. Дальнейшие рассуждения не изменятся, если правая часть будет содержать еще и слагаемое вида $\alpha_{11}x_1$. Аналогично правую часть второго уравнения обозначим через $L_2(x_1, x_2, \ldots, x_n)$ и т. д. Систему (2.14) можно тогда переписать в виде

$$\left.\begin{array}{l} x_1=L_1(x_1,\ x_2,\ \ldots,\ x_n), \\ x_2=L_2(x_1,\ x_2,\ \ldots,\ x_n), \\ \ldots\ldots\ldots\ldots\ldots \\ x_n=L_n(x_1,\ x_2,\ \ldots,\ x_n). \end{array}\right\} \quad (4.14)$$

Зададим какие-либо начальные значения неизвестных $x_1^{(0)}, x_2^{(0)}, \ldots$ $\ldots, x_n^{(0)}$. Подставляя эти значения в правые части системы (4.14),

получим первые приближения:

$$x_1^{(1)} = L_1(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}),$$
$$x_2^{(1)} = L_2(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}),$$
$$\ldots\ldots\ldots\quad\ldots\ldots$$
$$x_n^{(1)} = L_n(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}).$$

Полученные первые приближения могут быть таким же образом использованы для получения вторых, третьих и т. д., так что для любого ν можно получить ν-е приближение $x_1^{(\nu)}, x_2^{(\nu)}, \ldots, x_n^{(\nu)}$.

Естественно возникает вопрос об условиях, выполнение которых обеспечит сходимость получающихся приближений к истинному решению системы $x_1, x_2, \ldots, x_n$. Установим эти условия.

Обозначим через $\varepsilon_k^{(\nu)}$ ошибку ν-го приближения для неизвестного x_k

$$\varepsilon_k^{(\nu)} = x_k^{(\nu)} - x_k,$$

где x_k означает точное значение неизвестного в решении системы. Так как x_k удовлетворяет системе (2.14), то из соответствующего уравнения этой системы имеем

$$x_k = \alpha_{k1}x_1 + \alpha_{k2}x_2 + \ldots + \alpha_{kn}x_n + \alpha_{k,n+1}.$$

С другой стороны, по определению $(\nu+1)$-го приближения,

$$x_k^{(\nu+1)} = \alpha_{k1}x_1^{(\nu)} + \alpha_{k2}x_2^{(\nu)} + \ldots + \alpha_{kn}x_n^{(\nu)} + \alpha_{k,n+1}.$$

Вычитая первое уравнение из второго, находим

$$\varepsilon_k^{(\nu+1)} = \alpha_{k1}\varepsilon_1^{(\nu)} + \alpha_{k2}\varepsilon_2^{(\nu)} + \ldots + \alpha_{kn}\varepsilon_n^{(\nu)}. \tag{5.14}$$

Пусть $\varepsilon^{(\nu)}$ означает наибольшую из величин $|\varepsilon_1^{(\nu)}|, |\varepsilon_2^{(\nu)}|, \ldots, |\varepsilon_n^{(\nu)}|$, т. е.

$$\varepsilon^{(\nu)} = \max\left(|\varepsilon_1^{(\nu)}|, |\varepsilon_2^{(\nu)}|, \ldots, |\varepsilon_n^{(\nu)}|\right). \tag{6.14}$$

Тогда из равенства (5.14) вытекает, что

$$|\varepsilon_k^{(\nu+1)}| \leqslant C_k\varepsilon^{(\nu)}, \tag{7.14}$$

где

$$C_k = |\alpha_{k1}| + |\alpha_{k2}| + \ldots + |\alpha_{kn}|. \tag{8.14}$$

Неравенство (7.14) справедливо для всех $k=1, 2, \ldots, n$. Поэтому наибольшая из величин $|\varepsilon_1^{(\nu+1)}|, \ldots, |\varepsilon_n^{(\nu+1)}|$, которую мы обозначим через $\varepsilon^{(\nu+1)}$, будет удовлетворять неравенству

$$\varepsilon^{(\nu+1)} \leqslant C\varepsilon^{(\nu)}, \tag{9.14}$$

где

$$C=\max(C_1,\ C_2,\ \ldots,\ C_n).$$

Неравенство (9.14) справедливо при любых $\nu=0,\ 1,\ 2,\ \ldots$. Поэтому можно написать

$$\begin{aligned}
&\varepsilon^{(1)}\leqslant C\varepsilon^{(0)},\\
&\varepsilon^{(2)}\leqslant C\varepsilon^{(1)}\leqslant C^2\varepsilon^{(0)},\\
&\varepsilon^{(3)}\leqslant C\varepsilon^{(2)}\leqslant C^3\varepsilon^{(0)},\\
&\ldots\ldots\ldots\ldots\\
&\varepsilon^{(\nu)}\leqslant C\varepsilon^{(\nu-1)}\leqslant C^\nu\varepsilon^{(0)},\\
&\ldots\ldots\ldots\ldots
\end{aligned}$$

Если $C<1$, то, как известно, $\lim\limits_{\nu\to\infty} C^\nu=0$, откуда следует, что и

$$\lim_{\nu\to\infty}\varepsilon^{(\nu)}=0.$$

Но $\varepsilon^{(\nu)}$ есть максимальная из величин $\left|\varepsilon_k^{(\nu)}\right|$ для $k=1,\ 2,\ \ldots,\ n$. Отсюда вытекает, что для любого k справедливо

$$\lim_{\nu\to\infty}\varepsilon_k^{(\nu)}=0,$$

или

$$\lim_{\nu\to\infty}x_k^{(\nu)}=x_k. \tag{10.14}$$

Таким образом, выполнения условия $C<1$ достаточно для того, чтобы наш итерационный процесс сходился к точному решению системы. Чтобы выяснить, что означает условие $C<1$ для коэффициентов уравнения, заметим, что, по определению, C есть верхняя грань сумм коэффициентов системы (2.14) по строкам, которые в свою очередь являются отношениями элементов матрицы коэффициентов системы (1.14) к диагональным элементам. Таким образом, *все коэффициенты системы должны быть малы по сравнению с диагональными.*

Это условие можно сформулировать и более точно: *для того чтобы итерационный процесс сходился, достаточно, чтобы в любой строке сумма отношений коэффициентов системы к диагональным коэффициентам, взятым из той же строки, была строго меньше единицы.*

То же рассуждение, которое использовалось для установления достаточных условий сходимости итерационного процесса, можно применить для получения оценки погрешности очередного приближения. Обозначим разность между двумя последовательными значениями k-го неизвестного через

$$\delta_k^{(\nu)}=x_k^{(\nu+1)}-x_k^{(\nu)}.$$

Прибавляя и вычитая в правой части точное значение неизвестного x_k, найдем

$$\delta_k^{(\nu)}=\left(x_k^{(\nu+1)}-x_k\right)-\left(x_k^{(\nu)}-x_k\right)=\varepsilon_k^{(\nu+1)}-\varepsilon_k^{(\nu)}.$$

Отсюда имеем

$$\left|\delta_k^{(\nu)}\right| = \left|\varepsilon_k^{(\nu+1)} - \varepsilon_k^{(\nu)}\right| = \left|\varepsilon_k^{(\nu)} - \varepsilon_k^{(\nu+1)}\right|,$$

так что

$$\left|\delta_k^{(\nu)}\right| \geqslant \left|\varepsilon_k^{(\nu)}\right| - \left|\varepsilon_k^{(\nu+1)}\right|.$$

Воспользовавшись соотношениями (6.14) и (9.14), заменим это неравенство таким:

$$\left|\delta_k^{(\nu)}\right| \geqslant \left|\varepsilon_k^{(\nu)}\right| - C\varepsilon^{(\nu)}.$$

Полагая здесь $k=1, 2, \ldots, n$ и обозначая через $\delta^{(\nu)}$ наибольшую из величин $\left|\delta_k^{(\nu)}\right|$,

$$\delta^{(\nu)} = \max\left(\left|\delta_1^{(\nu)}\right|, \left|\delta_2^{(\nu)}\right|, \ldots, \left|\delta_n^{(\nu)}\right|\right),$$

найдем

$$\delta^{(\nu)} \geqslant \left|\varepsilon_k^{(\nu)}\right| - C\varepsilon^{(\nu)}.$$

Так как это неравенство справедливо для всех k, то из него выводим

$$\delta^{(\nu)} \geqslant \varepsilon^{(\nu)} - C\varepsilon^{(\nu)},$$

откуда, наконец,

$$\varepsilon^{(\nu)} \leqslant \delta^{(\nu)}/(1-C). \tag{11.14}$$

Неравенство (11.14) и дает оценку погрешности ν-й итерации через разности двух последовательных итераций.

Таким образом, оценка погрешностей $\varepsilon_k^{(\nu)}$ значений неизвестных на ν-м шаге итерации может быть получена так: сначала следует найти суммы абсолютных величин коэффициентов строк матрицы системы (2.14). Наибольшее из этих чисел принимаем за величину C. После этого находим величины $\delta_k^{(\nu)}$, являющиеся разностями значений неизвестных на ν-м и $(\nu+1)$-м шагах:

$$\delta_k^{(\nu)} = x_k^{(\nu+1)} - x_k^{(\nu)} \quad (k=1, 2, \ldots, n),$$

и

$$\delta^{(\nu)} = \max\left(\left|\delta_1^{(\nu)}\right|, \left|\delta_2^{(\nu)}\right|, \ldots, \left|\delta_n^{(\nu)}\right|\right).$$

После этого можно получить оценку погрешностей неизвестных, воспользовавшись неравенством (11.14).

Перейдем теперь к рассмотрению вычислительной схемы для решения линейной системы методом итераций. Для простоты и определенности ограничимся системой трех уравнений с тремя неизвестными, записав ее сразу в виде

$$\begin{aligned} x_1 &= \alpha_{12}x_2 + \alpha_{13}x_3 + \alpha_{14}, \\ x_2 &= \alpha_{21}x_1 + \alpha_{23}x_3 + \alpha_{24}, \\ x_3 &= \alpha_{31}x_1 + \alpha_{32}x_2 + \alpha_{34}. \end{aligned} \tag{12.14}$$

За нулевые приближения значений неизвестных можно принять свободные члены соответствующих уравнений

$$x_1^{(0)}=\alpha_{14},\quad x_2^{(0)}=\alpha_{24},\quad x_3^{(0)}=\alpha_{34}. \tag{13.14}$$

В процессе вычислений удобно находить не непосредственно значения неизвестных, а поправки к предыдущим приближениям. Начнем с вычисления первых поправок. Пусть

$$\delta_1^{(0)}=x_1^{(1)}-x_1^{(0)},\quad \delta_2^{(0)}=x_2^{(1)}-x_2^{(0)},\quad \delta_3^{(0)}=x_3^{(1)}-x_3^{(0)}. \tag{14.14}$$

При этом значения $x_1^{(0)}$, $x_2^{(0)}$, $x_3^{(0)}$ можно считать заданными равенствами (13.14), а первые приближения $x_1^{(1)}$, $x_2^{(1)}$, $x_3^{(1)}$ находятся по формулам

$$\begin{aligned}
x_1^{(1)}&=\alpha_{12}x_2^{(0)}+\alpha_{13}x_3^{(0)}+\alpha_{14},\\
x_2^{(1)}&=\alpha_{21}x_1^{(0)}+\alpha_{23}x_3^{(0)}+\alpha_{24},\\
x_3^{(1)}&=\alpha_{31}x_1^{(0)}+\alpha_{32}x_2^{(0)}+\alpha_{34}.
\end{aligned}$$

Вычитая из этих равенств равенства (13.14), находим

$$\left.\begin{aligned}
\delta_1^{(0)}&=\alpha_{12}x_2^{(0)}+\alpha_{13}x_3^{(0)},\\
\delta_2^{(0)}&=\alpha_{21}x_1^{(0)}+\alpha_{23}x_3^{(0)},\\
\delta_3^{(0)}&=\alpha_{31}x_1^{(0)}+\alpha_{32}x_2^{(0)}.
\end{aligned}\right\} \tag{15.14}$$

Равенства (14.14) можно переписать в виде

$$x_1^{(1)}=x_1^{(0)}+\delta_1^{(0)},\quad x_2^{(1)}=x_2^{(0)}+\delta_2^{(0)},\quad x_3^{(1)}=x_3^{(0)}+\delta_3^{(0)},$$

где $\delta_i^{(0)}$ ($i=1, 2, 3$) уже определены равенствами (15.14). Точно таким же путем можно получить и поправки для следующих шагов. Действительно, по определению,

$$\delta_1^{(\nu)}=x_1^{(\nu+1)}-x_1^{(\nu)},$$

откуда

$$x_1^{(\nu+1)}=x_1^{(\nu)}+\delta_1^{(\nu)}.$$

Вместе с тем, по построению,

$$\begin{aligned}
x_1^{(\nu+1)}&=\alpha_{12}x_2^{(\nu)}+\alpha_{13}x_3^{(\nu)}+\alpha_{14},\\
x_1^{(\nu)}&=\alpha_{12}x_2^{(\nu-1)}+\alpha_{13}x_3^{(\nu-1)}+\alpha_{14}.
\end{aligned}$$

Вычитая из первого уравнения второе, находим

$$x_1^{(\nu+1)}-x_1^{(\nu)}=\alpha_{12}\left(x_2^{(\nu)}-x_2^{(\nu-1)}\right)+\alpha_{13}\left(x_3^{(\nu)}-x_3^{(\nu-1)}\right),$$

т. е.

$$\delta_1^{(\nu)}=\alpha_{12}\delta_2^{(\nu-1)}+\alpha_{13}\delta_3^{(\nu-1)}, \tag{16.14}$$

а аналогично для других неизвестных

$$\left.\begin{aligned}
\delta_2^{(\nu)}&=\alpha_{21}\delta_1^{(\nu)}+\alpha_{23}\delta_3^{(\nu-1)},\\
\delta_3^{(\nu)}&=\alpha_{31}\delta_1^{(\nu-1)}+\alpha_{32}\delta_2^{(\nu-1)}.
\end{aligned}\right\} \tag{17.14}$$

Итак, мы установили формулы, позволяющие вычислять последующие поправки через предыдущие. Для определения искомых значений неизвестных можно воспользоваться тождеством

$$x_1 = x_1 + x_1^{(0)} + (x_1^{(1)} - x_1^{(0)}) + (x_1^{(2)} - x_1^{(1)}) + \ldots + (x_1^{(\nu)} - x^{(\nu-1)}) - x_1^{(\nu)},$$

которое можно записать в виде

$$x_1 = x_1^{(0)} + \delta_1^{(0)} + \delta_1^{(1)} + \ldots + \delta_1^{(\nu-1)} + (x_1 - x_1^{(\nu)}).$$

Как было доказано выше, при $C < 1$ последовательность $\{x_1^{(\nu)}\}$ сходится к x_1, поэтому разность $x_1 - x_1^{(\nu)}$ стремится к нулю. Следовательно, можно написать

$$x_1 = x_1^{(0)} + \delta_1^{(0)} + \delta_1^{(1)} + \delta_1^{(2)} + \ldots + \delta_1^{(\nu)} + \ldots \tag{18.14}$$

Для двух других неизвестных точно так же получаем

$$\left.\begin{aligned} x_2 &= x_2^{(0)} + \delta_2^{(0)} + \delta_2^{(1)} + \delta_2^{(2)} + \ldots + \delta_2^{(\nu)} + \ldots, \\ x_3 &= x_3^{(0)} + \delta_3^{(0)} + \delta_3^{(1)} + \delta_3^{(2)} + \ldots + \delta_3^{(\nu)} + \ldots \end{aligned}\right\} \tag{19.14}$$

Формулы (13.14)—(19.14) являются исходными для построения вычислительной схемы, приведенной в табл. 1.14 для случая системы третьего порядка. Часть A схемы содержит коэффициенты и свободные члены исходной системы (1.14), часть B — коэффициенты системы (2.14), приготовленной к итерации. Для получения строк части B таблицы строки части A делятся каждая на свой диагональный элемент, взятый с противоположным знаком. Поскольку диагональные элементы не участвуют в дальнейших вычислениях, то их в таблице и не пишут, а на их место ставят прочерк. Часть X таблицы содержит последовательные значения поправок и неизвестных.

Строка нулевого приближения $x^{(0)}$ образуется переписыванием столбца свободных членов приведенной системы. Элементы строки $\delta^{(0)}$ получаются как скалярные произведения строки $x^{(0)}$ и строк коэффициентов приведенной системы. Запись $x^{(0)} \cdot (\mathrm{I})$ означает

$$x^{(0)} \cdot (\mathrm{I}) = x_1^{(0)} \cdot 0 + x_2^{(0)} \cdot \alpha_{12} + x_3^{(0)} \cdot \alpha_{13}.$$

Аналогично,

$$x^{(0)} \cdot (\mathrm{II}) = x_1^{(0)} \cdot \alpha_{21} + x_2^{(0)} \cdot 0 + x_3^{(0)} \cdot \alpha_{23},$$
$$x^{(0)} \cdot (\mathrm{III}) = x_1^{(0)} \cdot \alpha_{31} + x_2^{(0)} \cdot \alpha_{32} + x_3^{(0)} \cdot 0.$$

Строка $x^{(1)}$ образуется сложением двух предыдущих строк, а строка $\delta^{(1)}$ из строк (I), (II), (III) и $x^{(1)}$ — тем же способом, что и строка $\delta^{(0)}$ из строк (I), (II), (III) и $x^{(0)}$.

Вычисления продолжают обычно до тех пор, пока поправки не станут меньше требуемых погрешностей. Последнюю строку x принимают после этого за окончательные значения неизвестных. Иногда используют также и оценки погрешностей с помощью формулы (11.14).

Пример 1.14. Решим способом итераций систему уравнений

$$\begin{aligned} 4x_1 + 0{,}24x_2 - 0{,}08x_3 - 8 &= 0, \\ 0{,}09x_1 + 3x_2 - 0{,}15x_3 - 9 &= 0, \\ 0{,}04x_1 - 0{,}08x_2 + 4x_3 - 20 &= 0. \end{aligned}$$

Таблица 1.14

A		a_{11}	a_{12}	a_{13}	a_{14}
		a_{21}	a_{22}	a_{23}	a_{24}
		a_{31}	a_{32}	a_{33}	a_{34}
B	I	—	α_{12}	α_{13}	α_{14}
	II	α_{21}	—	α_{23}	α_{24}
	III	α_{31}	α_{32}	—	α_{34}
$x^{(0)}$		$x_1^{(0)}$	$x_2^{(0)}$	$x_3^{(0)}$	
$\delta^{(0)}$		$\delta_1^{(0)}=x^{(0)}\cdot(\mathrm{I})$	$\delta_2^{(0)}=x^{(0)}\cdot(\mathrm{II})$	$\delta_3^{(0)}=x^{(0)}\cdot(\mathrm{III})$	
$x^{(1)}$		$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	
$\delta^{(1)}$		$\delta_1^{(1)}=\delta^{(0)}\cdot(\mathrm{I})$	$\delta_2^{(1)}=\delta^{(0)}\cdot(\mathrm{II})$	$\delta_3^{(1)}=\delta^{(0)}\cdot(\mathrm{III})$	X
$x^{(2)}$		$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	
$\delta^{(2)}$		$\delta_1^{(1)}\cdot(\mathrm{I})$	$\delta_2^{(1)}\cdot(\mathrm{II})$	$\delta_3^{(1)}\cdot(\mathrm{III})$	
...		...	...	...	

Вычисления приведены в табл. 2.14. Как видно из таблицы, поправки после трех шагов уже достаточно малы и вычисления на этом можно закончить.

Оценим погрешность полученных решений. Суммы модулей коэффициентов рассматриваемой системы по строкам равны 0,04, 0,08 и 0,03. Поэтому можно принять $C=0{,}08$. Далее, для Δ_ν находим при $\nu=3$ значение $\Delta_\nu=0{,}00055$. Поэтому формула (11.14) дает

$$\varepsilon^{(3)}=0{,}0006.$$

Заметим, что в способе итераций не рассматриваются контрольные соотношения. Дело в том, что этот способ, как принято говорить, устойчив относительно промежуточных просчетов, т. е. просчет в промежуточных вычислениях не отражается на точности получающегося в конечном счете решения, поскольку такой просчет означает лишь изменение исходных данных для итераций.

Таблица 2.14

	4	0,24	—0,08	—8
	0,09	3	—0,15	—9
	0,04	—0,08	4	—20
I	—	—0,06	0,02	2
II	—0,03	—	0,05	3
III	—0,01	0,02	—	5
$x^{(0)}$	2	3	5	
$\delta^{(0)}$	—0,08	0,19	0,04	
$x^{(1)}$	1,92	3,19	5,04	
$\delta^{(1)}$	—0,0106	0,0044	0,0046	
$x^{(2)}$	1,9094	3,1944	5,0446	
$\delta^{(2)}$	—0,000172	0,000548	0,000194	
$x^{(3)}$	1,909228	3,194948	5,044794	
$\delta^{(3)}$	.....	.....	.....	
$x^{(4)}$	.....	.....	.....	

§ 15. Способ Зейделя

В предыдущем параграфе при нахождении очередных приближений для значений неизвестных мы пользовались формулами

$$\left.\begin{aligned} x_1^{(\nu)} &= L_1\left(x_1^{(\nu-1)},\ x_2^{(\nu-1)},\ \ldots,\ x_n^{(\nu-1)}\right),\\ x_2^{(\nu)} &= L_2\left(x_1^{(\nu-1)},\ x_2^{(\nu-1)},\ \ldots,\ x_n^{(\nu-1)}\right),\\ &\ldots\ldots\ldots\ldots\ldots\ldots\\ x_n^{(\nu)} &= L_n\left(x_1^{(\nu-1)},\ x_2^{(\nu-1)},\ \ldots,\ x_n^{(\nu-1)}\right), \end{aligned}\right\} \qquad (1.15)$$

т. е. следующее приближение целиком определялось по значениям, полученным на предыдущем шаге. Между тем, при построении первого приближения можно, например, вычисляя значение $x_2^{(1)}$, воспользоваться для x_1 не нулевым приближением $x_1^{(0)}$, а уже полученным первым $x_1^{(1)}$. При вычислении $x_3^{(1)}$ можно пользоваться уже полученными новыми значениями $x_1^{(1)}$ и $x_2^{(1)}$ и т. д.

Иначе говоря, формулы (1.15), используемые для нахождения следующих значений неизвестных при итерации, предлагается изменить следующим образом:

$$\left.\begin{aligned}
x_1^{(\nu)} &= L_1\left(x_1^{(\nu-1)},\ x_2^{(\nu-1)},\ \ldots,\ x_n^{(\nu-1)}\right),\\
x_2^{(\nu)} &= L_2\left(x_1^{(\nu)},\ x_2^{(\nu-1)},\ \ldots,\ x_n^{(\nu-1)}\right),\\
x_3^{(\nu)} &= L_3\left(x_1^{(\nu)},\ x_2^{(\nu)},\ \ldots,\ x_n^{(\nu-1)}\right),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n^{(\nu)} &= L_n\left(x_1^{(\nu)},\ x_2^{(\nu)},\ \ldots,\ x_{n-1}^{(\nu)},\ x_n^{(\nu-1)}\right).
\end{aligned}\right\} \qquad (2.15)$$

Это изменение и составляет идею *итераций по способу Зейделя.*

Следует ожидать, что итерации по способу Зейделя дадут при том же числе шагов более точные результаты, чем обычные итерации, а такая же точность будет достигнута за меньшее число шагов, поскольку очередные значения неизвестных определяются здесь более точно. Эти ожидания, как правило, оправдываются, т. е. итерации по способу Зейделя, вообще говоря, действительно дают более точные результаты. Нужно, однако, иметь в виду, что условия сходимости обычного итерационного процесса и процесса Зейделя р а з л и ч н ы и один из этих процессов для данной системы может оказаться расходящимся тогда, когда другой сходится.

Мы не будем останавливаться на условиях сходимости процесса Зейделя и перейдем сразу к рассмотрению вычислительной схемы. Как и в предыдущем параграфе, рассмотрим эту схему на примере трех линейных уравнений с тремя неизвестными. Формулы для нахождения следующих приближений через предыдущие будут для

процесса Зейделя иметь вид

$$\left.\begin{aligned} x_1^{(\nu)} &= \alpha_{12}x_2^{(\nu-1)} + \alpha_{13}x_3^{(\nu-1)} + \alpha_{14},\\ x_2^{(\nu)} &= \alpha_{21}x_1^{(\nu)} + \alpha_{23}x_3^{(\nu-1)} + \alpha_{24},\\ x_3^{(\nu)} &= \alpha_{31}x_1^{(\nu)} + \alpha_{32}x_2^{(\nu)} + \alpha_{34}. \end{aligned}\right\} \qquad (3.15)$$

Формулы для вычисления поправок легко получить, исходя непосредственно из определения и используя (3.15). Действительно,

$$\begin{aligned} \delta_1^{(\nu)} = x_1^{(\nu+1)} - x_1^{(\nu)} &= (\alpha_{12}x_2^{(\nu)} + \alpha_{13}x_3^{(\nu)} + \alpha_{14}) -\\ &- (\alpha_{12}x_2^{(\nu-1)} + \alpha_{13}x_3^{(\nu-1)} + \alpha_{14}) =\\ &= \alpha_{12}(x_2^{(\nu)} - x_2^{(\nu-1)}) + \alpha_{13}(x_3^{(\nu)} - x_3^{(\nu-1)}), \end{aligned}$$

так что

$$\delta_1^{(\nu)} = \alpha_{12}\delta_2^{(\nu-1)} + \alpha_{13}\delta_3^{(\nu-1)}. \qquad (4.15)$$

Для $\delta_2^{(\nu)}$ формула выглядит несколько иначе. Так как

$$\begin{aligned} \delta_2^{(\nu)} = x_2^{(\nu+1)} - x_2^{(\nu)} &= (\alpha_{21}x_1^{(\nu+1)} + \alpha_{23}x_3^{(\nu)} + \alpha_{24}) -\\ &- (\alpha_{21}x_1^{(\nu)} + \alpha_{23}x_3^{(\nu-1)} + \alpha_{24}), \end{aligned}$$

то

$$\delta_2^{(\nu)} = \alpha_{21}\delta_1^{(\nu)} + \alpha_{23}\delta_3^{(\nu-1)}. \qquad (5.15)$$

Аналогично,

$$\delta_3^{(\nu)} = \alpha_{31}\delta_1^{(\nu)} + \alpha_{32}\delta_2^{(\nu)}. \qquad (6.15)$$

Формулы (4.15)—(6.15) справедливы для всех $\nu = 1, 2, \ldots$, но не для $\nu = 0$, как и в обычном итерационном процессе. Если, как мы это делали в предыдущем параграфе, принять за нулевые приближения неизвестных значения соответствующих свободных членов

$$x_1^{(0)} = \alpha_{14},\quad x_2^{(0)} = \alpha_{24},\quad x_3^{(0)} = \alpha_{34},$$

то для первых поправок получим выражения, аналогичные формулам (4.15):

$$\left.\begin{aligned} \delta_1^{(0)} &= \alpha_{12}x_2^{(0)} + \alpha_{13}x_3^{(0)},\\ \delta_2^{(0)} &= \alpha_{21}x_1^{(1)} + \alpha_{23}x_3^{(0)},\\ \delta_3^{(0)} &= \alpha_{31}x_1^{(1)} + \alpha_{32}x_2^{(1)}, \end{aligned}\right\} \qquad (7.15)$$

где $x_1^{(1)} = x_1^{(0)} + \delta_1^{(0)}$ и $x_2^{(1)} = x_2^{(0)} + \delta_2^{(0)}$.

Опишем теперь вычислительную схему. Исходные данные и величины α_{ij} располагаются так же, как и в обычном итерационном процессе. Поэтому первая часть вычислений может быть расположена так, как это показано в табл. 1.14, и мы ее здесь не приводим. Таблица для вычисления поправок имеет здесь вид, несколько отличный от табл. 1.14.

Таблица 1.15 показывает расположение вычислений поправок в способе Зейделя. Таблица может содержать любое число групп по четыре (в нашем примере) строки в каждой. Каждая такая группа предназначена для вычисления очередной поправки. Мы приводим две из них, вычисляя поправки $\delta^{(0)}$ и $\delta^{(1)}$. Следующие поправки вычисляются по той же схеме, что и $\delta^{(1)}$.

Изменением по сравнению со схемой обычного итерационного процесса является то, что в вычислениях очередных поправок используются, где возможно, уже полученные поправки того же этапа вычислений. Соответствующие места в табл. 1.15 подчеркнуты.

Таблица 1.15

	δ_1	δ_2	δ_3
(5)	—	$\underline{\alpha_{21}(\alpha_{14}+\delta_1^{(0)})}$	$\underline{\alpha_{31}(\alpha_{14}+\delta_1^{(0)})}$
(6)	$\alpha_{12}\alpha_{24}$	—	$\underline{\alpha_{32}(\alpha_{24}+\delta_2^{(0)})}$
(7)	$\alpha_{13}\alpha_{34}$	$\alpha_{23}\alpha_{34}$	—
(8) = (5) + (6) + (7)	$\delta_1^{(0)}$	$\delta_2^{(0)}$	$\delta_3^{(0)}$
(9)	—	$\underline{\alpha_{21}\delta_1^{(1)}}$	$\underline{\alpha_{31}\delta_1^{(1)}}$
(10)	$\alpha_{12}\delta_2^{(0)}$	—	$\underline{\alpha_{32}\delta_2^{(1)}}$
(11)	$\alpha_{13}\delta_3^{(0)}$	$\alpha_{23}\delta_3^{(0)}$	—
(12) = (9) + (10) + (11)	$\delta_1^{(1)}$	$\delta_2^{(1)}$	$\delta_3^{(1)}$
(13)	...	...	...

Как и при обычном итерационном процессе, вычисления продолжают до получения достаточно малых поправок. Окончательные значения неизвестных получаются по формулам

$$x_1 = \alpha_{14} + \delta_1^{(0)} + \delta_1^{(1)} + \ldots,$$
$$x_2 = \alpha_{24} + \delta_2^{(0)} + \delta_2^{(1)} + \ldots,$$
$$x_3 = \alpha_{34} + \delta_3^{(0)} + \delta_3^{(1)} + \ldots$$

Пример 1.15. Решим методом Зейделя систему уравнений

$$\begin{aligned} 10x_1 + x_2 + x_3 &= 12, \\ 2x_1 + 10x_2 + x_3 &= 13, \\ 2x_1 + 2x_2 + 10x_3 &= 14. \end{aligned}$$

Вычисления приведены в табл. 2.15 и 3.15.

§ 16. Способ Ньютона—Рафсона для нелинейных систем уравнений

Способы приближенного решения уравнений, рассмотренные в предыдущей главе, могут быть перенесены и на системы уравнений. Настоящий параграф посвящен рассмотрению метода решения систем, являющегося обобщением на случай нескольких неизвестных метода касательных, который для одного уравнения был разобран в § 6. Применительно к системам уравнений этот метод обычно называют методом Ньютона — Рафсона.

Ограничимся рассмотрением системы двух уравнений с двумя неизвестными, которую можно записать в виде

$$\left.\begin{aligned} f(x, y) &= 0, \\ \varphi(x, y) &= 0. \end{aligned}\right\} \qquad (1.16)$$

Считая заданными начальные приближения x_0, y_0 корней этой системы, будем искать поправки к этим приближенным значениям.

Если обозначить через h, k требуемые поправки, то точные значения корней можно записать в виде

$$x = x_0 + h, \quad y = y_0 + k.$$

Таблица 2.15

	10	1	1	12
	2	10	1	13
	2	2	10	14
(1)	—	—0,1	—0,1	1,2
(2)	—0,2	—	—0,1	1,3
(3)	—0,2	—0,2	—	1,4
(4)	1,2	1,3	1,4	

Таблица 3.15

	δ_1	δ_2	δ_3
(5)	—	—0,186	—0,186
(6)	—0,130	—	—0,196
(7)	—0,140	—0,140	—
(8)	—0,270	—0,326	—0,382
(9)	—	—0,0014	—0,0014
(10)	0,0326	—	—0,0074
(11)	0,0382	0,0382	—
(12)	0,0708	0,0368	—0,0088
(13)	—	—0,0006	—0,0006
(14)	—0,0037	—	0,0003
(15)	0,0009	0,0009	—
(16)	—0,0028	—0,0015	0,0006
x	0,9980	1,0183	1,0095

Таким образом, вместо системы (1.16) имеем

$$\left.\begin{aligned} f(x_0+h,\ y_0+k)&=0,\\ \varphi(x_0+h,\ y_0+k)&=0. \end{aligned}\right\} \tag{2.16}$$

Заменяя приращение функции $f(x,\ y)$ ее полным дифференциалом

$$\Delta f=f(x_0+h,\ y_0+k)-f(x_0,\ y_0)=h\frac{\partial f}{\partial x}+k\frac{\partial f}{\partial y}+o(h,\ k)$$

и поступая таким же образом с функцией $\varphi(x,\ y)$, перепишем уравнения (2.16) так

$$\left.\begin{aligned} f(x_0,\ y_0)+h\left(\frac{\partial f}{\partial x}\right)_0+k\left(\frac{\partial f}{\partial y}\right)_0+o_1(h,\ k)&=0,\\ \varphi(x_0,\ y_0)+h\left(\frac{\partial \varphi}{\partial x}\right)_0+k\left(\frac{\partial \varphi}{\partial y}\right)_0+o_2(h,\ k)&=0. \end{aligned}\right\} \tag{3.16}$$

Здесь символ $(\ )_0$ означает, что производная берется в точке x_0, y_0; o_1 и o_2 содержат члены более высокого порядка малости, нежели h и k. Пренебрегая в (3.16) членами высших порядков (т. е. считая, что h и k невелики), получим систему линейных уравнений для определения приближенных значений поправок:

$$\left.\begin{aligned} f(x_0,\ y_0)+h_1\left(\frac{\partial f}{\partial x}\right)_0+k_1\left(\frac{\partial f}{\partial y}\right)_0&=0,\\ \varphi(x_0,\ y_0)+h_1\left(\frac{\partial \varphi}{\partial x}\right)_0+k_1\left(\frac{\partial \varphi}{\partial y}\right)_0&=0. \end{aligned}\right\} \tag{4.16}$$

Из этой системы получаем значения поправок, которые легко выписать в явной форме через определители второго порядка

$$h_1=\frac{\begin{vmatrix} -f(x_0,\ y_0) & \left(\frac{\partial f}{\partial y}\right)_0\\ -\varphi(x_0,\ y_0) & \left(\frac{\partial \varphi}{\partial y}\right)_0 \end{vmatrix}}{\begin{vmatrix} \left(\frac{\partial f}{\partial x}\right)_0 & \left(\frac{\partial f}{\partial y}\right)_0\\ \left(\frac{\partial \varphi}{\partial x}\right)_0 & \left(\frac{\partial \varphi}{\partial y}\right)_0 \end{vmatrix}};\quad k_1=\frac{\begin{vmatrix} \left(\frac{\partial f}{\partial x}\right)_0 & -f(x_0,\ y_0)\\ \left(\frac{\partial \varphi}{\partial x}\right)_0 & -\varphi(x_0,\ y_0) \end{vmatrix}}{\begin{vmatrix} \left(\frac{\partial f}{\partial x}\right)_0 & \left(\frac{\partial f}{\partial y}\right)_0\\ \left(\frac{\partial \varphi}{\partial x}\right)_0 & \left(\frac{\partial \varphi}{\partial y}\right)_0 \end{vmatrix}}. \tag{5.16}$$

Более точные, чем x_0, y_0, значения корней получаются, как

$$x_1 = x_0 + h_1; \quad y_1 = y_0 + k_1.$$

Дальнейшие поправки можно получить тем же путем, отправляясь от точки x_1, y_1.

Пример 1.16. Решим систему уравнений

$$3x - 6y + 2 = 0,$$
$$x^3 + y^3 - 1 = 0.$$

Здесь $f(x, y) = 3x - 6y + 2$, $\varphi(x, y) = x^3 + y^3 - 1$. Построив графики для обеих функций (рис. 13), находим приближенные значения решения системы $x_0 = 0{,}83$, $y_0 = 0{,}75$.

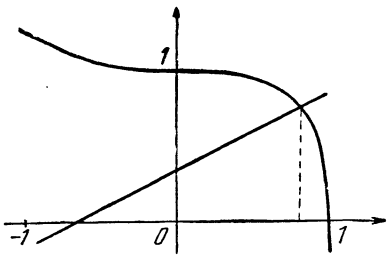


Рис. 13.

Найдем теперь поправки к этим приближенным значениям, пользуясь формулами (5.16).

Так как $\frac{\partial f}{\partial x} \equiv 3$, $\frac{\partial f}{\partial y} \equiv -6$, $\frac{\partial \varphi}{\partial x} = 3x^2$, $\frac{\partial \varphi}{\partial y} = 3y^2$, то

$$\left(\frac{\partial \varphi}{\partial x}\right)_0 = 2{,}07, \quad \left(\frac{\partial \varphi}{\partial y}\right)_0 = 1{,}69.$$

Значения функции в точке x_0, y_0 равны

$$f(x_0, y_0) = 3x_0 - 6y_0 + 2 = -0{,}01,$$
$$\varphi(x_0, y_0) = x_0^3 + y_0^3 - 1 = -0{,}01.$$

Следовательно, формулы (5.16) дают

$$h_1 = \frac{\begin{vmatrix} 0{,}01 & -6 \\ 0{,}01 & 1{,}69 \end{vmatrix}}{\begin{vmatrix} 3 & -6 \\ 2{,}07 & 1{,}69 \end{vmatrix}} = \frac{0{,}077}{17{,}5} = 0{,}0044,$$

$$k_1 = \frac{\begin{vmatrix} 3 & 0{,}01 \\ 2{,}07 & 0{,}01 \end{vmatrix}}{17{,}5} = \frac{0{,}0093}{17{,}5} = 0{,}0005.$$

Таким образом, после одного шага за решение данной системы можно принять точку

$$x_1 = 0{,}8344, \quad y_1 = 0{,}7505.$$

§ 17. Способ итераций для нелинейных систем уравнений

Способ итераций, рассмотренный в § 7 для одного уравнения и в § 14 для систем линейных уравнений, может быть с успехом применен для решения систем более общего вида. Рассмотрим применение способа итераций для системы двух уравнений с двумя неизвестными

$$\left.\begin{aligned} f(x, y) &= 0, \\ \varphi(x, y) &= 0. \end{aligned}\right\} \tag{1.17}$$

Предположим, что даны начальные приближения корней x_0, y_0. Приведем систему к виду

$$\left.\begin{aligned} x &= F(x, y), \\ y &= \Phi(x, y). \end{aligned}\right\} \tag{2.17}$$

Подставив в правые части системы (2.17) вместо x и y значения нулевых приближений x_0, y_0, получим первые приближения корней:

$$x_1 = F(x_0, y_0),$$
$$y_1 = \Phi(x_0, y_0).$$

Аналогично определяются вторые приближения:

$$x_2 = F(x_1, y_1),$$
$$y_2 = \Phi(x_1, y_1)$$

и, вообще,

$$x_n = F(x_{n-1}, y_{n-1}),$$
$$y_n = \Phi(x_{n-1}, y_{n-1}).$$

Легко установить, что если функции $F(x, y)$ и $\Phi(x, y)$ непрерывны и последовательности $x_1, x_2, \ldots, x_n, \ldots$ и $y_1, y_2, \ldots, y_n, \ldots$ сходятся, то пределы их являются корнями системы (2.17).

Сформулируем теперь условия, при которых описанный итерационный процесс является сходящимся.

Теорема. *Пусть* $\bar{x}$ *и* $\bar{y}$ *— истинные значения корней системы* (2.17) *и известно, что* $a<\bar{x}<b$, $c<\bar{y}<d$; *предположим также, что в прямоугольнике, ограниченном прямыми* $x=a$, $x=b$, $y=c$ *и* $y=d$, *других корней нет. Тогда, если в указанном прямоугольнике выполняются неравенства*

$$\left|\frac{\partial F}{\partial x}\right|\leqslant p_1,\quad \left|\frac{\partial F}{\partial y}\right|\leqslant q_1,\quad \left|\frac{\partial \Phi}{\partial x}\right|\leqslant p_2,\quad \left|\frac{\partial \Phi}{\partial y}\right|\leqslant q_2,$$

где $p_1+p_2\leqslant M<1$ *и* $q_1+q_2\leqslant N<1$, *то итерационный процесс сходится, причем в качестве нулевого приближения* $(x_0,\ y_0)$ *можно брать любую точку прямоугольника* *).

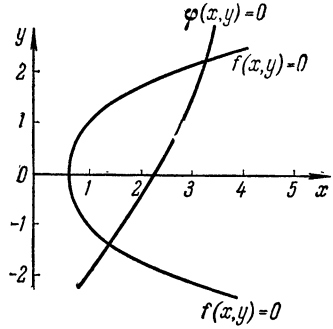


Рис. 14.

Доказательства этой теоремы мы не приводим.

Пример 1.17. Решим способом итераций систему

$$f(x,\ y)=x+3\lg x-y^2=0,$$
$$\varphi(x,\ y)=2x^2-xy-5x+1=0.$$

Построим кривые $f(x,\ y)=x+3\lg x-y^2=0$ и $\varphi(x,\ y)=2x^2-xy-5x+1=0$ и определим графически точки их пересечения (рис. 14). Это будут точки (1,4; —1,4) и (3,4; 2,2).

Рассмотрим вторую из этих точек и уточним соответствующие корни по способу итераций.

Прежде всего систему необходимо привести к виду (2.17), что можно сделать различными путями. Если приведем систему к виду

$$x=y^2-3\lg x,$$
$$y=2x+1/x-5,$$

то $F(x,\ y)=y^2-3\lg x$, $\Phi(x,\ y)=2x+1/x-5$. Здесь производные

$$\frac{\partial F}{\partial x}=-\frac{3\lg e}{x};\quad \frac{\partial F}{\partial y}=2y;\quad \frac{\partial \Phi}{\partial x}=2-\frac{1}{x^2};\quad \frac{\partial \Phi}{\partial y}=0.$$

*) В условии теоремы следовало бы еще добавить, что ни одно из последующих приближений $(x_n,\ y_n)$ не выходит за границы рассматриваемого прямоугольника; впрочем, обычно это условие выполняется (см. сноску на стр. 51).

Отсюда видно, что в окрестности точки $x_0=3{,}4$; $y_0=2{,}2$ будут иметь место неравенства

$$\left|\frac{\partial\Phi}{\partial x}\right|+\left|\frac{\partial F}{\partial x}\right|>1,\quad \left|\frac{\partial F}{\partial y}\right|>4.$$

Это показывает, что при таком виде системы итерационный процесс расходится.

Определим теперь x из второго уравнения, а y из первого:

$$x=\sqrt{\frac{x(5+y)-1}{2}},\quad y=\sqrt{x+3\lg x}.$$

Здесь

$$\frac{\partial F}{\partial x}=\frac{1}{\sqrt{2}}\frac{5+y}{2\sqrt{x(5+y)-1}};\quad \frac{\partial F}{\partial y}=\frac{1}{\sqrt{2}}\frac{x}{2\sqrt{x(5+y)-1}};$$

$$\frac{\partial\Phi}{\partial x}=\frac{1+\frac{3\lg e}{x}}{2\sqrt{x+3\lg x}};\quad \frac{\partial\Phi}{\partial y}=0.$$

За область изоляции корня можно принять прямоугольник $3<x<4$, $2<y<2{,}5$. Легко установить, что в этом прямоугольнике

$$\left|\frac{\partial F}{\partial x}\right|<0{,}60,\quad \left|\frac{\partial F}{\partial y}\right|<0{,}32,\quad \left|\frac{\partial\varphi}{\partial x}\right|<0{,}34.$$

Следовательно, процесс сходится, но так как сумма производных по x сравнительно велика, то скорость сходимости оказывается небольшой.

Вычисления с нулевым приближением $x_0=3{,}4$; $y_0=2{,}2$ дают

$$x_1=\sqrt{\frac{3{,}4\,(2{,}2+5)-1}{2}}=3{,}426,$$

$$y_1=\sqrt{3{,}426+3\lg 3{,}426}=2{,}243,$$

$$x_2=3{,}451,\qquad y_2=2{,}250,$$

$$x_3=3{,}466,\qquad y_3=2{,}255,$$

$$x_4=3{,}475,\qquad y_4=2{,}258,$$

$$x_5=3{,}480,\qquad y_5=2{,}259,$$

$$x_6=3{,}483\qquad y_6=2{,}260.$$

§ 18. Программирование итерационного процесса для системы линейных уравнений

Как и для одного уравнения (см. § 11), программирование итерационного процесса для системы сводится, в сущности, к итерационному циклу. Однако здесь имеются и некоторые особенности. В настоящем параграфе будет рассмотрена программа для решения системы линейных уравнений по способу Зейделя. Систему линейных уравнений n-го порядка удобно записывать в виде

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots & \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n. \end{aligned}\right\} \qquad (1.18)$$

При подготовке к итерации мы приводим ее к виду (2.15)

$$\begin{aligned} x_1 &= \alpha_{12}x_2 + \ldots + \alpha_{1n}x_n + \beta_1, \\ x_2 &= \alpha_{21}x_1 + \ldots + \alpha_{2n}x_n + \beta_2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n &= \alpha_{n1}x_1 + \alpha_{n2}x_2 + \ldots + \beta_n, \end{aligned}$$

разрешая первое уравнение относительно x_1, второе — относительно x_2 и т. д. Коэффициенты в этой системе находятся по формулам

$$\left.\begin{aligned} &\alpha_{11} = 0, \quad \alpha_{12} = -a_{12}/a_{11}, \ \ldots, \ \alpha_{1n} = -a_{1n}/a_{11}, \\ &\alpha_{21} = -a_{21}/a_{22}, \quad \alpha_{22} = 0, \quad \ldots, \ \alpha_{2n} = -a_{2n}/a_{22}, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &\alpha_{n1} = -a_{n1}/a_{nn}, \ \alpha_{n2} = -a_{n2}/a_{nn}, \ \ldots, \ \alpha_{nn} = 0. \end{aligned}\right\} \qquad (2.18)$$

Начальные приближения неизвестных берутся равными

$$x_1^{(0)} = \beta_1 = b_1/a_{11}, \quad x_2^{(0)} = \beta_2 = b_2/a_{22}, \ \ldots, \ x_n^{(0)} = b_n/a_{nn}. \qquad (3.18)$$

Поправки и очередные значения неизвестных вычисляются по формулам (4.15)—(6.15) и (8.15). Вычисления производятся до тех пор, пока все поправки не окажутся по абсолютной величине меньше заданной величины ε, т. е. до выполнения условий

$$|\delta_1^{(\nu)}| < \varepsilon, \quad |\delta_2^{(\nu)}| < \varepsilon, \ \ldots, \ |\delta_n^{(\nu)}| < \varepsilon. \qquad (4.18)$$

Пусть матрица коэффициентов системы записана подряд в ячейках a_{11}, a_{12}, ..., a_{nn}, а столбец свободных членов — в ячейках b_1, b_2, ..., b_n. Пусть, кроме того, в памяти отведено рабочее поле c_{11} — c_{nn}, размер которого равен размеру матрицы коэффициентов.

Программа решения системы по способу Зейделя состоит из двух блоков:

А) Нахождение коэффициентов α по формулам (2.18) и начальных приближений по формулам (3.18).

Б) Определение поправок и последовательных значений неизвестных и проверка окончания работы программы.

Прежде всего, чтобы не портить матрицу коэффициентов (иначе программа не будет самовосстанавливающейся), перешлем коэффициенты на рабочее поле:

Пересылка:

PA	0	0	0
		$a_{11}^* =$	c_{11}^*
$PA < \overline{n^2-1}$		$Я-1$	1^*

Начальное значение $x_1^{(0)}$ и первую строку коэффициентов α можно получить на рабочем поле с помощью программы:

Рабочая часть:

T_1	«1» :	c_{11}	$= R_0$
	0 —	R_0	$= R_1$
T_2	b_1 ·	R_0	$= x_1$
T_3		x_1	$= \delta_1$
	$[PA]$ 0	(0, $n-1$, 0)	0
T_4	c_{11}^* ·	R_1	$= c_{11}^*$
	$PA \geqslant 1$	T_4	F^*.
T_5		0	$= c_{11}$

Для того чтобы, применяя те же команды, получить $x_2^{(0)}$, $\delta_2^{(0)}$ и вторую строку коэффициентов α, команды $T_1 - T_5$ нужно переадресовать следующим образом:

Переадресация:

T_1+,	(0, $n+1$, 0) $= T_1$
T_2+,	(1, 0, 1) $= T_2$
T_3+,	(0, 1, 1) $= T_3$
T_4+,	(n, 0, n) $= T_4$
T_5+,	(0, 0, $n+1$) $= T_5$

При этом в командах T_2, T_3 происходит переход к следующему элементу матрицы, в команде T_4 — к следующей строке и в командах T_1, T_5 — от одного диагонального элемента к другому. Поэтому T_2, T_3 переадресуются на 1, T_4 — на n и T_1, T_5 — на $n+1$.

Теперь мы можем записать программу блока А) в виде блок-программы:

	Пересылка
	$(0,\ n,\ 0)-,\ \ (0,\ 1,\ 0)=k$
	Восстановление переменных команд с обходом переадресации и передачей управления на T_1
	Переадресация
	Рабочая часть
k	$-,\ (0,\ 1,\ 0)=k$
У0	Переадресация

Программирование блока Б) начнем с определения величины $\delta_1^{(1)}$, по формуле (4.15) в ячейке S:

T_6	$[PA]\,0$	$(0,\ n-1,\ 0)$	0
		0	$=S$
	$\delta_1^*\ \cdot$	c_{11}^*	$=R_0$
	$S\ +$	R_0	$=S$
	$PA \geqslant \bar{1}$	T_6	F^*

Теперь нужно переслать вычисленное в ячейке S значение в ячейку δ_1, использовать его при получении нового приближения и проверить выполнение неравенства (4.18).

Введем специальный счетчик — ячейку q, в которой будем считать, для какого числа поправок это неравенство уже выполняется. (Предварительно в ячейку q нужно записать нуль.) Перечисленные задачи выполняются программой

T_7		$x_1\ +\ S$	$=x_1$
		$\lvert S\rvert - \lvert\varepsilon\rvert$	$=0$
T_8	*У0*	$S\quad Я+2$	δ_1
		$q\ +,\ (0,\ 1,\ 0)=q$	

Для нахождения величин $x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}$ написанные две группы команд следует включить в цикл, причем команды $T_6 - T_8$ переадресуются следующим образом:

$$T_6+,\ (0,\ n,\ 0) = T_6$$
$$T_7+,\ (1,\ 0,\ 1) = T_7$$
$$T_8+,\ (0,\ 0,\ 1) = T_8$$

Окончание цикла итераций производится с помощью команд

$$q \not\sim (0,\ n,\ 0) = 0$$
У0 начало Б
стоп

так что программу блока Б) можно записать так:

				T_6^0	$= T_6$
				T_7^0	$= T_7$
				T_8^0	$= T_8$
начало Б)		$(0,\ n,\ 0) -,$		$(0,\ 1,\ 0)$	$= \rho$
	Б	0			q
		T_6	$+,$	$(0,\ n,\ 0)$	$= T_6$
		T_7	$+,$	$(1,\ 0,\ 1)$	$= T_7$
		T_8	$+,$	$(0,\ 0,\ 1)$	$= T_8$
	$[PA]$	0		$(0,\ n-1,\ 0)$	0
				0	$= S$
	T_6	$(\delta_1^*$	$\cdot$	c_{11}^*	$= R_0)$ н. п.
		S	$+$	R_0	$= S$
	$PA \geqslant \bar{1}$				F^*
	T_7	$(x_1$	$+$	S	$= x_1)$ н. п.
		$\lvert S \rvert$	$-$	$\lvert \varepsilon \rvert$	$= 0$
	T_8	$(У1$			$\delta_1)$ н. п.
		S			
		q	$+,$	$(0,\ 1,\ 0)$	$= q$
		ρ	$-,$	$(0,\ 1,\ 0)$	$= \rho$
	У0				
		q	$\not\sim$	$(0,\ n,\ 0)$	$= 0$
	У0		начало Б		
	стоп				

Если программу решения системы оформлять как стандартную, то большое число команд придется формировать. При этом, однако, отпадает необходимость восстановления переменных команд T_1—T_8, так как их можно формировать в начале программы в требуемом виде. Отметим еще, что можно организовать переадресацию команд T_7 и T_8 с помощью регистра адреса.

§ 19. Запись на алголе программы решения системы линейных уравнений по способу Гаусса

Способ последовательного исключения неизвестных, применяемый для решения линейных систем (способ Гаусса), может быть представлен в виде различных вычислительных схем. Характерной чертой рассмотренной в § 12 схемы единственного деления является разбиение процесса решения задачи на два этапа: прямой и обратный ход. Такое разбиение представляет некоторые неудобства для программирования. Более удобным для программирования является другой вычислительный вариант способа Гаусса, который называют *схемой Жордана*. В ней значения неизвестных получаются в результате прямого хода, так что обратный ход оказывается ненужным.

Рассмотрим схему Жордана на примере системы четырех уравнений с четырьмя неизвестными, которую мы будем предполагать записанной в виде:

$$\left.\begin{aligned} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + a_{14}^{(1)}x_4 &= b_1^{(1)},\\ a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + a_{24}^{(1)}x_4 &= b_2^{(1)},\\ a_{31}^{(1)}x_1 + a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + a_{34}^{(1)}x_4 &= b_3^{(1)},\\ a_{41}^{(1)}x_1 + a_{42}^{(1)}x_2 + a_{43}^{(1)}x_3 + a_{44}^{(1)}x_4 &= b_4^{(1)}. \end{aligned}\right\} \qquad (1.19)$$

Разделив первое уравнение системы на $a_{11}^{(1)}$ и подставляя затем найденное отсюда значение x_1 в остальные уравнения, получим:

$$\left.\begin{aligned} x_1 + a_{12}^{(2)}x_2 + a_{13}^{(2)}x_3 + a_{14}^{(2)}x_4 &= b_1^{(2)},\\ a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 + a_{24}^{(2)}x_4 &= b_2^{(2)},\\ a_{32}^{(2)}x_2 + a_{33}^{(2)}x_3 + a_{34}^{(2)}x_4 &= b_3^{(2)},\\ a_{42}^{(2)}x_2 + a_{43}^{(2)}x_3 + a_{44}^{(2)}x_4 &= b_4^{(2)}. \end{aligned}\right\} \qquad (2.19)$$

Коэффициенты системы (2.19) получаются по формулам:

$$a_{1j}^{(2)} = l_{1j}, \quad b_1^{(2)} = u_1 \quad (j = 2, 3, 4), \tag{3.19}$$

$$a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i1}^{(1)} l_{ij}, \quad b_i^{(2)} = b_i^{(1)} - a_{i1}^{(1)} u_1 \tag{4.19}$$

$$(i, j = 2, 3, 4),$$

где

$$l_{1j} = a_{1j}^{(1)} / a_{11}^{(1)}, \quad u_1 = b_1^{(1)} / a_{11}^{(1)} \quad (j = 2, 3, 4). \tag{5.19}$$

На следующем шаге разделим второе уравнение системы (2.19) на $a_{22}^{(2)}$, определим x_2 из этого уравнения и подставим его значение во все остальные уравнения системы, в том числе и в первое (в этом и состоит отличие схемы Жордана от рассмотренной ранее, где значение x_2 подставлялось только в два следующих уравнения). Тогда получим:

$$\left.\begin{aligned} x_1 + \quad a_{13}^{(3)} x_3 + a_{14}^{(3)} x_4 &= b_1^{(3)}, \\ x_2 + a_{23}^{(3)} x_3 + a_{24}^{(3)} x_4 &= b_2^{(3)}, \\ a_{33}^{(3)} x_3 + a_{34}^{(3)} x_4 &= b_3^{(3)}, \\ a_{43}^{(3)} x_3 + a_{44}^{(3)} x_4 &= b_4^{(3)}. \end{aligned}\right\} \tag{6.19}$$

Подобно предыдущему,

$$a_{2j}^{(3)} = l_{2j}, \quad b_2^{(3)} = u_2 \quad (j = 3, 4), \tag{7.19}$$

$$a_{ij}^{(3)} = a_{ij}^{(2)} - a_{i2}^{(2)} l_{2j}, \quad b_i^{(3)} = b_i^{(2)} - a_{i2}^{(2)} u_2 \tag{8.19}$$

$$(i = 1, 3, 4; \ j = 3, 4),$$

$$l_{2j} = a_{2j}^{(2)} / a_{22}^{(2)}, \quad u_2 = b_2^{(2)} / a_{22}^{(2)} \tag{9.19}$$

$$(j = 3, 4).$$

Разделив третье уравнение системы (6.19) на $a_{33}^{(3)}$ и исключив затем x_3 снова из всех уравнений системы, найдем:

$$\left.\begin{aligned} x_1 + \quad a_{14}^{(4)} x_4 &= b_1^{(4)}, \\ x_2 + \quad a_{24}^{(4)} x_4 &= b_2^{(4)}, \\ x_3 + a_{34}^{(4)} x_4 &= b_3^{(4)}, \\ a_{44}^{(4)} x_4 &= b_4^{(4)}. \end{aligned}\right\} \tag{10.19}$$

Коэффициенты этой системы получаются из коэффициентов предыдущей по формулам, подобным (7.19)—(9.19).

Исключая затем x_4 из системы (10.19), получим, наконец,

$$\left.\begin{aligned} x_1 &= b_1^{(5)},\\ x_2 &= b_2^{(5)},\\ x_3 &= b_3^{(5)},\\ x_4 &= b_4^{(5)}, \end{aligned}\right\} \tag{11.19}$$

так что исключение неизвестных в процессе прямого хода уже привело к получению их значений.

Благодаря отсутствию обратного хода схема Жордана программируется проще, хотя и требует примерно в полтора раза больше арифметических действий, чем ранее рассмотренная схема.

Рассмотрим алгольную программу схемы Жордана для системы n уравнений с n неизвестными:

$$\left.\begin{aligned} &c_{11}x_1 + c_{12}x_2 + \ldots + c_{1n}x_n = d_1,\\ &c_{21}x_1 + c_{22}x_2 + \ldots + c_{2n}x_n = d_2,\\ &\qquad\qquad\ldots\ldots\ldots\ldots\\ &c_{n1}x_1 + c_{n2}x_2 + \ldots + c_{nn}x_n = d_n. \end{aligned}\right\} \tag{12.19}$$

Из формул схемы Жордана видно, что в процессе исключения неизвестных коэффициенты каждой преобразованной системы можно помещать на место коэффициентов преобразуемой. Тогда в конце преобразований свободные члены оказываются равными значениям неизвестных. Для того чтобы иметь возможность повторить решение системы (12.19), следует предварительно образовать из матрицы ее коэффициентов c_{ij}, d_i (i, $j=1, 2, \ldots$) новый массив — копию a_{ij}, b_i (i, $j=1, 2, \ldots, n$) при помощи включенных в цикл операторов присваивания

```
for i: =1 step 1 until n do
  begin b[i]:=d[i];
      for j:=1 step 1 until n do
          a[i, j]:=c[i, j]
  end
```

Напишем теперь программу одного шага метода исключения, на котором из системы, содержащей неизвестные

x_k, x_{k+1}, ..., x_n, получается система с неизвестными x_{k+1}, x_{k+2}, ..., x_n.

Сопоставляя формулы (3.19)—(5.19) и (7.19)—(9.19), получим следующие соотношения для нужных коэффициентов

$$b_k^{(k+1)} = b_k^{(k)}/a_{kk}^{(k)}, \quad a_{kj}^{(k+1)} = a_{kj}^{(k)}/a_{kk}^{(k)}$$
$$(j = k+1, \ k+2, \ \ldots, \ n),$$
$$b_i^{(k+1)} = b_i^{(k)} - a_{ik}^{(k)} \cdot b_k^{(k+1)},$$
$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - a_{ik}^{(k)} \cdot a_{kj}^{(k+1)}$$
$$(i = 1, 2, \ \ldots, \ k-1, \ k+1, \ k+2, \ \ldots, \ n;$$
$$j = k+1, \ k+2, \ \ldots, \ n).$$

Алгольная программа, реализующая эти формулы, состоит из оператора присваивания для вычисления $b_k^{(k+1)}$, простого цикла для определения коэффициентов $a_{kj}^{(k+1)}$ и двойного цикла для нахождения матрицы $a_{ij}^{(k+1)}$ и вектора $b_i^{(k+1)}$

```
        b[k]: = b[k]/a[k, k];
for j: = k+1 step 1 until n do
        a[k, j]: = a[k, j]/a[k, k];
for i: = 1 step 1 until k-1,
        k+1 step 1 until n do
    begin b[i]: = b[i] - a[i, k]×b[k];
        for j: = k+1 step 1 until n do
        a[i, j]: = a[i, j] - a[i, k]×a[k, j]
    end
```

Теперь для получения неизвестных нам достаточно составленную программу одного шага метода исключения включить в цикл по k от 0 до $n-1$, а затем присвоить искомым величинам x_1, x_2, ..., x_n значения свободных членов b_1, b_2, ..., b_n. После оформления соответствующих

циклов мы получим алгольную программу схемы Жордана в следующем виде:

```
перенос: for i: =1 step 1 until n do
           begin b[i]: =d[i];
                 for j: =1 step 1 until n do
                   a[i, j]: =c[i, j]
           end
исключ: for k: =0 step 1 until n—1 do
  шаг:   begin b[k]: =b[k]/a[k, k];
         for j: =k+1 step 1 until n do
           a[k, j]: =a[k, j]/a[k, k];
         for i: =1 step 1 until k—1,
                k+1 step 1 until n do
         begin b[i]: =b[i]—a[i, k]×b[k];
               for j: =k+1 step 1 until n do
               a[i, j]: =a[i, j]—a[i, k]×a[k, j]
         end
         end
неизв.: for i: =1 step 1 until n do
           x[i]: =b[i];
```

Оформим теперь полученную программу в виде алгольной процедуры. Обращение к процедуре запишем в виде:

жордан (n, c, d, x, a, b).

Здесь n, c, d, x—фактические параметры, обозначающие соответственно порядок системы (n), идентификаторы матрицы коэффициентов системы (c), векторы правых частей (d) и векторы неизвестных (x); идентификаторы вспомогательных массивов a и b длины n^2 и n.

Описание процедуры (с комментариями) записывается в виде:

```
procedure жордан (n, c, d, x, a, b)
    begin value n, integer n,
          array c, d, x, a, b
```

comment по схеме Жордана решается система n линейных уравнений с n неизвестными $cx=d$, a и b — вспомогательные матрица и вектор порядка $n\uparrow 2$ и n соответственно;

```
begin integer i, j, k;
comment i — номер уравнения в системе,
        j — номер неизвестного,
        k — номер шага исключения;
comment присвоение начальных значений массивам
        a[1:n, 1:n] и b[1:n];
перенос: for i:=1 step 1 until n do
        begin b[i]:=d[i];
           for j:=1 step 1 until n do
                a[i, j]:=c[c, j]
        end переноса.

comment из уравнений системы a×x=b последовательно
исключаются неизвестные

                x[1], x[2], ..., x[n];
        исключ: for k:=0 step 1 until n−1 do
begin

comment двойной цикл нахождения x[k] из уравнения
с номером k и подстановки в уравнения с номерами
1, 2, ..., k−1, k+1, k+2, ..., n;

    b[k]:=b[k]/a[k, k];
    for j:=k+1 step 1 until n do
    begin a[k, j]:=a[k, j]/a[k, k];
      for i:=1 step 1 until k−1,
                k+1 step 1 until n do
        begin b[i]:=b[i]−a[i, k]×b[k];
          for j:=k+1 step 1 until n do
            a[i, j]:=a[i, j]−a[i, k]×a[k, j]
        end цикла по строке
    end цикла исключений;
```

comment полученные значения свободных членов равны неизвестным;
end исключения

неизв.: **for** i: $=1$ **step** 1 **until** n **do**
$x[i]$: $=b[i]$

end процедуры;

В качестве примера применения составленной процедуры напишем алгольную программу решения линейной системы вида:

$$\left.\begin{aligned} x_1+\ 2x_2+\ 3x_3+\ldots+15x_{15}&=0,\\ 2x_1+\ 3x_2+\ 4x_3+\ldots+16x_{15}&=1,\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots&\\ 15x_1+16x_2+17x_3+\ldots+29x_{15}&=14. \end{aligned}\right\}$$

Представим эту систему в виде:

$$\sum_{j=1}^{15} c_{ij}x_i=d_i \quad (j=1, 2, \ldots, 15),$$

где

$$c_{ij}=i+j-1, \quad d_i=i-1.$$

Тогда алгольный блок решения системы с печатью результатов запишется в виде:

```
begin
    array c[1 : 15,1 : 15], d[1 : 15], x[1 : 15];
    integer i, j;
      comment вычисление коэффициентов c[i, j], правых частей d[i] системы;
     for i: =1 step 1 until 15 do
       begin d[i]: =i—1;
             for j: =1 step 1 until 15 do
                 c[i, j]: =i+j—1
       end вычисления коэффициентов;
```

```
    жордан (15, c, d, x, c, d);
    comment при обращении к процедуре вспомогательные матрицы и векторы совмещены: a=c, b=d;
    печать неизвестных: for i:=1 step 1 do 15
                        print (x[i]);
    end
```

Заметим, что для превращения этого блока в алгольную программу следует включить его во внешний блок, описание которого состоит лишь из приведенного ранее описания процедуры решения системы по методу Жордана.

ГЛАВА IV

ИНТЕРПОЛЯЦИЯ

§ 20. Общая постановка задачи интерполяции

В § 5 первого выпуска мы уже встречались с задачей интерполяции и простейшим приемом ее решения — линейной интерполяцией. Под интерполяцией там понималось отыскание значений функции, соответствующих промежуточным значениям аргумента, отсутствующим в таблице. Первоначальной задачей интерполяции действительно являлось «искусство чтения между строками». В настоящее время задача интерполяции понимается шире.

В известном смысле можно сказать, что задача интерполяции обратна задаче табулирования функций. Именно, при табулировании по аналитическому выражению функции находится таблица ее значений, а при интерполяции, наоборот, по таблице значений функции строится ее аналитическое выражение. Поясним, что следует понимать под этими словами.

Пусть $y=f(x)$ — некоторая функция, для которой известна лишь таблица ее значений, т. е. известно, что при значениях аргумента $x=x_0, x_1, \ldots, x_n$ функция принимает значения $y_0, y_1, \ldots, y_n$:

$$\left.\begin{aligned} f(x_0)&=y_0,\\ f(x_1)&=y_1,\\ &\cdots\cdots\\ f(x_n)&=y_n. \end{aligned}\right\} \tag{1.20}$$

Геометрическая задача отыскания функции $f(x)$ по заданным ее частным значениям означает, что мы должны построить кривую, проходящую через точки плоскости

с координатами (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) (рис. 15). Читателю должно быть ясно, что через данные точки (пусть даже в большом количестве) можно провести бесчисленное множество различных кривых. Таким образом, задача отыскания функции $f(x)$ по конечному числу заданных ее значений слишком неопределенна; таких функций можно построить бесчисленное множество.

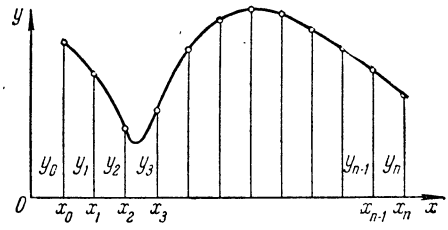


Рис. 15.

Мы будем в дальнейшем обозначать через $F(x)$ любую функцию, которая в заданных точках принимает заданные значения. Из сказанного выше следует, что функций $F(x)$ может быть сколько угодно.

Предположим теперь, что функция $F(x)$ не произвольная, а удовлетворяет некоторым дополнительным требованиям; тогда задача приобретает значительно более определенный характер. Чаще всего требуют, чтобы функция $F(x)$ была многочленом степени на единицу меньшей, чем число известных табличных значений.

Таким образом, мы приходим к следующей формулировке задачи.

Для данных значений $x=x_0, x_1, \ldots, x_n$ *и* $y=y_0, y_1, \ldots, y_n$ *найти многочлен* $y=F(x)$ *степени* n, *удовлетворяющий условиям*

$$\left.\begin{aligned} F(x_0) &= y_0, \\ F(x_1) &= y_1, \\ &\cdots\cdots \\ F(x_n) &= y_n. \end{aligned}\right\} \qquad (2.20)$$

Иначе говоря, речь идет об отыскании аналитического выражения для многочлена, принимающего в заданных точках заданные значения. Такую задачу называют задачей *параболической интерполяции*. Точки $x_0, x_1, \ldots, x_n$ называют *узлами интерполяции*.

Многочлен $F(x)$, удовлетворяющий условиям (2.20), называется *интерполяционным многочленом*, а формулы для его построения — *интерполяционными формулами*. Различным интерполяционным формулам будут посвящены следующие параграфы настоящей главы.

Основная идея применения интерполяционных формул состоит в том, что функция $y=f(x)$, для которой известна лишь таблица значений (1.20), заменяется интерполяционным многочленом, который рассматривается как приближенное аналитическое выражение для функции $f(x)$.

Замена функции $f(x)$ ее интерполяционным многочленом используется прежде всего для нахождения промежуточных значений функций, как о том говорилось в начале параграфа. С этим применением интерполяционных формул связано и их название. Но этим случаем их применение не ограничивается. Такая замена может потребоваться и тогда, когда аналитическое выражение для $f(x)$ известно, но является слишком сложным, а функция $f(x)$ должна подвергаться различным математическим операциям (например, интегрированию). Возможно также, что значения функции $f(x)$ получаются из экспериментальных данных, так что получение других промежуточных значений может оказаться затруднительным или невозможным, а аналитическое выражение функции — неизвестным.

Несмотря на указанную выше «противоположность» задач табулирования и интерполирования, интерполяционные формулы широко используются при составлении таблиц функций.

Именно, отдельные значения функции вычисляются (чаще всего с помощью степенных рядов) непосредственно с большой точностью, а затем полученные таблицы многократно уплотняются с помощью интерполяционных формул, благодаря чему получают более или менее подробные таблицы.

§ 21. Табличные разности и их свойства

Предположим, что *рассматриваемые значения аргумента являются равноотстоящими*, т. е. образуют арифметическую прогрессию. Такое предположение обычно имеет место при интерполяции функций, заданных в виде таблиц с постоянным шагом, где $x_1 = x_0 + h$, $x_2 = x_0 + 2h$, ..., $x_m = x_0 + mh$. Разность арифметической прогрессии h и называется *шагом таблицы*.

Прежде чем перейти к рассмотрению интерполяционных формул, познакомимся с понятием *конечных разностей*.

Пусть значения функции $f(x)$ заданы в точках x_0, $x_1 = x_0 + h$, $x_2 = x_0 + 2h$, ..., $x_n = x_0 + nh$ (*узлы интерполяции*). Составим разности значений функции:

$$y_1 - y_0 = f(x_0 + h) - f(x_0) = \Delta y_0 = \Delta f(x_0),$$
$$y_2 - y_1 = f(x_0 + 2h) - f(x_0 + h) = \Delta y_1 = \Delta f(x_1),$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$y_n - y_{n-1} = f(x_0 + nh) - f(x_0 + (n-1)h) = \Delta y_{n-1} = \Delta f(x_0 + (n-1)h).$$

Эти значения называют *первыми разностями* функции, или *разностями первого порядка*. По ним мы можем составить разности второго порядка, или *вторые разности*,

$$\Delta^2 y_0 = \Delta y_1 - \Delta y_0, \quad \Delta^2 y_1 = \Delta y_2 - \Delta y_1, \ldots, \Delta^2 y_m = \Delta y_{m+1} - \Delta y_m,$$

и, вообще, разности любого порядка k, или *k-е разности*

$$\Delta^k y_m = \Delta^{k-1} y_{m+1} - \Delta^{k-1} y_m. \qquad (1.21)$$

Последовательные разности располагают в форме таблицы. Употребляются две формы таблиц последовательных разностей. В табл. 1.21 разности в каждом столбце вписываются между соответствующими значениями уменьшаемого и вычитаемого; такая таблица называется *диагональной*. В таблице же 2.21 разности ставятся в одной строке с вычитаемым; в этом случае таблицу разностей называют *горизонтальной*. Так как разности функций обычно бывают невелики, то их принято записывать в единицах последнего знака, без нулей впереди.

Таблица 1.21

x	y	Разности			
		первые	вторые	третьи	...
x_0	y_0				
		Δy_0			
$x_1=x_0+h$	y_1		$\Delta^2 y_0$		
		Δy_1		$\Delta^3 y_0$	
$x_2=x_0+2h$	y_2		$\Delta^2 y_1$		
		Δy_2		$\Delta^3 y_1$	
$x_3=x_0+3h$	y_3		$\Delta^2 y_2$		
		Δy_3		$\Delta^3 y_2$	
$x_4=x_0+4h$	y_4		$\Delta^2 y_3$		
		Δy_4		...	
$x_5=x_0+5h$	y_5		...		
		...			
...	...				

Равенства (1.21) определяют разности различных порядков последовательно. Установим соотношения, дающие выражения для конечных разностей непосредственно через значения функции. Действительно, так как $\Delta y_0 = y_1 - y_0$ и $\Delta y_1 = y_2 - y_1$, то $\Delta^2 y_0 = (y_2 - y_1) - (y_1 - y_0)$, так что

$$\Delta^2 y_0 = y_2 - 2y_1 + y_0.$$

Точно так же

$$\Delta^3 y_0 = y_3 - 3y_2 + 3y_1 - y_0.$$

Нетрудно доказать, что для любого k

$$\Delta^k y_0 = y_k - k y_{k-1} + \frac{k(k-1)}{2!} y_{k-2} - \ldots + (-1)^{k-1} k y_1 + (-1)^k y_0. \tag{2.21}$$

Эта формула имеет место и для разности $\Delta^k y_m$, достаточно ко всем номерам значений функции прибавить m.

Таблица 2.21

x	y	Разности			
		первые	вторые	третьи	...
x_0	y_0	Δy_0	$\Delta^2 y_0$	$\Delta^3 y_0$	
$x_1 = x_0 + h$	y_1	Δy_1	$\Delta^2 y_1$	$\Delta^3 y_1$	
$x_2 = x_0 + 2h$	y_2	Δy_2	$\Delta^2 y_2$	$\Delta^3 y_2$	
$x_3 = x_0 + 3h$	y_3	Δy_3	$\Delta^2 y_3$	...	
$x_4 = x_0 + 4h$	y_4	Δy_4	...		
$x_5 = x_0 + 5h$	y_5	...			
...	...				

Формула (2.21) легко запоминается, если заметить, что ее выражение напоминает разложение бинома $(y-1)^k$ по формуле бинома Ньютона. Надо только вместо степеней y писать y с тем же индексом. Например, вместо y^k следует писать y_k, вместо y^{k-1} писать y_{k-1} и т. д.; в последнем слагаемом вместо $1 = y^0$ пишем y_0.

Нередко бывают полезны и другие формулы, дающие выражение значений функции через конечные разности. Именно, из $\Delta y_0 = y_1 - y_0$ вытекает, что

$$y_1 = y_0 + \Delta y_0.$$

Аналогично, $y_2 = y_1 + \Delta y_1 = (y_0 + \Delta y_0) + \Delta y_1$, но так как $\Delta^2 y_0 = \Delta y_1 - \Delta y_0$, то

$$y_2 = y_0 + 2\Delta y_0 + \Delta^2 y_0.$$

Как и выше, эти выкладки можно провести для любого k, что дает

$$y_k = y_0 + k\Delta y_0 + \frac{k(k-1)}{2!}\Delta^2 y_0 + \ldots + \Delta^k y_0. \qquad (3.21)$$

Полученная формула дает возможность определить значения y_k при любом k через значение y_0 и конечные разности до k-го порядка включительно. Число k необходимо является целым. Формулу (3.21) удобно записать символически следующим образом:

$$y_k = (1+\Delta)^k y_0. \qquad (4.21)$$

Здесь скобка $(1+\Delta)^k$ раскрывается по формуле бинома Ньютона, а полученные «произведения» $\Delta^r y_0$ означают разности соответствующих порядков.

Отметим некоторые простейшие свойства конечных разностей:

1°. Если C постоянно, то $\Delta C = 0$.

2°. $\Delta[Cf(x)] = C\Delta f(x)$.

3°. $\Delta[f_1(x) + f_2(x)] = \Delta f_1(x) + \Delta f_2(x)$.

Перечисленные свойства достаточно очевидны. Кроме них, нам потребуется выражение разности функции $f(x) = x^n$ для целого n:

$$4°.\ \Delta(x^n) = nhx^{n-1} + \frac{n(n-1)}{2!}h^2x^{n-2} + \ldots + nh^{n-1}x + h^n.$$

Для доказательства этого равенства достаточно написать

$$\Delta(x^n) = (x+h)^n - x^n$$

и воспользоваться формулой бинома Ньютона.

С помощью этих свойств легко получить выражения для последовательных разностей многочлена. Действительно, если

$$y = f(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n, \qquad (5.21)$$

то в силу свойств 3° и 2°

$$\Delta y = a_0\Delta(x^n) + a_1\Delta(x^{n-1}) + \ldots + a_{n-1}\Delta(x),$$

так что вследствие 4°

$$\Delta y = a_0\left[nhx^{n-1} + \frac{n(n-1)}{2!}h^2x^{n-2} + \ldots + nh^{n-1}x + h^n\right] +$$
$$+ a_1\left[(n-1)hx^{n-2} + \frac{(n-1)(n-2)}{2!}h^2x^{n-3} + \ldots\right] + \ldots + a_{n-1}n,$$

или, окончательно,

$$\Delta y = a_0 nhx^{n-1} + \left[a_0\frac{n(n-1)}{2!}h^2 + a_1(n-1)h\right]x^{n-2} + \ldots + a_{n-1}h. \tag{6.21}$$

Таким образом, *первая разность многочлена степени n* со старшим членом a_0x^n *есть многочлен степени* $n-1$ со старшим членом a_0nhx^{n-1}. Вычисляя аналогичным путем последовательные разности многочлена любого порядка, убедимся в справедливости следующего утверждения.

Если $f(x)$ *— многочлен степени n относительно x со старшим членом* a_0x^n, *то для любого* $m < n$ *разность* $\Delta^m f(x)$ *есть многочлен от x степени* $n-m$ *со старшим членом*

$$n(n-1)\ldots(n-m+1)\,a_0h^mx^{n-m}.$$

Для $m=n$ *разность* $\Delta^n f(x) = n!\,a_0h^n$ *и при* $m > n$ *разность* $\Delta^m f(x)$ *тождественно равна нулю.*

Подчеркнем, что этот вывод верен только тогда, когда h постоянно, т. е. значения x образуют арифметическую прогрессию.

Справедлива и обратная теорема, которую мы приведем без доказательства: *если n-е разности функции, образованные для равноотстоящих значений аргумента при любом шаге h, постоянны, то функция представляет собой многочлен степени n.*

Последнее утверждение позволяет при отыскании промежуточных значений функции (если шаг таблицы постоянен) ограничиваться вычислениями многочленов, степень которых равна порядку практически постоянных разностей. Способы построения таких многочленов будут указаны в следующем параграфе.

Пример 1.21. Рассмотрим таблицу значений многочлена $f(x) = x^2 - 3x + 2$ (табл. 3.21).

Согласно принятому условию, записываем разности в единицах последнего знака без нулей впереди. В соответствии с приведенной выше теоремой вторые разности оказываются постоянными, а третьи разности равны нулю.

Пример 2.21. Рассмотрим функцию, значения которой заданы таблицей 4.21, и составим разности функции.

Мы видим, что третьи разности можно считать практически постоянными, и, следовательно, четвертые — равными нулю. Поэтому на рассматриваемом участке функция ведет себя приблизительно как многочлен третьей степени.

Необходимо, однако, иметь в виду, что теорема о конечных разностях для многочленов относится к точным разностям функции. Если же допускать округления, то, как видно из приводимых примеров, порядок практически постоянных разностей зависит как от точности вычисления значений функции, так и от шага таблицы.

Пример 3.21. Рассмотрим многочлен четвертой степени

$$y = 0{,}02x^4 + 0{,}05x^3 + 0{,}04x^2 + 0{,}01x - 8{,}85.$$

Точные значения этого многочлена даны в табл. 5.21. Как видно из таблицы, разности четвертого порядка постоянны, что вполне соответствует утверждениям теоремы.

Если же рассматривать значения этого многочлена с точностью до четвертого десятичного знака, то, как показывает табл. 6.21, при сохранении того же шага $h = 0{,}1$ практически постоянными можно считать уже третьи разности. Более того, в табл. 7.21 приведены значения того же многочлена с точностью до одной сотой; из нее мы видим, что в этом случае практически постоянными оказываются уже вторые разности.

Опираясь на замечание, сделанное выше, можно сказать, что если бы мы заранее не знали, что таблицы 6.21 и 7.21 дают значение многочлена четвертой степени, то мы искали бы промежуточные значения с помощью многочленов третьей степени в первом случае и второй степени во втором случае. Это означает, что *чем менее*

Таблица 3.21

(1)	(2)	(3)	(4)	(5)
x	y	Разности		
		Δy	$\Delta^2 y$	$\Delta^3 y$
1,0	0,00			
		−16		
1,2	−0,16		8	
		−8		0
1,4	−0,24		8	
		0		0
1,6	−0,24		8	
		8		0
1,8	−0,16		8	
		16		0
2,0	0,00		8	
		24		0
2,2	0,24		8	
		32		0
2,4	0,56		8	
		40		0
2,6	0,96		8	
		48		
2,8	1,44			

Таблица 4.21

(1)	(2)	(3)	(4)	(5)
x	y	Разности		
		Δy	$\Delta^2 y$	$\Delta^3 y$
0,30	1,0313			
		−330		
0,35	0,9983		−38	
		−368		6
0,40	0,9615		−32	
		−400		5
0,45	0,9215		−27	
		−427		4
0,50	0,8788		−23	
		−450		7
0,55	0,8338		−16	
		−466		5
0,60	0,7872		−11	
		−477		6
0,65	0,7395		−5	
		−482		
0,70	0,6913			

Таблица 5.21

(1)	(2)	(3)	(4)	(5)	(6)
x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
4,0	0,150000				
		810 972			
4,1	0,960972		53 448		
		864 420		2292	
4,2	1,825392		55 740		48
		920 160		2340	
4,3	2,745552		58 080		48
		978 240		2388	
4,4	3,723792		60 468		48
		1 038 708		2436	
4,5	4,762500		62 904		48
		1 101 612		2484	
4,6	5,864112		65 388		48
		1 167 000		2532	
4,7	7,031112		67 920		48
		1 234 920		2580	
4,8	8,266032		70 500		48
		1 305 420		2628	
4,9	9,571452		73 128		
		1 378 548			
5,0	10,950000				

Таблица 6.21

(1)	(2)	(3)	(4)	(5)
x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
4,0	0,1500			
		8 110		
4,1	0,9610		534	
		8 644		24
4,2	1,8254		558	
		9 202		22
4,3	2,7456		580	
		9 782		25
4,4	3,7238		605	
		10 387		24
4,5	4,7625		629	
		11 016		25
4,6	5,8641		654	
		11 670		25
4,7	7,0311		679	
		12 349		27
4,8	8,2660		706	
		13 055		24
4,9	9,5715		730	
		13 785		
5,0	10,9500			

Таблица 7.21

(1)	(2)	(3)	(4)
x	y	Δy	$\Delta^2 y$
4,0	0,15		
		81	
4,1	0,96		6
		87	
4,2	1,83		5
		92	
4,3	2,75		5
		97	
4,4	3,72		7
		104	
4,5	4,76		6
		110	
4,6	5,86		7
		117	
4,7	7,03		7
		124	
4,8	8,27		6
		130	
4,9	9,57		8
		138	
5,0	10,95		

точно даны значения функции, тем более простые интерполяционные формулы надо выбирать.

Необходимо еще иметь в виду, что на разностях высших порядков заметно сказываются ошибки округления. Сравнение разностей в табл. 6.21 и 7.21 с разностями в табл. 5.21 наглядно это поясняет.

Чтобы выяснить влияние шага таблицы на разности, рассмотрим таблицу значений того же многочлена с шагом $h=0{,}01$. Как показывает табл. 8.21, при таком шаге вторые разности оказываются практически постоянными даже для таблицы с пятью знаками.

Таблица 8.21

(1)	(2)	(3)	(4)
x	y	Δy	$\Delta^2 y$
4,00	0,15000		
		7876	
4,01	0,22876		51
		7927	
4,02	0,30803		51
		7978	
4,03	0,38781		53
		8031	
4,04	0,46812		52
		8083	
4,05	0,54895		52
		8135	
4,06	0,63030		

Если функция на рассматриваемом участке не имеет аналитических особенностей, то обычно ее разности изменяются плавно. Поэтому нарушение плавного изменения разностей позволяет в ряде случаев обнаружить ошибки в отдельных значениях функции, заданной с помощью таблицы.

Пусть одно из значений функции содержит ошибку, равную ε. Влияние этой ошибки на разности функции можно проследить по табл. 9.21. Из нее видно, что ошибка тем больше, чем выше порядок разности.

Таблица 9.21

(1)	(2)	(3)	(4)	(5)
x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
...	...	...	...	...
x_{n-2}	y_{n-2}			
		Δy_{n-2}		$\Delta^3 y_{n-3}+\varepsilon$
x_{n-1}	y_{n-1}		$\Delta^2 y_{n-2}+\varepsilon$	
		$\Delta y_{n-1}+\varepsilon$		$\Delta^3 y_{n-2}-3\varepsilon$
x_n	$y_n+\varepsilon$		$\Delta^2 y_{n-1}-2\varepsilon$	
		$\Delta y_n-\varepsilon$		$\Delta^3 y_{n-1}+3\varepsilon$
x_{n+1}	y_{n+1}		$\Delta^2 y_n+\varepsilon$	
		Δy_{n+1}		$\Delta^3 y_n-\varepsilon$
x_{n+2}	y_{n+2}		...	
...	...	...		

Если предположить, что вторые разности в рассматриваемой таблице практически постоянны всюду, кроме выделенного участка, то в качестве исправленного значения второй разности $\Delta^2 y_{n-1}$ можно принять среднее

арифметическое трех выписанных в табл. 9.21 вторых разностей, потому что сумма этих трех разностей уже не содержит ошибки. Таким образом, полагаем

$$\overline{\Delta^2 y_{n-1}} = (1/3)\,[(\Delta^2 y_{n-2} + \varepsilon) + (\Delta^2 y_{n-1} - 2\varepsilon) + (\Delta^2 y_n + \varepsilon)] = \\ = (\Delta^2 y_{n-2} + \Delta^2 y_{n-1} + \Delta^2 y_n)/3, \qquad (7.21)$$

где $\overline{\Delta^2 y_{n-1}}$ означает исправленное значение второй разности. Напомним, что вторые разности предполагаются постоянными.

Зная исправленное значение второй разности, легко найдем ошибку значения y_n, находящегося в той же горизонтальной строке, что и исправляемая вторая разность

$$\varepsilon = (\overline{\Delta^2 y_{n-1}} - \Delta^2 y_{n-1})/2. \qquad (8.21)$$

Ошибка выражается в целых единицах последнего разряда, как и значения функции и все разности.

Исправив ошибочное значение функции, следует снова пересчитать все разности. Нужно только иметь в виду, что таким путем могут быть обнаружены лишь отдельные ошибки, находящиеся в таблице далеко друг от друга. Кроме того, функция может иметь на рассматриваемом участке такие особенности (например, резко выраженный максимум), что ее разности вообще не будут иметь плавного течения.

Пример 4.21. Функция задана таблицей (табл. 10.21), в которой вторые разности можно считать практически постоянными, кроме участка, соответствующего значениям аргумента 0,25, 0,30, 0,35. Естественно считать, что значение y, соответствующее $x = 0{,}30$, содержит ошибку.

Исправленное значение второй разности найдем по формуле (7.21)

$$\overline{\Delta^2 y} = (48 + 9 + 48)/3 = 35$$

в единицах седьмого знака. Ошибка найдется по формуле (8.21)

$$\varepsilon = (35 - 9)/2 = 13$$

единиц седьмого знака. Таким образом, исправленное значение функции при $x = 0{,}30$ должно быть $y = 0{,}302\,2637$

Таблица 10.21

(1)	(2)	(3)	(4)
x	y	Δy	$\Delta^2 y$
0,00	0,292 8341		
		15 632	
0,05	0,294 3973		33
		15 665	
0,10	0,295 9638		34
		15 699	
0,15	0,297 5337		34
		15 733	
0,20	0,299 1070		33
		15 766	
0,25	0,300 6836		48
		15 814	
0,30	0,302 2650		9
		15 823	
0,35	0,303 8473		48
		15 871	
0,40	0,305 4344		35
		15 906	
0,45	0,307 0250		35
		15 941	
0,50	0,308 6191		

вместо указанного в таблице 0,302 2650. Пересчитывая разности (табл. 11.21), убеждаемся, что у вновь полученной таблицы вторые разности практически постоянны всюду.

Таблица 11.21

(1)	(2)	(3)	(4)
x	y	Δy	$\Delta^2 y$
0,20	0,299 1070		
		15 766	
0,25	0,300 6836		35
		15 801	
0,30	0,302 2637		35
		15 836	
0,35	0,303 8473		35
		15 871	
0,40	0,305 4344		

Рассмотренный прием позволяет исправлять отдельные ошибки функции также и в том случае, когда постоянны третьи разности.

В заключение отметим связь между конечными разностями и производными. Из определения производной следует, что

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = \lim_{h \to 0} \frac{\Delta f(x)}{h}.$$

Отсюда вытекает, что при малых h имеем $f'(x) \approx \frac{\Delta f(x)}{h}$, или, в более короткой записи,

$$y' \approx \Delta y / h. \qquad (9.21)$$

Найдем теперь

$$\lim_{h\to 0}\frac{\Delta^2 y}{h^2}=\lim_{h\to 0}\frac{f(x+2h)-2f(x+h)+f(x)}{h^2}.$$

Считая функцию $f(x)$ дважды непрерывно дифференцируемой, применим два раза правило Лопиталя (при постоянном x и переменном h). Тогда

$$\lim_{h\to 0}\frac{f(x+2h)-2f(x+h)+f(x)}{h^2}=\lim_{h\to 0}\frac{f'(x+2h)\cdot 2-2f'(x+h)}{2h}=$$

$$=\lim_{h\to 0}\frac{f''(x+2h)\cdot 2-f''(x+h)}{1}=f''(x),$$

т. е.

$$\lim_{h\to 0}\frac{\Delta^2 y}{h^2}=f''(x).$$

Отсюда при малых h будем иметь

$$y''\approx\Delta^2 y/h^2.$$

Рассуждая аналогичным образом и дальше, получим приближенную формулу для любого n (предполагая, что функция $y=f(x)$ имеет n-ю непрерывную производную)

$$y^{(n)}\approx\Delta^n y/h^n. \qquad (10.21)$$

Формулы (9.21) и (10.21) могут быть использованы для приближенного нахождения производных. Однако их точность весьма невелика (особенно при $n>1$), поэтому на практике применяются более совершенные приемы.

§ 22. Точность линейной интерполяции. Квадратичная интерполяция. Интерполяция по схеме Эйткина

В § 5 первого выпуска мы рассматривали линейную интерполяцию, суть которой, как там было сказано, состоит в том, что на участке между двумя табличными значениями аргумента мы считаем функцию изменяющейся линейно. Пользуясь рассмотренными свойствами разностей функции, мы можем теперь сказать, что линейная интерполяция предполагает на рассматриваемом участке постоянство первых разностей. Во всех тех случаях, когда первые разности изменяются, линейная интерполяция будет давать некоторую погрешность.

Ясно, что речь идет не о точном постоянстве первых разностей, что возможно, как об этом было сказано в предыдущем параграфе, только в случае линейной функции, а о том, чтобы первые разности были практически постоянными. Во всех случаях, когда функция отлична от линейной и первые разности не являются точно постоянными, линейная интерполяция приводит к погрешностям в определении промежуточных значений функции. Эта погрешность будет тем больше, чем больше различаются между собой разности функции, т. е. чем больше ее вторые разности.

Общее рассмотрение вопросов оценки погрешности интерполяции мы отложим до § 25. Здесь мы ограничимся лишь приведенным без доказательства важным правилом, позволяющим оценить точность линейной интерполяции. Это правило гласит:

абсолютная ошибка линейной интерполяции по таблице с постоянным шагом и практически постоянными вторыми разностями не превосходит восьмой части абсолютной величины второй разности.

Из этого правила мы заключаем, что линейная интерполяция дает приблизительно на один верный знак больше, чем величина второй разности. Это означает, что если, например, вторая разность есть величина порядка десяти единиц последнего знака таблицы, то линейная интерполяция дает точность порядка единицы последнего знака. Если вторая разность существенно больше, то линейная интерполяция может оказаться уже непригодной. Она будет тем более непригодной, если вторые разности на рассматриваемом участке таблицы заметно изменяются. В таких случаях требуется интерполяция более высокой степени.

Наиболее часто используемым способом параболической интерполяции более высокой степени является *квадратичная интерполяция*, для которой требуются значения функции уже не в двух, а в трех точках. Геометрический смысл ее состоит в том, что через выбранные три точки проводится парабола вида $y = Ax^2 + Bx + C$ и принимается, что значения функции на рассматриваемом участке могут быть вычислены как значения построенного квадратного

трехчлена. Коэффициенты A, B, C трехчлена находятся из условия прохождения параболы через заданные точки.

При практическом применении квадратичной интерполяции, даже если не пользоваться специальными формулами, речь о которых будет идти в следующем параграфе, нет никакой нужды в действительном вычислении коэффициентов квадратного трехчлена.

Для вычисления значений функции, соответствующих промежуточным значениям аргумента, при квадратичной интерполяции удобно пользоваться схемой Эйткина, аналогичной схеме линейной интерполяции, рассмотренной в § 5 первого выпуска.

Пусть в точках x_0, x_1 заданы значения функции y_0, y_1. Линейная интерполяция по схеме Эйткина сводится к вычислению определителя второго порядка

$$P_{0,1}(x)=\frac{1}{x_1-x_0}\begin{vmatrix} y_0 & x_0-x \\ y_1 & x_1-x \end{vmatrix}.$$

Действительно, из выражения для $P_{0,1}(x)$ ясно, что это многочлен первой степени. Далее путем непосредственной подстановки значений читатель легко убедится в том, что $P_{0,1}(x_0)=y_0$ и $P_{0,1}(x_1)=y_1$.

Пусть теперь даны три значения аргумента x_0, x_1, x_2, при которых функция принимает соответственно значения y_0, y_1, y_2. Линейная интерполяция на каждом из этих участков осуществляется выражениями вида

$$P_{0,1}(x)=\frac{1}{x_1-x_0}\begin{vmatrix} y_0 & x_0-x \\ y_1 & x_1-x \end{vmatrix}, \tag{1.22}$$

$$P_{1,2}(x)=\frac{1}{x_2-x_1}\begin{vmatrix} y_1 & x_1-x \\ y_2 & x_2-x \end{vmatrix}. \tag{2.22}$$

Значение функции в точке x, соответствующее квадратичной интерполяции по указанным трем точкам, получается путем линейной интерполяции между значениями этих линейных выражений, т. е. по формуле

$$P_{0,1,2}(x)=\frac{1}{x_2-x_0}\begin{vmatrix} P_{0,1}(x) & x_0-x \\ P_{1,2}(x) & x_2-x \end{vmatrix}. \tag{3.22}$$

Действительно, так как $P_{0,1}$ и $P_{1,2}$ являются, как это видно из их выражений, многочленами первой степени, то $P_{0,1,2}$ — второй степени относительно x. Далее, по построению $P_{0,1}(x)$ и $P_{1,2}(x)$ имеем

$$P_{0,1}(x_0)=y_0; \quad P_{0,1}(x_1)=y_1; \quad P_{1,2}(x_1)=y_1; \quad P_{1,2}(x_2)=y_2.$$

Поэтому

$$P_{0,1,2}(x_0)=\frac{1}{x_2-x_0}\begin{vmatrix} P_{0,1}(x_0) & 0 \\ P_{1,2}(x_0) & x_2-x_0 \end{vmatrix}=P_{0,1}(x_0)=y_0.$$

Аналогично,

$$P_{0,1,2}(x_2)=\frac{1}{x_2-x_0}\begin{vmatrix} P_{0,1}(x_2) & x_0-x_2 \\ P_{1,2}(x_2) & 0 \end{vmatrix}=P_{1,2}(x_2)=y_2.$$

Кроме того, при $x=x_1$

$$P_{0,1,2}(x_1)=\frac{1}{x_2-x_0}\begin{vmatrix} P_{0,1}(x_1) & x_0-x_1 \\ P_{1,2}(x_1) & x_2-x_1 \end{vmatrix}=\frac{1}{x_2-x_0}\begin{vmatrix} y_1 & x_0-x_1 \\ y_1 & x_2-x_1 \end{vmatrix}=$$

$$=\frac{y_1}{x_2-x_0}[(x_2-x_1)-(x_0-x_1)]=y_1.$$

Итак, мы получили, что многочлен $P_{0,1,2}(x)$ принимает в точках x_0, x_1, x_2 соответственно значения y_0, y_1, y_2, так что он и дает требуемую нам параболу.

Пример 1.22. Функция $y=f(x)$ задана в таблице 1.22. С помощью квадратичной интерполяции по схеме Эйткина найдем значение функции $f(13{,}13)$. Шаг таблицы равен здесь $h=0{,}05$. За x_0 примем значение $x_0=13{,}10$.

Таблица 1.22

(1)	(2)
x	$f(x)$
13,00	1,49936 17229
13,05	1,50106 09921
13,10	1,50292 30057
13,15	1,50494 18872
13,20	1,50711 14191
13,25	1,50942 50600

На участке (13,10; 13,15) находим

$$P_{0,1}=\frac{1}{0{,}05}\begin{vmatrix}1{,}50292\,30057 & -0{,}03\\ 1{,}50494\,18872 & 0{,}02\end{vmatrix}=1{,}50413\,43346.$$

Это значение является искомым значением функции, полученным линейной интерполяцией. Для следующего участка

$$P_{1,2}=\frac{1}{0{,}05}\begin{vmatrix}1{,}50494\,18872 & 0{,}02\\ 1{,}50711\,14191 & 0{,}07\end{vmatrix}=1{,}50407\,40744.$$

Теперь для квадратичной интерполяции по формуле (3.22) получаем

$$P_{0,1,2}=\frac{1}{0{,}1}\begin{vmatrix}1{,}50413\,43346 & -0{,}03\\ 1{,}50407\,40744 & 0{,}07\end{vmatrix}=1{,}50411\,62565.$$

Это и есть искомое значение функции. Точное значение функции, взятое из более подробной таблицы, равно $f(13{,}13)=1{,}50411\,59009$. Сравнение результатов, полученных линейной и квадратичной интерполяцией с точным значением, показывает, что квадратичная интерполяция сохраняет здесь верными семь значащих цифр, тогда как линейная интерполяция только пять.

Рассмотренная здесь схема Эйткина может применяться не только для квадратичной интерполяции, но и для параболической интерполяции более высоких степеней. Например, если функция задана в четырех точках, то для интерполяции третьей степени можно воспользоваться формулой

$$P_{0,1,2,3}(x)=\frac{1}{x_3-x_0}\begin{vmatrix}P_{0,1,2}(x) & x_0-x\\ P_{1,2,3}(x) & x_3-x\end{vmatrix}, \qquad (4.22)$$

где $P_{0,1,2}$ и $P_{1,2,3}$ получены по формулам типа (3.22).

§ 23. Интерполяционные формулы Лагранжа и Ньютона

Задача параболической интерполяции в том общем виде, как она была поставлена в § 20, решается с помощью различных *интерполяционных формул*. Как уже было сказано, эту задачу можно сформулировать следующим образом: *рассматривается функция* $f(x)$, *для*

которой заданы значения $y_i = f(x_i)$ $(i = 0, 1, 2, \ldots, n)$, *причем все* x_i *и* y_i *известны. Требуется определить многочлен* $y = F(x)$ *степени* n, *для которого выполняется равенство:* $F(x_i) = f(x_i)$ $(i = 0, 1, 2, \ldots, n)$.

Мы можем написать

$$F(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n. \tag{1.23}$$

Используя написанные выше условия, получим для нахождения неизвестных коэффициентов $a_0, a_1, a_2, \ldots, a_n$ систему $n+1$ уравнений с $n+1$ неизвестными:

$$\left.\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= y_0, \\ a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n &= y_1, \\ \cdots\cdots\cdots\cdots\cdots\cdots & \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= y_n. \end{aligned}\right\} \tag{2.23}$$

Дальше будет показано, что эта система имеет единственное решение, если значения $x_0, x_1, \ldots, x_n$ отличны друг от друга (см. стр. 190); определив значения коэффициентов a_i из системы (2.23), получим интерполяционный многочлен $F(x)$, дающий решение поставленной задачи.

Перейдем теперь к решению общей задачи отыскания интерполяционного многочлена. Вместо того чтобы решать систему уравнений (2.23), мы поступим иначе, непосредственно построив многочлен $F(x)$, удовлетворяющий всем поставленным условиям.

Найдем прежде всего выражение для многочлена, принимающего в точке $x = x_0$ значение $y_0 = 1$, а в точках $x = x_1$, $x = x_2, \ldots,$ $x = x_n$ — значения $y_1 = y_2 = \ldots = y_n = 0$. Легко проверить, что такой многочлен будет иметь вид

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

В самом деле, значения $x = x_1$, $x = x_2, \ldots,$ $x = x_n$ являются корнями нашего многочлена, а при $x = x_0$ числитель равен знаменателю.

Теперь построим многочлен $y = F_0(x)$, принимающий в точке $x = x_0$ значение y_0 и обращающийся в нуль для значений $x = x_i$ $(i = 1, 2, \ldots, n)$. Учитывая предыдущее построение, легко заметить, что многочлен $F_0(x)$ должен

иметь вид

$$F_0(x)=\frac{(x-x_1)(x-x_2)\dots(x-x_n)}{(x_0-x_1)(x_0-x_2)\dots(x_0-x_n)}y_0.$$

После этих предварительных построений можно перейти к нахождению многочлена $F(x)$, принимающего в точках $x=x_i$ $(i=0, 1, 2, \dots, n)$ заданные значения $F(x_i)=y_i$ $(i=0, 1, 2, \dots, n)$. Для этого определим при фиксированном j $(0\leqslant j\leqslant n)$ многочлен $F_j(x)$, принимающий в точке $x=x_j$ значение $F_j(x_j)=y_j=F(x_j)$, а во всех остальных точках $x=x_i$ $(i=0, 1, 2, \dots, n, i\neq j)$ — значения $F_j(x_i)=0$. Это будет многочлен

$$F_j(x)=\frac{(x-x_0)(x-x_1)\dots(x-x_{j-1})(x-x_{j+1})\dots(x-x_n)}{(x_j-x_0)(x_j-x_1)\dots(x_j-x_{j-1})(x_j-x_{j+1})\dots(x_j-x_n)}y_j.$$

Искомый многочлен будет равен сумме

$$F(x)=\sum_{j=0}^{n}F_j(x),$$

так как в каждой точке x_j одно из слагаемых принимает нужное значение y_j, а все остальные слагаемые обращаются в нуль. Итак, многочлен, удовлетворяющий условиям поставленной задачи, найден. Подставляя вместо $F_j(x)$ их выражения, получим

$$F(x)=\sum_{j=0}^{n}\frac{(x-x_0)(x-x_1)\dots(x-x_{j-1})(x-x_{j+1})\dots(x-x_n)}{(x_j-x_0)(x_j-x_1)\dots(x_j-x_{j-1})(x_j-x_{j+1})\dots(x_j-x_n)}y_j, \tag{3.23}$$

или, в развернутом виде,

$$\begin{aligned}F(x)=&\frac{(x-x_1)(x-x_2)\dots(x-x_n)}{(x_0-x_1)(x_0-x_2)\dots(x_0-x_n)}y_0+\\&+\frac{(x-x_0)(x-x_2)\dots(x-x_n)}{(x_1-x_0)(x_1-x_2)\dots(x_1-x_n)}y_1+\dots+\\&+\frac{(x-x_0)(x-x_1)\dots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\dots(x_n-x_{n-1})}y_n.\end{aligned} \tag{4.23}$$

Полученная формула называется *интерполяционной формулой Лагранжа.*

Покажем теперь, что интерполяционный многочлен Лагранжа является единственным решением поставленной задачи. Действительно, пусть существует еще один многочлен $R(x)$ степени n, который принимает в заданных точках заданные значения. Тогда разность $F(x)-R(x)$ представляет собой многочлен степени не выше n, который обращается в нуль в точках $x=x_i$ $(i=0, 1, \ldots, n)$, т. е. имеет $n+1$ корень. Отсюда следует, что эта разность равна нулю тождественно, так как многочлен степени не выше n не может иметь $n+1$ корень.

Мы заключаем, таким образом, что, *каковы бы ни были значения* $x_0, x_1, \ldots, x_n$, *среди которых нет совпадающих, и совершенно произвольные значения* $y_0, y_1, \ldots, y_n$, *существует единственный многочлен* $F(x)$ *степени* n, *принимающий в заданных точках заданные значения, т. е. удовлетворяющий условиям* $F(x_i)=y_i$ $(i=0, 1, 2, \ldots, n)$.

Пример 1.23. Положим $n=1$. Ясно, что мы имеем в этом случае две точки и интерполяционная формула Лагранжа дает уравнение прямой, проходящей через две заданные точки. Обозначив абсциссы этих точек через x_0 и x_1, получим интерполяционный многочлен в виде

$$F(x)=\frac{x-x_1}{x_0-x_1}y_0+\frac{x-x_0}{x_1-x_0}y_1. \tag{5.23}$$

Очевидно, что он совпадает с многочленом $P_{0,1}(x)$, построенным по схеме Эйткина.

Пример 2.23. Примем $n=2$. Тогда мы получим уравнение параболы, проходящей через три точки. Оно будет иметь вид

$$F(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0+\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1+ \\ +\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2. \tag{6.23}$$

Читатель легко может убедиться, что формула (3.22) для многочлена $P_{0,1,2}(x)$, осуществляющего квадратичную интерполяцию, совпадает с формулой (6.23), так же как формула (1.22) для линейной интерполяции совпадает с формулой (5.23). Следовательно, приведенные в пре-

дыдущем параграфе формулы являются частными случаями интерполяционной формулы Лагранжа.

Интерполяционные формулы Ньютона, к рассмотрению которых мы переходим, предназначены для решения той же общей интерполяционной задачи, что и формула Лагранжа. При их выводе сделаем дополнительное предположение, что рассматриваются *равноотстоящие значения аргумента*.

Допустим, что значения функции $y=f(x)$ заданы для равноотстоящих значений аргумента x_0, $x_1=x_0+h$, $x_2=x_0+2h$, ..., $x_n=x_0+nh$. Обозначим эти значения соответственно $y_0=f(x_0)$, $y_1=f(x_1)$, ..., $y_n=f(x_n)$.

Как было показано выше, существует единственный многочлен $F(x)$ степени n, для которого $F(x_k)=y_k$ ($k=0, 1, \ldots, n$). Сейчас речь пойдет о другом способе записи и отыскания этого многочлена, который в конечном счете совпадает с многочленом, полученным по формуле Лагранжа. Запишем искомый многочлен в виде

$$\begin{aligned} F(x)=a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+\\ +a_3(x-x_0)(x-x_1)(x-x_2)+\ldots\\ \ldots+a_n(x-x_0)(x-x_1)\ldots(x-x_{n-1}). \end{aligned} \tag{7.23}$$

Для определения коэффициентов a_0, a_1, ..., a_n положим в (7.23) $x=x_0$. Тогда

$$y_0=F(x_0)=a_0.$$

Далее, полагая $x=x_1$, получаем

$$y_1=F(x_1)=a_0+a_1(x_1-x_0),$$

или, так как $x_1-x_0=h$,

$$y_1=y_0+a_1h,$$

откуда

$$a_1=(y_1-y_0)/h=\Delta y_0/h.$$

Продолжая вычисление коэффициентов, положим $x=x_2$. Тогда

$$y_2=F(x_2)=a_0+a_1(x_2-x_0)+a_2(x_2-x_0)(x_2-x_1).$$

Заменим найденные коэффициенты a_0, a_1 их значениями

$$y_2 = y_0 + (\Delta y_0/h) \cdot 2h + a_2 \cdot 2h \cdot h;$$

тогда

$$y_2 - 2\Delta y_0 - y_0 = y_2 - 2y_1 + y_0 = 2a_2h^2.$$

Воспользовавшись формулой (2.21), выражающей разности через значения функции, получим

$$a_2 = \frac{\Delta^2 y_0}{2!\, h^2}.$$

Точно так же получим

$$a_3 = \frac{\Delta^3 y_0}{3!\, h^3}.$$

Аналогичные дальнейшие вычисления позволяют записать общую формулу для нахождения коэффициентов

$$a_k = \frac{\Delta^k y_0}{k!\, h^k}.$$

Подставив найденные выражения коэффициентов в формулу (7.23), находим

$$F(x) = y_0 + \frac{\Delta y_0}{1!\, h}(x - x_0) + \frac{\Delta^2 y_0}{2!\, h^2}(x - x_0)(x - x_1) + \\ + \frac{\Delta^3 y_0}{3!\, h^3}(x - x_0)(x - x_1)(x - x_2) + \ldots + \\ + \frac{\Delta^n y_0}{n!\, h^n}(x - x_0)(x - x_1)\ldots(x - x_{n-1}). \qquad (8.23)$$

Полученную формулу и называют *первой интерполяционной формулой Ньютона.*

Теперь ясно видно различие между формулами Ньютона и Лагранжа. В формуле Лагранжа (4.23) каждое из слагаемых представляет многочлен n-й степени и все эти слагаемые равноправны. Поэтому мы не можем заранее (т. е. до вычислений) пренебрегать какими-либо из них. В формулу же Ньютона входят в качестве слагаемых многочлены повышающихся степеней, причем коэффициентами при них служат последовательные конечные разности, деленные на факториалы. Как мы уже видели в § 21, последовательные разности обычно

довольно быстро уменьшаются. Мы получаем поэтому возможность не учитывать в формуле Ньютона тех слагаемых, коэффициенты при которых становятся пренебрежимо малыми. Благодаря этому можно вычислять промежуточные значения функции достаточно точно, пользуясь простыми интерполяционными формулами.

Для практического использования формулу Ньютона (8.23) обычно записывают в несколько преобразованном виде. Чтобы получить его, введем обозначение

$$(x-x_0)/h=t \quad \text{или} \quad x=x_0+th.$$

Множители, входящие в формулу (8.23), выразятся через t следующим образом:

$$(x-x_1)/h=(x-x_0-h)/h=t-1,$$
$$(x-x_2)/h=(x-x_0-2h)/h=t-2.$$
$$\cdots\cdots\cdots\cdots\cdots\cdots$$
$$(x-x_{n-1})/h=(x-x_0-(n-1)\,h)/h=t-n+1.$$

Подставив эти выражения в формулу (8.23), приведем ее к виду

$$F(x_0+th)=y_0+\frac{t}{1!}\Delta y_0+\frac{t(t-1)}{2!}\Delta^2 y_0+\frac{t(t-1)(t-2)}{3!}\Delta^3 y_0+$$
$$+\ldots+\frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0. \qquad (9.23)$$

Это и есть окончательный вид первой интерполяционной формулы Ньютона. По причинам, которые будут указаны ниже, ее называют также *интерполяционной формулой Ньютона для интерполирования вперед*.

Если в формуле (9.23) положить t равным целому числу $k\,(k\leqslant n)$, то правая часть формулы будет содержать $k+1$ слагаемое, причем последнее из них будет

$$\frac{k(k-1)\ldots(k-k+1)}{k!}\Delta^k y_0=\Delta^k y_0.$$

Коэффициенты при всех последующих слагаемых будут содержать множитель $(k-k)$ и, следовательно, обратятся в нуль. Формула для $F(x_0+kh)$ примет тогда в точности тот же вид, что и формула (3.21), выражающая y_k через значение y_0 и последовательные разности.

Следовательно,

$$F(x_0+kh)=y_k.$$

Впрочем, последнее равенство ясно и из хода вывода формулы (3.21).

Если же k не равно целому числу, а этот случай и является наиболее интересным, то формула (9.23) дает значения функции $F(x)$ для значений аргумента, отсутствующих в исходной таблице, т. е. для значений x, не совпадающих ни с одним из x_k ($k=0, 1, \ldots, n$).

Формулу (9.23), как и формулу (3.21), можно записать в символической форме

$$F(x_0+th)=(1+\Delta)^t y_0, \qquad (10.23)$$

где выражение $(1+\Delta)^t$ нужно разложить в биномиальный ряд и оборвать его на члене, содержащем Δ^n, а каждое произведение Δ^r на y_0 заменить разностью $\Delta^r y_0$.

Формулой (9.23) следует пользоваться при вычислении значений функции $f(x)$ для значений аргумента, лежащих между x_0 и x_1, т. е. для $t<1$. Переходя к интервалу $x_1<x<x_2$, нецелесообразно брать ту же формулу; удобнее принимать за x_0 следующий узел интерполяции.

Приведем теперь пример применения интерполяционной формулы Ньютона.

Пример 3.23. По данной таблице значений семизначных логарифмов для чисел от 1000 до 1050 с шагом 10 (табл. 1.23) вычислить значения логарифмов всех целых чисел от 1000 до 1010.

Составленная таблица разностей, записанная как диагональная, показывает, что третьи разности функции можно считать практически постоянными. Полагаем $x_0=1000$, $y_0=3{,}0000000$, $\Delta y_0=0{,}0043214$, $\Delta^2 y_0=-0{,}0000426$, $\Delta^3 y_0=0{,}0000008$. Остается определить значение t. Так как здесь $h=10$, то для $x=1001$ имеем $t_1=0{,}1$, для $x=1002$ будет $t_2=0{,}2$ и т. д.

Для определения искомых значений y при выбранных t удобно расположить требуемые вычисления в таблицу (табл. 2.23). При этом промежуточные вычисления проводятся с одним запасным знаком, а окончательные результаты округлены.

Таблица 1.23

(1)	(2)	(3)	(4)	(5)
x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
1000	3,000 0000			
		43 214		
1010	3,004 3214		—426	
		42 788		8
1020	3,008 6002		—418	
		42 370		9
1030	3,012 8372		—409	
		41 961		8
1040	3,017 0333		—401	
		41 560		
1050	3,021 1893			

Проверив полученные значения по семизначной таблице логарифмов, убедимся, что полученные значения верны до последнего знака.

Для составления таблицы значений в интервале $1010 < x < 1020$ полагаем $x_0 = 1010$. Тогда $y_0 = 3{,}0043214$, $\Delta y_0 = 0{,}0042788$, $\Delta^2 y_0 = -0{,}0000418$, $\Delta^3 y_0 = 0{,}0000009$; далее поступаем так же, как и ранее.

Как видно из рассмотренного примера, в интерполяционной формуле (9.23) используются разности, идущие по диагонали вниз. Поэтому применение этой формулы удобно в начале таблицы, где имеется достаточное число разностей.

Наоборот, она оказывается непригодной в конце таблицы, где этих разностей слишком мало.

Таблица 2.23

(1)	(2)	(3)	(4)	(5)	(6)
t	x	$t\Delta y_0$	$\frac{t(t-1)}{2}\Delta^2 y_0$	$\frac{t(t-1)(t-2)}{6}\times$ $\times\Delta^3 y_0$	$y=y_0+(3)+$ $+(4)+(5)$
0	1000	0	0	0	3,000 0000
0,1	1001	43 214	192	2	3,000 4341
0,2	1002	86 428	341	4	3,000 8677
0,3	1003	129 642	447	5	3,001 3009
0,4	1004	172 856	511	5	3,001 7337
0,5	1005	216 070	532	5	3,002 1661
0,6	1006	259 284	511	4	3,002 5980
0,7	1007	302 498	447	4	3,003 0295
0,8	1008	345 712	341	3	3,003 4606
0,9	1009	388 926	192	1	3,003 8912
1,0	1010	432 140	0	0	3,004 3214

Так, в том же примере для $x_0=1030$ имеется только первая и вторая разности, а для $x_0=1040$ — только первая.

Таким образом, первая интерполяционная формула Ньютона наиболее удобна для отыскания значений функции, соответствующих бóльшим, нежели начальные, значениям аргумента, чем и объясняется приведенное выше ее другое название — интерполяционная формула для интерполирования вперед.

Для интерполирования в конце таблицы применяется иная формула, которую мы сейчас и рассмотрим. Напишем искомый интерполяционный многочлен в форме

$$F(x)=a_0+a_1(x-x_n)+a_2(x-x_n)(x-x_{n-1})+\ldots+ \\ +a_n(x-x_n)(x-x_{n-1})\ldots(x-x_1). \qquad (11.23)$$

Как и выше, коэффициенты a_0, a_1, ..., a_n определяются из условия $y_i=f(x_i)=F(x_i)$. Положим в (11.23) $x=x_n$. Тогда

$$a_0=y_n.$$

Точно так же, полагая $x=x_{n-1}$, имеем

$$y_{n-1}=y_n+a_1(x_{n-1}-x_n),$$

а так как $x_{n-1}-x_n=-h$, то

$$a_1=(y_n-y_{n-1})/h=\Delta y_{n-1}/h.$$

Далее, полагая в (11.23) $x=x_{n-2}$ и заменяя найденные коэффициенты a_0, a_1 их значениями, получаем

$$a_2=(y_n-2y_{n-1}+y_{n-2})/2h^2=\Delta^2 y_{n-2}/(2!\,h^2).$$

Продолжая аналогичные вычисления, получим общую формулу для коэффициентов

$$a_k=\frac{\Delta^k y_{n-k}}{k!\,h^k} \qquad (k=1,\ 2,\ \ldots,\ n).$$

После подстановки в (11.23) полученных значений коэффициентов формула примет вид

$$F(x)=y_n+\frac{\Delta y_{n-1}}{1!\,h}(x-x_n)+\frac{\Delta^2 y_{n-2}}{2!\,h^2}(x-x_n)(x-x_{n-1})+\ldots$$
$$\ldots+\frac{\Delta^n y_0}{n!\,h^n}(x-x_n)(x-x_{n-1})\ldots(x-x_1). \qquad (12.23)$$

Это и есть *вторая интерполяционная формула Ньютона.* Для применения, однако, ее предварительно преобразуют, как и первую. Введем обозначение

$$(x-x_n)/h=t \quad \text{или} \quad x=x_n+th.$$

Выразим теперь через t множители в формуле (12.23):

$$(x-x_{n-1})/h=(x-(x_n-h))/h=t+1,$$
$$(x-x_{n-2})/h=(x-(x_n-2h))/h=t+2,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$(x-x_1)/h=(x-(x_n-(n-1)h))/h=t+n-1.$$

Произведя такую замену, получим окончательно

$$F(x)=F(x_n+th)=y_n+t\Delta y_{n-1}+\frac{t(t+1)}{2!}\Delta^2 y_{n-2}+\ldots$$
$$\ldots+\frac{t(t+1)\ldots(t+n-1)}{n!}\Delta^n y_0. \qquad (13.23)$$

Формулу (13.23) называют *второй интерполяционной формулой Ньютона* или *интерполяционной формулой Ньютона для интерполирования назад.* Приведем пример использования формулы (13.23) для интерполяции.

Пример 4.23. По таблице, данной в примере 3.23 (табл. 1.23), найти значения логарифма для $x=1044$. Примем $x_n=1050$, $y_n=3{,}0211893$, $\Delta y_{n-1}=0{,}0041560$, $\Delta^2 y_{n-2}=-0{,}0000401$, $\Delta^3 y_{n-3}=0{,}0000008$. Для $x=1044$ получаем отрицательное значение $t=-0{,}6$. По формуле (13.23)

$$F(1044)=3{,}0211893+(-0{,}6)\cdot 0{,}004\ 1560+$$
$$+\frac{(-0{,}6)(-0{,}6+1)}{2}(-0{,}0000401)+$$
$$+\frac{(-0{,}6)(-0{,}6+1)(-0{,}6+2)}{2\cdot 3}0{,}0000008.$$

Произведя вычисления, получим

$$F(1044)=3{,}018\,7005.$$

Проверка по семизначным таблицам логарифмов показывает, что все семь знаков верные.

§ 24. Экстраполяция. Обратная интерполяция

В предыдущем параграфе были приведены примеры применения интерполяционных формул для отыскания значений функции, соответствующих промежуточным значениям аргумента, отсутствующим в таблице.

Эти же формулы могут быть использованы и для нахождения значений функции, соответствующих значениям аргумента, находящимся вне пределов таблицы, т. е. для *экстраполяции*.

Применение интерполяционных формул для экстраполяции ничем не отличается от рассмотренного в предыдущих примерах. Единственным различием является то, что при интерполяции по первой формуле Ньютона значение t оказывается положительным, а при экстраполяции — отрицательным. Для второй формулы Ньютона, наоборот, при интерполяции значение t отрицательно, а при экстраполяции — положительно.

Таким образом, *первая интерполяционная формула Ньютона применяется для интерполирования вперед и экстраполирования назад, а вторая — для интерполирования назад и экстраполирования вперед.*

Таблица 1.24

(1)	(2)	(3)	(4)	(5)
x	$\cos x$	Δ	Δ^2	Δ^3
65°00′	0,4226 1826			
		−52 7981		
65°20′	0,4173 3845		−1412	
		−52 9393		17
65°40′	0,4120 4452		−1395	
		−53 0788		19
66°00′	0,4067 3664		−1376	
		−53 2164		17
66°20′	0,4014 1500		−1359	
		−53 3523		18
66°40′	0,3960 7977		−1341	
		−53 4864		
67°00′	0,3907 3113			

Пример 1.24. Дана таблица значений функции

$$y = \cos x$$

с шагом 20′ с точностью до восьми знаков в пределах от 65° до 67° (табл. 1.24). Вычислить значения $\cos x$ для углов 64°40′ до 65° с шагом 5′.

Разности мы записали в диагональную таблицу в той же табл. 1.24. Как показывают вычисления, уже третьи разности можно считать практически постоянными.

Вычисления экстраполированных значений приведены в табл. 2.24. В той же таблице в последней колонке

Таблица 2.24

(1)	(2)	(3)	(4)
t	$t(t-1)/2$	$t(t-1)(t-2)/6$	(1) Δ
—0,25	0,15625	—0,117 1875	13 1995 2
—0,50	0,37500	—0,312 5000	26 3990 5
—0,75	0,65625	—0,601 5625	39 5985 8
—1	1	—1	52 7981 0

(5)	(6)	(7)	(8)
(2) Δ^2	(3) Δ^3	$y = y_0 + (4) + (5) + (6)$	$y = \cos x$
—220 6	—2 0	0,4239 3599	0,4239 3599
—529 5	—5 3	0,4252 5282	0,4252 5281
—926 6	—102	0,4265 6876	0,4265 6874
—1412 0	—17 0	0,4278 8378	0,4278 8376

справа приведены точные значения косинуса для соответствующих углов, взятые из более подробной таблицы. Сравнение этих значений со значениями, полученными экстраполяцией, показывает, что уже при экстраполяции больше чем на половину шага таблицы ошибка составляет две единицы последнего (восьмого) знака функции.

Экстраполяция по второй интерполяционной формуле Ньютона производится точно так же. Что касается формулы Лагранжа, то для вычисления значения функции при любом x (т. е. и для интерполирования, и для экстраполирования) требуется только подставить в формулу (4.23) соответствующее значение аргумента.

Отметим, что экстраполяция, вообще говоря, дает бо́льшие ошибки, нежели интерполяция, и пределы ее применения ограничены.

До сих пор рассматривались лишь задачи нахождения значений функции, соответствующих данным значе-

ниям аргумента, отсутствующим в таблице. Между тем, нередко приходится сталкиваться и с задачами иного, обратного характера: *по таблице функции отыскать значение аргумента x, которому соответствует данное значение функции, отсутствующее в таблице.* Так поставленную задачу называют *задачей обратной интерполяции.*

Таблица 3.24

x	$f(x)$
3,7	0,2160494
3,8	0,2129001
3,9	0,2098875
4,0	0,2070019
4,1	0,2042345

Таблица 4.24

x	$\varphi(x)$
0,2129001	3,8
0,2098875	3,9
0,2070019	4,0

Задачу обратной интерполяции можно легко обратить, считая значения функции, наоборот, значениями аргумента. Однако, так как разности функции не постоянны, то обратная интерполяция приводит к необходимости интерполировать по таблице, значения аргумента в которой не являются равноотстоящими. По этой причине для обратной интерполяции применяется обычно интерполяционная формула Лагранжа.

Пример 2.24. Функция $y=f(x)$ задана таблицей (табл. 3.24).

Определим, какому аргументу соответствует значение функции 0,2116793.

Как видно из табл. 3.24, значение аргумента заключено между 3,8 и 3,9. Поэтому задачу нахождения аргумента можно считать задачей интерполяции в таблице функции с переменным шагом, приведенной в табл. 4.24. Здесь мы поменяли местами функцию и аргумент.

Пользуясь формулой Лагранжа для $x=0{,}2116793$, находим

$$y=\frac{(0{,}2116793-0{,}2098875)\,(0{,}2116793-0{,}2070019)}{(0{,}2129001-0{,}2098875)\,(0{,}2129001-0{,}2070019)}\cdot 3{,}8+$$
$$+\frac{(0{,}2116793-0{,}2129001)\,(0{,}2116793-0{,}2070019)}{(0{,}2098875-0{,}2129001)\,(0{,}2098875-0{,}2070019)}\cdot 3{,}9+$$
$$+\frac{(0{,}2116793-0{,}2129001)\,(0{,}2116793-0{,}2098875)}{(0{,}2070019-0{,}2129001)\,(0{,}2070019-0{,}2098875)}\cdot 4{,}0=3{,}8400.$$

Таким образом, значение $f(x)=0{,}2116793$ следует считать соответствующим значению аргумента $x=3{,}8400$.

Можно получить тот же результат, пользуясь вычислениями по схеме Эйткина. Рекомендуем читателю проделать соответствующие вычисления самостоятельно. Для их контроля приводим значения многочленов первой степени

$$P_{0,1}=3{,}8405 \quad \text{и} \quad P_{1,2}=3{,}8379.$$

§ 25. Оценка точности интерполяционных формул

Замена функции интерполяционным многочленом, полученным с помощью какой-либо интерполяционной формулы, служит источником большого числа приближенных формул вычислительной математики. Оценка погрешности этих формул требует, в свою очередь, оценки погрешности, происходящей при замене функции интерполяционным многочленом. Рассмотрением этого вопроса мы и займемся в настоящем параграфе.

Как было указано в § 20, задача параболической интерполяции состоит в нахождении для функции $f(x)$ многочлена $F(x)$ степени n, удовлетворяющего равенствам $F(x_i)=f(x_i)$ $(i=0, 1, 2, \ldots, n)$. Обозначим разность

$$f(x)-F(x)=R(x).$$

Если $f(x)$ представляет собою многочлен n-й степени, то $F(x)\equiv f(x)$ и $R(x)\equiv 0$. В общем случае в точках x, отличных от узлов интерполяции, разность $R(x)$ отлична от нуля и представляет погрешность замены функции интерполяционным многочленом. Ее называют *остаточным членом интерполяции*.

Представим функцию $f(x)$ в виде

$$f(x)=F(x)+R(x), \tag{1.25}$$

где $F(x)$ — интерполяционный многочлен, $R(x)$ — остаточный член интерполяции, и найдем выражение для $R(x)$. Для этого зафиксируем некоторое значение x из интервала (x_0, x_n) и рассмотрим вспомогательную функцию $T(z)$ вспомогательной переменной z, опреде-

ленную равенством

$$T(z)=f(z)-F(z)-\frac{(z-x_0)(z-x_1)\ldots(z-x_n)}{(x-x_0)(x-x_1)\ldots(x-x_n)}R(x), \qquad (2.25)$$

где $x_0, x_1, \ldots, x_n$ — узлы интерполяции и z изменяется на интервале (x_0, x_n).

Допустим, что $f(x)$ непрерывна и имеет непрерывные производные до порядка $n+1$ включительно. Тогда этими же свойствами обладает и функция $T(z)$, так как x, а значит, и $R(x)$ здесь постоянны. Из выражения (2.25) видно, что $T(z)$ обращается в нуль при $z=x_0$, $z=x_1$, ..., $z=x_n$. Действительно, числитель дроби при $R(x)$ обратится в любом узле интерполяции в нуль, а $f(z)$ и $F(z)$ в узлах интерполяции совпадают. Кроме того, $T(z)$ обращается в нуль и при $z=x$, так как в этом случае дробь при $R(x)$ равна единице, а $F(x)+R(x)=f(x)$ вследствие равенства (1.25).

Таким образом, функция $T(z)$ обращается в нуль в $n+2$ точках. Применяя для каждого из $n+1$ частичных интервалов теорему Ролля, убедимся, что производная $T'(z)$ обратится в нуль в интервале (x_0, x_n) по крайней мере $n+1$ раз. Применяя далее теорему Ролля к производным $T'(z)$, $T''(z)$, ..., придем к заключению, что производная $T^{(n+1)}(z)$ обращается в нуль по крайней мере в одной точке $z=\xi$.

Вычислим теперь производную $T^{(n+1)}(z)$, для чего продифференцируем функцию (2.25) $n+1$ раз. Так как $F(z)$ — многочлен n-й степени, то его $(n+1)$-я производная равна нулю. Далее,

$$\frac{d^{n+1}}{dz^{n+1}}[(z-x_0)(z-x_1)\ldots(z-x_n)]=(n+1)!,$$

ибо выражение в квадратных скобках есть многочлен $n+1$-й степени относительно z со старшим членом z^{n+1}. Итак,

$$T^{(n+1)}(z)=f^{(n+1)}(z)-\frac{(n+1)!}{(x-x_0)(x-x_1)\ldots(x-x_n)}R(x).$$

Полагая здесь $z=\xi$, найдем

$$f^{(n+1)}(\xi)-\frac{(n+1)!}{(x-x_0)(x-x_1)\ldots(x-x_n)}R(x)=0,$$

откуда

$$R(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\ldots(x-x_n), \qquad (3.25)$$

где ξ — некоторая промежуточная точка интервала (x_0, x_n).

Выражение (3.25) можно преобразовать так, как это делалось в § 23 с формулами Ньютона. Положим $(x-x_0)/h=t$. Тогда получим

$$R(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}t(t-1)\ldots(t-n),$$

или, иначе,

$$R(x)=f^{(n+1)}(\xi)\,h^{n+1}\binom{t}{n+1},$$

где символ $\binom{t}{n+1}$ означает биномиальный коэффициент

$$\binom{t}{n+1}=\frac{t(t-1)\ldots(t-n)}{(n+1)!}.$$

Обычно в выражении для остаточного члена явно указывают в виде индекса степень интерполяционного многочлена. Поэтому окончательное выражение для остаточного члена пишут в виде

$$R_n(x)=\binom{t}{n+1}f^{(n+1)}(\xi)\,h^{n+1}. \tag{4.25}$$

Если удается получить оценку наибольшего значения $(n+1)$-й производной на рассматриваемом интервале в виде $|f^{(n+1)}(x)|<M$, то для остаточного члена мы получаем

$$|R_n(x)|<Mh^{n+1}\binom{t}{n+1}. \tag{5.25}$$

Однако чаще всего такую оценку для $f^{(n+1)}(\xi)$ получить не удается. Тогда можно получить представление о величине погрешности интерполяции, воспользовавшись приближенным равенством (10.21)

$$y^{(n+1)}\approx\Delta^{n+1}y/h^{n+1}.$$

Так как обычно интерполяционную формулу обрывают на членах, содержащих практически постоянные разности, то можно считать, что

$$R_n(x)\approx\binom{t}{n+1}\Delta^{n+1}y, \tag{6.25}$$

где $n+1$ — порядок постоянных разностей.

Формулу (6.25) можно использовать для доказательства приведенного в § 22 правила, относящегося к оценке линейной интерполяции. Приняв в (6.25) $n=1$, получим

$$R_2(x)\approx\frac{t(t-1)}{2!}\Delta^2y.$$

Так как при интерполяции между точками x_0, x_1 значение t удовлетворяет условиям $0<t<1$ и при этих значениях t справедливо неравенство $|t(t-1)|\leqslant 1/4$, то

$$R_2(x)<|\Delta^2y|/8,$$

что и утверждается в упомянутом правиле.

§ 26. Программирование прямой и обратной интерполяции

Из рассмотренных в предыдущих параграфах интерполяционных формул и схем наиболее удобной для программирования является схема Эйткина. Как показывают формулы (1.22), (3.22) и (4.22), линейная, квадратичная и кубическая интерполяции производятся по схеме

Эйткина единообразно, причем для интерполяции следующего порядка используются многочлены, полученные на предыдущем шаге. Это позволяет строить интерполяцию любого порядка в виде двойного цикла, в котором внутренний цикл подсчитывает все нужные интерполяционные многочлены данного порядка, а внешний ведется по порядку интерполяции.

Пусть заданы две группы по $n+1$ ячеек, в которых записаны подряд значения $x_0, x_1, \ldots, x_n$ и $y_0, y_1, \ldots, y_n$. Степень интерполяции n записана в ячейках (0, n, 0) и (n, 0, 0) в виде числа единиц адресов. Предположим, что в ячейке α лежит требуемое значение аргумента; составим программу, которая вычислит интерполированное значение функции и положит его в ячейку γ.

Из формул (1.22) и (2.22) видно, что роль многочленов нулевого порядка играют значения функции $y_0, y_1, \ldots, y_n$. Поэтому работу программы нужно начать с пересылки этих значений на рабочее поле, отведенное для счета многочленов. Это будут ячейки $P_0, P_1, \ldots, P_n$. Пересылку можно осуществить с помощью трех команд, пользуясь регистром адреса

1)	[PA]	0 (0, n, 0)		0
2)			y_0^*	$=P_0^*$
3)	$PA\geqslant\overline{1}$		2)	F^*

Напишем теперь цикл для вычисления многочленов $P_{0,1}(\alpha), P_{1,2}(\alpha), \ldots, P_{n-1,n}(\alpha)$ по формулам типа (1.22). Многочленов первой степени будет уже не $n+1$, а только n. При переходе ко второй степени число многочленов снова уменьшится на единицу. Поэтому можно организовать внешний цикл по счетчику ν, который будет при каждом шаге уменьшаться на единицу. Вычисление многочленов первой степени можно организовать так:

4)		(n, 0, 0)	—	(1, 0, 0)	$=\nu$
5)	PA	0		0	0
6)		x_0^*	—	α	$=R_0$
7)		x_1^*	—	α	$=R_1$
8)		P_0^*	·	R_1	$=R_2$
9)		P_1^*	·	R_0	$=R_3$

10) R_2 — R_3 $=R_4$
11) x_1^* — x_0^* $=R_5$
12) R_4 : R_5 $=P_0^*$
13) $PA<\overline{n-1}$ 6) $\overline{1}^*$

Команды 5) — 13) образуют внутренний цикл нашей программы. При переходе к следующему шагу внешнего цикла, при новом обращении к внутреннему нужно уменьшить значение ν и переадресовать команды 7) и 11), увеличив их левые адреса на единицу:

14) ν —, (1, 0, 0) $=\nu$
15) 7) +, (1, 0, 0) $=7)$
16) 11) +, (1, 0, 0) $=11)$

Команду 13) проверки окончания цикла можно сформировать из заготовки $13)_0$ $PA<0$ 6) $\overline{1}^*$ таким образом:

17) $13)_0+$, $\nu=13)$.

Обозначив переадресуемые команды 7) и 11) через T_1, T_2, а формируемую команду 13) через U, организуем внешний цикл следующим образом:

а) $(n,\ 0,\ 0)-,\ (1,\ 0,\ 0)=\nu$
б) T_1^0 $=T_1$
в) *Б* T_2^0 T_2
г) $T_1+,\ (1,\ 0,\ 0)=T_1$
д) $T_2+,\ (1,\ 0,\ 0)=T_2$
е) $U^0+,$ $\nu=U$
ж) *Внутренний цикл*
з) ν —, (1, 0, 0) $=\nu$
и) *У0* P_0 г) γ
к) *стоп*

Здесь команда и), одновременно с передачей управления, пересылает ответ в ячейку γ. Программа будет работать верно для любого n, кроме случая $n=0$, который необходимо исключить.

Окончательно программу интерполяции можно записать в виде

```
[PA]   0          (0, n, 0)    0
                  y0*          =P0*
PA≥    1          Я—1          F*
       114   —, (0, n, 0)      =(n, 0, 0)
(n, 0, 0) —, (1, 0, 0)        =ν
                  T1^0         =T1
       ──────────────────────────►
Б      T2^0       Δ            T2
∇      T1    +, (1, 0, 0)     =T1
       T2    +, (1, 0, 0)     =T2
Δ      U^0   +,   ν           =U
PA     0          0            0
       x0*   —    α            =R0      ↑
T1     (x1*  —    α            =R1)     | н. п.
       P0*   ·    R1           =R2      |
       P1*   ·    R0           =R3      |
       R2    —    R3           =R4      |
T2     (x1*  —    x0*          =R5)     | н. п.
       R4    ↓    R5           =P0*     ↓
U (PA<n—1                      1*)      н. п.
       ν     —, (1, 0, 0)     =ν
       ──────────────────────────►
УО     P0         ∇            γ
              стоп
T1^0   x1*   —    α            =R1
T2^0   x1*   —    x0*          =R5
U^0 PA <0         T1—1         1*.
```

Та же программа пригодна и для экстраполяции, поскольку никаких ограничений на значение аргумента в ней не содержится. Программа вычисляет значение многочлена n-й степени для любого x. Наоборот, в некоторых случаях нужно, чтобы программа не считала значений функции для аргументов, выходящих за пределы данной таблицы. Такая надобность возникает, например, при работе с табличной функцией (см. следующий

параграф). Тогда в программе надо предусмотреть специальную проверку.

Обратная интерполяция, т. е. нахождение промежуточного значения аргумента по значению функции, может быть выполнена с помощью той же программы, в которой, однако, требуется поменять местами значения x и y и заслать в ячейку α заданное значение функции. Программа выдаст в ячейке γ соответствующее значение аргумента.

В качестве примера обратной интерполяции рассмотрим программирование нахождения корня способом проведения параболы, который разбирался в § 5. Как мы теперь видим, это есть обычная задача обратной интерполяции: даны значения y_0, y_1, y_2 функции $f(x)$ в точках x_0, x_1, x_2; требуется найти такую точку x, в которой значение функции равно нулю. Поскольку теперь степень интерполяции мала и значение функции вычислять не требуется, то можно обойтись без двойного цикла. Нужно только предположить, как и раньше, что значения x_0, x_1, x_2 и y_0, y_1, y_2 расположены в ячейках памяти подряд.

Приближенное значение корня в ячейке γ можно получить с помощью такой программы:

PA	0	0	0	$P_0 \cdot y_2 = R_0$	
	$x_0^* \cdot y_1^* = R_0$			$P_1 \cdot y_0 = R_1$	
	$x_1^* \cdot y_0^* = R_1$			$R_0 - R_1 = R_2$	
	$R_0 - R_1 = R_2$			$y_2 - y_0 = R_3$	
	$y_1^* - y_0^* = R_3$			$R_2 \,:\, R_3 = \gamma$	
	$R_2 \,:\, R_3 = P_0^*$			*стоп*	
$PA < \overline{1}$			$\overline{1}^*$		

При фактическом нахождении корня эту программу следует включить в качестве блока в цикл, который будет находить аргументы x_0, x_1, x_2 и соответствующие значения y_0, y_1, y_2 в зависимости от заданной функции.

§ 27. Программа работы с табличной функцией

В практике вычислений приходится иметь дело с функциями, которые на различных участках определяются различными формулами. В этом случае часто бывает выгодно задавать функцию не совокупностью формул,

а таблицей значений в отдельных точках. Еще более важно задание функции таблицей значений, когда эти значения получены в результате эксперимента, так что аналитическое выражение функции может оказаться вообще неизвестным.

Если функция задана в виде таблицы значений, записанной в оперативной памяти машины, то для работы с такой функцией требуется иметь программу, работающую по образцу стандартных программ функций одного переменного. Задав в ячейке α аргумент и обратившись к этой программе, мы должны в ячейке γ получить соответствующее значение функции, либо взятое прямо из таблицы, либо, если это требуется, полученное путем интерполяции табличных значений. Такую программу называют *программой работы с табличной функцией*; ее составлению и посвящен настоящий параграф.

Пусть функция $y=f(x)$ определена на отрезке $[a, b]$ и задана в $n+1$ равноотстоящих точках отрезка $x_0=a$, $x_1=x_0+h$, $x_2=x_0+2h$, ..., $x_n=x_0+nh=b$. Предположим, что эти значения аргумента расположены в идущих подряд ячейках памяти x_0, x_1, ..., x_n, а соответствующие им значения функции — в идущих подряд ячейках памяти y_0, y_1, ..., y_n. Шаг таблицы будем считать настолько малым, чтобы для нахождения значений функции, соответствующих промежуточным значениям аргумента, было достаточно квадратичной интерполяции.

Нахождение значения функции по значению аргумента, находящемуся в ячейке α, состоит из двух этапов:

I) определение номера первого из трех табличных значений функции, нужных для интерполяции;

II) квадратичная интерполяция по найденным трем значениям.

Рассмотрим сначала программу, осуществляющую второй этап вычислений. Предположим, что требуемый на первом этапе вычислений номер i уже определен и записан в ячейке ν в виде числа единиц второго адреса, так что $\nu \equiv (0, i, 0)$. Интерполяцию будем осуществлять по первой интерполяционной формуле Ньютона (9.23), которая для случая квадратичной интерполяции имеет вид

$$y=F(x)=F(x_i+th)=y_i+t\Delta y_i+\frac{t(t-1)}{2}\Delta^2 y_i.$$

Нужные вычисления осуществляются программой:

Интерполяция

[PA]	0	v	0	$y_0^* + R_0$	$= R_0$
	α	$-x_0^*$	$= R_0$	t	$-$«1» $= R_1$
	R_0	: h	$= t$	t	$\cdot R_1 = R_1$
	y_1^*	$-y_0^*$	$= \Delta y_i$	R_1	$\cdot$«1/2» $= R_1$
	y_2^*	$-y_1^*$	$= \Delta y_{i+1}$	R_1	$\cdot \Delta^2 y_i = R_1$
	Δy_{i+1}	$-\Delta y_i$	$= \Delta^2 y_i$	$R_0 + R_1$	$= \gamma$
	t	$\cdot \Delta y_i$	$= R_0$	*стоп.*	

В этой программе первая команда заносит в регистр адреса номер i, который затем прибавляется к адресам встречающихся в программе ячеек x_0, y_0, y_1, y_2. Дальнейших пояснений к программе не требуется.

Заметим, кстати, что при $\alpha = x_i$ мы получим $t = 0$, вследствие чего $\gamma = y_i$. Аналогично, при $\alpha = x_{i+1}$ будем иметь $t = 1$ и $\gamma = y_{i+1}$, а при $\alpha = x_{i+2}$ получим $t = 2$ и $\gamma = y_{i+2}$. Таким образом, если заданное значение аргумента совпадает с одним из узлов интерполяции, то в качестве значения функции программа выдаст соответствующее табличное значение.

Перейдем теперь к рассмотрению первого этапа решения задачи — к отысканию номера i первого из трех значений y, нужных для интерполяции. Для того чтобы приведенная формула при заданном значении α осуществляла интерполяцию, а не экстраполяцию, должно быть выполнено условие

$$x_i \leqslant \alpha < x_{i+1}.$$

Так как $x_i = x_0 + ih$, $x_{i+1} = x_0 + (i+1)h$, то отсюда вытекает, что

$$(\alpha - x_0)/h - 1 < i \leqslant (\alpha - x_0)/h.$$

Из последнего неравенства видно, что i есть целая часть числа $(\alpha - x_0)/h$:

$$i = [(\alpha - x_0)/h].$$

Выделение целой и дробной частей числа уже рассматривалось нами в первом выпуске. Если произвести сдвиг мантиссы ячейки v по порядку этой же ячейки, т. е.

выполнить команду

$$v[\rightarrow,]v=R,$$

то в ячейке R выделится мантисса дробной части числа, лежащая в адресной части. Целая часть при этом теряется; если же произвести сдвиг всей ячейки, то целая часть числа попадет в кодовую часть и будет записана непосредственно перед первым адресом. Чтобы сдвинуть целую часть в средний адрес, надо произвести еще один сдвиг на два адреса, т. е. на $24_{10}=30_8$ разрядов вправо.

Поскольку сдвиг вправо является отрицательным, то можно сразу сдвинуть целую часть ячейки v во второй адрес, производя сдвиг мантиссы v не на порядок v, а на порядок, уменьшенный на 30_8. Таким образом, мы можем получить величину i во втором адресе ячейки ν с помощью команд

$$\begin{array}{llll} \alpha & — & x_0 & =u \\ u & : & h & =v \\ v & ,— & (30;\ 0,\ 0,\ 0) & =R_0 \\ R_0[\rightarrow,] & & v & =\nu. \end{array}$$

При этом в кодовой части ячейки ν останется порядок числа $v=(x-x_0)/h$, а в третьем адресе — начало мантиссы дробной части. Однако для операции взятия в регистр содержимое кодовой части не играет роли, а ячейка ν ни для чего другого не используется.

Мы не можем пока считать задачу окончательно решенной, так как необходимо рассмотреть еще два исключительных случая: 1) когда заданное значение аргумента выходит за пределы отрезка $[a, b]$, являющегося областью определения функции или 2) когда заданный аргумент попадает между последними узлами интерполяции. В первом случае (когда $\alpha<x_0$ либо $\alpha>x_n$) условимся засылать в ячейку γ вместо ответа число *Щ* (777, *F*, *F*, *F*) и выходить на *стоп*. Во втором случае, когда аргумент α удовлетворяет неравенству

$$x_{n-1}\leqslant\alpha<x_n,$$

в качестве i в среднем адресе ячейки ν появится число $n-1$. Вычисления с таким значением i невозможны, так как справа от точки x_i будет тогда лишь одно, а не два

значения функции. Поэтому значение i нужно в этом случае уменьшить на единицу.

Таким образом, с учетом всех указанных возможностей программу для осуществления первого этапа можно записать в виде

1)	α —	x_0	$=u$
2)	*У1* *Щ*	*стоп*	γ
3)	u :	h	$=v$
4)	v , — (30; 0, 0, 0)		$=R_0$
5)	$R_0[\rightarrow,]$	v	$=\nu$
6)	ν —, (0, n, 0)		$=R_0$
7)	*У0*	*стоп*	
8)	R_0+, (0, 1, 0)		$=0$
9)	*У0* *Интерполяция*		
10)	ν —, (0, 1, 0)		$=\nu$.

Если $\alpha < x_0$, то в результате команды 1) выработается сигнал $\omega=1$ и команда 2), заслав *Щ* в γ, передаст управление на *стоп*. В противном случае команды 3)—5) вычислят величину i и поместят ее во втором адресе ячейки ν. При $i \geqslant n$, что означает $\alpha \geqslant x_n$, команда 6) выработает сигнал $\omega=0$ и команда 7) передаст управление на *стоп*. При $i<n$ в левом адресе ячейки R_0 получится F, а в среднем F, если $i=n-1$, и число, меньшее F, если $i<n-1$.

Команда 8), прибавляя к R_0 единицу среднего адреса, проверяет это обстоятельство. Если $i<n-1$, то в среднем адресе R_0 мы имеем число, меньшее F, прибавление единицы не переполняет мантиссу R_0 и команда 8) вырабатывает сигнал $\omega=0$. Тогда в ячейке ν лежит нужное значение i и команда 9) передает управление на интерполяцию. Если же $i=n-1$, то в среднем адресе R_0 находится F и прибавление единицы переполняет мантиссу R_0, так что вырабатывается сигнал $\omega=1$. Поэтому команда 9) передает управление команде 10), которая уменьшает i на единицу. Команда 10) передает управление следующей ячейке, в которой либо должна начинаться программа интерполяции, либо находиться команда безусловной передачи управления программе интерполяции.

В составленную программу можно внести еще одно уточнение. При нахождении тройки табличных значений y_i, y_{i+1}, y_{i+2}, которые нужны для интерполяции, мы выбирали i так, чтобы $x_i \leqslant \alpha < x_{i+1}$. Однако очевидно, что если α ближе к x_i, чем к x_{i+1}, то квадратичная интерполяция будет точнее, если вместо значений, указанных выше, брать значения y_{i-1}, y_i, y_{i+1}, т. е. если уменьшить i на единицу, как и для последнего участка. Если α ближе к x_i, чем к x_{i+1},

то дробная часть числа $v=(\alpha-x_0)/h$ будет меньше половины, в противном случае — больше. Поскольку начало дробной части числа v размещается в третьем адресе ячейки ν, то об этом можно судить по первой цифре этого адреса, т. е. по двенадцатому разряду ячейки ν. Если в двенадцатом разряде стоит нуль, то дробная часть v меньше половины и α ближе к x_i, а если единица, то дробная часть v больше половины и α ближе к x_{i+1}. В первом случае i надо уменьшить на единицу (если $i>0$; при $i=0$, т. е. $v<1$, значение i не изменяется), во втором — оставить без изменения. Проверку можно осуществить высечением двенадцатого разряда ячейки ν, что достигается командой

$$\nu \wedge (0,\ 0,\ 4000)=0$$

и последующей условной передачей управления. При этом удобно также несколько изменить команды 8) — 10).

С внесенными изменениями программа работы с табличной функцией будет выглядеть так:

	α	—	x_0	$=u$
У1	*Щ*		*стоп*	γ
	u	:	h	$=v$
У0	0		*Интерполяция*	ν
	v	,	$-(30;\ 0\ ,0,\ 0,)$	$=R_0$
	R_0	[→,]	v	$=\nu$
	ν	—,	$(0,\ n,\ 0)$	$=R_0$
У0			*стоп*	
	R_0	+,	$(0,\ 1,\ 0)$	$=0$
У1				
	ν	$\wedge$	$(0,\ 0,4000)$	$=0$
У0				
	ν	—,	$(0,\ 1,\ 0)$	$=\nu$
Интерполяция [*РА*]	0		ν	0
	α	—	x_0^*	$=R_0$
	R_0	:	h	$=t$
	y_1^*	—	y_0^*	$=\Delta y_i$
	y_2^*	—	y_1^*	$=\Delta y_{i+1}$
	Δy_{i+1}	—	Δy_i	$=\Delta^2 y_i$
	t	·	Δy_i	$=R_0$
	y_0^*	+	R_0	$=R_0$
	t	—	«1»	$=R_1$
	t	·	R_1	$=R_1$
	R_1	·	«$^1/_2$»	$=R_1$
	R_0	+	R_1	$=\gamma$
	стоп			

ГЛАВА V

ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

§ 28. Формулы прямоугольников и трапеций

Формулы для приближенного вычисления определенных интегралов, называемые также *квадратурными формулами*, применяются очень часто. Дело в том, что для большого числа элементарных функций первообразные уже не выражаются через элементарные функции, в результате чего нельзя вычислить определенный интеграл с помощью формулы Ньютона—Лейбница. Встречаются также и случаи, когда приходится прибегать к формулам приближенного интегрирования даже для таких интегралов, которые могут быть найдены в конечном виде, но их выражение оказывается слишком сложным. Особенно важны формулы приближенного интегрирования при решении задач, содержащих функции, заданные таблично.

Наиболее простыми формулами для численного интегрирования являются формулы *прямоугольников* и *трапеций*. Вывод их основан на использовании геометрического смысла определенного интеграла, выражающего, как известно, площадь криволинейной трапеции.

Формула прямоугольников, собственно, есть не что иное, как интегральная сумма, составленная с учетом некоторых дополнительных предположений, впрочем, совершенно естественных.

Пусть требуется вычислить интеграл $\int_a^b f(x)\,dx$. Разобьем участок интегрирования $[a, b]$ на n равных частей и поместим точки, значения функции в которых входят в интегральную сумму, в левых концах полученных участков. Если считать, что n достаточно велико, т. е.

длина участков разбиения $h=(b-a)/n$ достаточно мала, то интегральная сумма должна уже мало отличаться от величины интеграла. Таким образом, мы получаем приближенное равенство

$$\int_a^b f(x)\,dx \approx h(y_0+y_1+\ldots+y_{n-1}) = h\sum_{k=0}^{n-1} y_k, \qquad (1.28)$$

которое и является *формулой прямоугольников*. Здесь, как и всюду в дальнейшем, через y_0, y_1, ..., y_n обозначены значения функции $y=f(x)$ в точках деления x_0, x_1, ..., x_n.

Аналогичная формула прямоугольников получится и в том случае, если брать для интегральной суммы значения функции не в левых, а в правых концах участков разбиения. Тогда формула примет вид

$$\int_a^b f(x)\,dx \approx h(y_1+y_2+\ldots+y_n) = h\sum_{k=1}^{n} y_k. \qquad (2.28)$$

Для функции, монотонной на отрезке интегрирования, всякая интегральная сумма, а значит, и определенный интеграл заключены между приближенными значениями, указанными в правых частях формул (1.28) и (2.28). Геометрическую иллюстрацию этого факта можно видеть на рис. 16, где взята возрастающая функция и слагаемые, входящие в различные суммы, показаны пунктиром и штриховкой. Благодаря этому представление о погрешностях формулы прямоугольников можно получить, рассматривая разность результатов, полученных по формулам (1.28) и (2.28).

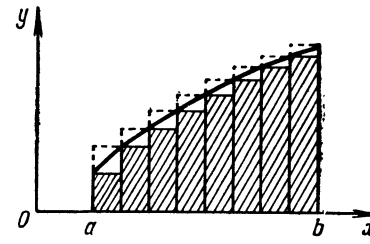


Рис. 16.

Если функция имеет на отрезке интегрирования конечное число экстремумов, то отрезки интегрирования можно разбить на участки монотонности и получить, таким образом, оценку погрешности формулы прямоугольников.

Пример 1.28. Вычислить по формулам прямоугольников интеграл $\int_0^1 \frac{dx}{1+x}$, разбив участок интегрирования на 10 частей.

Значения функции приведены в табл. 1.28. Применяя формулы (1.28) и (2.28), находим в первом случае

$$\int_0^1 \frac{dx}{1+x} \approx 0{,}1 \cdot (1{,}0000 + 0{,}9091 + \ldots + 0{,}5263) = 0{,}7188,$$

и во втором

$$\int_0^1 \frac{dx}{1+x} \approx 0{,}1 \cdot (0{,}9091 + 0{,}8333 + \ldots + 0{,}5000) = 0{,}6688.$$

Как известно, истинным значением этого интеграла является число $\ln 2 = 0{,}6931\ldots$, которое действительно заключено между двумя полученными приближениями. Зная это истинное значение, мы можем определить относительные погрешности приближенных значений интеграла; они равны соответственно $\delta_1 = 3{,}7\%$ и $\delta_2 = 3{,}5\%$. Но и имея только два вычисленных приближения, можно утверждать, что абсолютная погрешность каждого из них не превосходит $\Delta = 0{,}05$.

Таблица 1.28

(1)	(2)	(3)
x	«1» + (1)	y = «1» : (2)
0	1,0	1,0000
0,1	1,1	0,9091
0,2	1,2	0,8333
0,3	1,3	0,7692
0,4	1,4	0,7143
0,5	1,5	0,6667
0,6	1,6	0,6250
0,7	1,7	0,5882
0,8	1,8	0,5556
0,9	1,9	0,5263
1,0	2,0	0,5000

Легко заметить, что среднее арифметическое полученных по формулам прямоугольников приближений равно 0,6938, т. е. довольно точно совпадает уже с истинным значением интеграла: относительная погрешность равна здесь 0,1%.

Это наводит на мысль принимать в качестве приближенного значения интеграла среднее арифметическое приближений, полученных по формулам (1.28) и (2.28).

Заметим, что нет никакой нужды вычислять предварительно эти значения, так как можно сразу воспользоваться готовой формулой. Действительно, взяв среднее арифметическое правых частей формул (1.28) и (2.28), мы получим

$$\int_a^b f(x)\,dx \approx h\left(\frac{1}{2}y_0 + y_1 + \ldots + y_{n-1} + \frac{1}{2}y_n\right) = h\left(\frac{y_0}{2} + \sum_{k=1}^{n-1} y_k + \frac{y_n}{2}\right). \tag{3.28}$$

Это и есть *формула трапеций*.

Формулу трапеций (3.28) можно легко получить и непосредственно, исходя из ее геометрического смысла. Разобьем отрезок интегрирования на n равных частей точками $a = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = b$, проведем ординаты во всех точках деления и заменим каждую из полученных криволинейных трапеций прямолинейной, как это показано на рис. 17. Сторонами каждой из этих трапеций являются две соседние ординаты, участок оси Ox, длина которого $h = (b-a)/n$, и хорда кривой.

Площадь самой левой трапеции равна $\Delta s_1 = \frac{y_0 + y_1}{2}h$. Аналогично для трапеции, расположенной над участком (x_{i-1}, x_i), находим

$$\Delta s_i = \frac{y_{i-1} + y_i}{2}h. \tag{4.28}$$

Суммирование выражений (4.28) по всем i от 1 до n и приводит к формуле (3.28), так как все ординаты, кроме крайних, используются в выражениях вида (4.28) дважды.

Пример 2.28. Вычислим по формуле трапеций длину дуги параболы $y = x(1-x)$ между точками пересечения ее с осью Ox (рис. 18).

В интегральном исчислении доказывается, что длина дуги кривой $y = f(x)$ на участке $[a, b]$ выражается интегралом

$$l = \int_a^b \sqrt{1 + (y')^2}\,dx, \tag{5.28}$$

где $y' = f'(x)$ — производная функции $f(x)$. Парабола $y = x(1-x)$

пересекается с осью Ox в точках $x=0$ и $x=1$. Так как $y'=1-2x$, то длина дуги выразится интегралом

$$l=\int_0^1 \sqrt{1+(1-2x)^2}\,dx. \tag{6.28}$$

Разобьем отрезок интегрирования на четыре части и вычислим соответствующие значения функции (табл. 2.28). Пользуясь формулой

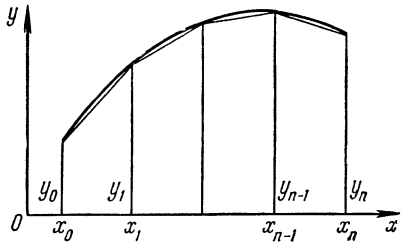


Рис. 17.

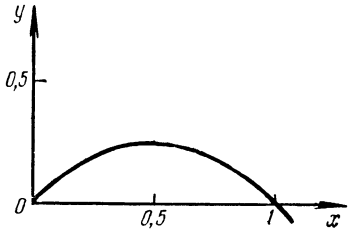


Рис. 18.

трапеций (3.28), находим

$$l\approx 0{,}25\left(\frac{1{,}414}{2}+1{,}118+1{,}000+1{,}118+\frac{1{,}414}{2}\right)=1{,}162.$$

Интеграл (6.28) можно выразить через элементарные функции. Опуская все промежуточные выкладки, можем написать

$$l=\int_0^1 \sqrt{1+(1-2x)^2}\,dx=$$

$$=\left[\frac{2x-1}{4}\sqrt{4x^2-4x+2}+\frac{1}{4}\ln\left(4\sqrt{4x^2-4x+2}+8x-4\right)\right]_0^1=$$

$$=\frac{\sqrt{2}}{2}+\frac{1}{2}\ln\left(\sqrt{2}+1\right)=1{,}148.$$

Отсюда видно, что полученное по формуле трапеций приближенное значение длины дуги l дает абсолютную погрешность $\Delta l=0{,}014$ или относительную погрешность $\delta=1{,}2\%$. Вместе с тем, вычисления, требуемые формулой трапеций, во много раз проще и короче, нежели вычисления по точной формуле.

Точность приближенного значения интеграла, полученного по формуле трапеций (3.28), можно улучшить, увеличивая число n участков разбиения отрезка интегрирования, хотя при этом, естественно, возрастает объем требуемых вычислений. Например, уже при $n=5$ и $h=0{,}2$ получаем в качестве приближенного значения по

Таблица 2.28

(1)	(2)	(3)	(4)	(5)	(6)
x	«2» · (1)	«1» — (2)	(3)²	«1» + (4)	$f(x)=\sqrt{(5)}$
0	0	1	1	2,000	1,414
0,25	0,500	0,500	0,250	1,250	1,118
0,50	1,000	0	0	1,000	1,000
0,75	1,500	—0,500	0,250	1,250	1,118
1	2,000	—1	1	2,000	1,414

формуле трапеций $l=1{,}157$ с абсолютной погрешностью $\Delta=0{,}009$ и относительной — 0,8%, а при $n=10$ и $h=0{,}1$ уже $l=1{,}150$, так что абсолютная погрешность составляет только 0,002, а относительная 0,2%. Вычисления для этого последнего случая приведены в табл. 3.28. Даже в этом последнем случае вычисления осуществляются проще и быстрее, нежели по точной формуле (включая нахождение неопределенного интеграла).

Таблица 3.28

(1)	(2)	(3)	(4)	(5)	(6)
x	«2» · (1)	«1» — (2)	(3)²	«1» + (4)	$f(x)=\sqrt{(5)}$
0	0	1	1	2,00	1,414
0,1	0,2	0,8	0,64	1,64	1,281
0,2	0,4	0,6	0,36	1,36	1,166
0,3	0,6	0,4	0,16	1,16	1,077
0,4	0,8	0,2	0,04	1,04	1,020
0,5	1,0	0	0	1,00	1,000
0,6	1,2	—0,2	0,04	1,04	1,020
0,7	1,4	—0,4	0,16	1,16	1,077
0,8	1,6	—0,6	0,36	1,36	1,166
0,9	1,8	—0,8	0,64	1,64	1,281
1	2	—1	1	2,00	1,414

§ 29. Формула Симпсона

Более точной, нежели рассмотренная в предыдущем параграфе формула трапеций, является формула Симпсона. Для достижения той же точности в ней можно

брать меньшее число n участков разбиения и соответственно больший шаг h, а при одном и том же шаге h, т. е. при том же объеме вычислений *), она дает меньшие абсолютную и относительную погрешности.

Формулу Симпсона можно получить с помощью того же приема, который уже дважды применялся в § 28. Разобьем участок $[a, b]$ на четное число $n=2m$ частей

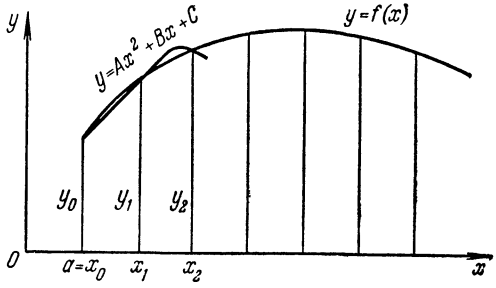


Рис. 19.

точками $a=x_0<x_1<\ldots<x_{n-1}<x_n=b$, обозначим ординаты в точках деления через $y_0, y_1, \ldots, y_n$ и рассмотрим пару соседних участков, например, с левым концом в точке $a=x_0$ (рис. 19). Проведем через три точки кривой с координатами (x_0, y_0), (x_1, y_1), (x_2, y_2) параболу с осью, параллельной оси Oy. Ее уравнение будет

$$y=Ax^2+Bx+C, \tag{1.29}$$

причем коэффициенты A, B, C остаются пока неизвестными **).

Заменив площадь заданной криволинейной трапеции на участке $[x_0, x_2]$ площадью криволинейной трапеции, ограниченной параболой (1.29), придем к приближен-

*) Как видно из приведенных в § 28 примеров, почти вся вычислительная работа идет на нахождение значений функции. Поэтому объем вычислений определяется числом требуемых ординат.

**) Для получения этого уравнения можно воспользоваться любой из интерполяционных формул, рассмотренных в § 23. Тогда не возникнет задача нахождения неизвестных коэффициентов A, B, C. Тем не менее, приведенный ниже способ вывода проще, чем при использовании интерполяционных формул.

ному равенству

$$\int_{x_0}^{x_2} f(x)\,dx \approx \int_{x_0}^{x_2} (Ax^2+Bx+C)\,dx = \left[A\frac{x^3}{3}+B\frac{x^2}{2}+Cx\right]_{x_0}^{x_2} =$$
$$= A\frac{x_2^3-x_0^3}{3}+B\frac{x_2^2-x_0^2}{2}+C(x_2-x_0).$$

Вынося за скобку общий множитель x_2-x_0 и приводя к общему знаменателю, получим

$$\int_{x_0}^{x_2} f(x)\,dx \approx \frac{x_2-x_0}{6}[2A(x_0^2+x_0x_2+x_2^2)+3B(x_0+x_2)+6C]. \tag{2.29}$$

Неизвестные коэффициенты A, B, C в уравнении (1.29) и формуле (2.29) находятся из условия, что при значениях x, равных x_0, x_1, x_2, функция $f(x)$ принимает соответственно значения y_0, y_1, y_2. Заметив, что $x_1=\frac{x_0+x_2}{2}$, запишем эти условия в виде

$$\left.\begin{aligned} y_0 &= Ax_0^2 + Bx_0 + C,\\ y_1 &= A\left(\frac{x_0+x_2}{2}\right)^2 + B\frac{x_0+x_2}{2} + C,\\ y_2 &= Ax_2^2 + Bx_2 + C. \end{aligned}\right\} \tag{3.29}$$

Умножая второе равенство (3.29) на четыре и складывая после этого все три равенства (3.29), находим

$$y_0+4y_1+y_2 =$$
$$= A[x_0^2+(x_0+x_2)^2+x_2^2]+B[x_0+2(x_0+x_2)+x_2]+6C =$$
$$= 2A(x_0^2+x_0x_2+x_2^2)+3B(x_0+x_2)+6C, \tag{4.29}$$

что совпадает с квадратной скобкой в правой части равенства (2.29).

Подставив (4.29) в правую часть равенства (2.29) и заметив, что $x_2-x_0=2h$, где $h=(b-a)/n$, придем к приближенному равенству

$$\int_{x_0}^{x_2} f(x)\,dx \approx \frac{h}{3}(y_0+4y_1+y_2). \tag{5.29}$$

Ясно, что для каждой следующей пары участков получается такая же формула

$$\int_{x_2}^{x_4} f(x)\,dx \approx \frac{h}{3}(y_2 + 4y_3 + y_4) \qquad (6.29)$$

. . . . . .

Суммируя равенства вида (5.29) и (6.29) по всем участкам, получим формулу

$$\int_a^b f(x)\,dx \approx \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + \ldots$$

$$\ldots + 2y_{2m-2} + 4y_{2m-1} + y_{2m}). \qquad (7.29)$$

Формула (7.29) и есть нужная нам *формула Симпсона*. Учитывая геометрический смысл формулы, ее называют также *формулой парабол*. В ней все ординаты с нечетными номерами умножаются на четыре, а все ординаты с четными номерами (кроме крайних) — на два. Крайние ординаты y_0 и y_{2m} входят в формулу с коэффициентами, равными единице.

Пример 1.29. Вычислим $\int_0^1 \frac{dx}{1+x^2}$ способом трапеций и парабол, разбив участок [0,1] на 10 частей, т. е. приняв $h = 0{,}1$. Вычисление значений функции приведено в табл. 1.29.

Таблица 1.29

(1)	(2)	(3)	(4)
x	$(1)^2$	«1» + (2)	$f(x) =$ «1» : (3)
0,0	0,00	1,00	1,000 0000
0,1	0,01	1,01	0,990 0990
0,2	0,04	1,04	0,961 5385
0,3	0,09	1,09	0,917 4312
0,4	0,16	1,16	0,862 0690
0,5	0,25	1,25	0,800 0000
0,6	0,36	1,36	0,735 2941
0,7	0,49	1,49	0,671 1409
0,8	0,64	1,64	0,609 7561
0,9	0,81	1,81	0,552 4862
1,0	1,00	2,00	0,500 0000

Просуммировав полученные значения функции с соответствующими коэффициентами, получим:

по формуле трапеций

$$\int_0^1 \frac{dx}{1+x^2} \approx$$

$$\approx 0{,}1\left(\frac{1{,}000\,000}{2} + 0{,}9900990 + \ldots + 0{,}5524862 + \frac{0{,}500\,0000}{2}\right) =$$

$$= 0{,}7849815;$$

по формуле Симпсона

$$\int_0^1 \frac{dx}{1+x^2} \approx \frac{0{,}1}{3}(1{,}0000000 + 4 \cdot 0{,}9900990 + 2 \cdot 0{,}9615385 + \ldots$$

$$\ldots + 4 \cdot 0{,}5524862 + 0{,}5000000) = 0{,}7853982.$$

Рассматриваемый интеграл равен

$$\int_0^1 \frac{dx}{1+x^2} = \operatorname{arctg} x \Big|_0^1 = \frac{\pi}{4},$$

поэтому можно считать, что, вычисляя этот интеграл, мы находим приближенное значение числа $\frac{\pi}{4}$. Так как истинное значение $\frac{\pi}{4} = 0{,}78539816$, то относительная погрешность при пользовании методом трапеций составляет 0,05%, а при пользовании методом парабол — практически отсутствует.

Пример 2.29. Вычислим с помощью формулы Симпсона длину дуги параболы из примера 2.28, разбивая отрезок интегрирования на четыре части.

Беря значения функции из табл. 2.28 и пользуясь формулой (7.29), находим

$$l = \frac{0{,}25}{3}(1{,}414 + 4 \cdot 1{,}118 + 2 \cdot 1{,}000 +$$

$$+ 4 \cdot 1{,}118 + 1{,}414) = 1{,}148.$$

Таким образом, формула Симпсона уже при $n = 4$ дает для длины дуги четыре значащих цифры точными.

Пример 3.29. Вычислим с помощью формулы Симпсона интеграл

$$\int_0^2 \frac{dx}{\sqrt{1+x^2+x^4}},$$

разбивая отрезок интегрирования на восемь частей.

Таблица 2.29

(1)	(2)	(3)	(4)	(5)	(6)
x	$(1)^2$	$(2)^2$	«1» + (2) + (3)	$\sqrt{(4)}$	$f(x)=$ «1»; (5)
0	0	0	1	1	1
0,25	0,0625	0,0039	1,0664	1,0327	0,9683
0,50	0,2500	0,0625	1,3125	1,1456	0,8729
0,75	0,5625	0,3164	1,8789	1,3707	0,7296
1,00	1,0000	1,0000	3,0000	1,7321	0,5773
1,25	1,5625	2,4414	5,0039	2,2369	0,4470
1,50	2,2500	5,0625	8,3125	2,8831	0,3468
1,75	3,0625	9,3789	13,4414	3,6662	0,2728
2,00	4,0000	16,0000	21,0000	4,5826	0,2182

Значения функции вычислены в табл. 2.29. Значение интеграла по формуле Симпсона равно

$$\int_0^2 \frac{dx}{\sqrt{1+x^2+x^4}} = \frac{0{,}25}{3}(1+4\cdot 0{,}9683+2\cdot 0{,}8729+\ldots$$

$$\ldots+0{,}2182) = 1{,}2069.$$

Поскольку точное значение интеграла нам в этом случае неизвестно, то указать возможную погрешность полученного результата мы сейчас не можем.

§ 30. Проверка точности результатов численного интегрирования. Остаточные члены квадратурных формул

Возможность оценить точность результатов, полученных по формулам численного интегрирования, имеет очень большое значение. Действительно, в примерах, рассматривавшихся в предыдущих параграфах (кроме примера 3.29), мы имели возможность сравнить получен-

ные по приближенным формулам значения интегралов с точными значениями. Однако практический интерес представляют, конечно, случаи, когда точное значение интеграла остается неизвестным, как это имело место в примере 3.29.

В рассмотренных примерах мы заранее задавались числом частей n, на которые разбивался участок интегрирования, что уже определяло выбор шага. Естественно возникает вопрос — какова точность полученного при этом приближения? Другой, тесно связанный с ним и наиболее практически важный вопрос — каким нужно выбрать шаг и число частей n, чтобы получить значение интеграла с погрешностью, не превосходящей заданного предела?

Вопросы эти весьма сложны и дать на них ответ в общем виде не представляется возможным. Если же речь идет о функциях, заданных аналитическими выражениями, для которых можно получить и оценить старшие производные, то для оценки точности формул трапеций и парабол можно воспользоваться выражением соответствующих *остаточных членов*, равных разности между интегралом и его приближенным значением.

Остаточный член формулы трапеций имеет вид *)

$$R = -\frac{(b-a)^3}{12n^2} M_2. \qquad (1.30)$$

Здесь $(b-a)$ — длина участка интегрирования, n — число частей, на которые разбит участок, и M_2 — наибольшее по абсолютной величине значение второй производной $f''(x)$ на рассматриваемом участке (взятое, тем не менее, со своим знаком).

Проанализируем влияние всех входящих в эту формулу членов. Длина участка интегрирования входит в числитель формулы (1.30). Из этого следует, что чем больше участок, тем больше погрешность формулы трапеций, и наоборот, при уменьшении участка интегрирования погрешность падает. Это обстоятельство вполне естественно. Столь же естественно и то, что число n частей

*) Это выражение представляет собой предельную абсолютную погрешность.

разбиения входит в знаменатель: при увеличении n погрешность быстро падает.

Легко уяснить себе также зависимость погрешности формулы трапеций от величины второй производной. Прежде всего, если $f'(x) \equiv 0$, то формула трапеций дает точный результат, поскольку погрешность обращается в нуль. Но вторая производная обращается тождественно в нуль лишь для линейной функции. Для нее формула трапеций и должна дать точный результат, поскольку она основана на замене функции линейной. Вторая производная характеризует кривизну графика функции, которая является мерой отклонения кривой от прямой линии. Поэтому ясно, что чем меньше вторая производная, тем меньше график функции $f(x)$ отличается от прямой линии и тем меньшую погрешность мы будем получать при интегрировании по формуле трапеций, заменяя функцию линейной.

Так же просто объяснить и знак минус в формуле (1.30). В самом деле, если график функции на рассматриваемом участке является выпуклым вверх, то хорды кривой, как это видно на рис. 17, лежат ниже кривой, и формула трапеций дает значение, меньшее истинного, и, значит, погрешность этой формулы положительна. Вместе с тем, вторая производная в этом случае отрицательна. Наоборот, для функции с графиком, выпуклым вниз, вторая производная положительна, а хорды проходят выше кривой, и формула трапеций имеет отрицательную погрешность. Таким образом, знак погрешности в обоих случаях противоположен знаку второй производной.

Рассмотрим теперь вывод выражения (1.30) для остаточного члена в формуле трапеций. Как уже говорилось, она получается заменой интегрируемой функции $f(x)$ на участке $[x_0, x_1]$ интерполяционным многочленом первой степени

$$F_1(x) = \frac{x-x_1}{x_0-x_1} y_0 + \frac{x-x_0}{x_1-x_0} y_1.$$

Пользуясь общим выражением для остаточного члена интерполяционной формулы (3.25), можем написать

$$f(x) = F_1(x) + R_1(x) = F_1(x) + \frac{f''(\xi)}{2}(x-x_0)(x-x_1). \qquad (2.30)$$

Интегрируя (2.30) по $[x_0, x_1]$, найдем

$$\int_{x_0}^{x_1} f(x)\,dx = \int_{x_0}^{x_1} F_1(x)\,dx + \frac{1}{2}\int_{x_0}^{x_1} f''(\xi)(x-x_0)(x-x_1)dx.$$

Так как первое слагаемое справа дает известный член формулы трапеций $h(y_0+y_1)/2$, в чем читатель легко может убедиться сам, выполнив интегрирование с учетом приведенного выше выражения для $F_1(x)$, то ошибка формулы трапеций для этого интервала выразится остаточным членом

$$r = \frac{1}{2}\int_{x_0}^{x_1} f''(\xi)(x-x_0)(x-x_1)dx. \qquad (3.30)$$

К интегралу (3.30) можно применить общую *теорему о среднем*, которая формулируется так: *если $g(x)$ и $h(x)$ непрерывны на $[a, b]$ и $h(x)$ сохраняет здесь постоянный знак, то найдется такая точка ξ, $a \leqslant \xi \leqslant b$, что будет справедливо равенство*

$$\int_a^b g(x)\,h(x)\,dx = g(\xi)\int_a^b h(x)\,dx.$$

В интеграле (3.30) можно принять производную $f''(\xi)$ за функцию $g(x)$, а произведение $(x-x_0)(x-x_1)$ — за функцию $h(x)$, сохраняющую постоянный знак на участке $[x_0, x_1]$. Тогда

$$r = \frac{1}{2}f''(\xi_1)\int_{x_0}^{x_1}(x-x_0)(x-x_1)\,dx = -\frac{h^3}{12}f''(\xi_1).$$

Полученная формула дает погрешность формулы трапеций для участка $[x_0, x_1]$. Просуммировав такие выражения для всех участков, получим выражение для общей ошибки

$$R = -\frac{h^3}{12}\sum_{i=1}^{n} f''(\xi_i).$$

Приняв среднее арифметическое значений вторых производных $\frac{1}{n}\sum_{i=1}^{n} f''(\xi_i)$ за значение второй производной в некоторой точке, что можно сделать вследствие ее непрерывности, можно заменить сумму на $\sum_{i=1}^{n} f''(\xi_i) = nf''(\xi)$. Так как, кроме того, $h=(b-a)/n$, то окончательное выражение для остаточного члена формулы трапеций можно

записать в виде

$$R = -\frac{(b-a)^3}{12n^2} f''(\xi),$$

что совпадает с приведенной выше формулой (1.30).

Практическое значение самой формулы (1.30) невелико, так как оценить величину M_2 удается далеко не всегда, особенно в тех случаях, когда функция задана таблицей. Однако в некоторых простых случаях удается получить довольно хорошую оценку погрешности, используя следующий прием.

Увеличим шаг формулы трапеций вдвое, т. е. уменьшим вдвое n (при этом первоначальное значение n должно быть четным) и найдем значение интеграла в этом случае, что очень просто, поскольку вычисления новых значений функции не требуется. Так как n входит в знаменатель в квадрате, то уменьшение n вдвое увеличит остаточный член вчетверо, и если разность между приближенным и истинным (неизвестным) значением интеграла в первом случае составляет R, то во втором это будет $4R$*). Если считать, что оба приближенных значения отклоняются от истинного в одну и ту же сторону, то разность между двумя приближенными значениями, которую можно найти, не зная точного значения, равна утроенной ошибке приближения, полученного с первоначальным n. Это дает возможность оценить порядок погрешности полученного приближения.

Возвратимся к примеру 2.28 и оценим таким способом погрешность приближения, полученного при $n=10$. По табл. 3.28 мы получили значение $I=1{,}150$, а при $n=5$ соответствующее значение интеграла $I=1{,}157$. Разность между этими значениями (0,007), как было сказано выше, равна утроенной ошибке, поэтому можно считать, что абсолютная погрешность значения $I=1{,}150$ не превышает 0,003, что, как мы знаем, дает вполне точное представление о действительной величине погрешности.

Аналогичное выражение имеет место и для формулы Симпсона. Именно, остаточный член формулы парабол

*) Напоминаем, что речь идет о предельной абсолютной погрешности.

имеет вид

$$R = -\frac{(b-a)^5}{180n^4} M_4, \tag{4.30}$$

где, как и выше, $(b-a)$ — длина участка интегрирования, $n=2m$ — общее число частей, на которые разбит участок, и M_4 — наибольшее по абсолютной величине значение четвертой производной функции, которую мы интегрируем. На выводе этой формулы мы не останавливаемся.

Формула парабол основана, как показывает само название, на замене интегрируемой функции участками парабол. Отсюда можно заключить, что формула парабол точна для всех многочленов второй степени. Из (4.30) вытекает, что формула парабол точна даже и для многочленов третьей степени, так как их четвертая производная тождественно равна нулю.

Сама по себе формула (4.30) практически бесполезна, так как найти и оценить четвертую производную весьма затруднительно. Однако сделанное выше замечание позволяет в ряде случаев так или иначе оценить точность полученных по формуле Симпсона приближений.

Прежде всего, о применимости формулы Симпсона с данным шагом h можно судить по разностям функции, составленным с тем же шагом. Если вторые или третьи разности функции практически постоянны, то функция достаточно хорошо изображается многочленом соответственно второй или третьей степени, для которых формула Симпсона дает точный результат; в этом случае можно считать, что погрешность результата выражается единицами тех же разрядов, что и третьи разности функции.

Если практически постоянны четвертые разности функции, то можно преобразовать формулу (4.30), заменяя четвертую производную разностями, как об этом было сказано в § 22. Именно, формула (10.21) с $n=4$ дает

$$f^{IV}(x) \approx \Delta^4 y/h^4.$$

Так как $h=(b-a)/n$, то вместо (4.30) получаем формулу

$$R \approx -\frac{b-a}{180}\Delta^4 y, \tag{5.30}$$

которой уже значительно проще воспользоваться, чем формулой (4.30).

Недостаток формулы (5.30) не только в том, что она применима лишь при практически постоянных четвертых разностях, но еще и в том, что вычисление всех разностей функции до четвертого порядка является довольно трудоемкой работой. Гораздо более простым и надежным является прием, который уже использовался нами выше для формулы трапеций, — удвоение шага.

При удвоении шага (для этого следует брать n кратным четырем) погрешность формулы Симпсона возрастает в 16 раз. Поэтому погрешность результата, вычисленного с шагом h, примерно в 15 раз меньше, чем разность между этим результатом и результатом, вычисленным с шагом $2h$.

Последнее заключение обычно выражают в форме следующего практического правила:

В интеграле I_{2n} верных знаков на один больше, чем совпадающих знаков в I_n и I_{2n}.

Обратившись в качестве примера к интегралу, рассмотренному в примере 3.29, заметим, что при интегрировании с двойным шагом ($h=0{,}5$) получается значение интеграла 1,2086. Таким образом, разность между этими значениями составляет 0,0017. Это означает, что погрешность полученного там значения около 0,0001, так что, во всяком случае, все знаки в числе 1,207 можно считать верными. Более точные вычисления показывают, что так оно и есть.

§ 31. Общая постановка задачи нахождения линейной квадратурной формулы. Чебышевские квадратуры

Выведенные в предыдущих параграфах квадратурные формулы имели вид суммы ординат, взятых с некоторыми коэффициентами. Основная вычислительная работа идет на нахождение значений функции. Поэтому весьма существенно, что при том же количестве ординат переход от коэффициентов формулы трапеций к коэффициентам формулы парабол позволяет значительно увеличить точность.

Естественно поинтересоваться, нельзя ли еще более увеличить точность за счет некоторого дальнейшего изменения коэффициентов в квадратурной формуле при тех же ординатах.

Другой резерв повышения точности квадратурных формул — в выборе самих ординат. Точки, в которых вычисляются значения функции для формул трапеций и парабол, определяются тем, что отрезок интегрирования делится на n равных частей. Между тем, такое деление далеко не всегда выгодно. На тех участках, где функция изменяется медленно и плавно, точки деления можно было бы брать более редкими, а на участках с быстрым изменением — более частыми. Во всяком случае, сами точки, ординаты в которых входят в квадратурную формулу, тоже можно выбирать в различных местах участков интегрирования.

Таким образом, мы приходим к мысли искать квадратурные формулы в виде

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i). \tag{1.31}$$

Формулу вида (1.31) называют *линейной квадратурной формулой*, коэффициенты w_i — *весами* этой формулы, а точки x_i — ее *узлами*.

В формулах трапеций и парабол узлы определялись их количеством n, так как были расположены равномерно. Заменяя функцию интерполяционным многочленом более высокой степени, можно получить новые квадратурные формулы с теми же узлами, но другими весами, которые будут давать бо́льшую точность, поскольку остаточный член интерполяции, а значит и остаточный член интегрирования, будет более высокого порядка. Исходя из этих соображений, квадратурные формулы строят таким образом, чтобы они точно интегрировали многочлены возможно более высокой степени. Наивысшая степень многочлена, точно интегрируемого данной квадратурной формулой, и служит характеристикой точности этой формулы.

Заметим, что нет никакой необходимости искать квадратурные формулы для произвольного отрезка $[a, b]$.

Действительно, если функция $f(x)$ определена и непрерывна на отрезке $[a, b]$, то, сделав замену переменных $t = 2(x-a)/(b-a) - 1$, получим:

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\int_{-1}^{1} f[x(t)]\,dt,$$

так что всегда можно ограничиваться рассмотрением квадратурных формул для отрезка $[-1, 1]$.

Вывод формул трапеций и парабол сводится к подбору весов при заданных узлах. Примером простой линейной квадратурной формулы, вывод которой исходит из иных соображений, является квадратурная формула Чебышева. Здесь, наоборот, ищутся наилучшие узлы при некоторых предположениях относительно весов.

Именно, в квадратурной формуле Чебышева предполагается, что все веса равны между собою. Такое предположение заметно облегчает вычисления. Итак, будем искать для функции $f(x)$, заданной на отрезке $[-1, 1]$, квадратурную формулу вида

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i) = w\sum_{i=1}^{n} f(x_i) \tag{2.31}$$

с фиксированным числом узлов n, полагая $w_1 = w_2 = \ldots = w_n = w$. Узлы для формулы (2.31) подбираются таким образом, чтобы равенство (2.31) было точным для многочленов возможно более высокой степени.

Так как в формулу (2.31) входят всего $n+1$ неизвестных (n узлов $x_1, \ldots, x_n$ и вес w), то мы имеем возможность удовлетворить $n+1$ уравнениям. Выберем их так, чтобы формула (2.31) была точна для функций

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

Полагая $f(x) \equiv 1$, найдем

$$\int_{-1}^{1} 1\,dx = w\sum_{i=1}^{n} 1,$$

откуда $w_n = 2$ и $w = 2/n$. Теперь остается найти узлы. Полагая $f(x) = x, x^2, \ldots, x^n$ и учитывая, что для нечетных степеней $k = 2s+1$

$$\int_{-1}^{1} x^k dx = \int_{-1}^{1} x^{2s+1} dx = 0,$$

а для четных $k=2s$

$$\int_{-1}^{1} x^k dx = \frac{2}{k+1},$$

придем к следующей системе уравнений относительно искомых узлов $x_1, \ldots, x_n$ чебышевской квадратуры:

$$x_1+x_2+\ldots+x_n=0,$$
$$x_1^2+x_2^2+\ldots+x_n^2=n/3,$$
$$\ldots\ldots\ldots\ldots$$
$$x_1^n+x_2^n+\ldots+x_n^n=\begin{cases} 0 & \text{при } n \text{ нечетном,} \\ n/(n+1) & \text{при } n \text{ четном.} \end{cases}$$

Таблица 1.31

(1)	(2)	(3)	(4)
n	k	x	w
2	1	−0,577350	1
	2	0,577350	
3	1	−0,707107	0,666667
	2	0	
	3	0,707107	
4	1	−0,794654	0,50
	2	−0,187592	
	3	0,187592	
	4	0,794654	
5	1	−0,832498	0,4
	2	−0,374541	
	3	0	
	4	0,374541	
	5	0,832498	
6	1	−0,866247	0,333333
	2	−0,422519	
	3	−0,266635	
	4	0,266635	
	5	0,422519	
	6	0,866247	
7	1	−0,883862	0,285714
	2	−0,529657	
	3	−0,323912	
	4	0	
	5	0,323912	
	6	0,529657	
	7	0,883862	

Найденную систему n уравнений с n неизвестными можно привести к одному алгебраическому уравнению n-й степени, корни которого служат узлами квадратурной формулы Чебышева. Не проводя этих преобразований, ограничимся готовыми результатами. В табл. 1.31 приведены значения узлов чебышевской квадратуры для некоторых значений n.

Отметим, что для $n=8$ корни соответствующего многочлена оказываются мнимыми, так что в этом случае построить квадратурную формулу не удается. При $n=9$ формула может быть построена (соответствующие узлы определены), а для бо́льших значений n вопрос о существовании действительных узлов квадратурной формулы Чебышева пока остается открытым.

Рассмотрим пример применения формулы Чебышева.

Пример 1.31. Вычислим с помощью квадратурной формулы Чебышева с 5 узлами интеграл

$$\int_{-1}^{1} \frac{dx}{1+x^2}.$$

Вычисления приведены в табл. 2.31. Как видим, результат имеет

Таблица 2.31

(1)	(2)	(3)	(4)	(5)
x	«1» + (1)²	«1» : (2)	w	(3) · (4)
−0,832498	1,693053	0,590649		
−0,374541	1,140281	0,876977		
0	1	1,000000		
0,374541	1,140281	0,876977		
0,832498	1,693053	0,590649		
	Σ	3,935252	0,4	1,574101

почти такую же точность, как при использовании формулы парабол с 10 узлами (точное значение интеграла $\pi/2$).

§ 32. Квадратурные формулы Гаусса

Если в линейной квадратурной формуле общего вида (1.31) считать все узлы и веса свободными параметрами, то можно получить квадратурные формулы, позволяющие точно интегрировать многочлены достаточно высокой степени. Такие формулы обычно называют квадратурными формулами *гауссового типа*.

Как и в предыдущем параграфе, мы можем ограничиться рассмотрением отрезка $[-1, 1]$, записывая квад-

ратурную формулу в виде

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i). \qquad (1.32)$$

Правая часть формулы (1.32) содержит $2n$ свободных параметров, что позволяет получить $2n$ различных условий. Поэтому естественно ожидать, что надлежащий выбор этих параметров позволит получить квадратурную формулу, точно интегрирующую многочлен степени $2n-1$, имеющий такое же количество коэффициентов. Как мы убедимся, это действительно возможно.

Начнем со следующего интересного результата, представляющего и самостоятельный интерес.

Теорема 1. *Каковы бы ни были произвольные различные узлы* $x_1, x_2, \ldots, x_n$ *на отрезке* $[-1, 1]$, *существует единственная система весов* $w_1, w_2, \ldots, w_n$, *обеспечивающая* (*с этими узлами*) *точное интегрирование по формуле* (1.32) *многочленов до* $(n-1)$-*й степени включительно.*

Доказательство. Пусть $x_1, x_2, \ldots, x_n$ — фиксированная система n узлов. Обозначим $f_k(x) = x^k$ $(k = 0, 1, \ldots, n-1)$ и потребуем, чтобы равенство (1.32) было точным для всех $f_k(x)$. Тогда для неизвестных узлов $w_1, w_2, \ldots, w_n$ получим систему n уравнений

$$\left.\begin{aligned} w_1 + w_2 + \ldots + w_n &= 2/1, \\ w_1 x_1 + w_2 x_2 + \ldots + w_n x_n &= 0, \\ w_1 x_1^2 + w_2 x_2^2 + \ldots + w_n x_n^2 &= 2/3, \\ &\ldots\ldots\ldots\ldots \end{aligned}\right\} \qquad (2.32)$$

Последнее уравнение этой системы, получающееся для функции $f_{n-1}(x) = x^{n-1}$, имеет вид

$$w_1 x_1^{n-1} + w_2 x_2^{n-1} + \ldots + w_n x_n^{n-1} = \begin{cases} 0 & \text{при } n \text{ четном}, \\ 2/n & \text{при } n \text{ нечетном}. \end{cases}$$

Итак, для нахождения весов w_k квадратурной формулы мы получили неоднородную систему из n линейных уравнений с n неизвестными. В соответствии с известным правилом Крамера, такая система совместна и имеет единственное решение тогда и только тогда, когда определитель системы отличен от нуля. Но определитель системы (2.32) имеет вид

$$D = \begin{vmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_n \\ x_1^2 & x_2^2 & \ldots & x_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ x_1^{n-1} & x_2^{n-1} & \ldots & x_n^{n-1} \end{vmatrix}. \qquad (3.32)$$

Такой определитель называется *определителем Вандермонда*. Вычислим его.

Вывести требуемую формулу проще всего индукцией по порядку определителя. Для определителей второго и третьего порядка легко непосредственным вычислением установить равенства

$$D_2=\begin{vmatrix} 1 & 1 \\ x_1 & x_2 \end{vmatrix}=x_2-x_1 \quad \text{и} \quad D_3=\begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{vmatrix}=$$

$$=(x_2-x_1)(x_3-x_1)(x_3-x_2)=\prod_{\substack{i>j \\ 1\leqslant i,\ j\leqslant 3}}(x_i-x_j).$$

Предположим, что равенство

$$D_n=\begin{vmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \\ \dots & \dots & \dots & \dots \\ x_1^{n-1} & x_2^{n-1} & \dots & x_n^{n-1} \end{vmatrix}=\prod_{\substack{i>j \\ 1\leqslant i,\ j\leqslant n}}(x_i-x_j)$$

доказано для определителей всех порядков до D_{n-1} включительно.

Рассмотрим определитель D_n как функцию переменной x_n. Очевидно, что это есть многочлен степени $n-1$. Столь же очевидно, что при $x_n=x_1$, $x_n=x_2$, ..., $x_n=x_{n-1}$ этот многочлен обращается в нуль, так как у него оказываются два равных столбца. Но тогда x_1, x_2, ..., x_{n-1} являются корнями этого многочлена и, как известно из алгебры, его можно представить в виде

$$D_n=A(x_n-x_1)(x_n-x_2)\dots(x_n-x_{n-1}),$$

где A — коэффициент при старшей степени многочлена.

Из выражения определителя D_n видно, что $A=D_{n-1}$. Таким образом, мы приходим к равенству

$$D_n=(x_n-x_1)\dots(x_n-x_{n-1})\,D_{n-1},$$

а так как для D_{n-1} его выражение через произведения разностей предполагается известным, то и для D_n получаем

$$D_n=\prod_{\substack{i>j \\ 1\leqslant i,\ j\leqslant n}}(x_i-x_j),$$

что и утверждалось.

Таким образом, определитель системы (2.32) наверняка отличен от нуля, если все заданные узлы р а з л и ч н ы. Тем самым теорема 1 доказана.

Для дальнейших построений нам понадобится знакомство с системой *многочленов Лежандра* и некоторыми их свойствами. Первые из этих многочленов определяются

равенствами

$$P_0(x) \equiv 1, \qquad P_3(x) = (5/2)\,x^3 - (3/2)x,$$
$$P_1(x) = x, \qquad P_4(x) = (35/8)\,x^4 - (15/4)\,x^2 + 3/8,$$
$$P_2(x) = (3/2)\,x^2 - 1/2, \quad \ldots\ldots\ldots\ldots\ldots$$

Для любого натурального индекса n многочлен Лежандра $P_n(x)$ можно определить равенством

$$P_n(x) = \frac{1}{2^n \cdot n!} \frac{d^n}{dx^n}[(x^2-1)^n]. \tag{4.32}$$

Из формулы (4.32) видно, что $P_n(x)$ есть многочлен n-й степени. В самом деле, в квадратной скобке стоит многочлен степени $2n$, который дифференцируется n раз, а при каждом дифференцировании степень многочлена понижается ровно на единицу.

Нам потребуются следующие свойства многочленов Лежандра.

1°. *Многочлен Лежандра $P_n(x)$ имеет n действительных корней на интервале* $(-1, 1)$.

Доказательство. Рассмотрим функцию $\varphi_0(x) = (x^2-1)^n$. Она обращается в нуль в точках $x = \pm 1$ и, следовательно, $\varphi_1(x) = \varphi_0'(x)$ обращается в нуль по крайней мере в одной точке внутри этого интервала, вследствие теоремы Ролля. Но $\varphi_1(x)$ содержит множитель $(x^2-1)^{n-1}$ и поэтому вместе с $\varphi_0(x)$ обращается в нуль при $x = \pm 1$.

Таким образом, для функции $\varphi_1(x)$ отрезок $[-1, 1]$ разбивается на два участка, на концах которых $\varphi_1(x)$ обращается в нуль. Отсюда, в силу теоремы Ролля, следует, что $\varphi_2(x) = \varphi_1'(x)$ имеет по крайней мере два нуля внутри интервала и по-прежнему нули на концах. Заметив, что $\varphi_2(x) = \varphi_0''(x)$ и продолжая аналогичные рассуждения, найдем, что для $\varphi_{n-1}(x) = \varphi_0^{(n-1)}(x)$ отрезок $[-1, 1]$ разбивается на n участков, на концах которых $\varphi_0^{(n-1)}(x)$ обращается в нуль. Отсюда следует, что $\varphi_n(x) = \varphi_0^{(n)}(x)$ имеет не менее n корней внутри отрезка $[-1, 1]$.

Как видно из формулы (4.32), многочлен $P_n(x)$ отличается от $\varphi_0^{(n)}(x)$ лишь постоянным множителем. Но тогда он имеет ровно n корней, так как является многочленом n-й степени.

2°. *Если n — натуральное и $k = 0, 1, 2, \ldots, n-1$, то*

$$\int_{-1}^{1} x^k P_n(x)\,dx = 0. \tag{5.32}$$

Доказательство. Подставив в интеграл (5.32) выражение (4.32) для $P_n(x)$, получим

$$\int_{-1}^{1} x^k P_n(x)\,dx = \frac{1}{2^n n!} \int_{-1}^{1} x^k \frac{d^n}{dx^n}\left[(x^2-1)^n\right] dx =$$

$$= \frac{1}{2^n \cdot n!} \int_{-1}^{1} x^k \varphi_0^{(n)}(x)\,dx = \frac{1}{2^n \cdot n!} \int_{-1}^{1} x^k \varphi_n(x)\,dx,$$

где $\varphi_0(x) = (x^2-1)^n$, как и при доказательстве предыдущего свойства. Интегрируя последний интеграл по частям, находим

$$\int_{-1}^{1} x^k P_n(x)\,dx = \frac{1}{2^n n!} x^k \varphi_0^{(n-1)}(x)\Big|_{-1}^{1} - \frac{k}{2^n \cdot n!} \int_{-1}^{1} x^{k-1} \varphi_{n-1}(x)\,dx.$$

Вследствие замеченного выше обращения в нуль функции $\varphi_0(x)$ вместе со своими производными до $\varphi_0^{(n-1)}(x) = \varphi_{n-1}(x)$ включительно в точках $x = \pm 1$, найдем, что проинтегрированный член обратится в нуль. Так как $k \leqslant n-1$, то $n-k-1 \geqslant 0$. Интегрируя по частям k раз и замечая, что проинтегрированный член каждый раз будет обращаться в нуль, получим, что интеграл (5.32) равен

$$\int_{-1}^{1} x^k P_n(x)\,dx = \frac{(-1)^k k!}{2^n \cdot n!} \varphi_{n-k-1}(x)\Big|_{-1}^{1} = 0.$$

Равенство (5.32) доказано.

Установив эти два свойства многочленов Лежандра, возвратимся к рассмотрению квадратурных формул.

Выберем n узлов квадратурной формулы (1.32) так, чтобы они совпадали с n действительными корнями многочлена Лежандра $P_n(x)$, и определим веса $\{w_i\}$ этой квадратурной формулы из системы (2.32). Такую квадратурную формулу назовем *квадратурой Гаусса* порядка n. Для нее справедливы следующие теоремы.

Теорема 2. *Квадратура Гаусса порядка n точна для многочленов степени до $2n-1$ включительно.*

Доказательство. По теореме 1 формула (1.32) с выбранными узлами и весами точна для многочленов до степени $n-1$. Остается доказать, что многочлены со степенями от n до $2n-1$ также интегрируются точно. Для этого покажем сначала, что любой

многочлен степени $2n-1$ можно представить в виде

$$Q_{2n-1}(x) = M_{n-1}(x) + \sum_{s=0}^{n-1} c_s x^s P_n(x). \tag{6.32}$$

В самом деле, разделив многочлен $Q_{2n-1}(x)$ степени $2n-1$ на многочлен $P_n(x)$ степени n, получим в частном и в остатке многочлены степени $n-1$. Обозначив остаток через $M_{n-1}(x)$ и записав частное в виде $\sum_{s=0}^{n-1} c_s x^s$, придем к равенству (6.32).

Заметим теперь, что всякий многочлен вида $x^s P_n(x)$ интегрируется нашей формулой точно. Действительно, вследствие (5.32)

$$\int_{-1}^{1} x^s P_n(x)\,dx = 0 \quad (s \leqslant n-1).$$

С другой стороны,

$$\sum_{i=1}^{n} w_i f(x_i) = \sum_{i=1}^{n} w_i x_i^s P_n(x_i) = 0,$$

так как x_i суть корни многочлена $P_n(x)$ и поэтому квадратурная формула дает точное значение интеграла. Но тогда каждое слагаемое правой части (6.32) интегрируется гауссовой квадратурой точно, а значит, точно интегрируется и любой многочлен $(2n-1)$-й степени.

Теорема 3. *Если линейная квадратурная формула с n узлами точно интегрирует произвольный многочлен $(2n-1)$-й степени, то она совпадает с формулой Гаусса.*

Доказательство. Обозначим через $\{\alpha_i\}$ узлы квадратуры Гаусса и через $\{\beta_i\}$ узлы некоторой другой квадратуры с весами $\{\overline{w}_i\}$, точно интегрирующей многочлены до степени $2n-1$. Объединив все точки $\{\alpha_i\}$ и $\{\beta_i\}$, получим узлы $x_1, x_2, \ldots, x_m$, причем $m \leqslant 2n$, так как некоторые узлы этих двух квадратур могут совпадать.

По теореме 1 найдем единственным образом веса $\{v_i\}$, обеспечивающие для квадратурной формулы

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{m} v_i f(x_i)$$

точное интегрирование для многочленов степени до $m-1 \leqslant 2n-1$. Но такую формулу можно получить из предыдущей, полагая $v_i = w_i$, если соответствующий узел x_i совпадает с узлом α, и $v_i = 0$ в противном случае: это есть первоначальная гауссова квадратура, обладающая нужной точностью. С другой стороны, квадратура такой же точности получится, если полагать $v_i = \overline{w}_i$ для узлов x_i, совпадающих

с β_i, и $v_i=0$ для остальных. Так как, вследствие теоремы 1, квадратура такой точности единственна, то вторая квадратурная формула должна совпадать с гауссовой, чем теорема 3 доказана.

Здесь, как и в предыдущем параграфе, мы брали в качестве «стандартного» участка интегрирования отрезок $[-1, 1]$. Нередко бывает удобно переходить к отрезку $[0, 1]$. Формула замены переменных выглядит здесь несколько проще. Именно, если переменная принадлежит $[a, b]$, то переменная t, определенная равенством

$$x=(b-a)\,t+a \tag{7.32}$$

или

$$t=(x-a)/(b-a), \tag{8.32}$$

будет изменяться уже на $[0, 1]$.

Уравнения для отыскания узлов и весов квадратурных формул будут в этом случае отличаться от уравнений для отрезка $[-1, 1]$. Мы не станем их выписывать; принципы их получения должны быть ясны читателю из предыдущего. Таблицы узлов и весов могут вычисляться как для $[-1, 1]$, так и для $[0, 1]$. Последние даже несколько предпочтительнее, благодаря более простым формулам перехода (7.32) и (8.32).

При практическом вычислении интегралов нет нужды преобразовывать выражение подынтегральной функции к новой переменной. Проще преобразовать узлы t_i для $[0, 1]$, взятые из таблицы, в узлы x_i на $[a, b]$ по формуле (7.32). Тогда квадратурная формула примет вид

$$\int_a^b f(x)\,dx=(b-a)\sum_{i=1}^{n} A_i f[a+(b-a)\,t_i]. \tag{9.32}$$

Пример 1.32. Вычислим с помощью формулы Гаусса с $n=3$ интеграл

$$\int_0^1 \frac{dx}{1+x^2}.$$

Вычисления приведены в табл. 1.32, где в колонках (1) – (5) приведены узлы и веса гауссовой квадратуры для этого случая. Более подробные таблицы приведены в сле-

Таблица 1.32

(1)	(2)	(3)	(4)	(5)
x_i	«1» + (1)²	«1» / (2)	A_i	(3) · (4)
0,1127 0167	1,0127 0167	0,9874 5764	0,2777 7778	0,2742 9379
0,5	1,25	0,8	0,4444 4444	0,3555 5556
0,8872 9833	1,7872 9833	0,5595 0368	0,2777 7778	0,1554 1769
				$\Sigma = 0{,}7852\ 6704$

дующем параграфе (см. табл. 1.32). Расчет показывает, что квадратурная формула Гаусса с тремя узлами дает в данном примере относительную погрешность $\delta = 0{,}02\%$.

§ 33. Практические приемы оценки точности. Уточняющие квадратуры

В § 30 мы уже рассматривали вопрос о точности квадратурных формул и приводили выражения для остаточного члена формул трапеций и парабол. Аналогичное выражение можно получить и для остаточного члена формулы Гаусса; он будет содержать производную от интегрируемой функции порядка $2n$, т. е. $f^{(2n)}(\xi)$. Ясно, что оценка этой производной более чем затруднительна.

Для практической оценки точности обычно пользуются приемом, также упомянутым нами в § 30, — сравнением величин интеграла, полученных с помощью различных квадратурных формул. Однако для формулы Гаусса, по сравнению с формулами трапеций и парабол, при таком сравнении возникает довольно серьезное затруднение. Как мы уже знаем, основная часть вычислительной работы при численном интегрировании падает на вычисление значений функции в узлах. Так как узлы формул трапеций и парабол расположены на отрезке интегрирования равномерно, то, выбрав их число четным (или кратным четырем), мы при уменьшении числа узлов вдвое получим возможность пользоваться уже вычисленными ординатами. Если же пользоваться этим приемом для формулы Гаусса, то все

вычисления придется проводить заново, так как узлы гауссовых квадратур с n и $2n$ узлами совсем не совпадают между собою.

Тем не менее, основным практически используемым приемом оценки точности численного интегрирования является сравнение результатов, полученных с помощью двух различных квадратурных формул. Поэтому, выбрав количество n узлов, в которых будут вычисляться значения функции, мы можем воспользоваться ими различными способами. Можно использовать все n узлов для одной квадратурной формулы. Тогда наилучшей будет формула Гаусса, которая позволяет точно интегрировать многочлены до степени $2n-1$ включительно, но в этом случае мы не будем в состоянии судить о точности полученного результата. Целесообразнее поступить иначе, распределив эти узлы между двумя различными квадратурными формулами. Тогда степень точно интегрируемых многочленов окажется существенно ниже, зато мы сможем получить представление о порядке допускаемой погрешности.

Естественно поставить следующий вопрос. Пусть некоторый интеграл вычислен по формуле Гаусса с n узлами. Как построить новую, возможно более точную квадратурную формулу с $2n+1$ узлами так, чтобы использовать уже имеющиеся n узлов (и уже вычисленные значения функции в этих узлах) и добавить лишь $n+1$ новых.

Решением этого вопроса является уточняющая квадратурная формула, предложенная А. С. Кронродом. Уточняющая квадратура имеет $2n+1$ узлов и является точной для многочленов до степени $3n+1$ при n четном и $3n+2$ при n нечетном. Узлы этой квадратуры являются корнями многочлена $K_{2n+1}(x)=P_n(x)\,Q_{n+1}(x)$, где $P_n(x)$ — многочлен Лежандра. Отсюда следует, что n узлов уточняющей квадратуры являются узлами квадратурной формулы Гаусса. Остальные $n+1$ узлов, являющиеся корнями многочлена $Q_{n+1}(x)$, расположены между ними.

В практике вычислений можно полагать, что погрешность, даваемая уточняющей квадратурой Кронрода, примерно на два порядка меньше разности между результатами, полученными по этой формуле и по квадратуре Гаусса с n узлами.

Узлы и веса квадратурных формул Гаусса и уточняющих квадратур Кронрода приведены в табл. 1.33. Веса гауссовой квадратуры приводятся лишь против тех узлов, которые одновременно являются гауссовыми узлами.

Заметим, кстати, что различие в трудоемкости вычислений значений функции в «круглых» и «некруглых» узлах,

Таблица 1.33

(1)	(2)	(3)	(4)	(5)
n	k	x	A_i	K_i
2	1	0,03708995		0,09898990
	2	0,21132487	0,50000000	0,24545455
	3	0,50000000		0,31111110
	4	0,78867513	0,50000000	0,24545455
	5	0,96291005		0,09898990
3	1	0,01975437		0,05232811
	2	0,11270167	0,27777778	0,13424404
	3	0,28287813		0,20069871
	4	0,50000000	0,44444444	0,22545828
	5	0,71712187		0,20069871
	6	0,88729833	0,27777778	0,13424404
	7	0,98024563		0,05232811
4	1	0,01171987		0,03148869
	2	0,06943184	0,17392742	0,08502680
	3	0,17985689		0,13339917
	4	0,33000948	0,32607258	0,16347459
	5	0,50000000		0,17322150
	6	0,66999052	0,32607258	0,16347459
	7	0,82014311		0,13339917
	8	0,93056816	0,17392742	0,08502680
	9	0,98828013		0,03148869
5	1	0,00795732		0,02129102
	2	0,04691008	0,11846344	0,05761666
	3	0,12291664		0,09340040
	4	0,23076534	0,23931434	0,12052017
	5	0,36018479		0,13642490
	6	0,50000000	0,28444444	0,14149370
	7	0,63981521		0,13642490
	8	0,76923466	0,23931434	0,12052017
	9	0,87708336		0,09340040
	10	0,95308992	0,11846344	0,05761666
	11	0,99204268		0,02129102

имеющее значение при ручных расчетах, при работе на электронных вычислительных машинах полностью исчезает. Поэтому при работе на машине формулы трапеций и парабол теряют и те последние преимущества, которые они имеют при ручных расчетах сравнительно с формулами §§ 31 — 33.

Пример 1.33. Вычислим тот же интеграл $\int_0^1 \frac{dx}{1+x^2}$ при $n=2$, пользуясь формулой Гаусса и уточняющей ква-

Таблица 2.33

(1)	(2)	(3)
x_i	«1» $+(1)^2$	$\frac{\text{«1»}}{(2)}$
0,0370 8995	1,0013 7566	0,9986 2623
0,2113 2487	1,0446 5820	0,9572 5090
0,5	1,25	0,8
0,7886 7513	1,6220 0846	0,6165 1960
0,9629 1005	1,9271 9576	0,5188 8865

(4)	(5)	(6)	(7)
A_i	(3) · (4)	K_i	(3) · (6)
		0,0989 8990	0,0988 5391
0,5	0,4786 2545	0,2454 5455	0,2349 6159
		0,3111 1110	0,2488 8888
0,5	0,3082 5980	0,2454 5455	0,1513 2754
		0,0989 8990	0,0513 6474
	$\Sigma=0,7868\ 8525$		$\Sigma=0,7853\ 9666$

дратурой Кронрода. Результаты вычислений приведены в табл. 2.33. Сравнение полученных результатов с истинным значением интеграла подтверждает сделанное выше замечание о погрешности уточняющей квадратуры.

§ 34. Программирование формулы Симпсона

Рассмотрим программу для вычисления интеграла $\int_a^b f(x)\,dx$ по формуле Симпсона, предполагая, что известны пределы интегрирования a, b и шаг интегрирования h, а блок вычисления подынтегральной функции $f(x)$ написан как стандартный блок с входной ячейкой α и выходной γ. Значение интеграла будем помещать также в ячейку γ. Программу можно написать в виде арифметического цикла с проверкой окончания по достижении абсциссой правого конца b отрезка интегрирования:

```
        Ω = конец
        0 = Σ
  h · «1/3» = Δ
        a = α
      f(x)
  Σ +   γ = Σ   ↑
  α +   h = α   |
      f(x)      |
    γ · «4» = R |
  Σ +   R = Σ   |
  α +   h = α   |
      f(x)      |
  Σ +   γ = Σ   |
  b −   α = 0   |
УО              |
  Σ · Δ = γ     |
    конец
```

Как видно из написанной программы, мы пользуемся формулой (5.29) для участка (x_0, x_0+2h), начиная с $x_0=a$. Если $\alpha<b$, то вычисленное для этой точки значение функции прибавляется к Σ снова, как значение на левом конце нового участка.

При целых или двоично-рациональных a, b и двоично-рациональном h эта программа будет работать без ошибок. В противном случае она может дать неверные результаты, так как возможны ошибки при проверке окончания цикла. Действительно, если a, b и h выражаются бесконечными двоичными дробями, то при записи в машине их придется округлить. При этом сумма $a+nh$ может не совпасть с b с машинной точностью, а оказаться хотя бы на единицу последнего знака меньше или больше, чем b.

Если случится, что

$$a+nh<b,$$

то проверка окончания снова передает управление на рабочую часть цикла, так что интеграл будет вычислен не по тому участку, по которому следует. Кроме того, может оказаться, что функция $f(x)$ не определена за пределами заданного участка, поэтому программа может дать совсем нелепый результат или выйти на аварийный останов в блоке счета $f(x)$. То же самое может случиться, если

$$a+nh>b,$$

так как придется вычислять функцию в точке за пределами участка интегрирования, т. е., быть может, за пределами области определения функции.

Чтобы избежать этих ошибок, необходимо изменить проверку окончания. Для этого нужно сравнивать очередное значение α не с b, а с некоторым эталоном, отличающимся от b меньше, чем на часть шага, например, на величину $\Delta=\frac{h}{3}$, которая все равно вычисляется в программе. Полезно, кроме того, заменить такую точку на b, чтобы не вычислять функцию вне участка интегрирования, если получится

$$a+nh>b.$$

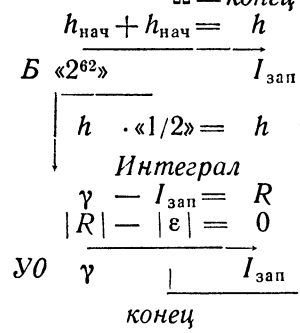
Программу с измененной проверкой окончания можно теперь написать так:

```
                Ω  = конец
                0  = Σ
          h · «1/3» = Δ
          b — Δ    = Эт
                a  = α
              f(x)
□         Σ + γ    = Σ
          α + h    = α
              f(x)
          γ · «4»  = R
          Σ + R    = Σ
          α + h    = α
          α — Эт   = 0
У1        ──────────────
                b  = α |
              f(x)     |
          Σ + γ    = Σ
          α ↛ b    = 0
У0          □
          Σ · Δ    = γ
            конец
```

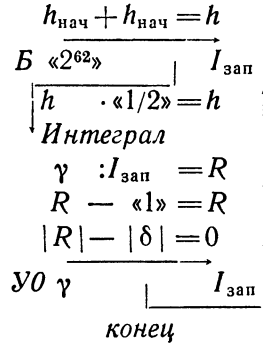
Написанная программа предполагает, что шаг интегрирования h уже выбран. Легко написать программу с автоматическим выбором шага для достижения нужной точности. Проверка точности результатов интегрирования производится путем сравнения результатов, полученных при интегрировании с различными шагами.

Удобно написать такой цикл сравнения с обращением к уже написанной программе как к блоку под названием *Интеграл*. Будем вычислять интеграл с некоторым шагом $h_{\text{нач}}$ и $2h_{\text{нач}}$. Если полученные значения с заданной точностью совпадают, то интеграл с шагом $h_{\text{нач}}$ дает ответ. В противном случае мы будем уменьшать шаг вдвое до тех пор, пока совпадение с нужной точностью не будет достигнуто.

Программу можно написать следующим образом:



Как видно из программы, проверка окончания производится по достижении заданной абсолютной точности ε совпадения интегралов, вычисленных с различным шагом. Не составляет труда написать такую же программу с проверкой окончания по достижении заданной относительной *) точности δ:



Приведенные нами программы трудно превратить в стандартные, так как слишком большое число команд приходится формировать. В блоке *Интеграл*, например, таких команд почти половина. Ясно, что формирование такого большого числа команд слишком удлинит про-

*) При этом, естественно, предполагается, что отличен от нуля не только вычисляемый интеграл, но и все его промежуточные значения.

грамму и для превращения в стандартную ее следует писать иначе.

Нетрудно написать программу интегрирования по Симпсону и в виде процедуры алгола. Особенно просто выглядит эта процедура в том случае, когда все значения функции уже вычислены и имеется массив значений y. Тогда процедуру интегрирования по формуле парабол можно записать так:

```
begin real procedure Sim (n, a, b, y);
        value a, b, n; real a, b; integer n; array y;
    begin real s; integer i;
        s:=(y[0]+y[n])/2;
        for i:=1 step 2 until n-1 do
        s:=s+2×y[i]+y[i+1];
        Sim:=2×(b-a)×s/(3×n)
    end Sim;
end
```

Впрочем, немногим сложнее написать указанную процедуру и для того случая, когда массив значений функции не вычислен заранее. Нужно только предположить, что в программе до обращения к процедуре *Sim* описан оператор функции $f(x)$, вычисляющий значение интегрируемой функции, где x играет роль формального параметра.

Будем считать, что заданное число участков разбиения записывается в виде $2n$. Требуемую процедуру можно представить следующим образом.

```
begin real procedure Sim (n, a, b); real a, b; integer n;
      begin real x, h, s; integer i;
          s:=0; x:=a; h:=(b-a)/(n×2);
          for i:=1 step 1 until n do
          begin s:=s+f(x)+4×f(x+h)+f(x+2×h);
              x:=x+2×h;
          end цикла;
          Sim:=s×h/3;
      end
end процедуры;
```

§ 35. Программирование квадратур Гаусса и уточняющих квадратур

Программировать квадратурную формулу Гаусса для определенного числа узлов весьма просто, однако требуется предварительно ввести в память узлы t_i и веса A_i для выбранного n. Как уже было сказано в § 32, формулу Гаусса для интеграла по произвольному отрезку $[a, b]$ можно записывать в виде

$$\int_a^b f(x)\,dx = (b-a)\sum_{i=1}^{n} A_i f[a + t_i(b-a)], \qquad (1.35)$$

где n — выбранное число узлов, t_i — абсциссы соответствующих узлов для отрезка $[0, 1]$, A_i — соответствующие веса.

Если считать, что написан стандартный блок «*Счет* $f(x)$», берущий аргумент в ячейке α, выдающий ответ в ячейку γ и сохраняющий регистр адреса, то требуемую программу, в свою очередь выдающую ответ в ячейку γ, для $n=5$ можно записать так

```
                Ω = конец
PA    0*    0   предконец
      b  —  a = R1
            0 = R2
      t1* · R1 = R3   ↑
      a  + R3 = α     |
        Счет f(α)     |
      γ0 · A1* = R3   |
      R2 + R3 = R2    |
                      |
PA < 4 |————————— 1*  ——|
      R2 · R1 = γ0
        предконец
        конец
```

Ясно, что при других значениях n достаточно в программе сменить лишь левый адрес в команде проверки окончания цикла и заменить узлы и веса. Написанную программу легко превратить в стандартную. Так как

адреса рабочих ячеек R, узлов t и весов A мы можем считать внутренними адресами программы, а значит известными, то формировать нужно всего лишь три команды, зависящие от внешней информации: две команды, содержащие пределы интегрирования, и команду обращения к счету функции $f(x)$.

Легко написать эту программу и в форме процедуры алгола. Так как эта процедура записывается для конкретного n, то ее формальными параметрами будут являться лишь пределы интегрирования. Кроме того, предполагается, что описан оператор функции $f(x)$, вычисляющий значения интегрируемой функции.

Процедуру интегрирования по Гауссу для $n=5$ можно записать так:

```
procedure GInt (a, b); real a, b;
begin real h, s; integer n; array t[1 : 5], A[1 : 5];
                h:=b−a; s:=0;
                for n:=1 step 1 until 5 do
                s:=s+A[n]×f(a+t[n]×h);
                GInt:=s×h;
end процедуры;
```

Трудности проверки точности интегрирования при использовании квадратурной формулы Гаусса были достаточно подробно разъяснены в § 33. Поэтому для автоматического достижения требуемой абсолютной или относительной точности следует пользоваться сравнением результатов, полученных по квадратурной формуле Гаусса и по соответствующей уточняющей квадратуре.

Гауссовы узлы находятся, как известно, среди узлов уточняющей квадратуры. Поэтому для программирования счета интеграла нужно иметь в памяти машины три массива констант: узлы уточняющей квадратуры t_i, среди которых находятся и гауссовы, веса гауссовых квадратур A_i и веса уточняющих квадратур K_i. Будем считать, что веса гауссовых квадратур занимают также $2n+1$ ячеек, причем веса для узлов, не являющихся гауссовыми, равны нулю.

Выбрав снова $n=5$, можем написать программу.

```
                    Интеграл
          Ω = конец                    γ0 · K1* = R6
   PA 0* 0    предконец                R2 + R5 = R2
        b — a = R1                     R3 + R6 = R3
            0 = R2                  PA < 12   M   1*
            0 = R3                     R2 · R1 = γ1
M     t1* · R1 = R4                    R3 · R1 = γ0
      a + R4 =  α                      предконец
F     Счет f (α)                       конец
      γ0 · A1* = R5
```

Эта программа, как легко убедится читатель самостоятельно, выдает в ячейку γ_1 интеграл, полученный по квадратурной формуле Гаусса, и в ячейку γ_0 — интеграл по уточняющей квадратуре Кронрода. Заметим, что программа содержит заведомо лишние умножения и сложения (при нулевых весах для негауссовых узлов), однако, организация обхода этих действий обойдется дороже.

Если сравнение полученных величин интегралов показывает, что достигнутая точность недостаточна, то переходить к большему n невыгодно, так как тогда пришлось бы держать в памяти машины большие таблицы узлов и весов для различных n. Обычно поступают иначе: разбивают участок интегрирования на части и применяют квадратурные формулы с тем же n для каждой из частей. Это можно сделать, включив написанный выше блок «Интеграл» внутрь цикла, пересылающего границы получающихся участков в ячейки $a_{\text{лев}}$, $a_{\text{прав}}$, заменяющие ячейки a и b в интеграле.

Некоторую трудность представляет при этом вычисление границ участков и их правильное расположение в памяти для удобства пересылки. Поручим эту работу отдельному блоку. Для работы этого блока, а также для формирования команды окончания наружного цикла понадобится счетчик числа участков. Сделаем счетчик фиксированным и для удобства формирования будем держать его в виде числа единиц первого (левого) адреса.

Программу интегрирования с нужной точностью можно записать так:

```
                                  Ω = конец
        РА          0*            0   предконец
                                  0 = s1
                                  0 = s2
                    (1,0,0)       |   Сч. Уч.
Нов     |           101 → ,Сч. Уч. = Сч. Уч.
        ↓             T0 + ,Сч. Уч. = T
ПГ                          Переработка границ
                                  a0* = a лев
                                  a1* = a прав
                                  Интеграл
                    s1 +          γ0 = s1
                    s2 +          γ1 = s2
T       (РА < n − 1   ПГ      1*)        н. п.
E                   s1 −      s2 = Δ
                    |Δ| −     |ε| = 0
        У0          Нов
                              s1 = γ0
                              s2 = γ1
                          предконец      н. п.
                            конец        н. п.
T0      РА < F            ПГ  1*
```

Написанная программа будет удваивать число частей участка интегрирования до тех пор, пока разность между значениями интеграла, полученного по формулам Гаусса и Кронрода, не сделается меньше требуемой погрешности ε. Величина ε выбирается программистом в зависимости от заданной точности, в соответствии со сказанным в § 33.

Для завершения программы остается написать блок «Переработка границ». Построим его таким образом. При значении счетчика числа участков *Сч. Уч.* = (1, 0, 0) этот блок должен лишь переслать в ячейки a_0, a_1 заданные границы первоначального участка интегрирова-

ния $[a, b]$. В дальнейшем между каждой парой точек вставляется их среднее арифметическое. Сначала новый массив строится на рабочем поле в ячейках m_0, m_1, ..., а затем снова пересылается в те же ячейки a_0, a_1, ...

Так как этот блок является внутренним и ни к какому другому не обращается, то для экономии места и времени мы не будем в нем пересылать Ω в *конец*, а закончим его передачей управления.

```
Переработка границ |  Сч. Уч. ↛ (1, 0, 0) = 0
                   |  ──────────────────────→
A                  |  У0   a          Я + 2   = a_0
                   |  ──────────────────────→
B                  |  Б    b          Ω       a_1
                   |  РА   0*         0       Ω − 1
                   |       77   →,    Сч. Уч. = R_0
                   |       M_0  +,    R_0     = M
                   |       50   →,    Сч. Уч. = R_0
                   |       U_0  +,    R_0     = W
                   |       N_0  +,    Сч. Уч. = N
                   |                   U_0     = U
                   |  ──────────────────────→
                   |  Б    V_0                 V
                   |  ┌──────────────────┐
□                  |  │    U  +, (0, 0, 2) = U
                   |  │    V  +, (0, 0, 2) = V
U                  |  │                (a_0^*  = m_0)   н. п.
                   |  │    a_0^*  +    a_1^*   = R_0
V                  |  │    (R_0   ·    «1/2»   = m_1)   н. п.
                   |  │                ┌──────────┐
M                  |  (РА < n − 1      │          1*)    н. п.
W                  |  │                (a_0^*  = m_посл) н. п.
                   |  РА   0            0       0
                   |  │                m_0^*   = a_0^*
N                  |  (РА < n          Я − 1    1*)     н. п.
                   |  Б                Ω − 1
U_0                |  │                a_0^*   = m_0
V_0                |  │     R_0   ·    «1/2»   = m_1
M_0                |  РА   < F         □        1*
N_0                |  РА   < 0         Я − 1    1*
```

Чтобы превратить программу в стандартную, нужно сформировать команды, зависящие от внешней информации. Их будет всего лишь четыре: команда обращения к функции в блоке «*Интеграл*», две команды, содержащие пределы интегрирования a, b, в блоке «*Переработка границ*» и команда проверки достижения нужной точности ($|\Delta|-|\varepsilon|=0$) в программе интегрирования с данной точностью.

Выпишем их отдельно. Это будут команды

F	*БВ*	*Я*+1	Счет $f(\alpha)$	Ω
A	*У0*	a	*Я*+2	a_0
B	*Б*	b	Ω	a_1
E		$\|\Delta\|$	$-\|\varepsilon\|=$	0

Будем считать, что обращение к программе интегрирования с нужной точностью имеет вид

БВ	*Я*+3	$\int_5$	Ω
	a	$f(\alpha)$	b
	0	0	ε

где $\int_5$ — заголовок программы, a, b — адреса ячеек, в которых записаны пределы интегрирования, $f(\alpha)$ — заголовок программы счета функции и ε — ячейка, по содержимому которой проверяется достижение нужной точности.

Для формирования команд воспользуемся заготовками

F_0	*БВ*	*Я*+1	0	Ω
A_0	*У0*	0	*Я*+2	a_0
B_0	*Б*	0	Ω	a_1
E_0		$\|\Delta\|-$	$\|0\|=$	0

Остается выполнить следующую программу

$$
\begin{array}{rrrr}
[PA] & 0^* & \Omega & BP \\
 & & F^* & =j \\
 & & (F-1)^* & =i \\
 & i \ \wedge & (0,\ F,\ 0) & =R \\
 & F_0 \ + , & R & =F \\
 & i \ \wedge & (F,\ 0,\ 0) & =R \\
 & A_0 \ + , & R & =A \\
 & \overline{130} \ \rightarrow , & i & =R \\
 & B_0 \ + , & R & =B \\
 & \overline{114} \ \rightarrow , & j & =R \\
 & E_0 \ + , & R & =E
\end{array}
$$

в работе которой читатель легко разберется самостоятельно.

ГЛАВА VI

ЧИСЛЕННОЕ РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

§ 36. Постановка задачи численного решения дифференциального уравнения с начальным условием. Метод Эйлера и его уточнение

Как известно из курса дифференциальных уравнений, лишь небольшое число типов уравнений первого порядка допускает интегрирование в квадратурах, т. е. сведéние к обычной операции интегрирования. Еще реже удается получить решение в элементарных функциях. Тем большее значение имеют численные методы решения дифференциальных уравнений, позволяющие вручную или с помощью вычислительной машины получить таблицу значений функции в требуемых точках.

Рассмотрим дифференциальное уравнение первого порядка, разрешенное относительно производной,

$$y' = f(x, y). \tag{1.36}$$

Общим решением такого уравнения является семейство функций $y = \varphi(x, C)$, зависящее от произвольного постоянного. Чтобы иметь возможность вычислить значение функции-решения в какой-либо точке x, необходимо выделить из этого семейства *частное решение*. Это делается с помощью задания начального условия вида

$$y|_{x=x_0} = y_0. \tag{2.36}$$

Нахождение решения, удовлетворяющего такому условию, называют *задачей Коши*. В дальнейшем, говоря о решении дифференциального уравнения первого порядка,

мы всегда будем иметь в виду уравнение вида (1.36) с начальным условием (2.36).

Итак, задача численного решения дифференциального уравнения первого порядка ставится следующим образом: требуется построить таблицу значений функции $y=\varphi(x)$, удовлетворяющей уравнению (1.36) и начальному условию (2.36) на отрезке $[a, b]$ с некоторым шагом h (обычно можно считать, что $a=x_0$, т. е. что начальное условие задано в левом конце заданного отрезка).

Простейшим из численных методов интегрирования дифференциальных уравнений является *метод Эйлера*. Обычно он применяется только для прикидочных расчетов, но идеи, положенные в его основу, являются исходными для широкого класса численных методов. Метод Эйлера основан на замене искомой функции многочленом первой степени, т. е. на линейной интерполяции. Впрочем, правильней говорить о линейной экстраполяции, так как речь идет о нахождении значений функции в соседних узлах, а не между узлами.

Выберем шаг h настолько малым, чтобы для всех x между x_0 и $x_1=x_0+h$ значения функции y мало отличались от линейной функции. Тогда на указанном интервале

$$y=y_0+(x-x_0)\,y_0'=y_0+(x-x_0)\,f(x_0,\ y_0),$$

где $y_0'=f(x_0, y_0)$ есть значение производной y' в точке $x=x_0$. Таким образом, кривая заменяется на этом участке отрезком прямой (*касательной* к кривой в начале участка). Для точки $x_1=x_0+h$ получим

$$y\,|_{x=x_1}=y_1=y_0+hy_0'.$$

Точно так же, для $x=x_2=x_1+h$ можно написать

$$y_2=y_1+hy_1'=y_1+hf(x_1,\ y_1).$$

Продолжая таким же способом строить дальнейшие значения функции, убедимся, что метод Эйлера можно представить в виде последовательного применения формул

$$\left.\begin{aligned}\Delta y_k&=y_k'h=f(x_k,\ y_k)\cdot h,\\ y_{k+1}&=y_k+\Delta y_k.\end{aligned}\right\}\qquad(3.36)$$

Геометрический смысл формулы Эйлера ясен из рис. 20. Интегральная кривая заменяется здесь ломаной, звенья которой имеют постоянную горизонтальную проекцию h. Первое звено касается искомой интегральной кривой в точке (x_0, y_0).

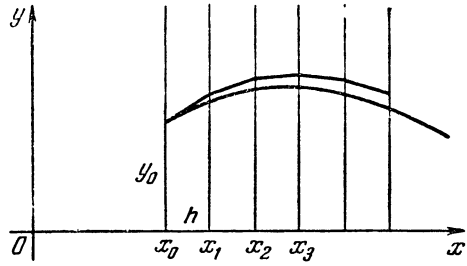


Рис. 20.

Формулы (3.36) можно получить и из иных соображений. Из (1.36) следует

$$y_{k+1}=y_k+\int\limits_{x_k}^{x_{k+1}} y'\,dx=y_k+\int\limits_{x_k}^{x_{k+1}} f(x,\ y(x))\,dx. \quad (4.36)$$

Полагая под интегралом функцию $f(x, y(x))$ равной постоянному значению $f(x_k, y_k)$ в левом конце интервала интегрирования (x_k, x_{k+1}), получим, что интеграл равен $f(x_k, y_k)h$, так что формула (4.36) превратится в (3.36).

Пример 1.36. Проинтегрируем методом Эйлера уравнение $y'=x-y$ с начальным условием $x_0=0$, $y_0=0$ на отрезке [0, 1] с шагом $h=0{,}1$.

Вычисления приведены в табл. 1.36. Первая строка в столбцах (1) и (2) заполнена по начальным данным. Затем вычисляется производная y' по уравнению $y'=x-y$, а затем — приращение $\Delta y=y'\cdot h$ в столбце (3). Теперь можно заполнить следующую строку в столбце (2), затем в столбце (4) и снова возвратиться к столбцу (3) и т. д.

Столбец (5) содержит таблицу значений точного решения $y=x-1+e^{-x}$ с двумя знаками. Сравнение показывает, что при $x=1$ относительная ошибка метода Эйлера составляет около 5,4%. В столбце (6) приведены значения, вычисленные по методу Эйлера с шагом $h=0{,}05$,

Таблица 1.36

(1)	(2)	(3)	(4)	(5)	(6)
x	y	Δy	y'	$y_{\text{точн}}$	
0	0		0	0	0
		0			
0,1	0,00		0,10	0,00	0,00
		0,01			
0,2	0,01		0,19	0,02	0,02
		0,02			
0,3	0,03		0,27	0,04	0,04
		0,03			
0,4	0,06		0,34	0,07	0,07
		0,03			
0,5	0,09		0,41	0,11	0,11
		0,04			
0,6	0,13		0,47	0,15	0,15
		0,05			
0,7	0,18		0,52	0,20	0,19
		0,05			
0,8	0,23		0,57	0,25	0,25
		0,06			
0,9	0,29		0,61	0,31	0,31
		0,06			
1	0,35			0,37	0,37

которые практически совпадают со значениями точного решения.

Уменьшение шага интегрирования, как показывает рассмотренный выше пример, является эффективным средством повышения точности. В принципе, таким путем можно достичь довольно хорошей точности. Иногда, при простой правой части уравнения, идут по этому пути, особенно если можно воспользоваться вычислительными машинами. Все-таки чаще стараются иметь дело с более точными методами.

Уточненный метод Эйлера при практически том же объеме вычислительной работы дает погрешность порядка h^2, вместо h в обычном методе Эйлера *), что достигается с помощью очень простого приема.

Возвратимся к формуле (4.36). При получении отсюда формулы (3.36) метода Эйлера мы полагали подынтегральную функцию $f(x, y(x))$ постоянной, равной ее значению $f(x_k, y_k)$ на левом конце участка. Более точное значение получится, если полагать $f(x, y(x))$ равной значению в центре участка. Так как значение производной между точками x_k и x_{k+1} не вычисляется, то мы возьмем двойной участок (x_{k-1}, x_{k+1}), заменив формулу (4.36) следующей:

$$y_{k+1} - y_{k-1} = \int_{x_{k-1}}^{x_{k+1}} f(x, y(x))\, dx. \qquad (5.36)$$

Центром интервала (x_{k-1}, x_{k+1}) является точка x_k. Поэтому, заменив в формуле (5.36) под интегралом функцию $f(x, y(x))$ ее значением в точке x_k, равным $y'_k = f(x_k, y_k)$, придем к формуле

$$y_{k+1} = y_{k-1} + 2hy'_k. \qquad (6.36)$$

Эта формула и выражает уточненный метод Эйлера. Однако она применима лишь при $k \geqslant 1$, а значение y_1 по ней получить нельзя. Следовательно, мы не можем выполнять и дальнейших вычислений, так как для нахождения y_2 надо иметь y'_1, для чего, в свою очередь, надо иметь значение y_1.

*) О погрешностях методов численного решения дифференциальных уравнений см. ниже, § 41.

Значение y_1 можно найти с помощью обычного метода Эйлера. Более точные результаты получатся, если сначала найти $y_{1/2}$ в точке $x_{1/2}=x_0+h/2$ по формуле (3.36), а затем искать y_1 по формуле (6.36) с шагом $h/2$. Иначе говоря, рекомендуется следующая последовательность действий:

$$y_{1/2}=y_0+(h/2)\,y_0'=y_0+(h/2)\,f(x_0,\ y_0),$$
$$y_1=y_0+hy_{1/2}'=y_0+hf(x_{1/2},\ y_{1/2}),$$
$$y_2=y_0+2hy_1'=y_0+2hf(x_1,\ y_1).$$

После этого дальнейшие вычисления идут уже по формуле (6.36) без изменений.

Таблица 2.36

(1)	(2)	(3)	(4)	(5)
x	y	Δy	y'	$y_{\text{точн}}$
0	0		0	0
		0		
0,05	0	0,005	0,050	0,001
0,1	0,005	0,019	0,095	0,005
0,2	0,019	0,036	0,181	0,019
0,3	0,041	0,052	0,259	0,041
0,4	0,071	0,066	0,329	0,070
0,5	0,107	0,079	0,393	0,107
0,6	0,150	0,090	0,450	0,149
0,7	0,197	0,101	0,503	0,197
0,8	0,251	0,110	0,549	0,249
0,9	0,307	0,119	0,593	0,307
1,0	0,370			0,368

Пример 2.36. Для сравнения рассмотрим то же уравнение $y'=x-y$ с теми же начальными условиями $x_0=0$, $y_0=0$ и проинтегрируем его уточненным методом Эйлера. Схема вычислений и расчеты для отрезка [0, 1] с шагом $h=0{,}1$ приведены в табл. 2.36. Разности здесь выписаны так, что они по-прежнему находятся между значениями функции, к которым они относятся.

Уточненный метод Эйлера дает при вычислениях с двумя знаками после запятой при $x=1$ значение $y=0{,}37$, что совпадает со значением точного решения. Таким образом, при той же вычислительной работе, что и в обычном методе Эйлера, его уточнение дает много лучшие результаты.

§ 37. Метод Адамса—Крылова

Рассмотренные в предыдущем параграфе метод Эйлера и его уточнение были основаны на замене искомой функции линейной функцией и экстраполяции этой последней за пределы того участка, где она известна. Метод Адамса, рассмотрению которого посвящен настоящий параграф, основан на той же идее экстраполяции, но берет за основу интерполяционный многочлен более высокой степени. Собственно говоря, существует целая серия формул, которые используют интерполяционные многочлены различных степеней и носят одно и то же название формул метода Адамса.

Будем рассматривать дифференциальное уравнение первого порядка $y'=f(x, y)$ с начальным условием $x=x_0$, $y=y_0$, выберем некоторый шаг интегрирования h и введем вспомогательную величину

$$\eta(x)=y'(x)\,h=f(x, y)\,h, \tag{1.37}$$

которую мы будем рассматривать лишь в узлах нашей таблицы.

Формулу (4.36) можно представить в виде

$$y_{k+1}=y_k+\int_{x_k}^{x_{k+1}} y'(x)\,dx=y_k+\frac{1}{h}\int_{x_k}^{x_{k+1}}\eta(x)\,dx. \tag{2.37}$$

Заменим теперь функцию $\eta(x)$ ее интерполяционным многочленом. Предположив, что $\eta(x)$ имеет на рассматриваемом участке таблицы постоянные третьи разности, построим для нее на отрезке $[x_{k-3}, x_k]$ интерполяционный многочлен с узлами $x_{k-3}, x_{k-2}, x_{k-1}, x_k$, воспользовавшись второй интерполяционной формулой Ньютона (13.23). Приняв $t=(x-x_k)/h$, получим

$$\eta(x)=\eta(x_k+th)=$$
$$=\eta_k+t\Delta\eta_{k-1}+\frac{t(t+1)}{2}\Delta^2\eta_{k-2}+\frac{t(t+1)(t+2)}{6}\Delta^3\eta_{k-3}. \quad (3.37)$$

Произведя в формуле (2.37) замену переменных $x=x_k+th$, найдем

$$\Delta y_k=\int_0^1 \eta(x_k+th)\,dt.$$

В эту формулу можно подставить выражение (3.37) для $\eta(x_k+th)$, что дает

$$\Delta y_k=\int_0^1\left[\eta_k+t\Delta\eta_{k-1}+\frac{t(t+1)}{2}\Delta^2\eta_{k-2}+\frac{t(t+1)(t+2)}{6}\Delta^3\eta_{k-3}\right]dt.$$

Остается вычислить интеграл. Интегрирование по t приводит нас к формуле

$$\Delta y_k=\left[\eta_k t+\frac{1}{2}t^2\Delta\eta_{k-1}+\frac{1}{2}\left(\frac{t^3}{3}+\frac{t^2}{2}\right)\Delta^2\eta_{k-2}+\right.$$
$$\left.+\frac{1}{6}\left(\frac{t^4}{4}+t^3+t^2\right)\Delta^3\eta_{k-3}\right]_0^1,$$

или, окончательно,

$$\Delta y_k=\eta_k+(1/2)\,\Delta\eta_{k-1}+(5/12)\,\Delta^2\eta_{k-2}+(3/8)\,\Delta^3\eta_{k-3}. \quad (4.37)$$

Это и есть основная в рассматриваемом методе *формула Адамса для продолжения таблицы.*

Формула (4.37) получена нами в предположении постоянства третьих разностей величины η. В тех случаях, когда можно ограничиться вторыми разностями, многочлен (3.37) можно писать без третьего слагаемого. Подстановка этого многочлена в (2.37) и интегрирование приводит к формуле Адамса с вторыми разностями

$$\Delta y_k=\eta_k+(1/2)\,\Delta\eta_{k-1}+(5/12)\,\Delta^2\eta_{k-1}. \quad (5.37)$$

Наоборот, если рассмотрение третьих разностей является недостаточным, то тот же метод позволяет получить более точную формулу Адамса, содержащую разности величины η более высокого порядка. Например, формула Адамса с разностями пятого порядка имеет вид

$$\Delta y_k = \eta_k + \frac{1}{2}\Delta\eta_{k-1} + \frac{5}{12}\Delta^2\eta_{k-2} + \frac{3}{8}\Delta^3\eta_{k-3} + \\ + \frac{251}{720}\Delta^4\eta_{k-4} + \frac{95}{288}\Delta^5\eta_{k-5}. \qquad (6.37)$$

Так как все разности входят в интерполяционную формулу (13.23) независимо друг от друга, то более простые формулы Адамса получаются из более сложных путем простого отбрасывания членов со старшими разностями. Дальнейшие рассуждения будут вестись, в основном, для формулы (4.37) с третьими разностями.

Использование формулы (4.37) возможно лишь тогда, когда известны все входящие в нее разности, т. е. при $k \geqslant 3$. Иначе говоря, таблица значений функции должна уже иметь вид табл. 1.37.

Таблица 1.37

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
n	x	y	Δy	η	$\Delta\eta$	$\Delta^2\eta$	$\Delta^3\eta$
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
$n-3$	x_{n-3}	y_{n-3}	...	η_{n-3}	...		
			Δy_{n-3}		$\Delta\eta_{n-3}$		...
$n-2$	x_{n-2}	y_{n-2}		η_{n-2}		$\Delta^2\eta_{n-3}$	
			Δy_{n-2}		$\Delta\eta_{n-2}$		$\Delta^3\eta_{n-3}$
$n-1$	x_{n-1}	y_{n-1}		η_{n-1}		$\Delta^2\eta_{n-2}$	
			Δy_{n-1}		$\Delta\eta_{n-1}$		
n	x_n	y_n		η_n			

Поэтому (4.37) называют формулой *для продолжения таблицы*. Чтобы иметь таблицу вида табл. 2.37, надо каким-либо образом совершить *вход в таблицу*, для чего вычислить разности Δy_0, Δy_1, Δy_2.

Значения разностей Δy_0, Δy_1, Δy_2 можно получить различными способами. Мы рассмотрим метод, основанный на преобразовании формулы (4.37), благодаря которому можно построить итерационный процесс нахождения требуемых разностей. Такое начало таблицы было предложено А. Н. Крыловым. Вывод нужных формул основан на постоянстве третьих разностей.

Полагая

$$\Delta^3\eta_k = \Delta^3\eta_{k-1} = \Delta^3\eta_{k-2} = \Delta^3\eta_{k-3},$$

заменим в (4.37) $\Delta^3\eta_{k-3}$ на $\Delta^3\eta_{k-2}$. Тогда

$$\Delta y_k = \eta_k + (1/2)\,\Delta\eta_{k-1} + (5/12)\,\Delta^2\eta_{k-2} + (3/8)\,\Delta^3\eta_{k-2}. \qquad (7.37)$$

Далее, из определения конечных разностей вытекает, что

$$\Delta^2\eta_{k-2} = \Delta^2\eta_{k-1} - \Delta^3\eta_{k-2}.$$

Подставив это выражение в (7.37) и заменив в полученной формуле $\Delta^3\eta_{k-2}$ на $\Delta^3\eta_{k-1}$, придем к формуле

$$\Delta y_k = \eta_k + (1/2)\,\Delta\eta_{k-1} + (5/12)\,\Delta^2\eta_{k-1} - (1/24)\,\Delta^3\eta_{k-1}. \qquad (8.37)$$

Так как

$$\Delta^2\eta_{k-1} = \Delta^2\eta_k - \Delta^3\eta_{k-1},$$
$$\Delta\eta_{k-1} = \Delta\eta_k - \Delta^2\eta_{k-1} = \Delta\eta_k - \Delta^2\eta_k + \Delta^3\eta_{k-1},$$

то после очередной замены $\Delta^3\eta_{k-1}$ на $\Delta^3\eta_k$ формула (8.37) приведется к виду

$$\Delta y_k = \eta_k + (1/2)\,\Delta\eta_k - (1/12)\,\Delta^2\eta_k + (1/24)\,\Delta^3\eta_k. \qquad (9.37)$$

Положим теперь в формулах (9.37), (8.37), (7.37) соответственно $k=0$, 1, 2. Это дает

$$\left.\begin{aligned}
\Delta y_0 &= \eta_0 + (1/2)\,\Delta\eta_0 - (1/12)\,\Delta^2\eta_0 + (1/24)\,\Delta^3\eta_0,\\
\Delta y_1 &= \eta_1 + (1/2)\,\Delta\eta_0 + (5/12)\,\Delta^2\eta_0 - (1/24)\,\Delta^3\eta_0,\\
\Delta y_2 &= \eta_2 + (1/2)\,\Delta\eta_1 + (5/12)\,\Delta^2\eta_0 + (3/8)\,\Delta^3\eta_0.
\end{aligned}\right\} \qquad (10.37)$$

Формулы (10.37) и являются исходными для входа в таблицу *формулами Крылова*. Рассмотрим теперь более подробно, как с их помощью начинать таблицу для метода Адамса—Крылова.

Зная из начального условия $x=x_0$ и $y=y_0$, легко находим

$$y_0' = f(x_0, y_0), \qquad \eta_0 = y_0' h,$$

так что можно заполнить первую строку таблицы (см. табл. 2.37).

Таблица 2.37

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
n	x	y	Δy	η	$\Delta\eta$	$\Delta^2\eta$	$\Delta^3\eta$
0	x_0	y_0		η_0			
			Δy_0		$\Delta\eta_0$		
1	x_1	y_1		η_1		$\Delta^2\eta_0$	
			Δy_1		$\Delta\eta_1$		$\Delta^3\eta_0$
2	x_2	y_2		η_2		$\Delta^2\eta_1$	
			Δy_2		$\Delta\eta_2$		
3	x_3	y_3		η_3			

Таблица 3.37

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
n	x	y	Δy	η	$\Delta\eta$	$\Delta^2\eta$	$\Delta^3\eta$
0	x_0	y_0		η_0			
			$\underline{\Delta y_1}$		$\underline{\Delta\eta_0}$		
1	x_1	$\underline{y_1}$		$\underline{\eta_1}$			

В качестве первого приближения берем один член первой формулы (10.37), т. е. полагаем $\Delta y_0 = \eta_0$. Это дает возможность определить последовательно

$$y_1 = y_0 + \Delta y_0,$$
$$y_1' = f(x_1, y_1),$$
$$\eta_1 = y_1' \cdot h,$$
$$\Delta\eta_0 = \eta_1 - \eta_0,$$

т. е. получить табл. 3.37. Вычисленные величины мы подчеркиваем в ней одной чертой, чем отмечается, что они получены в первом приближении и по одночленной формуле (10.37).

Для второго приближения берем две формулы (10.37), и в каждой из них сохраняем уже по два члена. При этом мы перевычисляем значение Δy_0 и вновь вычисляем Δy_1. Последовательно получаем:

$$\begin{aligned}
\Delta y_0 &= \eta_0 + (1/2)\,\Delta\eta_0, & y_2' &= f(x_2,\ y_2),\\
\Delta y_1 &= \eta_1 + (1/2)\,\Delta\eta_0, & \eta_2 &= y_2' h,\\
y_1 &= y_0 + \Delta y_0, & \Delta\eta_0 &= \eta_1 - \eta_0,\\
y_2 &= y_1 + \Delta y_1 & \Delta\eta_1 &= \eta_2 - \eta_1,\\
y_1' &= f(x_1,\ y_1), & \Delta^2\eta_0 &= \Delta\eta_1 - \Delta\eta_0.\\
\eta_1 &= y_1' h,
\end{aligned}$$

Все вычисления сводим в табл. 4.37, где величины, полученные по одночленной и двучленным формулам, подчеркнуты соответственно одной и двумя чертами.

Теперь, когда мы имеем не только первые, но и вторую разность $\Delta^2\eta_0$, мы можем снова перевычислить Δy_0, Δy_1 и найти Δy_2, пользуясь трехчленными формулами, т. е. сохраняя во всех трех формулах (10.37) по три члена. Как и ранее, последовательно находим:

$$\begin{aligned}
\Delta y_0 &= \eta_0 + \frac{1}{2}\,\Delta\eta_0 - \frac{1}{12}\,\Delta^2\eta_0, & \eta_1 &= y_1' h = f(x_1,\ y_1)\,h,\\
\Delta y_1 &= \eta_1 + \frac{1}{2}\,\Delta\eta_0 + \frac{5}{12}\,\Delta^2\eta_0, & \eta_2 &= y_2' h = f(x_2,\ y_2)\,h,\\
\Delta y_2 &= \eta_2 + \frac{1}{2}\,\Delta\eta_1 + \frac{5}{12}\,\Delta^2\eta_0, & \eta_3 &= y_3' h = f(x_3,\ y_3)\,h,\\
y_1 &= y_0 + \Delta y_0, & \Delta\eta_0 &= \eta_1 - \eta_0,\\
y_2 &= y_1 + \Delta y_1, & \Delta\eta_1 &= \eta_2 - \eta_1,\\
y_3 &= y_2 + \Delta y_2, & \Delta\eta_2 &= \eta_3 - \eta_2,\\
& & \Delta^2\eta_0 &= \Delta\eta_1 - \Delta\eta_0,\\
& & \Delta^2\eta_1 &= \Delta\eta_2 - \Delta\eta_1,\\
& & \Delta^3\eta_0 &= \Delta^2\eta_1 - \Delta^2\eta_0.
\end{aligned}$$

Таблица значений функции приобретает уже новый вид (табл. 5.37), и будет содержать уже первые, вторые и третьи приближения.

После этих вычислений мы можем воспользоваться полными формулами (10.37), которые позволяют перевычислить все имеющиеся в таблице значения. Впрочем, довольно часто разность $\Delta_3\eta_0$ оказывается настолько малой, что перевычисление Δy_0, Δy_1, Δy_2 не изменяет их значений. В таких случаях можно не выписывать следующего приближения, а сразу продолжать полученную таблицу по формуле Адамса (4.37).

Если, наоборот, полные формулы Крылова дают заметное расхождение с результатами третьего приближения, то разности Δy_0, Δy_1, Δy_2 перевычисляются до тех пор, пока эти значения не установятся. После этого можно уже продолжать таблицу по формуле Адамса (4.37). Если же результаты пересчета по формулам Крылова (10.37) не устанавливаются, то необходимо менять шаг интегрирования h.

При работе с формулой Адамса (5.37), содержащей лишь вторые разности, достаточно для входа в таблицу знать лишь Δy_0, Δy_1. Их

Таблица 4.37

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	n	x	y	Δy	η	$\Delta\eta$	$\Delta^2\eta$	$\Delta^3\eta$
I	0	x_0	y_0		η_0			
				$\underline{\Delta y_0}$		$\underline{\Delta\eta_0}$		
	1	x_1	$\underline{y_1}$		$\underline{\eta_1}$			
II	0	x_0	y_0		η_0			
				$\underline{\underline{\Delta y_0}}$		$\underline{\underline{\Delta\eta_0}}$		
	1	x_1	$\underline{\underline{y_1}}$		$\underline{\underline{\eta_1}}$		$\underline{\underline{\Delta^2\eta_0}}$	
				$\underline{\underline{\Delta y_1}}$		$\underline{\underline{\Delta\eta_1}}$		
	2	x_2	$\underline{\underline{y_2}}$		$\underline{\underline{\eta_2}}$			

можно получить с помощью такого же итерационного процесса, пользуясь упрощенными формулами Крылова,

$$\left.\begin{aligned} \Delta y_0 &= \eta_0 + \frac{1}{2}\Delta\eta_0 - \frac{1}{12}\Delta^2\eta_0, \\ \Delta y_1 &= \eta_1 + \frac{1}{2}\Delta\eta_0 + \frac{5}{12}\Delta^2\eta_0. \end{aligned}\right\} \qquad (11.37)$$

Для более сложной формулы Адамса, наоборот, необходима бóльшая начальная таблица и более сложные формулы Крылова.

Пример 1.37. С помощью метода Адамса—Крылова найдем решение дифференциального уравнения

$$y' = xy^3$$

с начальными условиями $x_0 = 0$, $y_0 = 1$ для участка $[0; 0{,}8]$, приняв $h = 0{,}1$. Для входа в таблицу воспользуемся формулами Крылова (10.37).

Схема расчета и все требуемые вычисления приведены в табл. 6.37, в которой отдельные этапы входа в таблицу разделены жирной горизонтальной чертой.

Таблица 5.37

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	n	x	y	Δy	η	$\Delta\eta$	$\Delta^2\eta$	$\Delta^3\eta$
I	0	x_0	y_0		η_0			
				Δy_0		$\Delta\eta_0$		
	1	x_1	y_1		η_1			
II	0	x_0	y_0		η_0			
				Δy_0		$\Delta\eta_0$		
	1	x_1	y_1		η_1		$\Delta^2\eta_0$	
				Δy_1		$\Delta\eta_1$		
	2	x_2	y_2		η_2			
III	0	x_0	y_0		η_0			
				Δy_0		$\Delta\eta_0$		
	1	x_1	y_1		η_1		$\Delta^2\eta_0$	
				Δy_1		$\Delta\eta_1$		$\Delta^3\eta_0$
	2	x_2	y_2		η_2		$\Delta^2\eta_1$	
				Δy_2		$\Delta\eta_2$		
	3	x_3	y_3		η_3			

Таблица 6.37

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
x	y	Δy	η	$\Delta\eta$	$\Delta^2\eta$	$\Delta^3\eta$	$y_{\text{точн}}$
0	1		0				
		0		100			
0,1	1,0000		0,0100				
0	1		0				
		0,0050		102			
0,1	1,0050		0,0102		8		
		0,0150		110			
0,2	1,0200		0,0212				
0	1		0				
		0,0050		102			
0,1	1,0050		0,0102		9		
		0,0156		111		12	
0,2	1,0206		0,0213		21		
		0,0270		132			
0,3	1,0476		0,0345				
0	1		0				
		0,0050		102			
0,1	1,0050		0,0102		9		
		0,0157		111		13	
0,2	1,0207		0,0213		22		
		0,0277		133			
0,3	1,0484		0,0346				

Продолжение табл. 6.37

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
x	y	Δy	η	$\Delta\eta$	$\Delta^2\eta$	$\Delta^3\eta$	$y_{\text{точн}}$
0	1		0				1
		0,0050		102			
0,1	1,0050		0,0102		9		1,0050
		0,0157		111		13	
0,2	1,0207		0,0213		22		1,0206
		0,0277		133		18	
0,3	1,0484		0,0346		40		1,0483
		0,0426		173		36	
0,4	1,0910		0,0519		76		1,0911
		0,0629		249		72	
0,5	1,1539		0,0768		148		1,1547
		0,0938		397		182	
0,6	1,2477		0,1165		330		1,2500
		0,1452		727			
0,7	1,3929		0,1892				1,4003
		0,2460					
0,8	1,6389						1,6667

Так как в данном случае нахождение y' и затем η не требует сложных выкладок, все вычисления ведутся на одном бланке. Для уравнений с более сложной правой частью вычисление производной и величины y переносится на другой бланк, который называют *вспомогательным*. Бланк, содержащий расчеты типа 6.37, называют тогда *основным*. Заполнение основного и вспомогательного бланка ведутся при этом параллельно.

§ 38. Метод Рунге — Кутта

Основным недостатком методов Адамса является трудность входа в таблицу. В каждом методе из этой группы для получения следующего значения функции используется весьма обширная информация о ее поведении в предыдущих узлах. Поэтому для работы с формулой Адамса с любыми разностями нужно предварительно найти еще несколько значений функции. При этом необходимо эти значения иметь с достаточно большой точностью, так как заметная погрешность в начальных значениях может свести на нет точность любого метода нахождения следующих.

Итерационный вход в таблицу по формулам Крылова достаточно удобен лишь для ручного счета. Если же иметь в виду программирование метода Адамса, то использование формул Крылова удлиняет программу приблизительно вчетверо, так как вместо расчета по одной формуле (4.37) прибавляется итерационный цикл еще с тремя формулами (10.37).

Метод Рунге — Кутта, изложению которого посвящен настоящий параграф, очень часто используется для нахождения значений функции в нескольких начальных точках, благодаря той его особенности, что он совсем не использует предыдущей информации. Каждый шаг в методе Рунге — Кутта делается как бы заново, и для вычисления значения функции в точке x_{n+1} используется лишь ее значение в точке x_n.

Платой за столь малую информативность метода является его трудоемкость. Для получения следующего значения функции требуется несколько раз вычислять значение производной y', т. е. обращаться к правой части

дифференциального уравнения. Если эта правая часть слишком громоздка, то трудоемкость метода будет в несколько раз превосходить соответствующую трудоемкость метода Адамса.

Существует несколько методов Рунге — Кутта различных порядков. Наиболее распространенным является метод четвертого порядка. Перейдем к его рассмотрению.

Дано дифференциальное уравнение первого порядка $y'=f(x, y)$ и предполагается известным значение $y_n=y(x_n)$. Не важно, является y_n заданным начальным значением или оно, в свою очередь, получено в процессе вычислений. Для получения значения функции $y_{n+1}=y(x_{n+1})$ по методу Рунге—Кутта выполняется следующая последовательность операций:

$$\left.\begin{aligned} k_1&=hf(x_n, y_n),\\ k_2&=hf(x_n+h/2, y_n+k_1/2),\\ k_3&=hf(x_n+h/2, y_n+k_2/2),\\ k_4&=hf(x_n+h, y_n+k_3). \end{aligned}\right\} \qquad (1.38)$$

После этого приращение функции находится по формуле

$$\Delta y_n=(1/6)(k_1+2k_2+2k_3+k_4), \qquad (2.38)$$

а

$$y_{n+1}=y_n+\Delta y_n. \qquad (3.38)$$

Геометрический смысл этого метода легко прослеживается по последовательности формул (1.38), из которых видно, что каждый шаг расчета представляет собою, в сущности, шаг по методу Эйлера.

Сначала следует сделать шаг величины $h/2$ из точки $M_0(x_n, y_n)$ под углом α_1, $y'_1=\operatorname{tg}\alpha_1=k_1/h$, и мы приходим в точку $M_1(x_n+h/2, y_n+k_1/2)$. В этой точке вычисляется направление $\operatorname{tg}\alpha_2=k_2/h$ и, делая шаг в этом направлении, снова из точки M_0 попадаем в точку $M_2(x_n+h/2, y_n+k_2/2)$. Затем по направлению $\operatorname{tg}\alpha_3=k_3/h$ снова из точки M_0 делается шаг величины h, который приводит в точку $M_3(x_n+h, y_n+k_3)$, в которой вычисляется направление $\operatorname{tg}\alpha_4=k_4/h$. Полученные четыре тангенса усредняются с весами 1/6, 2/6, 2/6, 1/6 по формуле (2.38) и по этому окончательному направлению мы делаем окончательный шаг из (x_n, y_n) в (x_{n+1}, y_{n+1}).

Таблица 1.38

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
x	«1»—(1)²	(1) : (2)	y	$y'=(3)\cdot(4)$	k	w	(6) · (7)
0	1	0	1	0	0	1/6	0
0,05	0,9975	0,0501	1	0,0501	0,0050	1/3	0,0016
0,05			1,0025	0,0502	0,0050	1/3	0,0017
0,1	0,9900	0,1010	1,0050	0,1015	0,0102	1/6	0,0017
0,1			1,0050	0,1015	0,0102	1/6	0,0017
0,15	0,9775	0,1535	1,0101	0,1551	0,0155	1/3	0,0052
0,15			1,0128	0,1555	0,0156	1/3	0,0052
0,2	0,9600	0,2083	1,0206	0,2126	0,0213	1/6	0,0036
0,2			1,0207	0,0126	0,0213	1/6	0,0036
0,25	0,9375	0,2667	1,0313	0,2750	0,0275	1/3	0,0092
0,25			1,0345	0,2759	0,0276	1/3	0,0092
0,3	0,9100	0,3297	1,0483	0,3456	0,0346	1/6	0,0058
0,3			1,0485	0,3457	0,0346	1/6	0,0058
0,35	0,8775	0,3989	1,0658	0,4251	0,0425	1/3	0,0142
0,35			1,0697	0,4267	0,0427	1/3	0,0142
0,4	0,8400	0,4762	1,0912	0,5196	0,0520	1/6	0,0087
0,4			1,0914	0,5197	0,0520	1/6	0,0087
0,45	0,7975	0,5643	1,1174	0,6305	0,0630	1/3	0,0210
0,45			1,1229	0,6337	0,0634	1/3	0,0211
0,5	0,7500	0,6667	1,1548	0,7699	0,0770	1/6	0,0128
0,5			1,1550				

Пример 1.38. Решим методом Рунге – Кутта уравнение

$$y' = xy/(1 - x^2)$$

с начальными условиями $x_0 = 0$, $y_0 = 1$ для участка [0; 0,5] с шагом $h = 0{,}1$.

Все вычисления приведены в табл. 1.38.

§ 39. Методы прогноза и коррекции. Метод Милна

Наиболее простые методы решения уравнений, рассмотренные в § 36, могут быть существенно улучшены с помощью следующего приема. Переписав формулу (4.36) в виде

$$y_{k+1} = y_k + \int_{x_k}^{x_{k+1}} y'(x)\,dx, \tag{1.39}$$

применим к интегралу, стоящему справа, какую-либо из простых квадратурных формул, например, формулу трапеций. Тогда получим равенство

$$y_{k+1} = y_k + (h/2)\,(y'_k + y'_{k+1}). \tag{2.39}$$

Использование формулы (2.39) для получения следующего значения функции требует знания производной не только в точке x_k, но и в точке x_{k+1}. С другой стороны, для получения y'_{k+1} из дифференциального уравнения нужно знать значение функции y_{k+1}. Встреченную трудность можно обойти следующим образом.

Сначала найдем предварительное, хотя бы и с заметной погрешностью, значение y_{k+1}, пользуясь каким-либо простым методом, например, по формуле (6.36) уточненного метода Эйлера

$$y_{k+1} = y_{k-1} + 2hy'_k. \tag{3.39}$$

Затем можно из уравнения $y' = f(x, y)$ вычислить величину y'_{k+1}, после чего уточнить значение y_{k+1} по формуле (2.39), получив новое, более точное значение. По исправленному значению y_{k+1} можно снова пересчитать y'_{k+1} и вновь исправить y_{k+1} по формуле (2.39). Такой

метод называют *уточненным методом Эйлера с итерациями*.

Заметим только, что, как и для уточненного метода Эйлера, для использования формулы (3.39) необходимо предварительно найти y_1. Это можно сделать обычным методом Эйлера или через вспомогательную половинную точку, как в § 36.

Пример 1.39. Найдем решение дифференциального уравнения

$$y' = 2xy$$

с начальным условием $x_0 = 0$, $y_0 = 1$ на участке [0; 1] с шагом $h = 0{,}1$.

Вычисления приведены в табл. 1.39. Первая строка вычислена, как в § 36: для $x = 0{,}05$ значение найдено по методу Эйлера, а для $x = 0{,}1$ – по формуле $y_1 = y_0 + hy'_{1/2}$. Столбцы (2) и (4) содержат соответственно предварительное значение y, полученное по формуле (3.39), и найденное через него значение производной y'. В столбцах (5) и (6) записано исправленное по формуле (2.39) значение функции и новое значение y'.

Нужно только иметь в виду, что для нахождения предварительного следующего значения функции в столбце (2) используются уже исправленные значения функции и производной из столбцов (5) и (6), так что каждое значение y из столбца (2) используется лишь один раз для вычислений в той же строке.

В столбце (7) приведены значения точного решения $y = e^{x^2}$. Сравнение показывает, что относительная погрешность решения в точке $x = 1$ составляет менее $0{,}4\%$.

Методы такого типа, в которых сначала находится предварительное значение функции в следующей точке по одной формуле, а затем исправляется с помощью производной по другой, объединяются под общим названием *методов прогноза и коррекции*. Эти методы особенно хороши тем, что дают возможность судить о погрешностях получающегося решения по разностям между прогнозируемыми и скорректированными значениями функции. Более подробно речь об этом будет идти в §§ 41 и 43.

Одним из наиболее распространенных методов прогноза и коррекции является *метод Милна*, к рассмотрению

Таблица 1.39

(1)	(2)	(3)	(4)	(5)	(6)	(7)
x	$y_{\text{предв}}$	Δy	$y'_{\text{предв}}$	$y_{\text{испр}}$	$y'_{\text{испр}}$	$y_{\text{точн}}$
0	1		0	1	0	1
		0				
0,05	1,000	0,010	0,100			
0,1	1,010	0,040	0,202	1,010	0,202	1,010
0,2	1,040	0,083	0,416	1,041	0,416	1,041
0,3	1,093	0,131	0,656	1,095	0,657	1,094
0,4	1,172	0,188	0,938	1,175	0,940	1,174
0,5	1,283	0,257	1,283	1,286	1,286	1,284
0,6	1,432	0,345	1,718	1,436	1,723	1,433
0,7	1,631	0,458	2,283	1,636	2,290	1,632
0,8	1,894	0,609	3,030	1,902	3,043	1,897
0,9	2,245	0,812	4,041	2,256	4,061	2,250
1,0	2,714		5,428	2,729		2,718

которого мы сейчас перейдем. В нем прогноз производится по формуле, близкой к (3.39), а коррекция — с помощью квадратурной формулы Симпсона.

Для получения формулы прогноза воспользуемся, как и при выводе формулы Адамса, экстраполяцией производной y' по ее интерполяционному многочлену третьей степени. Запишем

$$y_{n+1} - y_{n-3} = \int_{x_{n-3}}^{x_{n+1}} y'(x)\,dx \qquad (4.39)$$

и построим для y' интерполяционный многочлен третьей степени на участке (x_{n-3}, x_n) по узлам x_{n-3}, x_{n-2}, x_{n-1}, x_n. Первая интерполяционная формула Ньютона (9.23) дает

$$F(x_{n-3}+th)=y'_{n-3}+t\,\Delta y'_{n-3}+\frac{t(t-1)}{2}\,\Delta^2 y'_{n-3}+ \\ +\frac{t(t-1)(t-2)}{6}\,\Delta^3 y'_{n-3}.$$

Заменив y' под интегралом в формуле (4.39) выписанным интерполяционным многочленом и выполнив замену переменных $x=x_{n-3}+th$, получим

$$y_{n+1}-y_{n-3}=\int_{x_{n-3}}^{x_{n+1}} y'(x)\,dx\approx h\int_0^4 F(x_{n-3}+th)\,dt= \\ =h\,[4y'_{n-3}+8\Delta y'_{n-3}+(20/3)\,\Delta^2 y'_{n-3}+ \\ +(8/3)\Delta^3 y'_{n-3}].$$

В полученном равенстве заменим разности значениями производных с помощью формулы (2.21) и придем к формуле прогноза вида

$$y_{n+1}=y_{n-3}+\frac{4h}{3}\,(2y'_{n-2}-y'_{n-1}+2y'_n). \qquad (5.39)$$

Этот прогноз может быть использован для $n\geqslant 3$. Поэтому для решения уравнения по методу Милна необходимо предварительное вычисление значений y_1, y_2, y_3. Это можно сделать по формулам Крылова для входа в таблицу или по методу Рунге — Кутта. Полученные значения y_2 и y_3 могут быть исправлены с помощью формулы коррекции, которая, как уже говорилось, производится с помощью квадратурной формулы Симпсона.

Именно, так как

$$y_{n+1}-y_{n-1}=\int_{x_{n-1}}^{x_{n+1}} y'(x)\,dx,$$

то, вычислив интеграл справа по формуле Симпсона, получим

$$y_{n+1}=y_{n-1}+\frac{h}{3}\,(y'_{n-1}+4y'_n+y'_{n+1}). \qquad (6.39)$$

Таблица 2.39

(1)	(2)	(3)	(4)	(5)	(6)	(7)
x	y	Δy	η	$\Delta\eta$	$\Delta^2\eta$	$\Delta^3\eta$
0	1		0			
		0		—200		
0,1	1,0000		—0,0200			
0	1		0			
		—0,0100		—196		
0,1	0,9900		—0,0196		24	
		—0,0300		—172		
0,2	0,9600		—0,0368			
0	1		0			
		—0,0100		—196		
0,1	0,9900		—0,0196		22	
		—0,0284		—174		17
0,2	0,9616		—0,0370		39	
		—0,0444		—135		
0,3	0,9172		—0,0505			
0	1		0			
		—0,0099		—196		
0,1	0,9901		—0,0196		22	
		—0,0286		—174		17
0,2	0,9615		—0,0370		39	
		—0,0442		—135		
0.3	0,9173		—0,0505			

Таблица 3.39

(1)	(2)	(3)	(4)	(5)
x	$y_{\text{предв}}$	$\Delta y^{\text{предв}}_{-2}$	$y'_{\text{предв}}$	$^4y'_{-1}$
0	1		0	
0,1	0,9901		—0,1961	
0,2	0,9615		—0,3698	—0,7844
0,3	0,9173	—0,1377	—0,5049	—1,4792
0,4	0,8623	—0,1898	—0,5948	—2,0200
0,5	0,8003	—0,2261	—0,6405	—2,3776
0,6	0,7354	—0,2461	—0,6490	—2,5600
0,7	0,6713	—0,2524	—0,6309	—2,5944
0,8	0,6096	—0,2475	—0,5946	—2,5228
0,9	0,5525	—0,2355	—0,5495	—2,3792
1	0,4997		—0,4994	—2,1988

(6)	(7)	(8)	(9)	(10)
y'_{-2}	$\Delta y^{\text{испр}}_{-2}$	$y_{\text{испр}}$	$y'_{\text{испр}}$	$y_{\text{точн}}$
		1	0	1
		0,9901	—0,1961	0,9901
0	—0,0385	0,9615	—0,3698	0,9615
—0,1961	—0,0727	0,9174	—0,5050	0,9174
—0,3698	—0,0995	0,8620	—0,5944	0,8621
—0,5050	—0,1174	0,8000	—0,6400	0,8000
—0,5944	—0,1268	0,7352	—0,6486	0,7353
—0,6400	—0,1288	0,6712	—0,6307	0,6711
—0,6486	—0,1255	0,6097	—0,5948	0,6098
—0,6307	—0,1186	0,5526	—0,5497	0,5525
—0,5948	—0,1098	0,4999		0,5000

Формула (6.39) и есть нужная нам формула коррекции. Значение y'_{n+1}, входящее в нее, получается из дифференциального уравнения $y'_{n+1}=f(x_{n+1}, y_{n+1})$, где y_{n+1} — прогнозированное значение, полученное из (5.39).

Пример 2.39. Применим метод Милна для решения дифференциального уравнения

$$y' = -2xy^2$$

с начальным условием $x_0=0$, $y_0=1$ на участке [0, 1] с шагом $h=0{,}1$.

Рассчитаем первые значения с помощью формул Крылова (10.37). Соответствующие вычисления приведены в табл. 2.39. Из нее видно, что для входа в таблицу вполне достаточно трех итераций.

Дальнейшие вычисления читатель найдет в табл. 3.39.

§ 40. Системы дифференциальных уравнений и уравнения высших порядков

До сих пор речь шла у нас о численном решении одного дифференциального уравнения первого порядка с одной неизвестной функцией $y(x)$, т. е. уравнения вида

$$y'=f(x, y). \tag{1.40}$$

Нередко приходится встречаться с задачей, в которой приходится решать *систему нескольких дифференциальных уравнений с несколькими искомыми функциями.*

Мы ограничимся рассмотрением *нормальных систем* дифференциальных уравнений, в которых уравнения разрешены относительно производных и число уравнений равно числу неизвестных функций. Например, система двух уравнений с двумя неизвестными функциями y, z от одного и того же аргумента x в нормальной форме имеет вид

$$\left.\begin{aligned} y' &= f_1(x, y, z), \\ z' &= f_2(x, y, z), \end{aligned}\right\} \tag{2.40}$$

причем штрих означает производную по x. Общий вид нормальной системы трех уравнений с тремя неизвест-

ными функциями x, y, z от переменного t

$$\left.\begin{aligned}\frac{dx}{dt}&=f_1(t, x, y, z),\\ \frac{dy}{dt}&=f_2(t, x, y, z),\\ \frac{dz}{dt}&=f_3(t, x, y, z).\end{aligned}\right\} \tag{3.40}$$

Рассмотренные в §§ 36—39 настоящей главы численные методы решения дифференциального уравнения вида (1.40) без труда переносятся на системы вида (2.40), (3.40) или более общие: каждый раз при переходе к следующей точке параллельно вычисляются приращения каждой из неизвестных функций по аналогичным формулам.

Пусть, например, дана нормальная система двух уравнений (2.40) с начальными условиями

$$\left.\begin{aligned}y(x_0)&=y_0,\\ z(x_0)&=z_0.\end{aligned}\right\} \tag{4.40}$$

Тогда для решения этой системы по методу Эйлера вычисляем

$$\Delta y_0=y_0'h,$$
$$\Delta z_0=z_0'h,$$

где y_0', z_0' — значения соответствующих производных в точке $x=x_0$, которые находятся из уравнений (2.40):

$$y_0'=f_1(x_0, y_0, z_0),$$
$$z_0'=f_2(x_0, y_0, z_0).$$

Затем мы получаем значения искомых функций в точке $x=x_1=x_0+h$:

$$y_1=y_0+\Delta y_0,$$
$$z_1=z_0+\Delta z_0.$$

В общем виде формулы метода Эйлера для системы двух уравнений, аналогичные (3.36), можно записать

так:

$$\left.\begin{aligned} \Delta y_k &= y'_k \cdot h = f_1(x_k, y_k, z_k) \cdot h, \\ \Delta z_k &= z'_k \cdot h = f_2(x_k, y_k, z_k) \cdot h, \\ y_{k+1} &= y_k + \Delta y_k, \\ z_{k+1} &= z_k + \Delta z_k. \end{aligned}\right\} \qquad (5.40)$$

Аналогично поступают и для всех других методов. Мы не станем выписывать всех требуемых формул, перенеся их в приведенные ниже примеры.

Таким образом, между решением одного дифференциального уравнения первого порядка и решением системы имеется лишь чисто количественное различие — число требуемых вычислений растет пропорционально числу неизвестных функций. Качественных же различий здесь нет, поскольку применяются те же методы и те же формулы.

Для уравнений высших порядков тоже не приходится создавать новых методов. Как известно из курса дифференциальных уравнений, дифференциальное уравнение n-го порядка легко сводится к системе n дифференциальных уравнений первого порядка в нормальной форме. Поэтому для численного решения дифференциальных уравнений высших порядков также применимы все рассмотренные ранее методы.

Пример 1.40. Решим методом Эйлера уравнение Бесселя с индексом $m=1$

$$x^2y'' + xy' + (x^2 - 1)\,y = 0$$

с начальными условиями $x_0=1$, $y_0=0{,}4401$, $y'_0=0{,}3251$ на участке $[1{,}0;\ 1{,}5]$ с шагом $h=0{,}1$.

Уравнение второго порядка сводится к системе двух уравнений с помощью подстановки $y'=z$. Тогда получим $y''=z'$ и уравнение дает

$$z' = -z/x - (x^2-1)y/x^2.$$

Следовательно, требуемая система имеет вид

$$\begin{aligned} y' &= z, \\ z' &= -z/x - (x^2-1)\,y/x^2. \end{aligned}$$

Таблица 1.40

(1)	(2)	(3)	(4)	(5)
x	y	Δy	$y' = z$	Δz
1	0,4401		0,3251	
		0,0325		—0,0325
1,1	0,4726		0,2926	
		0,0293		—0,0348
1,2	0,5019		0,2578	
		0,0258		—0,0368
1,3	0,5277		0,2210	
		0,0221		—0,0386
1,4	0,5498		0,1824	
		0,0182		—0,0400
1,5	0,5680		0,1424	

(6)	(7)	(8)	(9)	(10)
$\frac{\text{«1»}}{(1)^2}$	«1» — (6)	$-\frac{(4)}{(1)}$	— (2) · (7)	z' (8) + (9)
1	0	—0,3251	0	—0,3251
0,8264	0,1736	—0,2660	—0,0820	—0,3480
0,6944	0,3056	—0,2148	—0,1534	—0,3682
0,5917	0,4083	—0,1700	—0,2155	—0,3855
0,5102	0,4898	—0,1303	—0,2693	—0,3996

(1)	(2)	(3)	(4)	(5)	(6)	(7)
x	«1» : (1)^2	«1» — (2)	y	$y' = z$	k	— (5) : (1)
1	1	0	0,4401	0,3251	0,0325	—0,3251
1,05	0,9070	0,0930	0,4563	0,3089	0,0309	—0,2942
1,05			0,4555	0,3083	0,0308	—0,2936
1,1	0,8264	0,1736	0,4709	0,2915	0,0292	—0,2650
1,1			0,4710	0,2915	0,0292	—0,2650
1,15	0,7561	0,2439	0,4856	0,2741	0,0274	—0,2383
1,15			0,4847	0,2737	0,0274	—0,2380
1,2	0,6944	0,3056	0,4984	0,2559	0,0256	—0,2132
1,2			0,4984	0,2558	0,0256	—0,2132
1,25	0,6400	0,3600	0,5112	0,2375	0,0238	—0,1900
1,25			0,5103	0,2371	0,0237	—0,1897
1,3	0,5917	0,4083	0,5221	0,2185	0,0218	—0,1681
1,3			0,5221	0,2184	0,0218	—0,1680
1,35	0,5487	0,4513	0,5330	0,1994	0,0199	—0,1477
1,35			0,5321	0,1990	0,0199	—0,1474
1,4	0,5102	0,4898	0,5420	0,1796	0,0180	—0,1283
1,4			0,5419	0,1796	0,0180	—0,1283
1,45	0,4756	0,5244	0,5509	0,1599	0,0160	—0,1103
1,45			0,5499	0,1596	0,0160	—0,1101
1,5	0,4444	0,5556	0,5579	0,1398	0,0140	—0,0932
1,5			0,5578	0,1397		

Таблица 2.40

(8)	(9)	(10)	(11)	(12)	(13)
$-(3)\cdot(4)$	$z'(7)+(8)$	l	w	$(6)\cdot(11)$	$(10)\cdot(11)$
0	—0,3251	—0,0325	1/6	0,0054	—0,0054
—0,0424	—0,3366	—0,0337	1/3	0,0103	—0,0112
—0,0424	—0,3360	—0,0336	1/3	0,0103	—0,0112
—0,0817	—0,3467	—0,0347	1/6	0,0049	—0,0058
—0,0818	—0,3468	—0,0347	1/6	0,0049	—0,0058
—0,1184	—0,3567	—0,0357	1/3	0,0091	—0,0119
—0,1182	—0,3562	—0,0356	1/3	0,0091	—0,0119
—0,1523	—0,3655	—0,0366	1/6	0,0043	—0,0061
—0,1523	—0,3655	—0,0366	1/6	0,0043	—0,0061
—0,1840	—0,3740	—0,0374	1/3	0,0079	—0,0125
—0,1837	—0,3734	—0,0373	1/3	0,0079	—0,0124
—0,2132	—0,3813	—0,0381	1/6	0,0036	—0,0064
—0,2132	—0,3812	—0,0381	1/6	0,0036	—0,0064
—0,2405	—0,3882	—0,0388	1/3	0,0066	—0,0129
—0,2401	—0,3875	—0,0388	1/3	0,0066	—0,0129
—0,2655	—0,3938	—0,0394	1/6	0,0030	—0,0066
—0,2654	—0,3937	—0,0394	1/6	0,0030	—0,0066
—0,2889	—0,3992	—0,0399	1/3	0,0053	—0,0133
—0,2884	—0,3985	—0,0398	1/3	0,0053	—0,0133
—0,3100	—0,4032	—0,0403	1/6	0,0023	—0,0067

Теперь остается лишь применить формулы (5.40), принимая начальные условия $x_0=1$, $y_0=0{,}4401$, $z_0=0{,}3251$. Вычисления приведены в табл. 1.40. Для значения $x=1{,}5$ находим $y_5=0{,}5680$, тогда как точное значение функции Бесселя по таблице равно $J_1(1{,}5)=0{,}5579$. Относительная ошибка составляет здесь 1,8%.

Пример 2.40. Применим метод Рунге — Кутта для решения того же уравнения с теми же начальными условиями.

Выпишем все формулы, требуемые для выполнения одного полного шага для системы вида (2.40):

$$\left.\begin{aligned}
k_1&=hf_1(x_n,\ y_n,\ z_n),\\
l_1&=hf_2(x_n,\ y_n,\ z_n),\\
k_2&=hf_1(x_n+h/2,\ y_n+k_1/2,\ z_n+l_1/2),\\
l_2&=hf_2(x_n+h/2,\ y_n+k_1/2,\ z_n+l_1/2),\\
k_3&=hf_1(x_n+h/2,\ y_n+k_2/2,\ z_n+l_2/2),\\
l_3&=hf_2(x_n+h/2,\ y_n+k_2/2,\ z_n+l_2/2),\\
k_4&=hf_1(x_n+h,\ y_n+k_3,\ z_n+l_3),\\
l_4&=hf_2(x_n+h,\ y_n+k_3,\ z_n+l_3).
\end{aligned}\right\} \qquad (6.40)$$

После выполнения вычислений по формулам (6.40) полученные приращения усредняются по формулам

$$\left.\begin{aligned}
k&=(1/6)\,(k_1+2k_2+2k_3+k_4),\\
l&=(1/6)\,(l_1+2l_2+2l_3+l_4),
\end{aligned}\right\} \qquad (7.40)$$

и значения y, z в точке x_{n+1} находятся как

$$\begin{aligned}
y_{n+1}&=y_n+k,\\
z_{n+1}&=z_n+l.
\end{aligned}$$

Вычисления по приведенной схеме приведены в табл. 2.40. Как показывает сравнение с таблицей функций Бесселя, все значения функции $J_1(x)$ получаются точными.

§ 41. О погрешностях методов численного решения дифференциальных уравнений

Оценка точности методов численного решения дифференциальных уравнений представляет серьезные трудности. Задачи с начальными условиями, которые мы здесь рассматривали, с этой точки зрения наиболее неблагоприятны. Ввиду наличия «обратной связи» от значений производной к значению функции, погрешности, допущенные при определении первоначальных значений, вызывают систематическое нарастание погрешностей при дальнейших вычислениях, причем скорость роста может оказаться довольно большой.

Причины такого нарастания очень хорошо иллюстрируются таким примером. Уравнение

$$y'' = 10y' + 11y \tag{1.41}$$

с начальными условиями $x_0 = 0$, $y_0 = 1$, $y'_0 = -1$ имеет точное решение $y = e^{-x}$. Решение этой задачи численными методами на участке не слишком малой длины дает результаты совершенно неприемлемые. В таблице 1.41 приведены значения решения уравнения (1.41), полученного методом Рунге—Кутта с шагом $h = 0{,}125$ для участка [0,3], и его точного решения. При этом для $x < 2$ значения приводятся с пропусками. Относительная погрешность в конце участка составляет, как мы видим, много более 100%.

Легко понять, в чем здесь дело. Общее решение этого уравнения имеет вид $y = C_1 e^{-x} + C_2 e^{11x}$, где C_1, C_2 — произвольные постоянные. При заданных начальных условиях $C_1 = 1$, $C_2 = 0$, т. е. компонента $C_2 e^{11x}$ отсутствует. Если применение какого-либо численного метода дает для этой компоненты в первой же точке $x = h$ значение ε, то для $x = 2{,}5$ это значение вырастает до $\varepsilon e^{11\,(2{,}5h)}$. Так как $e^{27} \approx 0{,}9 \cdot 10^{12}$, то ясно, что эта компонента полностью перекрывает точное значение результата.

Разумеется, приведенный пример специально подобран и является исключительным. В большинстве случаев дело обстоит более благополучно. Однако в благополучных случаях фактически получающиеся погрешности много меньше тех границ, которые установлены

Таблица 1.41

(1)	(2)	(3)
y	$y_{\text{рк}}$	$y_{\text{точн}}$
0	1	1
0,125	0,88250	0,88250
0,250	0,77880	0,77880
0,375	0,68729	0,68729
0,5	0,60653	0,60653
1,0	0,36788	0,36788
1,5	0,22313	0,22313
1,75	0,17376	0,17377
2,0	0,13507	0,13534
2,125	0,11840	0,11943
2,250	0,10137	0,10540
2,375	0,07728	0,09301
2,500	0,02069	0,08208
2,625	—0,16717	0,07244
2,750	—0,87116	0,06393
2,875	—3,59279	0,05642
3,000	—14,19131	0,04979

для них теоретическими оценками. Это обстоятельство объясняется теми же причинами: поскольку точная теоретическая оценка должна выполняться и в неблагоприятных случаях, нельзя ожидать, чтобы для практически малых погрешностей эта оценка достаточно хорошо указывала ее порядок.

Кроме сказанного, следует еще отметить, что теоретические оценки, получающиеся на базе оценок точности интерполяционных формул, рассматривавшихся в § 25, содержат производные высоких порядков искомой функции. Поэтому практическое значение таких оценок и с этой точки зрения весьма невелико.

В соответствии со сказанным, мы не станем приводить точных выражений для оценки погрешности численных методов решения дифференциальных уравнений, ограничившись необходимыми общими соображениями.

Если разложить искомое решение $y(x)$ по формуле Тейлора в окрестности точки x_1 и положить затем в разложении $x=x_{k+1}$, то мы получим

$$y_{k+1}=y_k+y'_k h+\frac{1}{2!}y''(\xi)\,h^2,$$

где ξ — точка интервала (x_k, x_{k+1}). Сравнение этой формулы с формулой (3.36), определяющей метод Эйлера, показывает, что погрешность метода Эйлера возникает от отбрасывания члена вида $\frac{1}{2!}y''(\xi)\,h^2$. Этот член при малых h будет являться величиной порядка h^2. Следовательно, погрешность метода Эйлера есть величина порядка h^2.

Аналогично показывается, что погрешность уточненного метода Эйлера есть величина уже порядка h^3, как об этом было сказано в § 36. Из этих общих рассуждений становится ясно, почему обычный и уточненный методы Эйлера применяются, как правило, лишь для прикидочных расчетов.

Остальные рассмотренные в настоящей главе численные методы имеют погрешности более высокого порядка. Не останавливаясь на выводах, укажем, что погрешность метода Адамса — Крылова со вторыми разностями (формула (5.37)) имеет порядок h^4; тот же метод с третьими

разностями (формула (4.37)) имеет погрешность порядка h^5. Такой же порядок погрешности имеют методы Милна и Рунге — Кутта.

При практическом применении численных методов решения дифференциальных уравнений обычно ограничиваются ориентировочными представлениями, среди которых основную роль играет сравнение приближений, полученных с различным шагом. Для численного интегрирования эти приемы нами уже рассматривались в §§ 30—33.

Автоматический выбор шага для достижения нужной точности при машинных вычислениях требует в этом случае, по меньшей мере, двойной работы. Поэтому особенно удобным является применение методов прогноза и коррекции, рассмотренных нами в § 39, так как для этих методов о погрешности каждого очередного значения можно судить по разности между прогнозированным и скорректированным. Подробнее об использовании методов прогноза и коррекции при текущем контроле точности речь будет идти в § 44.

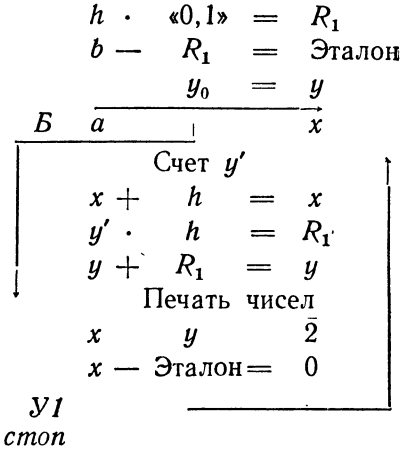
§ 42. Программирование элементарных методов интегрирования

Программирование метода Эйлера для численного решения дифференциального уравнения первого порядка с заданным шагом h является совершенно элементарным. Пусть требуется проинтегрировать методом Эйлера на отрезке $[a, b]$ с заданным шагом h уравнение

$$y' = f(x, y),$$

и задано начальное условие $y(a) = y_0$. Предположим, что написана подпрограмма счета $y' = f(x, y)$, берущая аргументы из ячеек x и y и кладущая ответ в ячейку y'. Тогда можно написать обычный арифметический цикл, каждый шаг которого будет содержать вычисление очередного значения y и печать. Проверку окончания следует организовать по эталону, как о том говорилось в § 34. Для удобства печати будем считать, что x и y расположены в соседних ячейках.

Требуемый цикл можно написать так:



Несколько сложнее программирование уточненного метода Эйлера, хотя бы и при фиксированном шаге, так как там фактически приходится программировать три формулы: сначала

$$y_{1/2} = y_0 + (h/2) f(x_0, y_0), \tag{1.42}$$

потом

$$y_1 = y_0 + hf(x_{1/2}, y_{1/2}), \tag{2.42}$$

и, наконец,

$$y_{k+1} = y_{k-1} + 2hf(x_k, y_k) \tag{3.42}$$

для $k = 1, 2, \ldots$.

Можно было бы запрограммировать подсчет по формулам (1.42) и (2.42) отдельно в подготовке цикла, и только счет по формуле (3.42) включить в цикл. Однако, введя счетчик и меняя, в зависимости от него, содержимое ячеек Δ_1, Δ_2, которые будут использоваться взамен h, можно включить в цикл и первые две формулы. Для этого только, кроме ячейки y, мы будем пользоваться еще одной ячейкой для хранения предыдущего значения решения, которую будем обозначать y_{-1}.

Читатель самостоятельно сумеет разобраться в работе программы, которую мы приводим:

		h ·	«0,1»	$=R_1$
		b —	R_1	=Эталон
			a	$=x$
			y_0	$=y_{-1}$
			y_0	$=y$
		h ·	«1/2»	$=\Delta_1$
			Δ_1	$=\Delta_2$
	Б	0	□	k
⊕		k +	«1»	$=k$
			Счет y'	
		y ·	Δ_1	$=R_1$
		y_{-1} +	R_1	$=R_1$
		x +	Δ_2	$=x$
		k —	«3»	$=0$
	У0		T	
		k ↛	«1»	$=0$
	У1			
		·	Δ_1	$=\Delta_2$
		Δ_2 +	Δ_2	$=\Delta_1$
	Б	R_1	□	y
Т			y	$=y_{-1}$
			R_1	$=y$
□			Печать чисел	
		x	y	2
		x —	Эталон	$=0$
	У1		⊕	
			стоп	

Написанные программы, разумеется, не являются стандартными, хотя число команд, требующих формирования, здесь невелико, и это формирование отняло бы немного места. Тем не менее, мы не станем рассматривать этого вопроса, поскольку методы Эйлера применяются обычно лишь для прикидочных расчетов.

§ 43. Программа метода Рунге—Кутта

Метод Рунге — Кутта для численного решения дифференциальных уравнений очень часто применяется при машинных расчетах, главным образом, благодаря тому, что он является «самоначинающимся»: формулы метода применимы, начиная с $n=0$, и никакого предварительного «входа в таблицу» не требуется. Более того, часто даже при использовании других методов, например, Адамса или Милна, формулы Рунге—Кутта применяются для получения требуемых начальных «разгонных» значений.

Начнем с уравнения первого порядка

$$y'=f(x,\ y) \tag{1.43}$$

с начальным условием

$$x=x_0,\ y=y_0.$$

Выделим три пары ячеек x_0, y_0, $\tilde{x}$, $\tilde{y}$ и $x_{\text{раб}}$, $y_{\text{раб}}$. Ячейки x_0, y_0 содержат, очевидно, заданные начальные значения, в ячейках $\tilde{x}$, $\tilde{y}$ будут записаны очередные значения аргумента и функции, а $x_{\text{раб}}$, $y_{\text{раб}}$ служат для промежуточных вычислений.

Таким образом, мы будем составлять программу для одного шага интегрирования. При интегрировании по участку ее следует включать в цикл в качестве рабочей части.

Запишем сначала программирование по формулам (1.38) «в лоб». При этом заметим только, что вычисление приращения k по формуле (2.38) не обязательно производить лишь после того, как вычислены все промежуточные величины k_1, ..., k_4; можно прибавлять в ячейку $\tilde{y}$ величины $k_1/6$, $k_2/3$, ... по мере их получения. По этой причине удобнее вычислять не k_1, а $k_1/2$.

Для вычисления значения производной предположим, что написан отдельный блок, который мы будем называть СПЧ (*счет правой части*); он должен брать аргументы из ячеек $x_{\text{раб}}$, $y_{\text{раб}}$ и помещать ответ в ячейку f.

Программу одного шага вычислений по формулам (1.38) можно теперь записать следующим образом:

1)	$\Omega = конец$	17)	$\tilde{y} + R_2 = \tilde{y}$
2)	$x_0 = \tilde{x}$	18)	$x_0 + h/2 = x_{раб}$
3)	$y_0 = \tilde{y}$	19)	СПЧ
4)	$x_0 = x_{раб}$	20)	$f \cdot h = R_1$
5)	$y_0 = y_{раб}$	21)	$y_0 + R_1 = y_{раб}$
6)	$h \cdot$ «1/2» $= h/2$	22)	$R_1 \cdot$ «1/3» $= R_2$
7)	СПЧ	23)	$\tilde{y} + R_2 = \tilde{y}$
8)	$f \cdot h/2 = R_1$	24)	$x_0 + h = x_{раб}$
9)	$y_0 + R_1 = y_{раб}$	25)	СПЧ
10)	$R_1 \cdot$ «1/3» $= R_2$	26)	$f \cdot h/2 = R_1$
11)	$\tilde{y} + R_2 = \tilde{y}$	27)	$y_0 + R_1 = y_{раб}$
12)	$x_0 + h/2 = x_{раб}$	28)	$R_1 \cdot$ «1/3» $= R_2$
13)	СПЧ	29)	$\tilde{y} + R_2 = \tilde{y}$
14)	$f \cdot h/2 = R_1$	30)	$x_0 + h/2 = x_{раб}$
15)	$y_0 + R_1 = y_{раб}$	31)	$\tilde{x} + h = \tilde{x}$
16)	$R_1 \cdot$ «2/3» $= R_2$	32)	$конец$

Прослеживая работу приведенной программы, читатель заметит стремление программиста сделать как можно более похожими друг на друга четыре группы команд 8)—12), 14)—18), 20)—24) и 26)—30), идущие вслед за обращением к счету правых частей. На самом деле, команда 18) не нужна, так как команда 12) уже обеспечила в ячейке $x_{раб}$ нужное содержимое, которое не изменялось. Так же не нужны и команды 27) и 30), поскольку после обращения к правым частям по команде 25) значения $x_{раб}$, $y_{раб}$ уже не потребуются.

Однако добавление этих команд позволяет достигнуть нужного единообразия. Посмотрим теперь, можно ли для сокращения программы включить эти группы в цикл. Этим условиям удовлетворяют пока только вторая и четвертая команды группы (например, 9) и 11)), так как они неизменны во всех группах. В третьей команде

(в первой группе — 10)) второй множитель равен 1/3 во всех группах, кроме второй, где он равен 2/3. Аналогично изменяются и крайние команды групп — первая и последняя: второй адрес этих команд содержит $h/2$, тогда как в третьей группе он заменяется на h.

Из сказанного ясно, что эти группы можно объединить в цикл, заменив вторые адреса в изменяющихся командах адресами рабочих ячеек, содержимое которых будет меняться нужным способом, в зависимости от счетчика. Для экономии ячеек арифметический цикл организуем с обратным изменением счетчика. Тогда программу можно записать так:

```
                                  Ω    = конец
                                  x0   = x̃
                                  y0   = ỹ
                                  x0   = x раб
                                  y0   = y раб
          h     •        «1/2»   = h/2
                       (0, 0, 3) = s
                          СПЧ                  ↑
          s     ↛      (0, 0, 1) = 0           |
          ------------------------>            |
У0        h/2            Я + 2     R0         |
          R0    +        R0      = R0         |
          s     ↛      (0, 0, 2) = 0           |
          ------------------------>            |
У0        «1/3»          Я + 2     R1         |
          R1    +        R1      = R1         |
          f     ·        R0      = R2         |
          y0    +        R2      = y раб      |
          R2    ·        R1      = R3         |
          ỹ     +        R3      = ỹ          |
          x0    +        R0      = x раб      |
          s     —,     (0, 0, 1) = s          |
У0                                             |
          ------------------------------------
          x̃     +          h     = x̃
                конец
```

Теперь мы можем перейти к задаче решения по методу Рунге—Кутта системы дифференциальных уравнений. Это делается по формулам типа (6.40). Рассмотрим сразу достаточно общую задачу системы n уравнений с n неизвестными функциями $x, y, \ldots, z$ от аргумента t:

$$\left.\begin{aligned} \frac{dx}{dt} &= f_1(t, x, y, \ldots, z), \\ \frac{dy}{dt} &= f_2(t, x, y, \ldots, z), \\ &\cdots\cdots\cdots\cdots \\ \frac{dz}{dt} &= f_n(t, x, y, \ldots, z), \end{aligned}\right\} \qquad (2.43)$$

с начальными условиями при $t=t_0$:

$$x=x_0,\ y=y_0,\ \ldots,\ z=z_0. \qquad (3.43)$$

Выделим для программы три группы по $n+1$ ячеек

$$t_0,\ x_0,\ y_0,\ \ldots,\ z_0$$

для начальных значений,

$$\tilde{t},\ \tilde{x},\ \tilde{y},\ \ldots,\ \tilde{z}$$

для очередных значений и

$$t_{\text{раб}},\ x_{\text{раб}},\ y_{\text{раб}},\ \ldots,\ z_{\text{раб}}$$

для рабочих значений переменных. Кроме того, предположим заданным шаг h по независимой переменной t и блок счета правых частей, который берет аргументы из ячеек $t_{\text{раб}}, x_{\text{раб}}, y_{\text{раб}}, \ldots, z_{\text{раб}}$ и помещает ответы в ячейки $f_1, f_2, \ldots, f_n$. Порядок системы будем предполагать записанным в адресной форме: в виде числа единиц второго адреса ячейки n.

Для одного шага Рунге—Кутта можно воспользоваться предыдущей программой, заменив команды пересылки и образования новых значений $\tilde{y}$ и $y_{\text{раб}}$ циклами с переадресацией для получения соответствующих значений всех требуемых функций. Переадресацию можно организовать с помощью регистра адреса.

Получаем такую программу:

				Ω	=	*конец*
	[*РА*]	0*		*n*		*предконец*
А				t_0^*	=	t^*
Б				t_0^*	=	$t_{раб}^*$
	РА ⩾	$\bar{1}$		*Я* — 2		$\bar{F}^*$
В		h	·	«1/2»	=	$h/2$
		(0, 0, 4) —,		(0, 0, 1)	=	s
Г	*БВ*			СПЧ		
		s	$\not\sim$	(0, 0, 1)	=	0
	У0	$h/2$		*Я* + 2		R_0
		R_0	+	R_0	=	R_0
		s	$\not\sim$	(0, 0, 2)	=	0
	У0	«1/3»		*Я* + 2		R_1
		R_1	+	R_1	=	R_1
	[*РА*]	0		*n*		0
Д		f_0^*	·	R_0	=	R_2
Е		t_0^*	+	R_2	=	$t_{раб}^*$
		R_2	·	R_1	=	R_3
Ж		$\tilde{t}^*$	+	R_3	=	$\tilde{t}^*$
	РА ⩾	$\bar{2}$				F^*
З		t_0	+	R_0	=	$t_{раб}$
		s	—,	(0, 0, 1)	=	s
	У0			*Г*		
И		$\tilde{t}$	+	h	=	$\tilde{t}$
				предконец		
				конец		

Написанная программа не является стандартной. Ее команды, помеченные русскими буквами *А* — *И*, зависят от размещения групп ячеек $t_0, \ldots, t, \ldots, t_{раб}, \ldots, f_1, \ldots$. Но эти команды нетрудно сформировать, превратив программу в стандартную программу с информацией. Назовем ее *Шаг РК* и допустим, что обращение к ней

имеет вид

БВ	*Я*+4	*Шаг РК*	Ω
	t_0	$t_{раб}$	$\tilde{t}$
	h	СПЧ	f_1
	0	0	n

Здесь t_0, $t_{раб}$, $\tilde{t}$ — адреса первых из соответствующих групп по $n+1$ ячеек подряд, f_1 — адрес первой ячейки из группы в n ячеек (в команде *Д* встречается адрес f_0; это адрес ячейки, предшествующей f_1. Как легко убедиться, сама эта ячейка в вычислениях не участвует). Порядок системы предполагается заданным адресно в третьем адресе третьей ячейки информации.

Выпишем сначала начальные состояния формируемых команд — «заготовки». Заметим только, что при выбранном способе задания информации команду *А* удобнее формировать в виде

$$A \mid t_0^* \vee 0 = \tilde{t}^*.$$

В соответствии с этим требуемые заготовки можно представить в виде

A_0	0*	$\vee$	0	$=0^*$	E_0	$0^*+R_2=0^*$
$Б_0$	0	$\vee$	0*	$=0^*$	$Ж_0$	$0^*+R_3=0^*$
B_0	0	·	«1/2»	$=h/2$	$З_0$	$0+R_0=0$
$Г_0$	*БВ*	*Я*+1	0	Ω	$И_0$	$0+0=0$
$Д_0$	F^*	·	$R_0=R_2$			

Формирующую часть следует начать переносом ячеек информации в рабочие ячейки программы, которые мы обозначим i, j, n. Тогда начало программы будет выглядеть так:

		Ω	=	*конец*
[*PA*]	0*	Ω		*предконец*
	$\overline{114}$	→, F^*	=	n
		$(F-1)^*$	=	j
		$(F-2)^*$	=	i

Остается написать формирование, что делается уже

довольно просто:

i	$\wedge$	$(F, 0, F) = R_0$		E_0	$+,$	R_0	$= E$
A_0	$+,$	R_0	$= A$	$З_0$	$+,$	R_0	$= З$
$\overline{64}$	$\rightarrow,$	i	$= R_0$	i	$\wedge$	$(0, 0, F) = R_0$	
$Б_0$	$+,$	R_0	$= Б$	$\overline{130}$	$\rightarrow,$	R_0	$= R_1$
j	$\wedge$	$(0, F, 0) = R_0$		R_0	$\vee$	R_1	$= R_0$
$Г_0$	$+,$	R_0	$= Г$	$Ж_0$	$+,$	R_0	$= Ж$
$\overline{130}$	$\rightarrow,$	j	$= R_0$	j	$\wedge$	$(F, 0, 0) = R_1$	
$Д_0$	$+,$	R_0	$= Д$	B_0	$+,$	R_1	$= B$
i	$\wedge$	$(F, 0, 0) = R_0$		$\overline{64}$	$\rightarrow,$	R_1	$= R_1$
$\overline{64}$	$\rightarrow,$	i	$= R_1$	R_0	$\vee$	R_1	$= R_0$
R_1	$\wedge$	$(0, 0, F) = R_1$		$И_0$	$+,$	R_0	$= И$
R_0	$\vee$	R_1	$= R_0$				

После формирующей части можно помещать рабочую часть программы, которая будет теперь иметь вид

```
      | [PA]  0     n      0
   A  |          н. п.
   Б  |          н. п.
      |  . . . . . . . . . . .
   И  |          н. п.
      |        предконец
      |          конец
```

после чего следует разместить команды-заготовки $A_0 — И_0$, выписанные выше.

§ 44. Программирование методов прогноза и коррекции. Текущий контроль точности вычислений

Методы прогноза и коррекции программируются сложнее метода Рунге — Кутта, в первую очередь потому, что последний является «самоначинающимся» и не требует различного подхода для различных шагов. Уже при программировании уточненного метода Эйлера в § 42 мы видели, что необходимость получения очередных значений по различным формулам ведет к значительному

усложнению и удлинению программы. В отношении простоты программы метод Рунге — Кутта представляет наибольшие удобства и не случайно он является одним из наиболее распространенных. Стандартная программа интегрирования по методу Рунге — Кутта имеется практически в любой библиотеке стандартных программ.

Однако при интегрировании уравнений по методу Рунге — Кутта очень затруднен контроль точности. Так как применение теоретических оценок исключено, то практический контроль точности опирается лишь на сравнение результатов численного интегрирования, полученных с различным шагом.

Ясно, что такая проверка точности требует увеличения расхода машинного времени по крайней мере вдвое, так как, в лучшем случае, все интегрирование по требуемому участку придется произвести дважды. В ряде случаев это увеличение может быть значительно бо́льшим: придется либо интегрировать по всему требуемому участку с очень маленьким шагом, нужным для наиболее неблагоприятной его части, либо по нескольку раз пересчитывать решение, увеличивая или уменьшая шаг для каждой части участка интегрирования.

Существенным преимуществом методов прогноза и коррекции перед методами типа Рунге — Кутта является возможность судить о́ точности каждого шага по разности между прогнозированным и скорректированным значениями решения. При этом можно добиваться повышения точности и без смены шага интегрирования, пользуясь лишь итерациями формулы коррекции. Это преимущество искупает трудности программирования.

Исходя из сказанного выше, можно при интегрировании уравнений каким-либо методом прогноза и коррекции заранее фиксировать шаг интегрирования, и для достижения требуемой точности итерировать формулу коррекции до тех пор, пока очередные значения y_n и y_n' не перестанут изменяться больше, чем на заданное ε. Но такое прямолинейное решение вопроса может в отдельных случаях потребовать чересчур большого числа итераций. Поэтому наиболее разумным представляется компромиссное решение этого вопроса, изложенное ниже.

Программистом задается первоначальный шаг интегрирования и требуемая точность получаемого решения. Формула коррекции итерируется до тех пор, пока разность между последовательными значениями y_n не сделается меньше заданного ε или пока число итераций не достигнет заданного в программе (или самим программистом) эталона. В первом случае программа переходит к следующему шагу интегрирования, во втором — изменяет шаг заданным образом, обычно — уменьшает его вдвое.

Рассмотрим программирование методов прогноза и коррекции. Для простоты ограничимся простейшим из них, основанным на уточненном методе Эйлера и квадратурной формуле трапеций (см. § 39, стр. 230). При этом мы не будем касаться начала решения, программирование которого рассмотрено в § 42, а напишем программу для одного шага, с использованием указанных выше приемов текущего контроля точности вычислений.

Итак, предположим, что в ячейках x_n, y_n находятся очередные значения аргумента x и функции $y(x)$, являющейся решением уравнения

$$y' = f(x, y), \tag{1.44}$$

в ячейке y_{n-1} — значение $y(x)$ на предыдущем шаге, используемое в уточненном методе Эйлера, в ячейке h — выбранный шаг интегрирования, и имеется блок *Счет y'*, вычисляющий по уравнению (1.44) значение производной y' в заданной точке. Этот блок берет аргументы из ячеек x, y и выдает значение y' в ячейку γ.

Нашей задачей является получение в ячейках x, y очередных значений аргумента и решения, пользуясь последовательностью формул (3.39) и (5.39). Кроме того, мы будем предполагать заданными эталоны точности (ε) и числа итераций (*ЭИ*), которые записаны в ячейках, обозначенных так же.

Первые команды программы, вычисляющие в ячейке y прогнозированное значение и запоминающие значение y'_n в ячейке R_0, достаточно ясны и комментариев не

требуют:

	$\Omega =$ *конец*	$\gamma = R_0$
Н)	$x_n = x$	$R_0 \cdot h = R_1$
	$y_n = y$	$R_1 + R_1 = R_1$
Счет	y'	$y_{n-1} + R_1 = y$

Следующие команды имеют целью вычислить скорректированное значение y, которое сперва помещается в ячейку R_2 для сравнения, и подготовить цикл для итерирования формулы коррекции. Счетчик числа итераций обозначен буквой k.

	$x_n + h = x$	$R_0 + \gamma = R_1$
Б	«1» $\overrightarrow{Я + 2\ k}$	$R_1 \cdot «1/2» = R_1$
И)	$k + «1» = k$	$R_1 \cdot h = R_1$
	Счет y'	$y_n + R_1 = R_2$

Вычислив скорректированное значение y, мы можем теперь проверять окончание цикла. При этом, как сказано выше, цикл носит двойной характер: он является одновременно и итерационным, и арифметическим. В соответствии с этим, здесь необходима двойная проверка окончания.

	$y - R_2 = R_3$
	$\lvert R_3\rvert - \lvert\varepsilon\rvert = 0$
У1	$\overrightarrow{R_2\ \text{конец}\ y}$
	$k - ЭИ = 0$
У1	*И)*
БВ Н)	*Изменение шага* Ω
	конец

Блок *Изменение шага* должен изменить определенным образом шаг интегрирования h. В простейшем случае, как уже было замечено выше, изменение шага сводится к его уменьшению вдвое. В таком случае нет нужды выделять это изменение в отдельный блок, и соответствующую команду можно заменить двумя такими:

$$h \cdot «1/2» = h$$

Б *Н)*

Теперь, после выхода этого блока на конец, можно снова обращаться к нему же для вычисления значений x, y уже в следующей точке, выполнив предварительно команды

$$y_n = y_{n-1}$$
$$x = x_n$$
$$y = y_n$$

которые, впрочем, можно было бы поместить и в конце рассматриваемого блока.

Рассмотренная выше структура текущего контроля точности при численном решении дифференциальных уравнений представляется наиболее рациональной. В ряде случаев полезно перед переходом к вычислению следующих значений x, y выяснить возможность увеличения шага h. Это можно сделать, проверив состояние счетчика k числа итераций. При $k=1$ имеет смысл пытаться производить интегрирование с двойным шагом, так как это равенство означает, что уже первое применение формулы коррекции дает значение y, отличающееся от прогнозированного меньше, чем на ε.

Конечно, метод, который мы программировали, дает не слишком большую точность. Более целесообразно пользоваться методом Милна. Но для этого достаточно лишь заменить в рассматриваемой программе группы команд, соответствующие вычислениям по формулам прогноза и коррекции, оставив общую схему программы, по существу, без изменения. Нужно только держать в памяти несколько предыдущих значений функции и предыдущих значений производной, используемых в методе Милна, и пересылать их в соответствующие ячейки после каждого вычисления очередного значения y. Мы полагаем, что читатель справится с этим самостоятельно.

§ 45. Запись алгоритмов численного решения дифференциальных уравнений на алголе

Запись алгоритмов численного решения дифференциальных уравнений в виде процедур алгола проста и никаких затруднений не вызывает. Нужно только иметь в виду, что наиболее удобно оформлять в качестве про-

цедуры стандартный шаг интегрирования, т. е. переход от значений x_n, y_n к значениям в следующей точке. В тех случаях, когда метод требует начала решения по другим формулам, целесообразно помещать это начало в общей программе до обращения к соответствующей стандартной процедуре метода.

Начнем с рассмотрения процедуры метода Эйлера. Формальными параметрами этой процедуры естественно считать x, y, шаг интегрирования h и функцию $f(x, y)$, выражающую производную, т. е. представляющую конкретное дифференциальное уравнение. Тогда процедуру одного шага решения уравнения $y'=f(x, y)$ по методу Эйлера можно написать так

```
procedure Euler (x, y, h, f); value x, h; real x, y, h;
                    real procedure f;
                    y: =y+h×f;
```

Так как любой шаг интегрирования по методу Эйлера совершается одинаково, с помощью одной и той же формулы, то для решения конкретного уравнения достаточно написать цикл, содержащий на каждом шагу обращение к описанной процедуре. В этом случае программу решения уравнения с правой частью и начальными условиями $x_{нач}$, $y_{нач}$ на отрезке $[x_{нач}, x_{посл}]$ можно написать следующим образом:

```
begin real x, y; real x нач, x посл, y нач, h;
                    real procedure fi; fi: =...;
      read (x нач, x посл, y нач, h);
      for x: =x нач step h until x посл − h do
      begin Euler (x, y, h, fi); print (x+h, y)
      end шага
end программы;
```

В приведенной программе, естественно, пропущено тело процедуры *fi*, поскольку мы не имели в виду никакого конкретного уравнения. Оператор присваивания *fi*: =... определяется правой частью уравнения, которое необходимо интегрировать. Например, для уравнения $y'=\frac{2xy}{x^2+y^2}$ он имеет вид

```
fi: =2 × x × y/(x↑2+y↑2);
```

Несколько сложнее будет обстоять дело для уточненного метода Эйлера. Прежде всего, число формальных параметров для соответствующей процедуры необходимо увеличить, так как, кроме текущего значения y, необходимо держать в памяти также и его предыдущее значение. При каждом шаге интегрирования необходимо, вычислив новое значение функции, переслать старое на место предыдущего; поэтому одним оператором присваивания, как в обычном методе Эйлера, здесь обойтись не удастся.

Процедура уточненного метода Эйлера имеет вид

```
procedure Ут Е (x, y, y пред, h, f); value x, h;
                real x, y, y пред, h; real procedure f;
                begin real k; k:=y пред+2×h×f;
                                  y пред:=y; y:=k;
                end
```

При этом по-прежнему предполагается, что параметрами процедуры f, т. е. аргументами этой функции, являются x, y.

Следующее осложнение состоит в том, что программу уточненного метода Эйлера нельзя писать в виде обычного цикла, как мы это сделали в предыдущем случае так как процедуру *Ут Е* можно включать в цикл, лишь начиная со второго шага. Поэтому первый шаг необходимо написать отдельно. Тогда программу решения уравнения с помощью уточненного метода Эйлера можно написать в следующем виде (тело процедуры *fi* по тем же причинам, что и раньше, пропущено):

```
begin real x, y, y пред; real x нач, x посл, y нач, h;
                         real procedure fi; fi:=...;
    read (x нач, x посл, y нач, h);
    x:=x нач; y:=y нач; y:=y+h×fi/2; x:=x+h/2;
                  y:=y нач+h×fi;
    y пред:=y нач; print (x нач+h,y);
    for x:=x нач+h step h until x посл-h do
        begin Ут Е (x, y, y пред, h, fi); print (x+h, y)
        end шага
end программы.
```

Для метода Рунге — Кутта, рассмотренного в §§ 38, 43, процедура пишется дольше, чем для двух рассмотренных выше методов, но зато программу интегрирования можно писать в виде простого цикла, благодаря тому, что метод Рунге — Кутта является самоначинающимся, и любой шаг интегрирования здесь совершается одинаково. По этой же причине формальные параметры для процедуры метода Рунге — Кутта будут такими же, как и для метода Эйлера.

Приведем процедуру одного шага метода Рунге — Кутта.

```
procedure RK (x, y, h, f); value x, h; real x, y, h;
                                real procedure f;
  begin real k1, k2, k3, k4, y0;
    k1:=h×f;   x:=x+h/2; y0:=y; y:=y0+k1/2
    k2:=h×f;   y:=y0+k2/2;
    k3:=h×f;   y:=y0+k3; x:=x+h/2;
    k4:=h×f;
    y:=y0+(k1+2×k2+2×k3+k4)/6
  end;
```

Здесь мы снова предполагаем, что функция f описана как процедура без параметров с аргументами x, y. Поэтому значения аргументов нужно изменять программно после каждого шага. Если бы f описывалась как процедура с параметрами $f(x, y)$, то можно было бы сразу писать, например,

$$k3:=h\times f(x+h/2,\ y+k2/2),$$

не вычисляя этих значений отдельно. Тогда не понадобилось бы использовать идентификатор $y0$ для запоминания первоначального значения y.

Написанную процедуру можно включить в цикл вместо процедуры *Euler*, и мы получим программу для решения уравнения методом Рунге — Кутта. Предоставляем проделать это читателю.

РЕКОМЕНДОВАННАЯ ЛИТЕРАТУРА

Брудно А. Л., Программирование в содержательных обозначениях, изд. 2-е, исправл., «Наука», 1968.

Брудно А. Л., Алгол, изд. 2-е, перераб., «Наука», 1971.

Гутер Р. С., Арлазаров В. Л., Усков А. В., Практика программирования, «Наука», 1965.

Гутер Р. С., Овчинский Б. В., Элементы численного анализа и математической обработки результатов опыта, изд. 2-е, перераб., «Наука», 1970.

Гутер Р. С., Овчинский Б. В., Резниковский П. Т., Программирование и вычислительная математика, «Наука», 1965.

Демидович Б. П., Марон И. А., Основы вычислительной математики, изд. 4-е, исправл., «Наука», 1970.

Демидович Б. П., Марон И. А., Шувалова Э. З., Численные методы анализа, изд. 3-е, перераб., «Наука», 1967.

Кронрод А. С., Узлы и веса квадратных формул, «Наука», 1964.

Лавров С. С., Универсальный язык программирования, изд. 2-е, перераб., «Наука», 1967.

Ляшенко В. Ф., Программирование для цифровых вычислительных машин М-20, БЭСМ-3М, БЭСМ-4, М-220, «Советское радио», 1967.

Мак-Кракен Д. Д., Программирование на алголе, «Мир», 1964.

Резниковский П. Т., Монахов В. М., Программирование для одноадресных машин, «Просвещение», 1968.

Хемминг Р. В., Численные методы (для научных работников и инженеров), «Наука», 1968.

Шилтере М. Я., Программирование на БЭСМ-2 «Просвещение», 1966.

Рафаил Самойлович Гутер,
Павел Тувьевич Резниковский

ПРОГРАММИРОВАНИЕ
И ВЫЧИСЛИТЕЛЬНАЯ
МАТЕМАТИКА

вып. 2

М., 1971 г., 264 стр. с илл.

Редакторы: *И. М. Овчинникова* и *И. А. Румянцев*

Техн. редактор *Л. А. Пыжова*

Корректор *И. Б. Мамулова*

Сдано в набор 5/IV 1971 г. Подписано к печати 17/VIII 1971 г. Бумага 84×108$^{1}/_{32}$. Физ. печ. л. 8,25. Условн. печ. л. 13,86. Уч.-изд. л. 11,87. Тираж 70 000 экз. Т-14309. Цена книги 49 коп. Заказ № 1685.

Издательство «Наука»
Главная редакция
физико-математической литературы
Москва, В-71, Ленинский проспект, 15.

Ордена Трудового Красного Знамени Ленинградская типография № 1 «Печатный Двор» имени А. М. Горького Главполиграфпрома Комитета по печати при Совете Министров СССР, г. Ленинград, Гатчинская ул., 26.