МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

Кафедра автоматики и телемеханики

В. В. ЧУРКИН

ЧИСЛЕННЫЕ МЕТОДЫ

(с алгоритмами и программами в среде C++Builder)

Учебно-методическое пособие

Киров

2014

УДК 519.6:004.43(07)

Ч-933

Допущено к изданию методическим советом
факультета автоматики и вычислительной техники
ФГБОУ ВПО «ВятГУ» в качестве учебно-методического пособия для
студентов направления 220400.62 «Управление в технических системах»
профиля «Управление и информатика в технических системах»
всех форм обучения

Рецензент
доктор технических наук, профессор кафедры электронных
вычислительных машин ФГБОУ ВПО «ВятГУ» Д. А. Страбыкин

Чуркин, В.В.

Ч-933 Численные методы (с алгоритмами и программами в среде C++Builder): учебно-методическое пособие / В. В. Чуркин. – Киров: ФГБОУ ВПО «ВятГУ», 2014. – 247 с.

УДК 519.6:004.43(07)

В издании излагаются численные методы согласно программе дисциплины «Численные методы». В учебно-методическом пособии приведены основные математические сведения, алгоритмы и примеры использования методов в виде лабораторных работ в среде *Mathcad* и в интегрированной среде разработки приложений *C++Builder 6*.

Оглавление

ВВЕДЕНИЕ

При составлении данного учебно-методического пособия была поставлена задача – при изучении дисциплины обеспечить условия приобретения студентами практических навыков использования численных методов в объеме, предусмотренном программой курса «Численные методы».

Математические описания методов, основные положения которых сопровождаются доказательствами, даны как подготовка к разработке алгоритмов, реализующих методы. Ряд основных методов реализован в лабораторных работах, выполняемых в среде *MathCad* и в среде C++Builder, причем результаты, полученные в *MathCad*'е, используются для тестирования и отладки кодов в среде *C++Builder*.

В разделе «Численные методы решения нелинейных уравнений» приведена иллюстрация и алгоритм отделения корней нелинейного уравнения, формулы, иллюстрации и алгоритмы методов уточнения корней – деления пополам, хорд, касательных, секущих и итераций. Представлены также алгоритмы комбинированных методов уточнения корней. В лабораторной работе этого раздела выполнено в *MathCad*'е построение графика функции и найден корень уравнения, в *C++Builder* исследованы методы деления пополам и секущих.

Раздел «Численные методы интерполяции и экстраполяции» содержит формулы и алгоритмы алгебраической интерполяции – прямой и обратной по Лагранжу, формулы и алгоритм тригонометрической интерполяции, описание и использование интерполяций – кусочно-линейной и сплайновой в *MathCad*'е. В лабораторной работе применены кусочно-линейная и сплайновая интерполяции в *MathCad*'е и тригонометрическая интерполяция в среде *C++Builder*.

В третьем разделе пособия «Алгебра, формирование, разложение и обращение матриц» содержатся, кроме определений, перечня функций и операций с векторами и матрицами, в том числе и в *MathCad*'е, алгоритмы формирования различных видов матриц, формулы и алгоритмы методов разложения матриц и описания методов обращения матриц, реализуемых машинными алгоритмами. В лабораторной работе, выполняемой в среде *C++Builder*, формируются разнотипные числовые матрицы и вычисляется матрица-произведение.

Раздел «Численные методы решения систем линейных алгебраических уравнений (СЛАУ)» содержит описания прямых и итерационных методов решения СЛАУ, алгоритм прямого метода Гаусса по схеме с частичным выбором ведущего элемента по столбцу и алгоритм итерационного метода Гаусса-Зейделя. На конкретном примере даны рекомендации к вычислению матричных выражений. Показано, как решать СЛАУ в *MathCad*'е. В лабораторной работе показана методика исследования решения СЛАУ прямым методом Гаусса.

«Численные методы аппроксимации по критерию наименьших квадратов» содержатся в пятом разделе пособия. Здесь рассмотрены: аппроксимация алгебраическим полиномом произвольной степени с выводом формул для вычисления параметров полинома и разработкой алгоритма, в *MathCad*'е – аппроксимации – линейная и линейная общего вида. Все эти методы аппроксимации использованы в лабораторной работе раздела.

В разделе «Численные методы вычисления определенных интегралов» рассмотрены методы Ньютона-Котеса и Гаусса в общем случае и в частных случаях – формулы прямоугольников, трапеций, Симпсона, Ньютона. Приведены две группы алгоритмов. В первой группе содержатся алгоритмы для получения зависимостей фактической ошибки вычисления определенного интеграла от шага интегрирования, во

второй – алгоритмы для получения зависимостей затрат машинного времени и фактической ошибки вычисления интеграла от задаваемой ошибки. Выполняемое в лабораторной работе исследование позволяет сравнивать по эффективности методы Ньютона-Котеса и Гаусса равных порядков.

Пособие завершается разделом «Численные методы интегрирования обыкновенных дифференциальных уравнений (ОДУ) и систем ОДУ». Здесь описана методика сравнения методов интегрирования ОДУ, приведены иллюстрации методов Эйлера – базового и модификаций, простого метода прогноза и коррекции. Приведены также формулы для практических одношаговых и многошаговых методов, даны примеры разработки алгоритмов этих методов. Приведены примеры интегрирования ОДУ в *MathCad*'е. В лабораторной работе исследованы одношаговый метод РК3 и метод прогноза и коррекции – метод Адамса с разгоном методом РК4.

Пособие содержит восемь заданий по 30 вариантов; два задания находятся в разделе «Численные методы решения систем линейных алгебраических уравнений (СЛАУ)».

Каждый раздел завершается перечнем контрольных вопросов, необходимых для проверки и контроля знаний студентов.

ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЙ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Краткие сведения

Рассматриваемые ниже методы относятся к алгебраическим и трансцендентным уравнениям, а также к уравнениям, содержащим нелинейные функции, например, показательную, логарифмическую и т. д.

Алгебраическое уравнение

$$a_0 x^m + a_1 x^{m-1} + \ldots + a_m = 0$$

имеет m корней. В общем случае коэффициенты a_i могут быть действительными и комплексными; здесь используется случай действительных коэффициентов.

Трансцендентные уравнения имеют бесконечное множество корней, например, уравнение $arctgx - x = 0$, которое содержит периодическую тригонометрическую функцию.

Процесс вычисления корней нелинейных уравнений состоит из двух этапов:

1) отделение корней, т. е. установление возможно более тесных промежутков $[a,b]$, в которых содержится один и только один корень уравнения $f(x) = 0$;

2) уточнение отделенных корней, т. е. доведение значений корней до заданной степени точности.

Отделение корней

Для отделения корней можно воспользоваться методом линейного поиска, в котором диапазон поиска $[x_н, x_в]$ проходится с шагом $\Delta x = b - a$; при выполнении условия $f(x) * f(x + \Delta x) < 0$ принимается решение о наличии корня в промежутке $[a,b]$. В общем случае в

диапазоне поиска может оказаться несколько корней ($m \succ 1$), к каждому из которых следует применить операцию уточнения. Иллюстрация и алгоритм отделения корней представлены соответственно на рис. 1.1 и 1.2 приложения.

Уточнение корней

Существует несколько основных методов уточнения корней уравнений.

1. Метод деления пополам (метод дихотомии, метод бисекций)

Это наиболее надежный алгоритм, особенно когда о поведении $f(x)$ известно только, что $f(x)$ – функция действительной переменной x и известен интервал $[a,b]$, на котором $f(x)$ меняет знак (рис. 1.3). Следовательно, между a и b существует точка, в которой функция обращается в нуль. Если разделить интервал пополам и узнать, больше

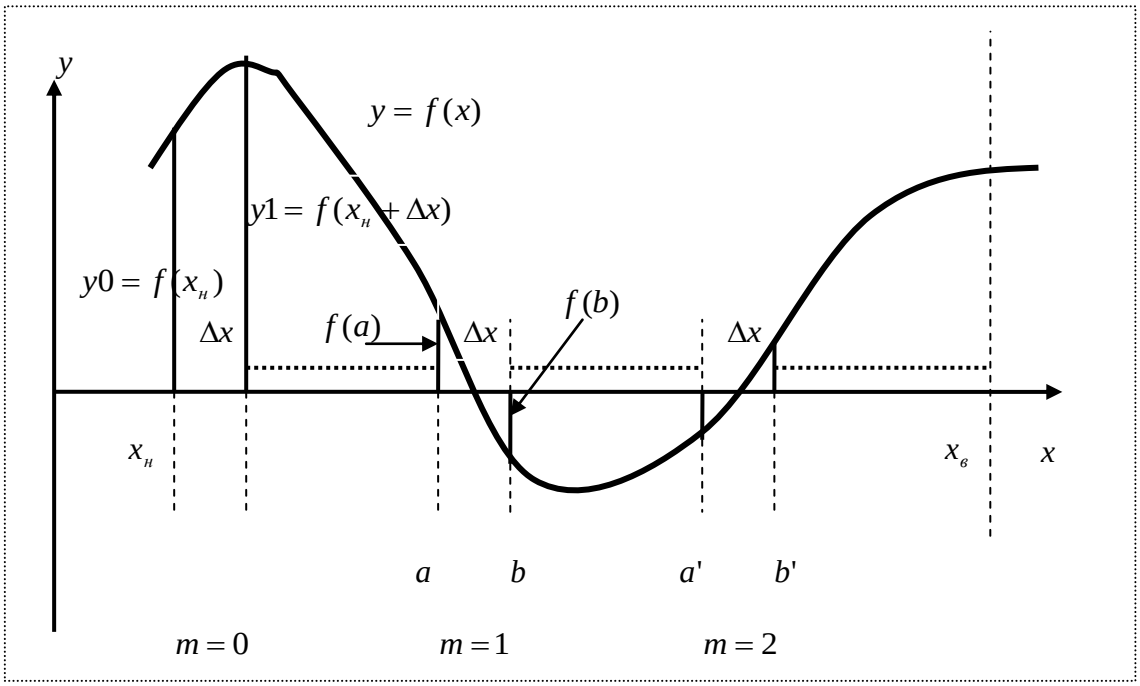


Рис. 1.1. Иллюстрация к отделению корней

нуля или меньше нуля функция в точке деления, то можем указать подынтервал, в котором функция меняет знак. Последующим делением

указываемых подынтервалов можно сколь угодно близко подойти к корню: например, за 10 шагов интервал с корнем будет уменьшен в 1024 раза. При заданной абсолютной точности ε алгоритм метода деления пополам состоит из следующих шагов (рис. 1.4).

1. Вычислить $y0 = f(a)$ и $y1 = f((a+b)/2)$. Затраты машинного времени на уточнение корня оценивают косвенно, по количеству обращений к функции $f(x)$ – n, следовательно, n будет инкрементирован дважды.

2. Если знаки $y0$ и $y1$ не совпадают, т. е. $y0 * y1 \le 0$, и $|a-(a+b)/2| \ge \varepsilon$, то нужно заменить b на $(a+b)/2$ и перейти к п.1.

3. Если же при $y0 * y1 \le 0$ $|a-(a+b)/2| \prec \varepsilon$, следует прекратить вычисления, т. к. достигнута заданная точность.

4. Если $y0 * y1 \succ 0$, и $|b-(a+b)/2| \ge \varepsilon$, то нужно заменить a на $(a+b)/2$ и перейти к п.1; в противном случае – прекратить вычисления, так как достигнута заданная точность. Любой из концов отрезка $[a,b]$, а лучше его середина, может быть использована в качестве корня x^* уравнения $f(x)=0$.

Отметим основные достоинства метода деления пополам: 1) абсолютно надежен; 2) скорость сходимости не зависит от вида $f(x)$.

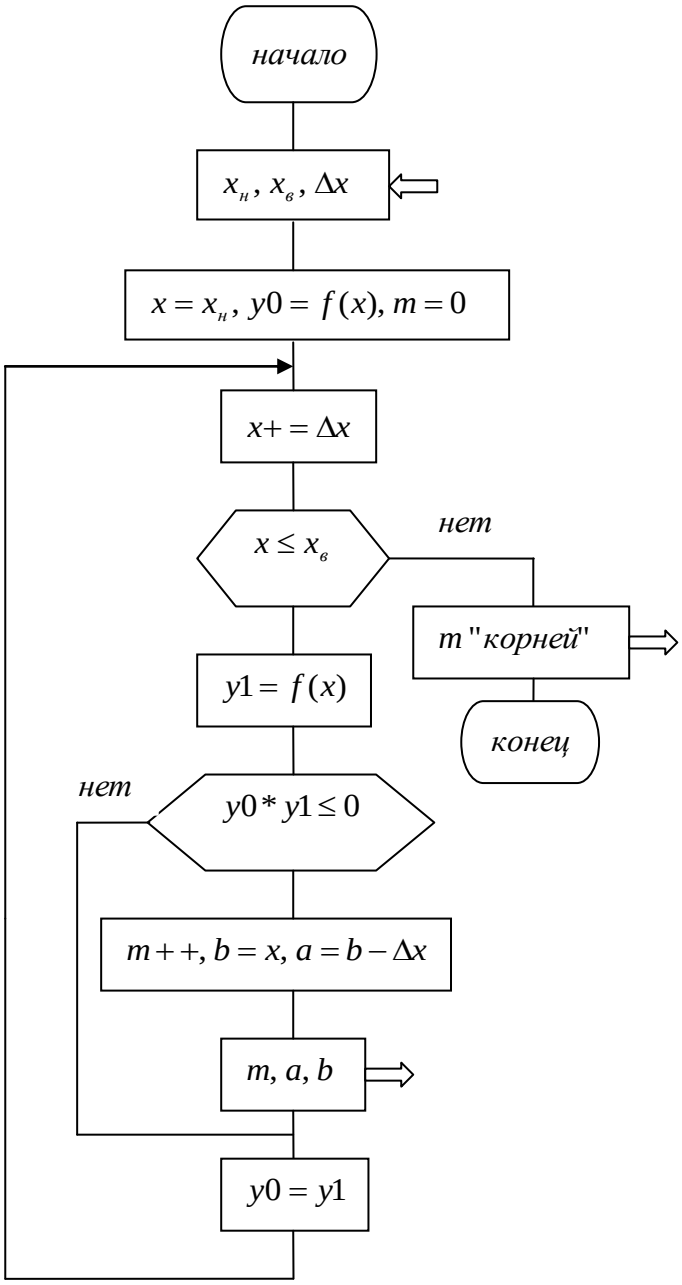


Рис. 1.2. Алгоритм отделения корней

2. Метод хорд

Метод деления пополам будет улучшен, если для следующего вычисления использовать не середину отрезка $[a, b]$, а то значение x, в

котором дает нуль линейная интерполяция между двумя известными значениями функции $f(x)$ противоположного знака (рис. 1.5).

Геометрически способ линейной интерполяции эквивалентен замене кривой $y = f(x)$ хордой, проходящей через точки $A(a, f(a))$ и $B(b, f(b))$.

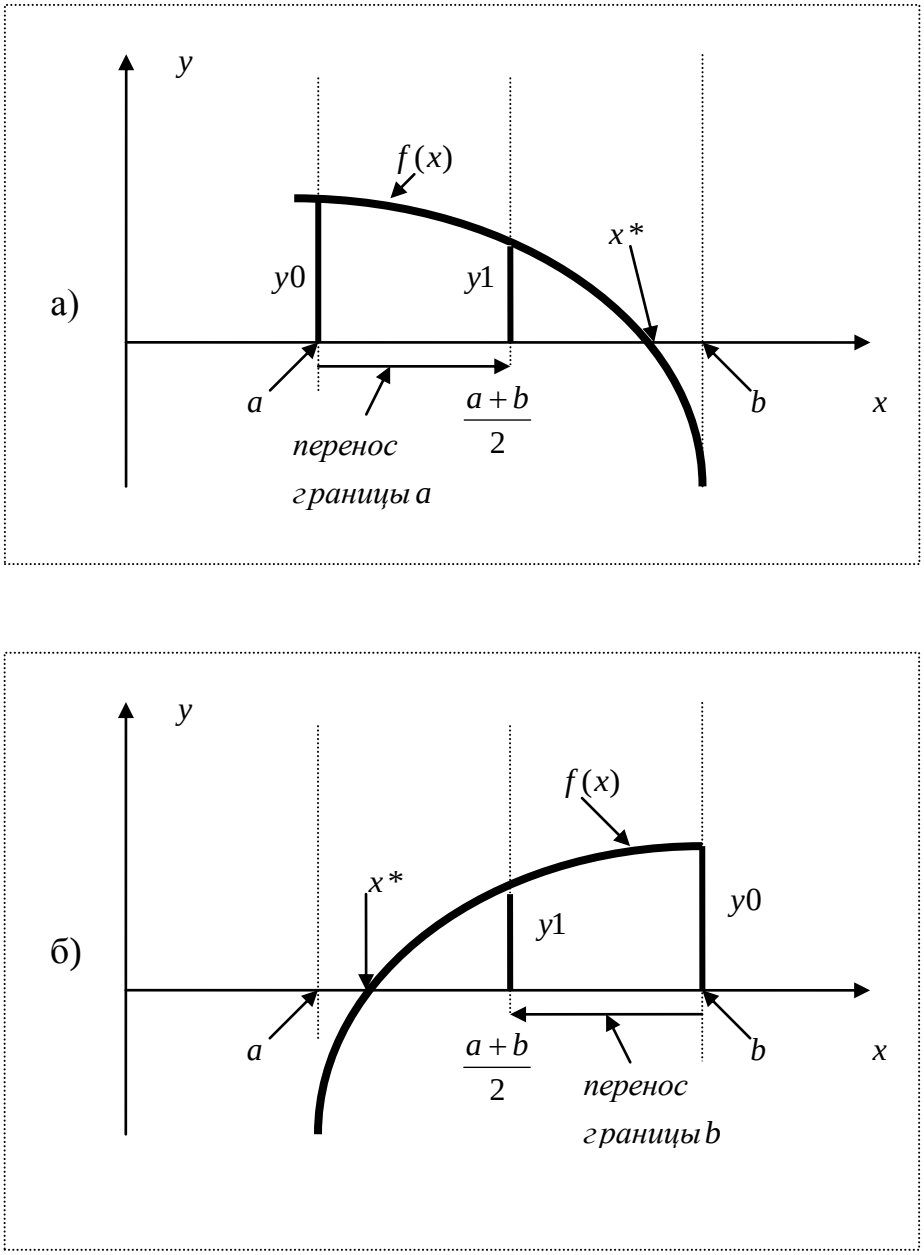


Рис. 1.3 а, б. Иллюстрации к методу деления пополам

Уравнение хорды:

$$(x0 - a)/(b - a) = (y0 - f(a))/(f(b) - f(a)).$$

Полагая $y0 = 0$ и $x0 = c,$ получаем приближение к корню:

$$c = a - f(a) * (b - a)/(f(b) - f(a)). \qquad (1)$$

Алгоритм метода хорд (рис. 1.6):

1. Вычислить $fa = f(a)$ и $fb = f(b)$.

2. Вычислить c по формуле (1) и $fc = f(c)$.

3. Если знаки fa и fc совпадают, т. е. $sign(f(c)) = sign(f(a)),$ то конец b неподвижен. В этом случае приняты: $a = c,$ $fa = fc,$ если $|b - c| \geq \varepsilon$. Затем перейти к п. 2. В противном случае, т. е. при $|b - c| \prec \varepsilon$ вычисления завершены, т. к. заданная точность достигнута.

4. Если $fa * fc \leq 0,$ неподвижен конец a. В случае $|c - a| \geq \varepsilon$: $b = c,$ $fb = fc.$ Затем перейти к п. 2. Иначе – вычисления завершены. Значение c используется как корень уравнения.

Достоинства метода хорд: 1) абсолютно надежен; 2) в большинстве случаев имеет более быструю сходимость, чем метод деления пополам.

Недостаток: скорость сходимости зависит от вида $f(x)$, и поэтому для некоторых функций число шагов на уточнение корня может оказаться большим, чем в методе деления пополам.

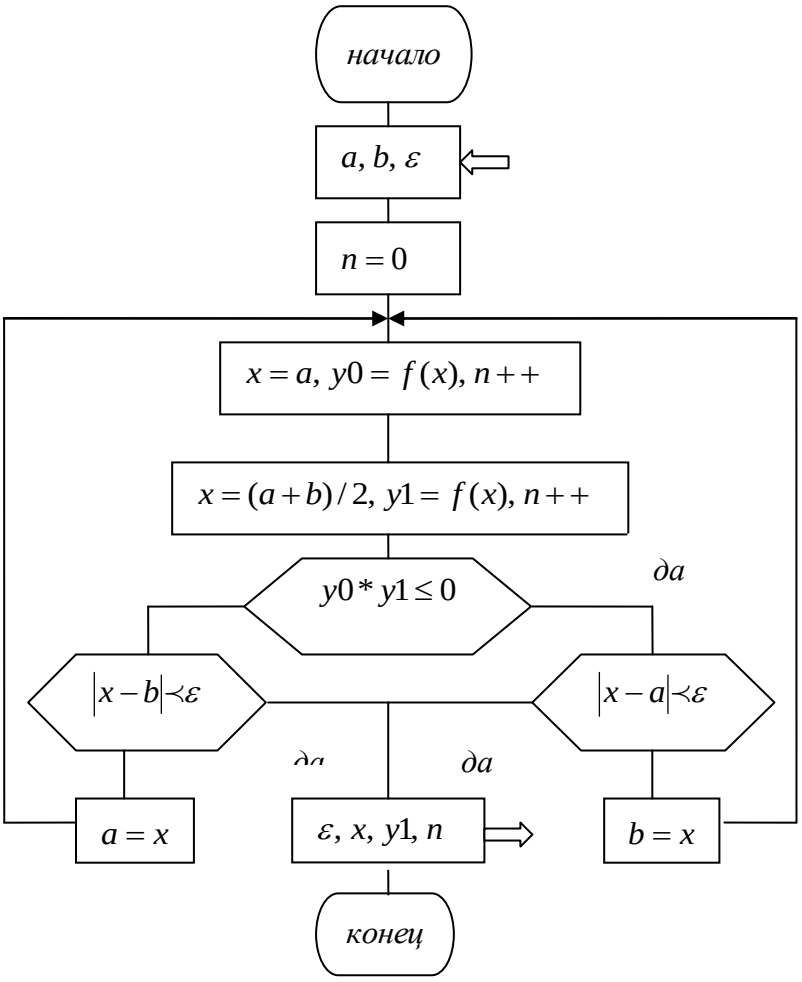


Рис. 1.4. Алгоритм метода деления пополам

3. Метод касательных (метод Ньютона)

Если $f(x)$ имеет одну и более непрерывных производных (т. е. $f(x)$ достаточно гладкая), то можно применить метод Ньютона (метод касательных) и метод секущих, позволяющие сократить число вычислений функции по сравнению с методом деления пополам и методом хорд, т. е. уменьшить затраты машинного времени.

В методе Ньютона каждое новое приближение x_{k+1} вычисляется как единственный нуль касательной прямой к функции $f(x)$ в точке x_k :

$$x_{k+1} = x_k - f(x_k)/f'(x_k),\ \ f'(x_k) \neq 0,\ \ k = 0, 1, 2\ldots .$$

Это итерационная формула метода Ньютона. Каждая итерация требует вычисления не только $f(x)$, но и её производной $f'(x)$.

Иллюстрация к методу касательных представлена на рис. 1.7, а алгоритм метода – на рис. 1.8.

Метод Ньютона обладает хорошей сходимостью. Основная трудность заключается в выборе начального приближения x_0, которое ведет к сходящемуся итерационному процессу. Поэтому методу Ньютона часто предшествует какой-нибудь глобально сходящийся алгоритм типа деления пополам.

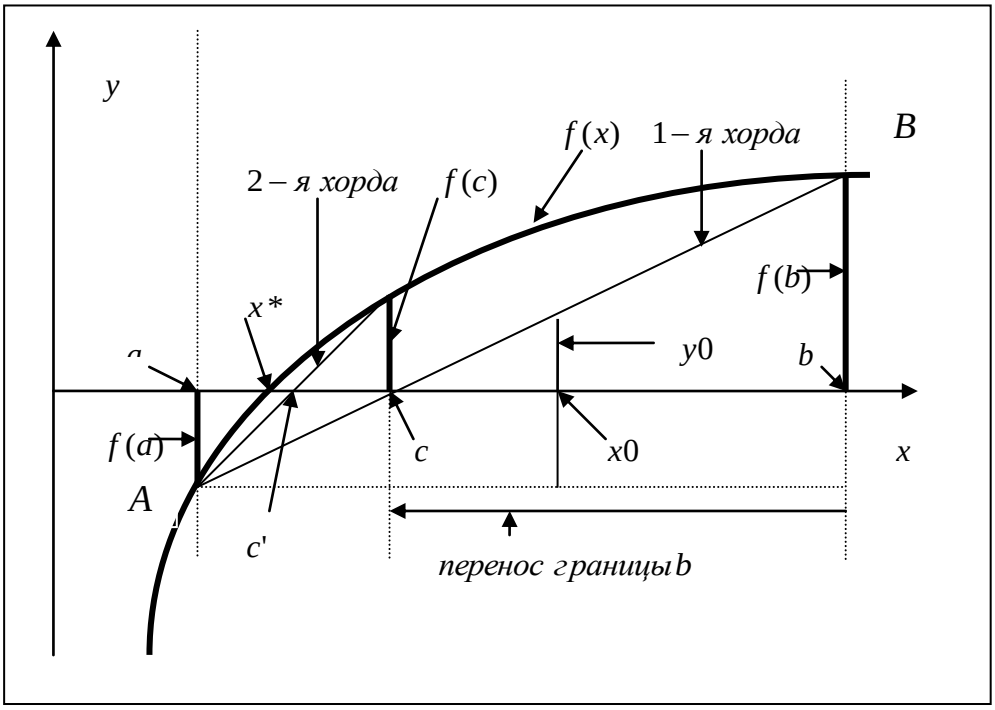


Рис. 1.5. Иллюстрация к методу хорд

4. Метод секущих

Данный метод заменяет производную первой разностью, найденной по двум последним итерациям. Итерационная формула метода имеет вид

$$x_{k+1} = x_k - f(x_k)/s_k,\ \ s_k = (f(x_k) - f(x_{k-1}))/(x_k - x_{k-1}),$$

$$k = 1, 2, \ldots,\ \ f(x_0) * f(x_1) < 0.$$

В этом алгоритме начинают с двумя исходными числами x_0 и x_1. На каждом шаге x_{k+1} получают как единственный нуль секущей прямой к функции $f(x)$, проходящей через точки с абсциссами x_{k-1} и x_k (рис. 1.9). Алгоритм метода секущих приведен на рис. 1.10.

Метод секущих имеет хорошую сходимость. Недостаток – в назначении x_0 и x_1, достаточно близких к корню для того, чтобы могла начаться сходимость.

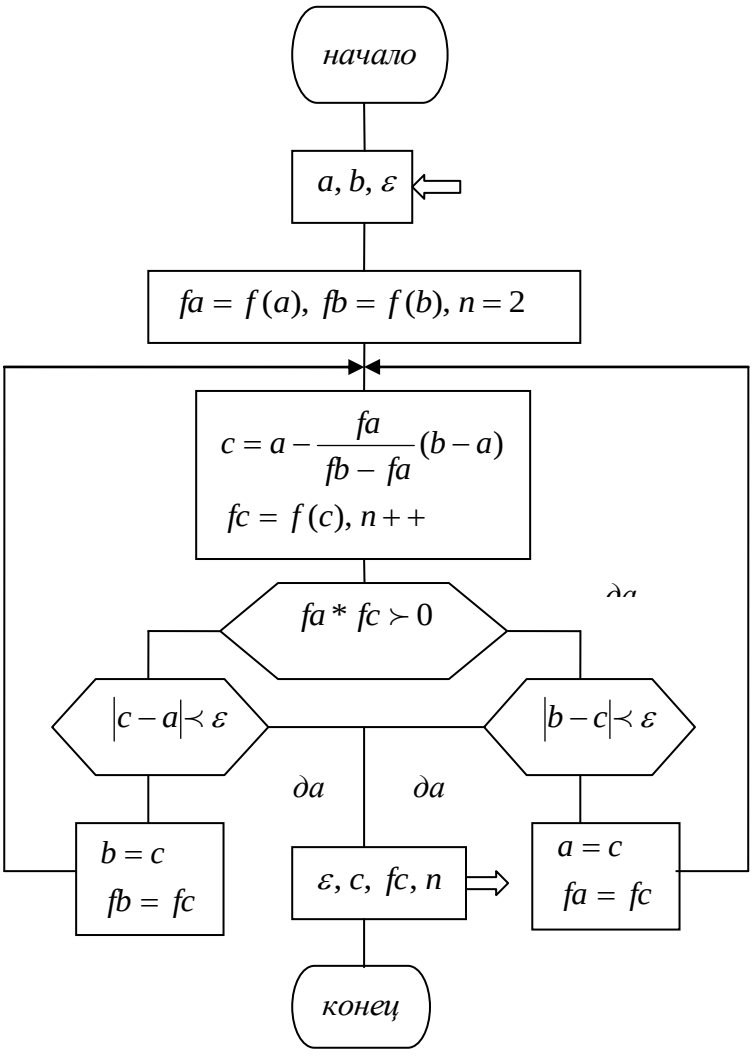


Рис. 1.6. алгоритм метода хорд

5. Метод итераций

Уравнение $f(x)=0$ заменяют равносильным $x=\varphi(x)$. Выбирают каким-либо способом приближенное значение корня x_0 и по нему находят $x_1=\varphi(x_0)$. Повторяя процесс, получают последовательность чисел:

$$x_n=\varphi(x_{n-1}),\ n=1,2,\ldots$$

Если эта последовательность – сходящаяся, то предел $x^*=\lim\limits_{n\to\infty} x_n$ является корнем равносильного уравнения и может быть вычислен по итерационной формуле $x_n=\varphi(x_{n-1}),\quad n=1,2,\ldots$с любой степенью точности.

Процесс итераций следует продолжать до тех пор, пока для двух последовательных приближений не будет выполнено неравенство $|x_n-x_{n-1}|\le\varepsilon(1-q)/q,$ где ε – заданная абсолютная точность вычисления корня и $q\ge|\varphi'(x)|$.

Поэтому в методе итераций при переходе от уравнения $f(x)=0$ к уравнению $x=\varphi(x)$ следует выбирать такое представление $x=\varphi(x)$, при котором $|\varphi'(x)|\le q<1,$ что является условием сходимости метода. Чем меньше q, тем быстрее последовательные приближения сходятся к корню x^*. Иллюстрации к методу итераций даны на рис. 1.11, алгоритм – на рис. 1.12

В заключение следует отметить, что не существует метода, который имел бы явное преимущество перед остальными для произвольного класса функций.

5. Комбинированные методы решений нелинейных уравнений

Методы комбинируют для повышения эффективности: комбинированный метод должен обеспечить при той же величине ошибки

меньшие затраты машинного времени по сравнению с любым из комбинируемых методов. Примеры алгоритмов комбинированных методов представлены на рис. 1.13 и 1.14.

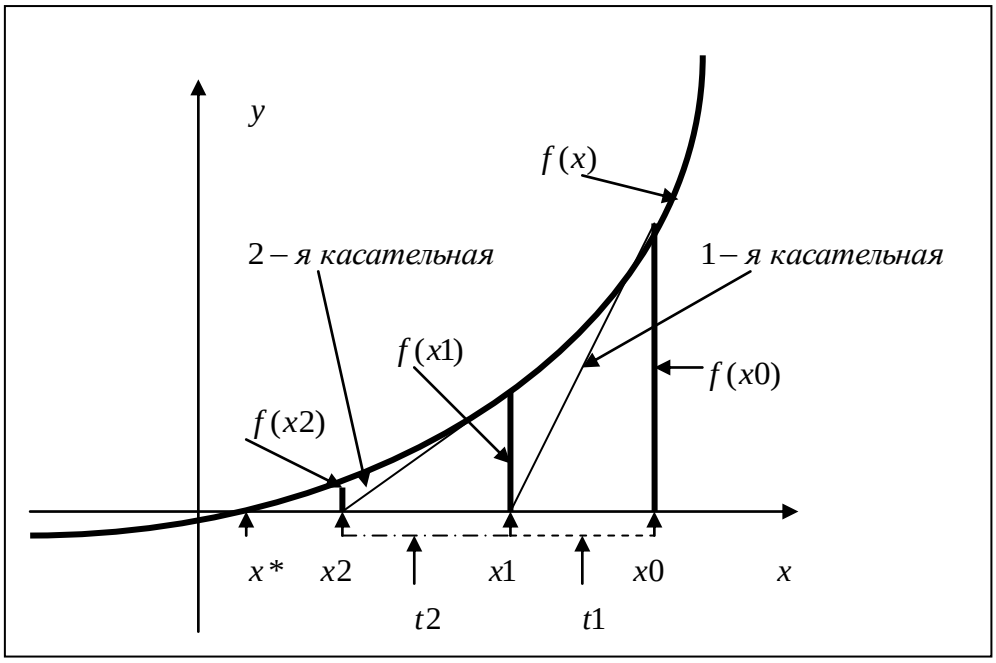


Рис. 1.7. Иллюстрация к методу касательных

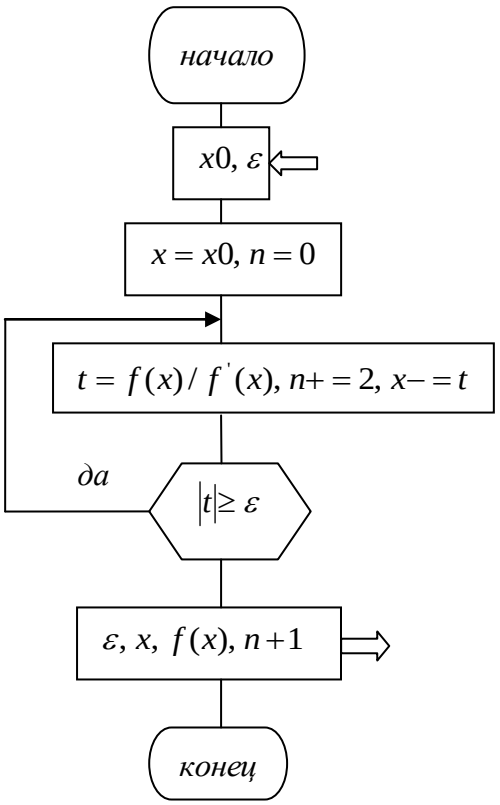


Рис. 1.8. Алгоритм метода касательных

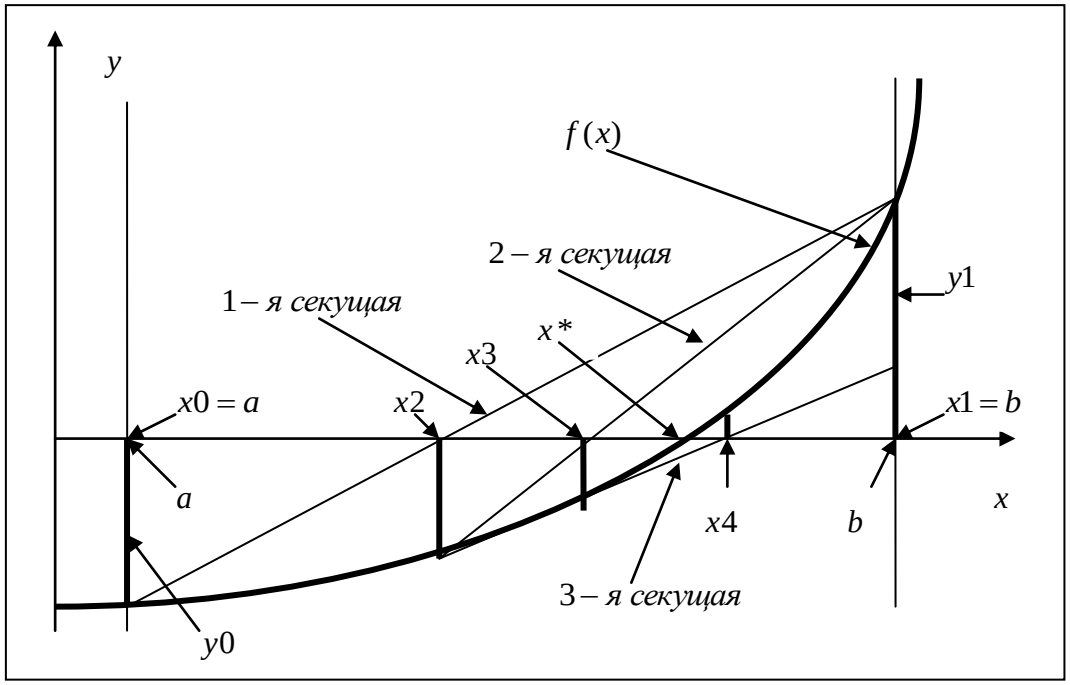


Рис. 1.9. Иллюстрация к методу секущих

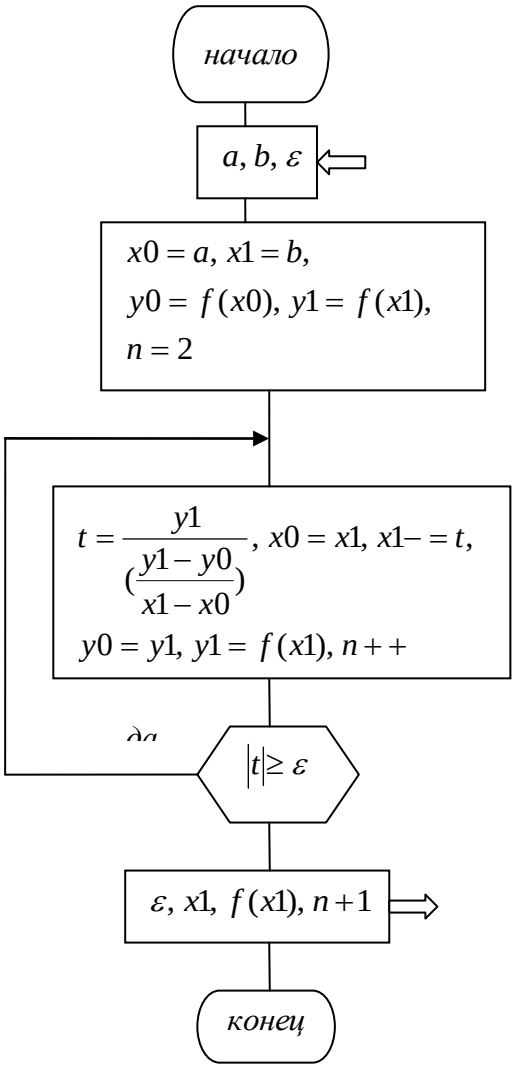


Рис. 1.10. Алгоритм метода секущих

Решение нелинейных уравнений в системе Mathcad

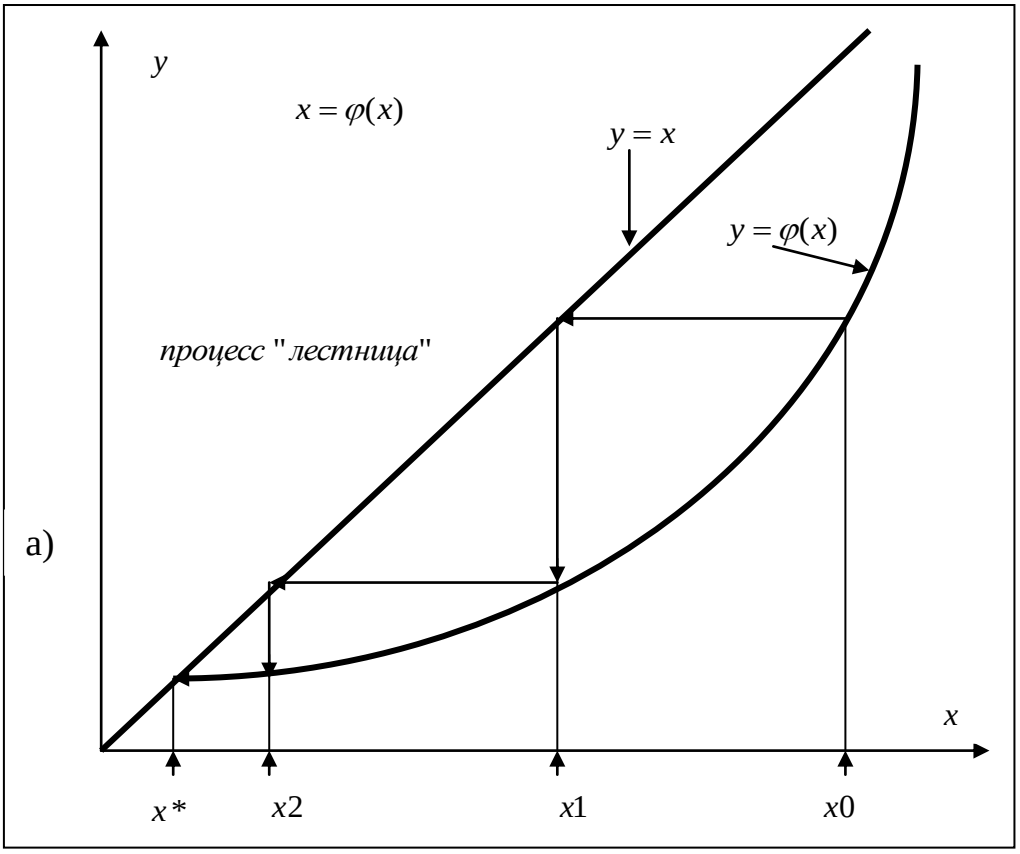
Для уравнений вида $f(x)=0$ корень находится с помощью функции $root(выражение, имя\_переменной, a, b)$, где выражение - $f(x)$; a, b - нижняя и верхняя границы диапазона значений аргумента.

При решении уравнений полезно построение графика функции $f(x)$. Для этого достаточно выполнить следующие действия.

1. На панели математических знаков щелкнуть на кнопке с изображением графика – на экране появится палитра графиков.

2. В палитре графиков щелкнуть на кнопке с изображением двумерного графика – на экране появится шаблон графика.

3. В место ввода шаблона по оси ординат ввести функцию, набрав её выражение, например, $3 \cdot \cos(2 \cdot x) + \exp(-0.5 \cdot x) - 2$.



а)

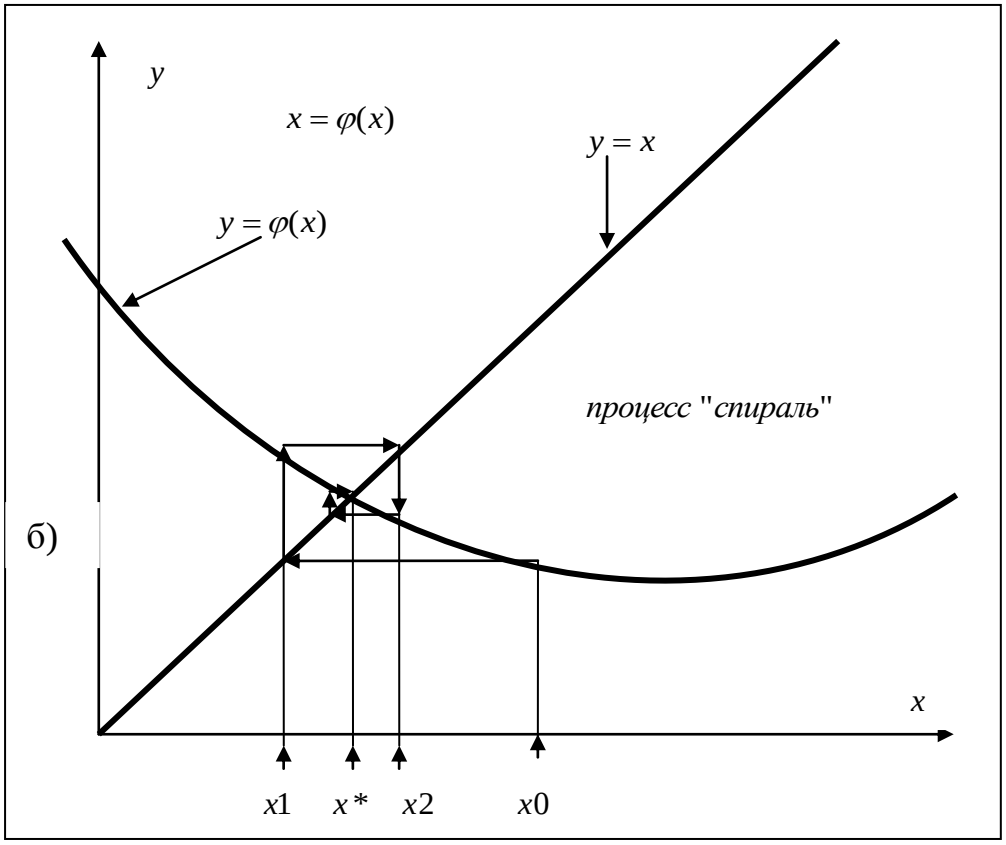


Рис. 1.11 а, б. Иллюстрации к методу итераций

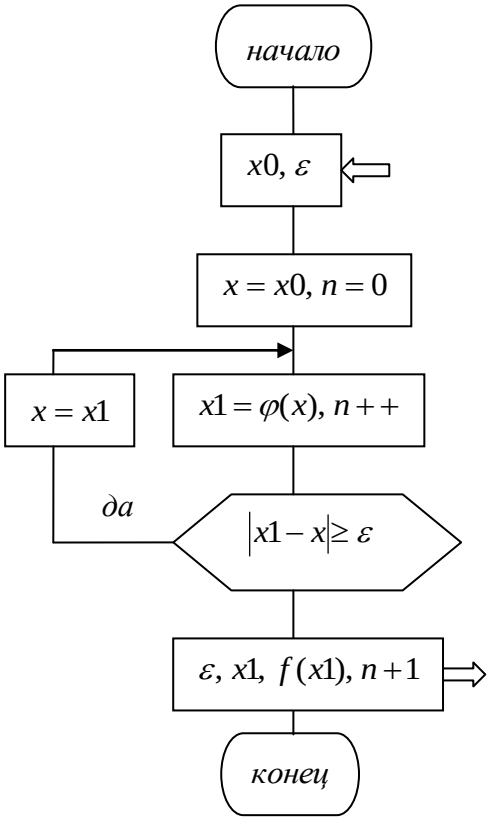


Рис. 1.12. Алгоритм метода итераций

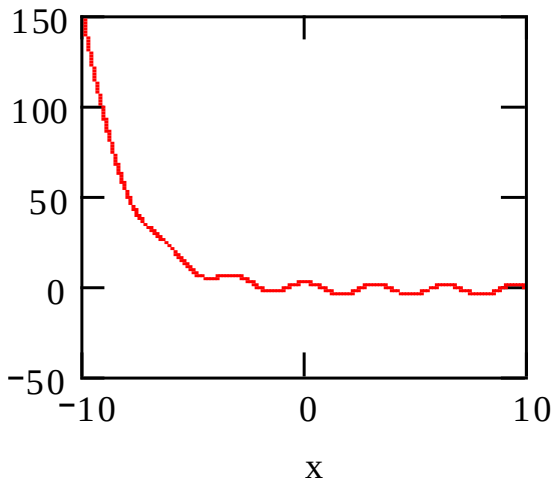
4. Ввести в место ввода шаблона по оси абсцисс имя аргумента – x.

5. Щелкнуть вне пределов графика левой кнопкой мыши – график построен.

Примечание. В *Mathcad* операция присваивания := вводится как : , а операция умножения изображается точкой после ввода *.

Пример построения графика функции и решения нелинейного уравнения

$3\cdot\cos(2\cdot x)+\exp(-0.5\cdot x)-2$



$x1:=\text{root}(3\cdot\cos(2\cdot x)+\exp(-0.5x)-2, x, -10, 10)$ $x1=0.571$

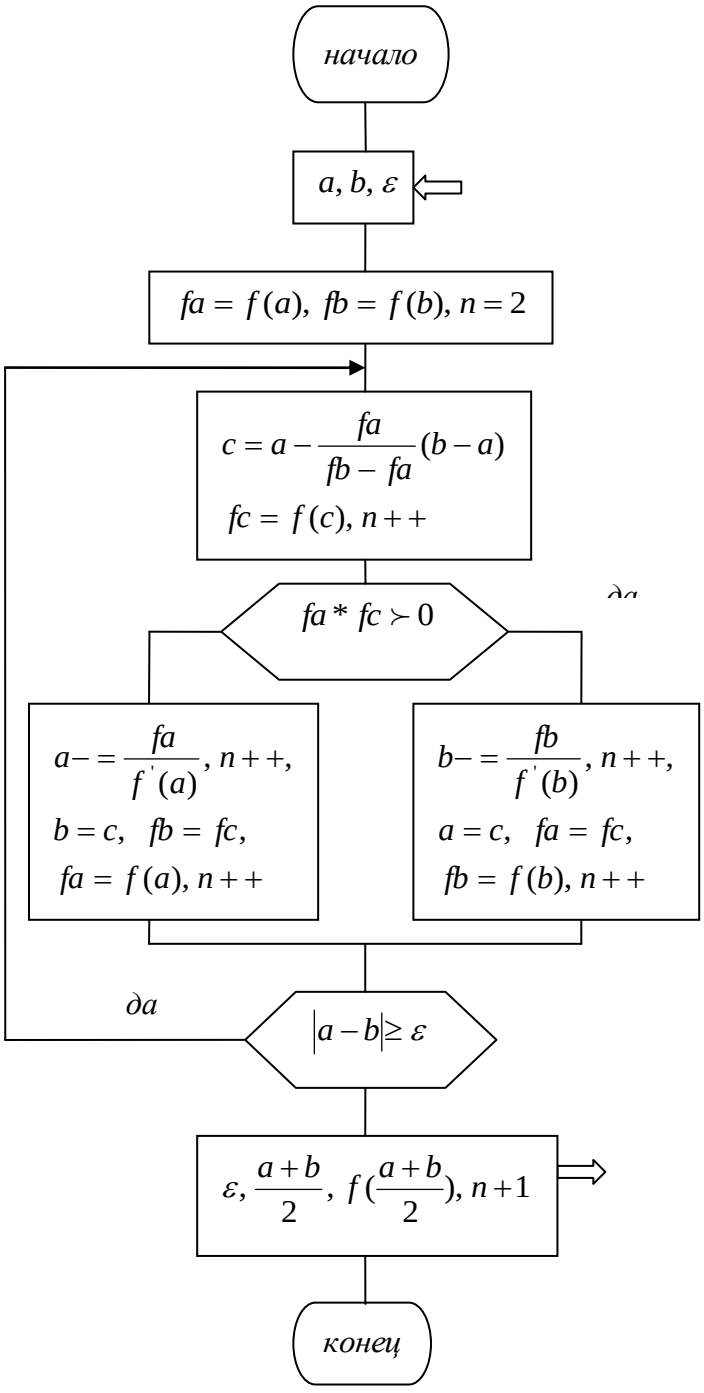


Рис. 1.13. Алгоритм комбинированного метода хорд и касательных

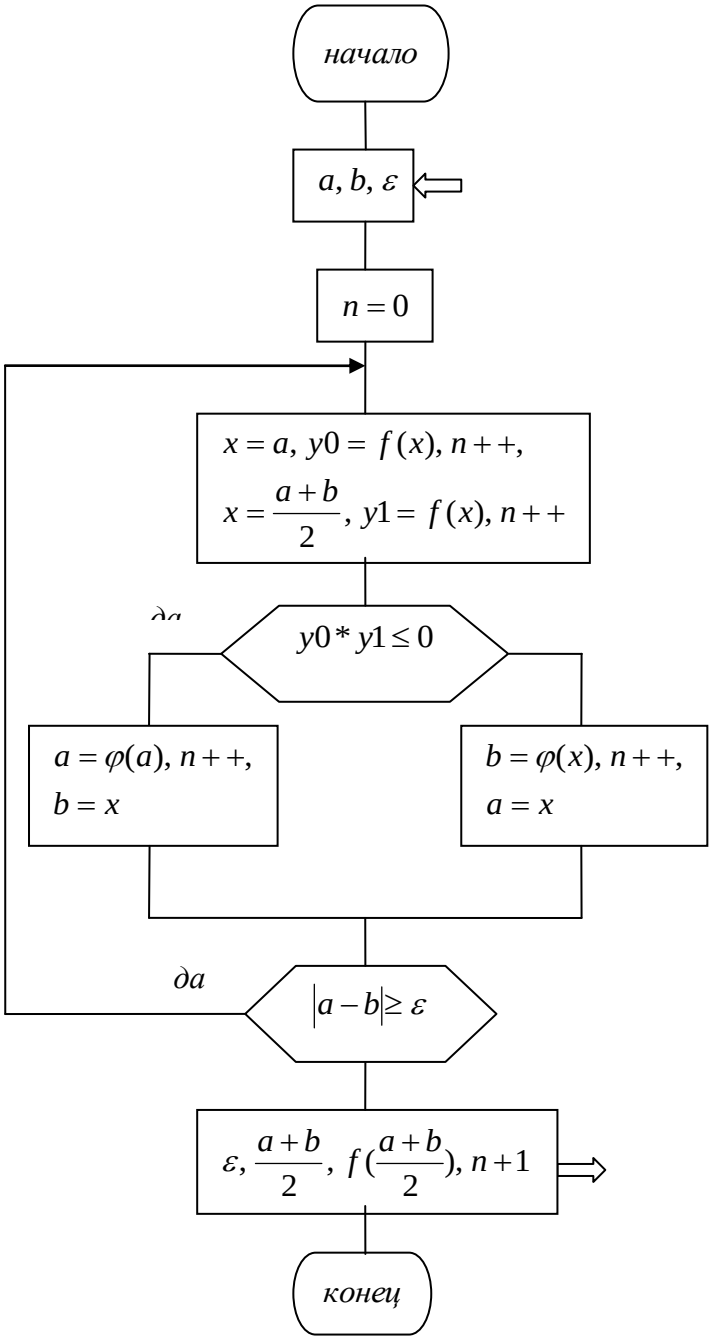


Рис. 1.14. Алгоритм комбинированного метода деления пополам и итераций

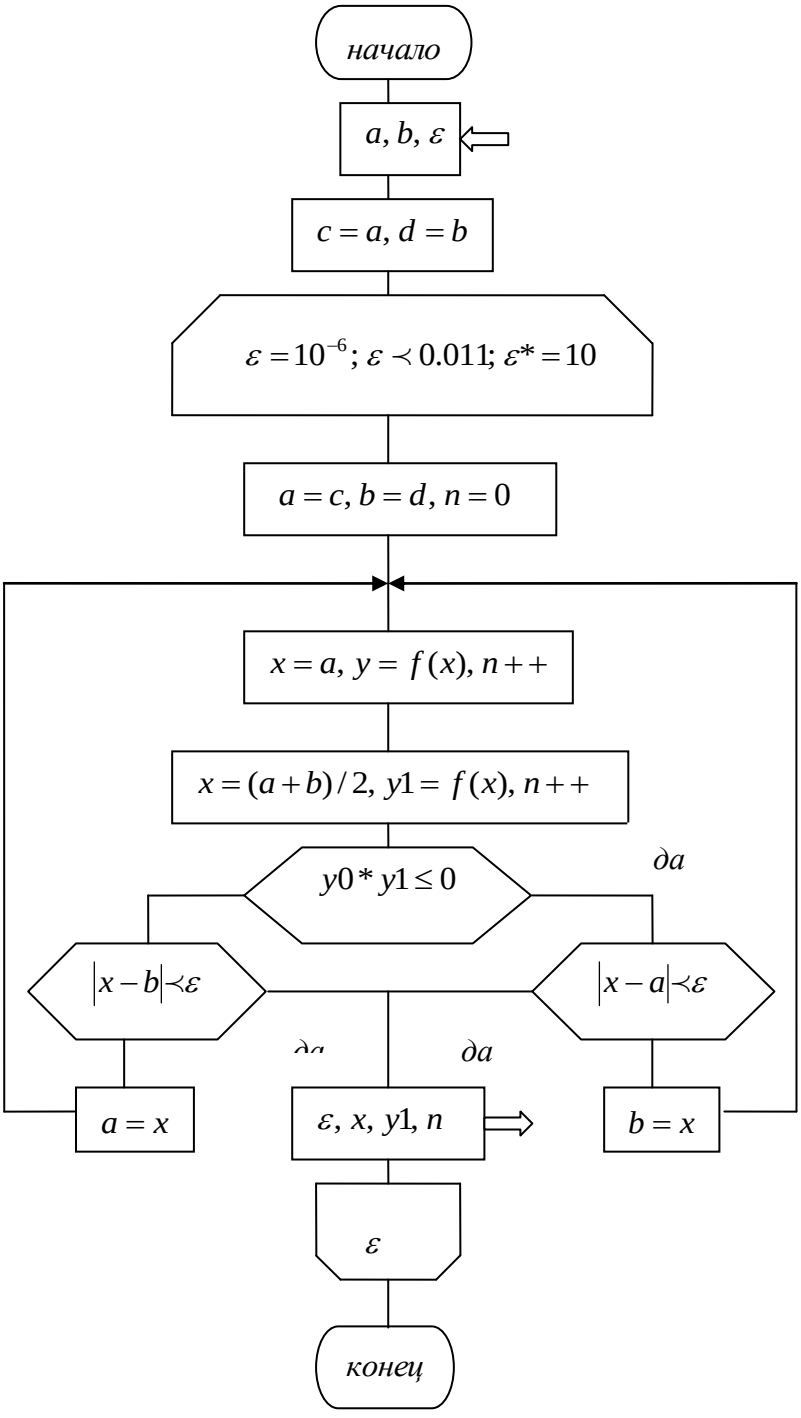


Рис. 1.15. Алгоритм расчета зависимости затрат машинного времени от задаваемой ошибки для метода деления пополам

ЛАБОРАТОРНАЯ РАБОТА 1

ИССЛЕДОВАНИЕ МЕТОДОВ РЕШЕНИЙ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

ЗАДАНИЕ

1. В *Mathcad'*е по заданному из таблицы варианту (см. *примечание*) уравнения $f(x)=0$ построить график $f(x)$ в диапазоне значений аргумента $-10 \le x \le 10$ и найти значение корня $x1$ (корней, если их несколько). Затем построить график $f(x)$ в диапазоне значений аргумента $x1-0.5 \le x \le x1+0.5$ и нанести на график линии сетки так, чтобы одна из горизонтальных линий проходила через нуль по оси ординат.

Примечание: 1 – метод деления пополам, 2 – метод хорд, 3 – метод касательных, 4 – метод секущих, 5 – метод итераций.

2. Составить алгоритм и написать код для отделения корня (корней) уравнения в диапазоне значений аргумента $-10 \le x \le 10$ с шагом $\ge 0{,}1$.

3. Составить алгоритм и написать код для уточнения значения корня (или одного из корней, если их несколько) заданным методом (методами). Получить таблицу и графики зависимостей временных затрат на уточнение корня от задаваемой погрешности ε (диапазон изменения $0{,}00001 \le \varepsilon \le 0{,}01$). Пример алгоритма представлен на рис. 1.15.

Таблица

Вариант	Уравнение $f(x)=0$	Используемый метод
1	$x^3+1=0$	1,3 (комбинация)
2	$x^3-6x+2=0$	2,4 (комбинация)
3	$x^3+x-500=0$	1,4 (комбинация)
4	$2x^3+3x^2+4x+5=0$	2,5 (комбинация)

5	$x^4 - 4x - 1 = 0$	1,5 (комбинация)
6	$2x^4 - 3x^2 + 75x - 10000 = 0$	1,3,5
7	$x^5 - x - 0.2 = 0$	2,4
8	$x^5 + x^4 + x - 1 = 0$	1,2,3
9	$x^5 - 10x + 128 = 0$	2,3 (комбинация)
10	$x + e^x = 0$	4,5
11	$x * \ln(x) - 1 = 0$	1,3,4
12	$tg(x) - x = 0$	2,3
13	$x - \sin(x) - 0.25 = 0$	1,5
14	$2x - e^{-0.1x} = 0$	3,5
15	$\ln(8x) - 10x + 2 = 0$	2,4
16	$10 * \sin(7x) - 8x + 1 = 0$	1,2
17	$x - e^{-0.1x} = 0$	1,3 (комбинация)
18	$x + \ln(x) = 0$	1,5 (комбинация)
19	$x - 5 * \ln(x) = 0$	2,3,5
20	$x + \ln(x) - 0.125 * e^x = 0$	1,4,5
21	$\sin(x) + \cos(2x) = 0$	2,3,4
22	$x^7 - 2x^6 + 7x - 8 = 0$	1,3
23	$x^9 - 2x^8 + 9x - 10 = 0$	2,4
24	$tg(x) - 2x + 0.5 = 0$	1,4
25	$0.5 * e^{-0.8x} - x = 0$	2,5
26	$2x^3 - 6x^2 - 3x + 15 = 0$	1,3
27	$\ln(2x) - e^{-x} - 1 = 0$	2,4
28	$2 * \sin(3x) + e^x - 5 = 0$	1,2,3
29	$5 * \cos(2x) + e^{-x} - 3 = 0$	2,4,5
30	$tg(2x) - 3x + 0.2 = 0$	1,3 (комбинация)

Рассмотрим пример проектирования приложения для исследования методов деления пополам и секущих при решении нелинейного уравнения $3*\cos(2*x)+\exp(-0.5*x)-2=0$. Решение уравнения выполняют в два этапа.

Этап 1. Отделить корень (корни) уравнения в диапазоне значений аргумента $-5 \le x \le 5$ с шагом 0,1.

Этап 2. Уточнить значение корня (или одного из корней, если их несколько) методами деления пополам и секущих. Получить таблицы и графики зависимостей временных затрат на уточнение корня от задаваемой погрешности ε (диапазон изменения $0{,}00001 \le \varepsilon \le 0{,}01$).

1. Создайте для проектов приложений по численным методам папку (каталог) с именем ЧИСЛ_МЕТ и запустите C++Builder 6.

2. Создайте новый проект командой *Файл/Новый/Приложение.*

3. Сохраните файлы модуля и проекта командой *Файл*/*Сохранить все* под именами LR1 и PR_LR1. Для этого удобно использовать соответствующую быструю кнопку (*Сохранить все*). В последующих сеансах работы сохраненный проект можно открыть командой *Файл/Открыть проект* (или *Повторно открыть*). Теперь перейдем к проектированию приложения – переносам на форму необходимых компонентов и заданию их свойствам значений, а в обработчиках событий – размещению кодов соответствующих алгоритмов. (Рекомендуется нажимать кнопку *Сохранить все* по окончании работы с каждым компонентом.) В результате проектирования получим форму, представленную на рис. 1.16. Выделите форму, щелкнув на ней левой кнопкой мыши, и в свойство **Caption** (надпись) впишите *МЕТОДЫ РЕШЕНИЙ НЕЛИНЕЙНЫХ УРАВНЕНИЙ***.**

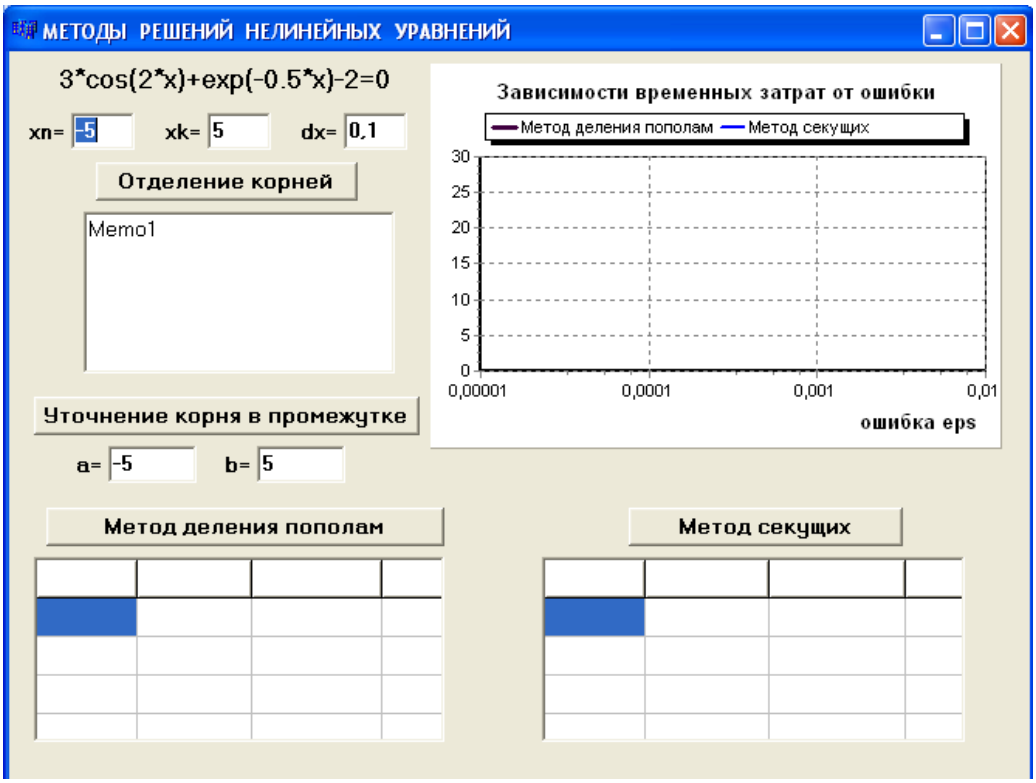


Рис. 1.16. Форма по окончании проектирования

4. В левый верхний угол формы поместите метку **Label1** (со страницы *Стандарт* библиотеки компонентов). Выделив метку щелчком на ней, установите свойство **Font** – жирный, размер 12. В свойство **Caption** метки впишите $3*\cos(2*x)+\exp(-0.5*x)-2=0$.

5. Ниже метки, по горизонтали, расположите на форме три метки: **LabeledEdit1, LabeledEdit2** и **LabeledEdit3** (страница *Дополнительно*). Для всех трех меток свойству **LabelPosition** присвойте значение **lpLeft** (из выпадающего списка), свойству **Font** – жирный, размер 10, свойству **Text** – значения –5; 5; 0,1 соответственно. Раскрывая свойство **EditLabel**, для всех трех меток установите подсвойство **Font** – жирный, размер 10, а в подсвойстве **Caption** впишите соответственно **xn=**, **xk=**, **dx=**.

6. Ниже меток поместите кнопку **Button1** (страница *Стандарт*), для которой установите свойство **Font** – жирный, размер 10, а в свойство **Caption** впишите **Отделение корней**.

7. Ниже кнопки поместите окно редактирования многострочного текста - компонент **Memo1** (страница *Стандарт*), для которого установите шрифт **Font** – обычный, размер 10.

8. Двойным щелчком на кнопке **Button1** перейдите в обработчик события – щелчка на кнопке. Перед обработчиком впишите определение функции, а в обработчик щелчка на кнопке **Button1** поместите код алгоритма отделения корней (*курсив*):

```
//---------------------------------------------------
float f(float x)
{return 3*cos(2*x)+exp(-0.5*x)-2;}
//---------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  float xn,xk,dx,x,y,y1,a,b;
  int m;
  xn=StrToFloat(LabeledEdit1->Text);
  xk=StrToFloat(LabeledEdit2->Text);
  dx=StrToFloat(LabeledEdit3->Text);
  x=xn; y=f(x); m=0;
  Memo1->SetFocus();
  Memo1->Clear();
  AnsiString s;
P:x+=dx;
  s="";
  if(x>xk){s=s+"Всего корней: "+IntToStr(m);
        Memo1->Lines->Add(s);
        LabeledEdit4->SetFocus();
        return;}
  else
```

```
    { y1=f(x);
      if(y*y1<=0) {m++; b=x; a=b-dx;
        s=s+IntToStr(m)+"-й корень:";
        if(a>=0)s=s+"  a= ";
          else s=s+"  a=";
        s=s+FloatToStrF(a,ffFixed,4,2);
         if(b>=0)s=s+"   b= ";
          else s=s+"   b=";
        s=s+FloatToStrF(b,ffFixed,4,2);
         Memo1->Lines->Add(s);}}
 y=y1; goto P;
}
```

Кроме того, в этом же файле, файле *LR1.cpp*, после директивы *#include "LR1.h"* поместите директиву *#include<math.h>*.

9. Быстрой кнопкой *“Переключатель Форма/Модуль F12”* или клавишей *F12* вызовите на экран форму. Ниже окна редактирования **Memo1** поместите кнопку **Button2**, в свойство **Caption** которой впишите **Уточнение корня в промежутке** жирным шрифтом размера 10.

10. В обработчике щелчка на кнопке **Button2** напишите (*курсив*):

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 StringGrid1->Cells[0][0]="   eps";
 StringGrid1->Cells[1][0]=" корень x";
 StringGrid1->Cells[2][0]="   f(x)";
 StringGrid1->Cells[3][0]="   n";
 StringGrid2->Cells[0][0]="   eps";
 StringGrid2->Cells[1][0]=" корень x";
 StringGrid2->Cells[2][0]="   f(x)";
 StringGrid2->Cells[3][0]="   n";
```

}

11. Под кнопкой **Button2** по горизонтали разместите две метки **LabeledEdit** (**4** и **5**). Свойству **LabelPosition** меток присвойте значение **lpLeft** (из выпадающего списка), свойству **Font** – жирный, размер 10, свойству **Text** – значения –5 и 5 соответственно. Раскрывая свойство **EditLabel**, для обеих меток установите подсвойство **Font** – жирный, размер 10, а в подсвойстве **Caption** впишите соответственно **a=**, **b=**.

12. Внизу формы, по всей ее ширине, разместите две таблицы - два компонента **StringGrid1** и **StringGrid2** (страница *Дополнительно*), а над ними – соответственно – кнопки **Button3** и **Button4**. В свойство **Caption** кнопок внесите соответственно (шрифт **Font** – жирный, размер 10) **Метод деления пополам** и **Метод секущих**, а в обработчиках щелчка на кнопках напишите коды соответствующих алгоритмов (*курсив*):

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{ double a,b,c,d,eps,x,y,y1;
  int n,i;
    c=StrToFloat(LabeledEdit4->Text);
    d=StrToFloat(LabeledEdit5->Text);
    Series1->Clear();
 for(eps=0.01,i=1;eps>0.000001;eps*=0.1,i++)
 { a=c; b=d;
   n=0;
A:  x=a;
   y=f(x); n++;
   x=(a+b)/2; y1=f(x); n++;
   if(y*y1<=0) {if(fabs(x-a)>=eps){b=x; goto A;}}
       else  {if(fabs(x-b)>=eps){a=x; goto A;}}
     StringGrid1->Cells[0][i]=FloatToStrF(eps,ffFixed,5,5);
     StringGrid1->Cells[1][i]=FloatToStrF(x,ffFixed,9,8);
```

```
      StringGrid1->Cells[2][i]=FloatToStrF(y1,ffExponent,5,2);
      StringGrid1->Cells[3][i]=IntToStr(n);
      Series1->AddXY(eps,n,"",clBlack);
   }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{  double a,b,eps,x0,x1,y0,y1,t;
   int n,i;
     a=StrToFloat(LabeledEdit4->Text);
     b=StrToFloat(LabeledEdit5->Text);
     Series2->Clear();
 for(eps=0.01,i=1;eps>0.000001;eps*=0.1,i++)
  { x0=a;  x1=b;
    y0=f(x0); y1=f(x1); n=2;
    do{
     t=y1/((y1-y0)/(x1-x0));
     x0=x1; x1-=t;
     y0=y1; y1=f(x1); n++;
       }while(fabs(t)>=eps);
      StringGrid2->Cells[0][i]=FloatToStrF(eps,ffFixed,5,5);
     StringGrid2->Cells[1][i]=FloatToStrF(x1,ffFixed,9,8);
     StringGrid2->Cells[2][i]=FloatToStrF(y1,ffExponent,5,2);
     StringGrid2->Cells[3][i]=IntToStr(n);
     Series2->AddXY(eps,n,"",clBlue);
  }
}
```

13. Теперь файл *LR1.cpp* будет выглядеть следующим образом:

```
//---------------------------------------------------------------------------
```

```
#include <vcl.h>
#pragma hdrstop
#include "LR1.h"
#include<math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
float f(float x)
{return 3*cos(2*x)+exp(-0.5*x)-2;}
//-------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{float xn,xk,dx,x,y,y1,a,b;
  int m;
  xn=StrToFloat(LabeledEdit1->Text);
  xk=StrToFloat(LabeledEdit2->Text);
  dx=StrToFloat(LabeledEdit3->Text);
  x=xn; y=f(x); m=0;
  Memo1->SetFocus();
  Memo1->Clear();
  AnsiString s;
P:x+=dx;
  s="";
```

```
  if(x>xk){s=s+"Всего корней: "+IntToStr(m);
           Memo1->Lines->Add(s);
           LabeledEdit4->SetFocus();
           return;}
  else
      { y1=f(x);
        if(y*y1<=0) {m++; b=x; a=b-dx;
           s=s+IntToStr(m)+"-й корень:";
           if(a>=0)s=s+"  a= ";
              else s=s+"  a=";
           s=s+FloatToStrF(a,ffFixed,4,2);
            if(b>=0)s=s+"   b= ";
              else s=s+"   b=";
           s=s+FloatToStrF(b,ffFixed,4,2);
             Memo1->Lines->Add(s);}}
  y=y1; goto P;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{StringGrid1->Cells[0][0]="   eps";
  StringGrid1->Cells[1][0]=" корень x";
  StringGrid1->Cells[2][0]="   f(x)";
  StringGrid1->Cells[3][0]="   n";
  StringGrid2->Cells[0][0]="   eps";
  StringGrid2->Cells[1][0]=" корень x";
  StringGrid2->Cells[2][0]="   f(x)";
  StringGrid2->Cells[3][0]="   n";
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{
 double a,b,c,d,eps,x,y,y1;
 int n,i;
      c=StrToFloat(LabeledEdit4->Text);
      d=StrToFloat(LabeledEdit5->Text);
      Series1->Clear();
  for(eps=0.01,i=1;eps>0.000001;eps*=0.1,i++)
   { a=c; b=d;
     n=0;
A:   x=a;
     y=f(x); n++;
     x=(a+b)/2; y1=f(x); n++;
     if(y*y1<=0) {if(fabs(x-a)>=eps){b=x; goto A;}}
          else   {if(fabs(x-b)>=eps){a=x; goto A;}}
        StringGrid1->Cells[0][i]=FloatToStrF(eps,ffFixed,5,5);
        StringGrid1->Cells[1][i]=FloatToStrF(x,ffFixed,9,8);
        StringGrid1->Cells[2][i]=FloatToStrF(y1,ffExponent,5,2);
        StringGrid1->Cells[3][i]=IntToStr(n);
        Series1->AddXY(eps,n,"",clBlack);
   }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{ double a,b,eps,x0,x1,y0,y1,t;
   int n,i;
   a=StrToFloat(LabeledEdit4->Text);
   b=StrToFloat(LabeledEdit5->Text);
   Series2->Clear();
```

```
  for(eps=0.01,i=1;eps>0.000001;eps*=0.1,i++)
   { x0=a;  x1=b;
    y0=f(x0); y1=f(x1); n=2;
    do{
     t=y1/((y1-y0)/(x1-x0));
     x0=x1; x1-=t;
     y0=y1; y1=f(x1); n++;
       }while(fabs(t)>=eps);
     StringGrid2->Cells[0][i]=FloatToStrF(eps,ffFixed,5,5);
     StringGrid2->Cells[1][i]=FloatToStrF(x1,ffFixed,9,8);
     StringGrid2->Cells[2][i]=FloatToStrF(y1,ffExponent,5,2);
     StringGrid2->Cells[3][i]=IntToStr(n);
     Series2->AddXY(eps,n,"",clBlue);
   }
}
//---------------------------------------------------------------------------
```

14. Установим в компонентах **StringGrid1** и **StringGrid2** следующие значения свойств: шрифт **Font** – обычный, размер 10, **ColCount** – 4, **RowCount** – 5, **FixedCols** – 0, **FixedRows** – 1, **FixedColor – clWhite**, **ScrollBars** – **ssNone**.

15. В правую верхнюю часть формы поместите компонент **Chart1** (страница *Additional*). Установим в нем те значения свойств, которые могут быть заданы или известны к данному моменту времени. Щелкните правой кнопкой мыши на компоненте **Chart1** и в появившемся меню выберите *Edit Chart*…. На экране появится окно *Редактора Диаграмм* (*Editing Chart1*) с открытой страницей *Chart*, которая имеет несколько закладок. На закладке *Panel*, нажав кнопку *Panel Color*…, выберите белый цвет. На закладке *Series* щелкните на кнопке *Add*… – добавить серию. В появившемся окне выберите тип графика – *Line* и выключите индикатор

3D. После щелчка на *OK* снова появится окно *Editing Chart1*. Перейдите на закладку *Titles*. В окне редактирования, которое в данный момент соответствует *Title* – заголовку графика, сотрите *TChart* и напишите (шрифт *Font*… – черный, жирный, размер 10) **Зависимости временных затрат от ошибки**. В группе кнопок *Alignment* оставьте нажатой кнопку *Center*. Цвет фона *Back Color*.. установите белый. В выпадающем списке от окна редактирования *Title* перейдите в окно редактирования *Foot* и напишите тем же шрифтом **ошибка eps**. В группе кнопок *Alignment* нажмите кнопку *Right*. Цвет фона *Back Color*.. также установите белый. Перейдите на закладку *Axis* для задания координатных характеристик осей. Оставьте в группе *Axis* нажатой кнопку *Left* и включенным индикатор *Automatic*. Затем нажмите в группе *Axis* кнопку *Bottom*, выключите индикатор *Automatic* и включите индикатор *Logarithmic*. Нажмите нижнюю кнопку *Change*…и в окне *Value* редактора *Minimum Bottom Axis* впишите значение 0,00001. Аналогичные действия выполните с верхней кнопкой *Change*…, но впишите число 0,01. Затем перейдите на закладку *Labels* и в окне редактирования *Values Format* значение # ##0,### замените, добавив в него справа ##, на значение # ##0,#####. Перейдите на закладку *Series*, кнопкой *Add*…добавьте еще один график *Line* при выключенном индикаторе *3D* . Выделив *Series 1*, кнопкой *Title*… вызовите *Change Series Title*, где дайте заголовок первому графику – *Метод деления пополам*. Действуя подобным образом, т. е. выделив *Series 2*, дайте заголовок второму графику – *Метод секущих*. На вкладке *Legend* нажмите кнопку *Top*. Перейдите со страницы *Chart* на страницу *Series*. Здесь на закладке *Format* задают параметры линий графиков. Требуемый график (*Метод деления пополам, Метод секущих*) выбирается из выпадающего списка. В группе *Line* через кнопку *Border*…устанавливается толщина (*Width*) линии графика, а через кнопку *Color*… – цвет линии графика. Для обоих графиков следует выключить индикатор *Color Each*. Задайте для

первого графика тройную толщину линии черного цвета, для другого – двойную толщину линии черного цвета.

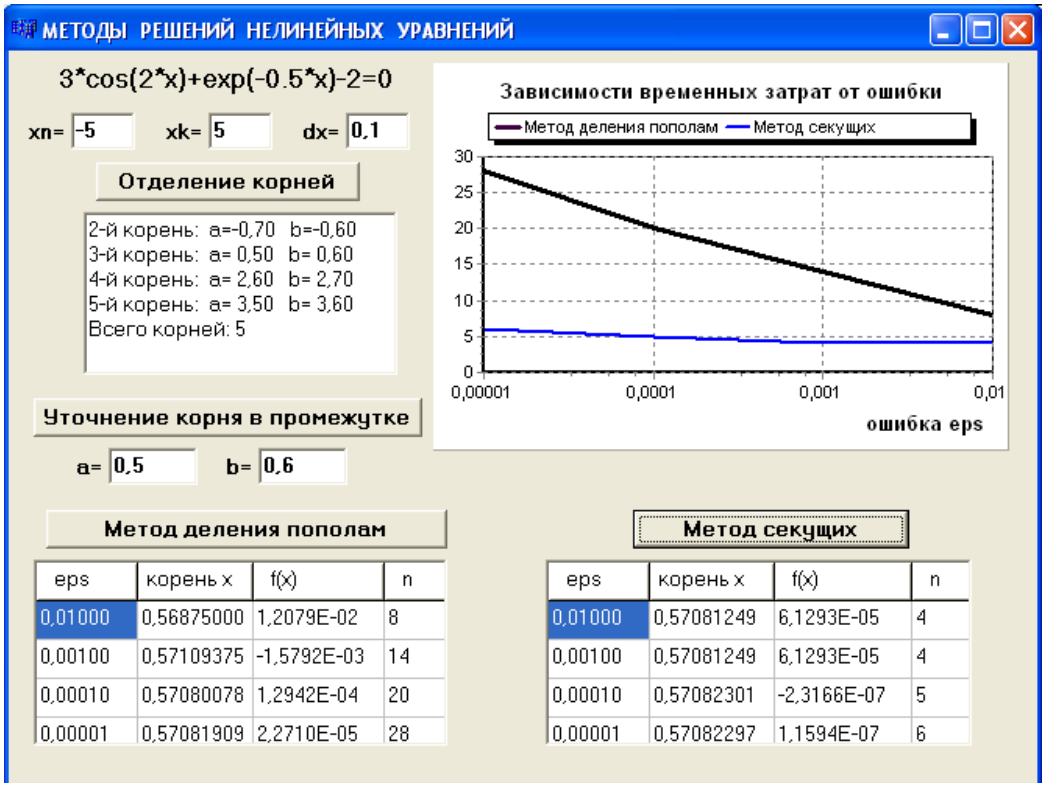


Рис. 1.17. Результаты выполнения задания

16. Выше на закладке *Axis* задавались координатные характеристики осей. При этом в группе *Axis* при нажатой кнопке *Left* был оставлен включенным индикатор *Automatic*, поскольку по оси ординат (*Left*) неизвестен диапазон откладываемых значений *n*. Чтобы установить его и этим завершить проектирование приложения, запустим приложение на выполнение, нажав быстрые кнопки *Сохранить все* и *Запуск*. (Забегая вперед, приведем форму по окончании выполнения задания – рис. 1.17).

17. После компиляции щелкните на кнопке **Отделение корней**. Результат отделения корней будет выведен в **Memo1**. Введите границы промежутка для одного из корней, например, **a**=0,5 **b**=0,6. Щелкните на кнопке **Уточнение корня в промежутке**, что вызовет появление

заголовков таблиц. Затем щелкните на кнопках **Метод деления пополам** и **Метод секущих**. Из таблиц видно, что максимальное значение *n* оказалось равным 28.

18. Щелкните на кнопке формы «*Закрыть*», чтобы продолжить проектирование приложения. Вернитесь к заданию координатных характеристик осей компонента **Chart1** на закладке *Axis*. В группе *Axis* при нажатой кнопке *Left* выключите индикатор *Automatic*. Нажав нижнюю кнопку *Change…*, выставите минимальное значение, равное 0, а затем при нажатой верхней кнопке *Change…*выставите максимальное значение, равное 30. На этом проектирование приложения завершается.

19. Сохранив изменения, запустите проект на выполнение. Результаты выполнения задания представлены на рис. 1.17.

20. Для завершения работы щелкните на кнопке формы «*Закрыть*» и выйдите из среды *Builder*.

Содержание отчета

1. Задание.
2. Формулы и иллюстрации методов.
3. Результаты выполнения задания в *Mathcad'*е .
4. Блок-схема алгоритма с таблицей идентификаторов.
5. Исходный код.
6. Результаты выполнения работы в виде таблицы и графиков.
7. Библиографический список.

Контрольные вопросы

1. Какие виды нелинейных уравнений можно решать численными методами?

2. Расскажите об отделении корней, приведя иллюстрацию для своего задания. Как выбирается величина Δx?

3. Сравните методы деления пополам и хорд.

4. Сравните методы касательных и секущих.

5. Как перейти от уравнения $f(x)=0$ к равносильному ему уравнению $x=\varphi(x)$? Объясните алгоритм метода итераций.

6. Расскажите об условиях применения методов уточнения корней.

7. Как зависит в численных методах значение функции в корне от величины задаваемой ошибки?

8. Приведите примеры комбинации методов. Поясните их целесообразность.

9. Приведите алгоритмы для получения зависимостей затрат машинного времени от ошибки для методов в задании.

10. Объясните полученные результаты.

11. Как построить класс для выполнения задания? В каких файлах и как можно разместить объявление класса и его реализацию?

ЧИСЛЕННЫЕ МЕТОДЫ ИНТЕРПОЛЯЦИИ И ЭКСТРАПОЛЯЦИИ

Краткие сведения

Интерполяция – это нахождение по ряду данных значений функции промежуточных её значений. Экстраполяция позволяет найти значения функции вне интервала интерполяции. Исходными данными для решения задач интерполяции и экстраполяции являются значения таблично заданной функции:

x	x_0	x_1	….	x_i	….	x_n
y	y_0	y_1	….	y_i	….	y_n

Узлы интерполяции могут быть неравноотстоящими $x_{i+1} = x_i + h_i$ и равноотстоящими $x_{i+1} = x_i + h$, где h_i, h – шаг интерполяции.

Решением задачи интерполяции (экстраполяции) является интерполирующая (экстраполирующая) функция $y = F(x)$, удовлетворяющая условиям

$$F(x_0) = y_0, \quad F(x_1) = y_1, \quad ...., \quad F(x_i) = y_i, \quad ..., \quad F(x_n) = y_n.$$

1. Алгебраическая интерполяция

При алгебраической интерполяции интерполирующая функция может быть написана непосредственно по таблице значений в виде интерполяционного полинома Лагранжа n-ой степени

$$y = Ln(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)(x - x_1)....(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)....(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}.$$

Узлы интерполяции при этом могут быть как равноотстоящими, так и неравноотстоящими. Представленный полином позволяет выполнять прямую интерполяцию. По таблице можно написать также полином Лагранжа для обратной интерполяции

$$x = Ln(y) = \sum_{i=0}^{n} x_i \frac{(y - y_0)(y - y_1)....(y - y_{i-1})(y - y_{i+1})...(y - y_n)}{(y_i - y_0)(y_i - y_1)....(y_i - y_{i-1})(y_i - y_{i+1})...(y_i - y_n)}.$$

Для разработки алгоритмов полиномы удобнее представить в виде

$$y = Ln(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j},$$

$$x = Ln(y) = \sum_{i=0}^{n} x_i \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{y - y_j}{y_i - y_j}.$$

Алгоритмы прямой и обратной интерполяций по полиномам Лагранжа представлены на рис. 2.1 и 2.2.

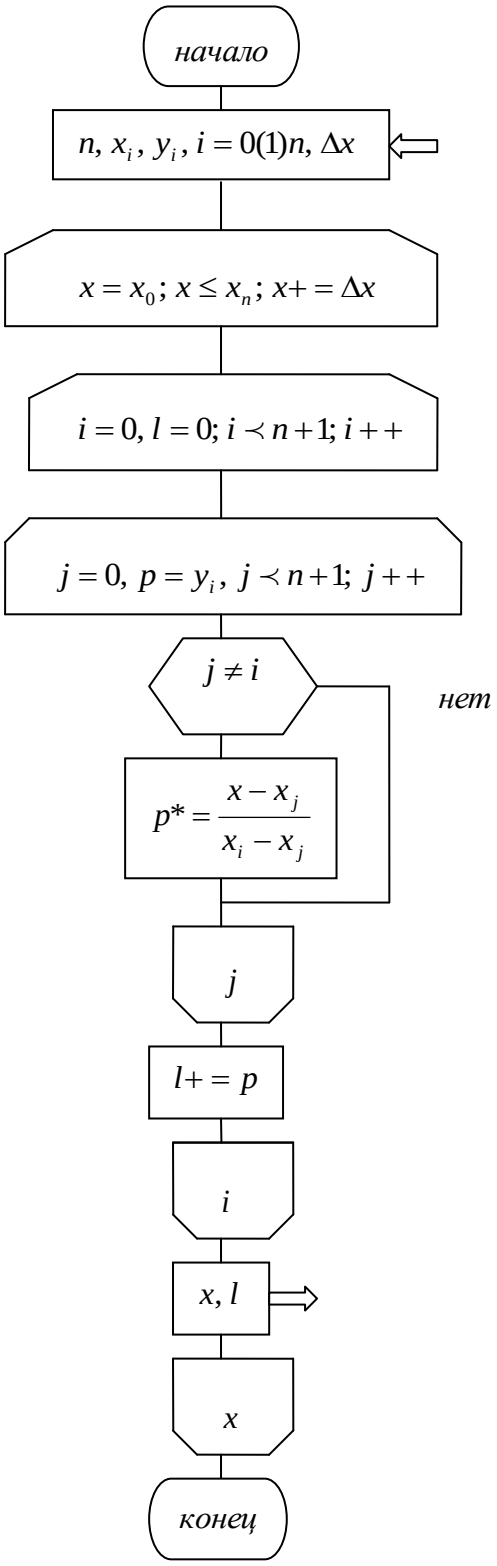


Рис. 2.1. Алгоритм прямой интерполяции по Лагранжу

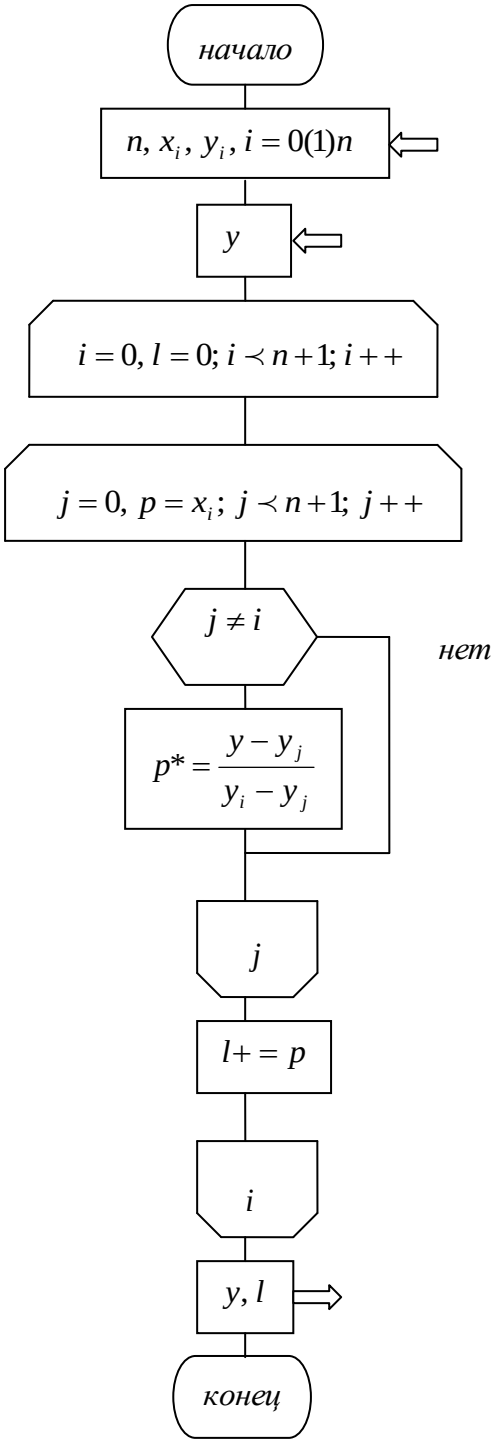


Рис. 2.2. Алгоритм обратной интерполяции по Лагранжу

2. Кусочно-линейная интерполяция

При кусочно-линейной интерполяции вычисление дополнительных точек выполняется по линейной зависимости.

Графически это означает соединение узловых точек отрезками прямых (рис. 2.3). При экстраполяции используются отрезки прямых, проведенных через две крайние точки.

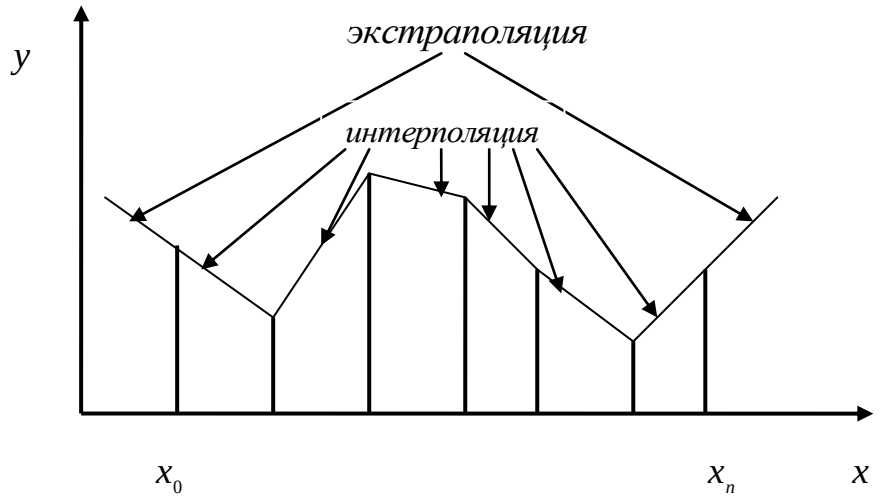


Рис. 2.3. Кусочно-линейная интерполяция и экстраполяция

При небольшом числе узлов (<10) кусочно-линейная интерполяция оказывается довольно грубой. При ней даже первая производная интерполирующей функции испытывает резкие скачки в узловых точках.

3. Сплайн-интерполяция

Гораздо лучшие результаты, по сравнению с кусочно-линейной, дает сплайн-интерполяция (*spline – гибкая линейка*). Здесь исходная функция заменяется отрезками кубических полиномов, проходящих через три смежные узловые точки. Коэффициенты полиномов $a_0x^3 + a_1x^2 + a_2x + a_3$, т. е. a_0, a_1, a_2, a_3, рассчитываются так, чтобы первая и вторая производные были непрерывными. Линия, которую описывает сплайн-функция, напоминает по форме гибкую линейку, закрепленную в узловых точках.

4. *Тригонометрическая интерполяция*

Пусть функция $f(x)$ задана на отрезке $[0,2\pi]$ таблицей значений $f(x_i)$ в равноотстоящих узлах $x_i = 2\pi i/(2n+1),\ i = 0,1,\ldots,2n.$

Тригонометрическим многочленом степени m называют многочлен

$$P_m(x) = a_0 + 2\sum_{k=1}^{m}(a_k \cos kx + b_k \sin kx).$$

Задача тригонометрической интерполяции состоит в построении тригонометрического интерполяционного многочлена наименьшей степени, удовлетворяющего условиям $P_m(x_i) = f(x_i),\quad i = 0,1,\ldots,2n.$

Решением этой задачи является тригонометрический многочлен

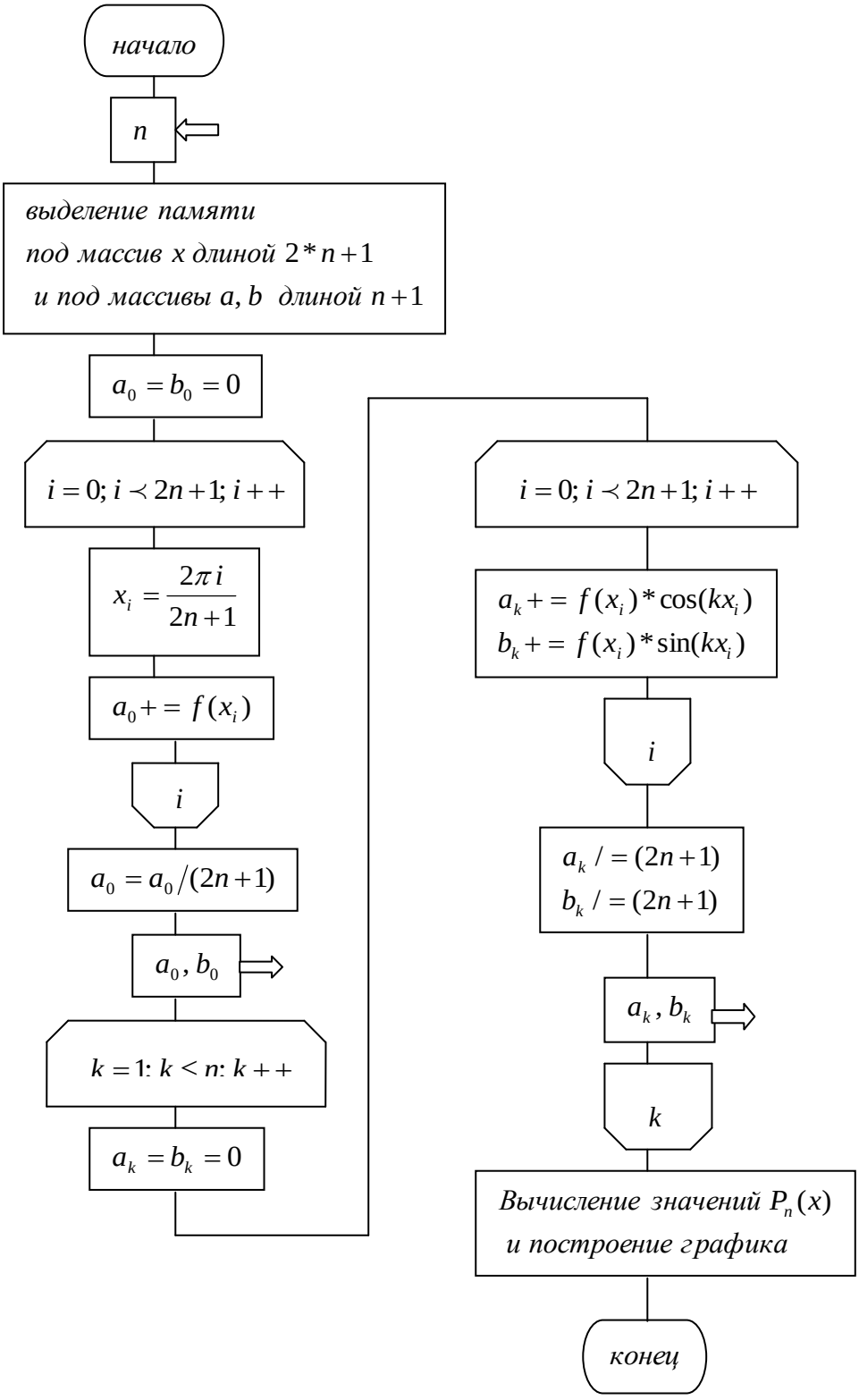


Рис. 2.4. Алгоритм тригонометрической интерполяции

$$P_n(x) = a_0 + 2\sum_{k=1}^{n}(a_k \cos kx + b_k \sin kx),$$

коэффициенты которого вычисляются по следующим формулам:

$$a_0 = \frac{1}{2n+1}\sum_{i=0}^{2n} f(x_i),$$

$$a_k = \frac{1}{2n+1}\sum_{i=0}^{2n} f(x_i)\cos\frac{2\pi ki}{2n+1},$$

$$b_k = \frac{1}{2n+1}\sum_{i=0}^{2n} f(x_i)\sin\frac{2\pi ki}{2n+1}.$$

Широкие возможности тригонометрической интерполяции следуют из того факта, что с возрастанием n многочлен $P_n(x)$ аппроксимирует $f(x)$ с возрастающей точностью. Это справедливо для достаточно широкого класса функций. Этим тригонометрическая интерполяция существенно отличается от алгебраической интерполяции на системе равноотстоящих узлов. При алгебраическом интерполировании разность между функцией $f(x)$ и интерполяционным многочленом может быть как угодно большой всюду, кроме узлов интерполяции. Тригонометрическое интерполирование свободно от этого недостатка.

Алгоритм тригонометрической интерполяции представлен на рис. 2.4.

Интерполирование в системе Mathcad

Система *Mathcad* позволяет интерполировать двумя практически важными типами функций: кусочно-линейной и сплайновой.

При кусочно-линейной интерполяции используется функция *linterp(VX,VY,x)*. Для заданных векторов *VX* и *VY* узловых точек и заданного аргумента *x* функция *linterp(VX,VY,x)* возвращает значение функции при ее линейной интерполяции. Для экстраполяции функция

linterp(VX,VY,x) не предназначена и за пределами области определения может вести себя непредсказуемо.

Для сплайн-интерполяции используется четыре встроенные функции. Три из них служат для получения векторов вторых производных сплайн-функций при различном виде интерполяции: *cspline(VX,VY)* – возвращает вектор *VS* вторых производных при приближении в краевых точках к кубическому полиному; *pspline(VX,VY)* – возвращает вектор *VS* вторых производных при приближении в краевых точках к параболической кривой; *lspline(VX,VY)* – возвращает вектор *VS* вторых производных при приближении в краевых точках к прямой. Четвертая функция *interp(VS,VX,VY,x)* возвращает значение *y(x)* для заданных векторов *VS,VX,VY* и заданного значения *x*.

Таким образом, сплайн-интерполяция проводится в два этапа. Сначала с помощью функций *cspline(VX,VY), pspline(VX,VY)* или *lspline(VX,VY)* отыскивается вектор вторых производных функции *y(x)*, заданной векторами *VX* и *VY* ее значений (абсцисс и ординат). Затем для каждой искомой точки вычисляется значение *y(x)* с помощью функции *interp(VS,VX,VY,x).*

Пример кусочно-линейной и сплайн-интерполяций

$$\text{data} := \begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 3 & 4 \\ 4 & 1 \\ 4.5 & 2 \\ 5 & 5 \end{pmatrix} \qquad \text{data} := \text{csort}(\text{data}, 0) \qquad X := \text{data}^{\langle 0 \rangle} \qquad Y := \text{data}^{\langle 1 \rangle}$$

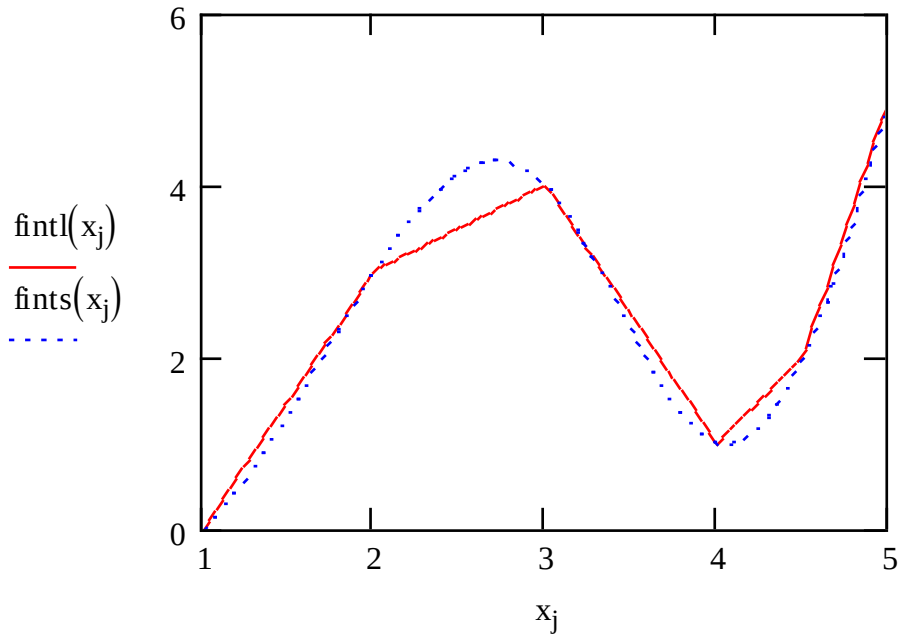
$$\text{fintl}(x) := \text{linterp}(X, Y, x) \qquad \text{fintl}(2) = 3 \qquad \text{fintl}(7.71) = 21.26$$

$$S := \mathrm{cspline}(X, Y) \qquad \mathrm{fints}(x) := \mathrm{interp}(S, X, Y, x)$$

$$\mathrm{fints}(2) = 3 \qquad \mathrm{fints}(7.71) = 18.893$$

$$\mathrm{scale} := 100 \qquad j := 0..\mathrm{scale}$$

$$x_j := \min(X) + j \cdot \frac{\max(X) - \min(X)}{\mathrm{scale}}$$



Примечания. 1. Функция *csort*(M,n) переставляет строки матрицы M таким образом, чтобы отсортированным оказался n-й столбец. 2. Функция *max(X)* возвращает максимальный, а *min(X)* - минимальный по значению элемент вектора X (или матрицы). 3. Операция .. (две точки) вводится как ; (точка с запятой) , а шаблон дроби – символом / (арифметическая операция деления).

ЛАБОРАТОРНАЯ РАБОТА 2

МЕТОДЫ ИНТЕРПОЛЯЦИИ И ЭКСТРАПОЛЯЦИИ

ЗАДАНИЕ

1. В *Mathcad'*е по заданному из таблицы варианту выполнить кусочно-линейную и сплайн-интерполяции для функции, заданной таблицей значений в точках $x_i = 2\pi i/(2n+1)$, где $i = 0, 1, \ldots, 2n$.

2. По заданному варианту составить алгоритм и написать код для определения параметров и построения графика интерполяционного тригонометрического многочлена. Нанести на график значения функции из таблицы.

Таблица

№	Значения функций
1	1.00; 1.803; 3.085; 4.776; 6.434; 7.347; 7.027; 5.652; 3.897; 2.381; 1.347; 0.722; 0.419; 0.256; 0.176; 0.142; 0.136; 0.155; 0.209; 0.324; 0.554
2	7.38; 6.76; 5.22; 3.47; 2.07; 1.16; 0.64; 0.36; 0.23; 0.16; 0.13; 0.13; 0.16; 0.23; 0.37; 0.64; 1.16; 2.08; 3.48; 5.22; 6.76
3	–1.24; –1.17; –1.08; –0.96; –0.84; –0.79; –0.8; –0.9; –1.1; –1.21; –1.02; –1.28; –1.32; –1.34; –1.36; –1.37; –1.37; –1.36; –1.35; –1.33; –1.30
4	–3.0; –3.58; –4.12; –4.56; –4.86; –4.99; –4.94; –4.73; –4.36; –3.86; –3.30; –2.7; –2.13; –1.64; –1.26; –1.05; –1.00; –1.13; –1.43; –1.87; –2.43
5	1.0; 1.05; 90.6; 520.4; 1714.7; 2915.0; 2439.2; 1020.6; 230.7; 32.17; 3.29; 0.3; 0.03; 0.004; 0.001; 0.0003; 0.0006; 0.002; 0.01; 0.09; 0.9
6	2980.1; 2089.3; 742.4; 146.6; 18.6; 1.8; 0.16; 0.02; 0.003; 0.001; 0.001; 0.001; 0.002; 0.003; 0.018; 0.9; 1.22; 18.6; 146.6; 742.5; 2089.7

7	1.0; 1.34; 1.75; 2.18; 2.53; 2.71; 2.65; 2.37; 1.97; 1.54; 1.16; 0.86; 0.64; 0.5; 0.42; 0.37; 0.36; 0.39; 0.45; 0.56; 0.74
8	2.71; 2.6; 2.28; 1.86; 1.44; 1.07; 0.8; 0.46; 0.42; 0.4; 0.37; 0.37; 0.4; 0.48; 0.6; 0.8; 1.07; 1.44; 1.86; 2.28; 2.6
9	–1.32; –1.28; –1.26; –1.24; –1.25; –1.25; –1.25; –1.26; –1.27; –1.29; –1.29; –1.33; –1.34; –1.37; –1.37; –1.37; –1.37; –1.36; –1.36; –1.35; –1.34
10	–4.0; –4.2; –4.5; –4.7; –4.9; –5.0; –4.9; –4.8; –4.6; –4.4; –4.1; –3.8; –3.5; –3.1; –3.0; –3.0; –3.0; –3.1; –3.2; –3.4; –3.7
11	1.0; 2.4; 5.4; 10.4; 16.3; 19.9; 18.6; 13.4; 7.7; 3.6; 1.6; 0.64; 0.27; 0.13; 0.07; 0.05; 0.05; 0.06; 0.09; 0.18; 0.4
12	20.0; 17.5; 11.9; 6.4; 2.9; 1.2; 0.5; 0.2; 0.1; 0.06; 0.05; 0.05; 0.06; 0.1; 0.5; 1.0; 1.2; 2.9; 6.4; 11.9; 17.5
13	–1.1; –0.8; –0.3; 0.3; 0.7; 0.8; 0.7; 0.5; 0.04; –0.6; –0.9; –1.1; –1.27; –1.32; –1.35; –1.37; –1.37; –1.36; –1.34; –1.3; –1.2
14	–2.0; –2.8; –3.7; –4.3; –4.7; –4.9; –4.9; –4.5; –4.1; –3.3; –2.4; –1.5; –0.6; –0.04; 0.6; 0.92; 0.99; 0.79; 0.34; –0.3; –1.1
15	1.1; 3.2; 9.5; 22.8; 41.4; 53.9; 49.4; 31.9; 15.2; 5.7; 1.8; 0.55; 0.17; 0.06; 0.03; 0.02; 0.01; 0.02; 0.04; 0.1; 0.3
16	–0.78; –1.22; –1.34; –1.39; –1.42; –1.43; –1.42; –1.41; –1.37; –1.3; –1.1; –0.1; 1.1; 1.2; 1.33; 1.36; 1.37; 1.35; 1.3; 1.17; 0.65
17	54.5; 45.7; 27.2; 12.1; 4.3; 1.3; 0.4; 0.13; 0.05; 0.03; 0.02; 0.02; 0.03; 0.05; 0.13; 0.41; 1.3; 4.3; 12.1; 21.2; 45.7
18	–0.78; 0.18; 0.89; 1.13; 1.21; 1.24; 1.23; 1.18; 1.04; 0.63; –0.38; –1.01; –1.22; –1.3; –1.35; –1.36; –1.37; –1.36; –1.33; –1.27; –1.1

19	–1.0; –2.1; –3.2; –4.1; –4.7; –4.9; –4.8; –4.4; –3.7; –2.7; –1.6; –0.4; 0.7; 1.7; 2.4; 2.9; 3.0; 2.7; 2.1; 1.2; 0.2
20	1.0; 4.36; 16.7; 49.8; 105.0; 146.3; 130.9; 75.9; 30.0; 8.75; 2.1; 0.47; 0.11; 0.03; 0.01; 0.007; 0.006; 0.009; 0.02; 0.05; 0.2
21	148.4; 118.8; 62.6; 22.5; 6.21; 1.45; 0.33; 0.08; 0.02; 0.01; 0.007; 0.007; 0.01; 0.02; 0.08; 0.32; 1.45; 6.2; 22.6; 62.2; 119.0
22	0.0; 0.97; 1.23; 1.32; 1.36; 1.37; 1.36; 1.34; 1.28; 1.13; 0.64; –0.64; –1.13; –1.28; –1.34; –1.37; –1.36; –1.32; –1.23; –0.9; –0.2
23	–0.0001; –1.47; –2.8; –3.9; –4.65; –4.98; –4.87; –4.33; –3.4; –2.16; –0.74; 0.74; 2.17; 3.14; 4.33; 4.87; 4.98; 4.65; 3.9; 2.8; 1.4
24	1.0; 5.8; 29.3; 108.9; 266.4; 396.7; 347.1; 180.5; 59.2; 13.5; 2.4; 0.4; 0.07; 0.01; 0.005; 0.003; 0.002; 0.004; 0.009; 0.03; 0.1
25	403.4; 309.0; 142.2; 42.1; 8.9; 1.56; 0.26; 0.05; 0.01; 0.0044; 0.0026; 0.0026; 0.0044; 0.01; 0.05; 0.263; 1.56; 8.95; 42.1; 142.2; 309.9
26	0.78; 1.22; 1.34; 1.39; 1.42; 1.43; 1.42; 1.41; 1.37; 1.3; 1.1; 0.1; –1.1; –1.2; –1.33; –1.36; –1.37; –1.35; –1.3; –1.17; –0.65
27	1.0; –0.77; –2.3; –3.6; –4.6; –4.9; –4.8; –4.1; –3.1; –1.6; 0.1; 1.9; 3.6; 5.1; 6.2; 6.84; 6.98; 6.58; 5.69; 4.4; 2.7
28	1.0; 7.8; 51.5; 238.1; 675.9; 1075.4; 920.1; 429.3; 110.8; 20.8; 2.83; 0.35; 0.04; 0.01; 0.002; 0.001; 0.001; 0.001; 0.004; 0.02; 0.12
29	1.10; 1.32; 1.40; 1.43; 1.45; 1.46; 1.46; 1.44; 1.42; 1.37; 1.25; 0.76; –0.8; –1.22; –1.33; –1.36; –1.37; –1.35; –1.29; –1.1; –0.1
30	2.0; –0.06; –1.9; –3.4; –4.9; –4.8; –4.0; –2.7; –1.1; 0.95; 3.0; 5.0; 6.7; 8.1; 8.8; 8.9; 8.5; 7.47; 5.94; 4.06; 2.43

В качестве примера возьмем функцию, которая задана в точках $x_i = 2\pi i/(2n+1)$, где $i = 0,1,\ldots,2n$, значениями: 2.0; –0.06; –1.9; –3.4; –4.9; –4.8; –4.0; –2.7; –1.1; 0.95; 3.0; 5.0; 6.7; 8.1; 8.8; 8.9; 8.5; 7.47; 5.94; 4.06; 2.43.

1. Создайте новый проект командой *Файл/Новый/Приложение.*

2. Сохраните файлы модуля и проекта командой *Файл*/*Сохранить все* под именами LR2 и PR_LR2. Для этого удобно использовать соответствующую быструю кнопку (*Сохранить все*). В последующих сеансах работы сохраненный проект можно открыть командой *Файл/Открыть проект* (или *Повторно открыть*). Теперь перейдем к проектированию приложения – переносам на форму необходимых компонентов и заданию их свойствам значений, а в обработчиках событий – размещению кодов соответствующих алгоритмов. (Рекомендуется нажимать кнопку *Сохранить все* по окончании работы с каждым компонентом.) В результате проектирования получим форму, представленную на рис. 2.5.

3. Выделите форму, щелкнув на ней левой кнопкой мыши, и в свойство **Caption** (надпись) впишите *ТРИГОНОМЕТРИЧЕСКАЯ ИНТЕРПОЛЯЦИЯ***.**

4. В верхней части формы, по центру, поместите кнопку **Button1** (страница *Стандарт*)**,** для которой установите свойство **Font** – жирный, размер 10, а в свойство **Caption** впишите *ИСХОДНЫЕ ДАННЫЕ*.

5. Под кнопкой разместите компонент **StringGrid1** (страница *Дополнительно*). Установите следующие значения свойств компонента **StringGrid1**: **ColCount** – 22, **DefaultColWidth** – 28, **FixedCols** – 1, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 3. Раскрыв свойство **Options**, установите значение подсвойства **goEditing** – **true**, что даст возможность редактировать содержимое таблицы в **StringGrid1**.

6. Ниже, по центру, поместите кнопку **Button2** (страница *Стандарт*)**,** для которой установите свойство **Font** – жирный, размер 10, а в свойство **Caption** впишите *КОЭФФИЦИЕНТЫ ИНТЕРПОЛЯЦИОННОГО МНОГОЧЛЕНА***.**

7. Под кнопкой **Button2** разместите компонент **StringGrid2** (страница *Дополнительно*). Установите следующие значения свойств компонента **StringGrid1**: **ColCount** – 12, **DefaultColWidth** – 48, **FixedCols** – 1, **FixedRows** – 0, **Font** – черный, обычный, размер 8, – **RowCount** – 3.

8. Ниже компонента **StringGrid2**, по центру, поместите кнопку **Button3** (страница *Стандарт*), для которой установите свойство **Font** – жирный, размер 10, а в свойство **Caption** впишите *ПОСТРОЕНИЕ ГРАФИКОВ***.**

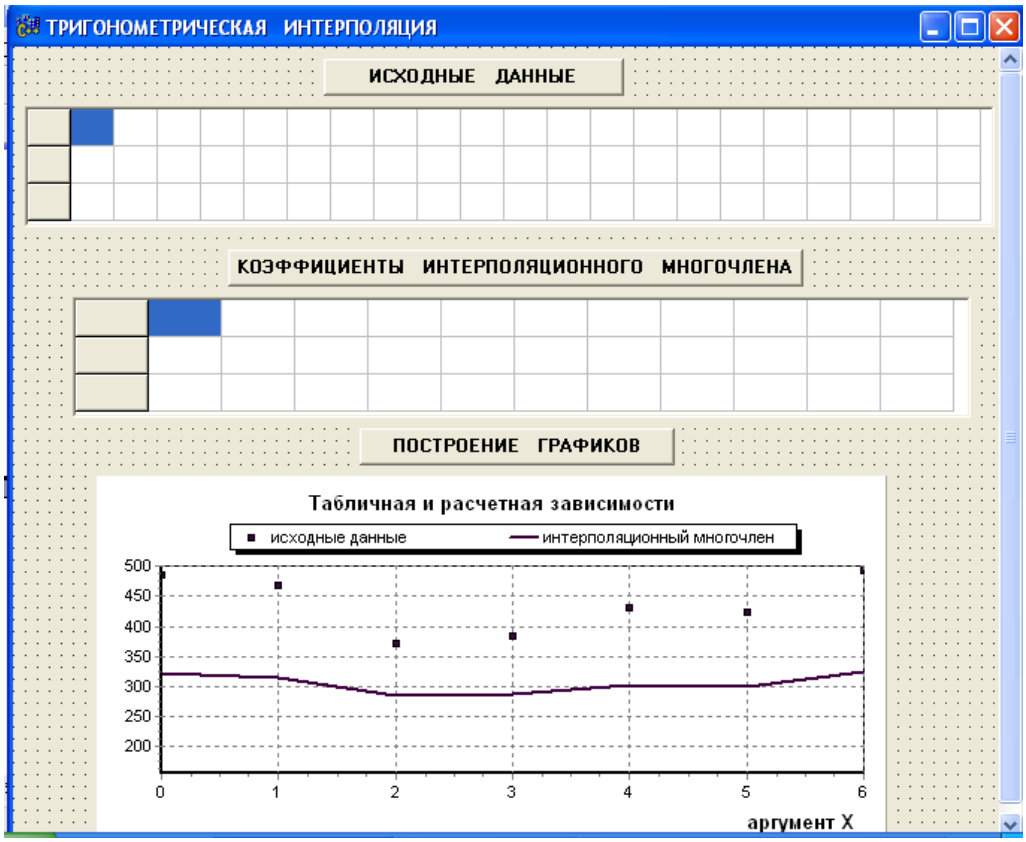


Рис. 2.5. Форма по окончании проектирования

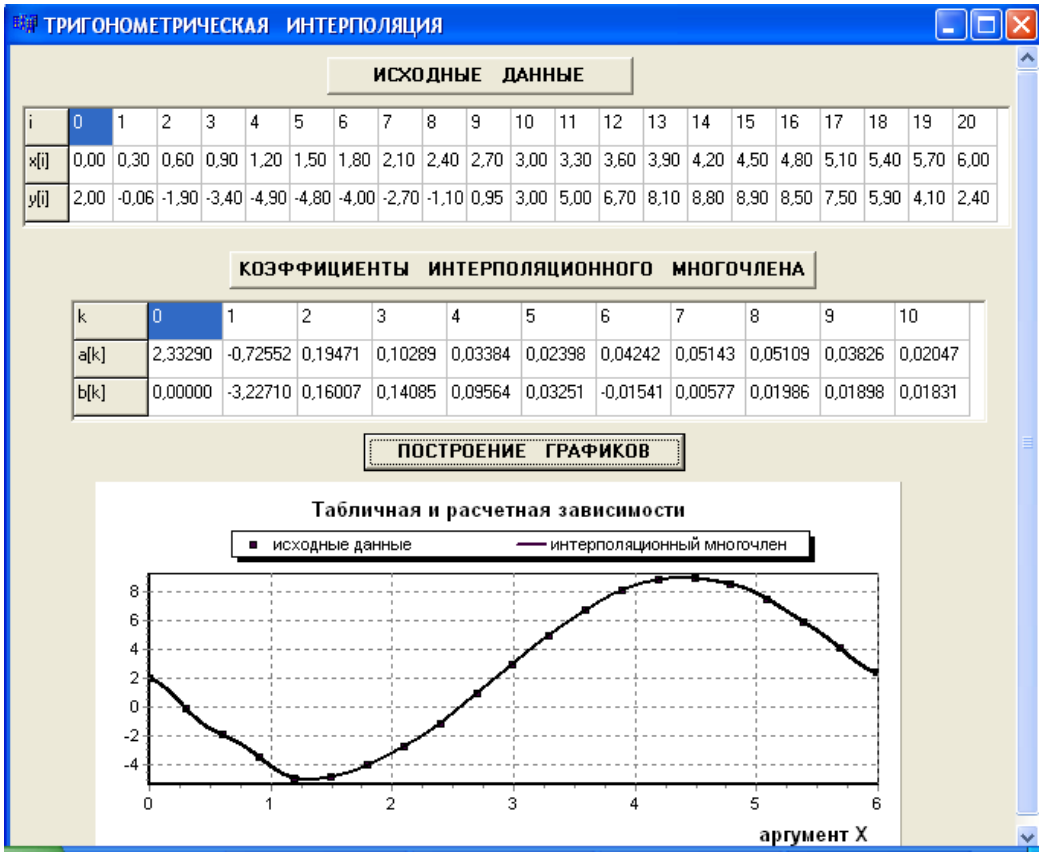


Рис. 2.6. Результаты выполнения задания

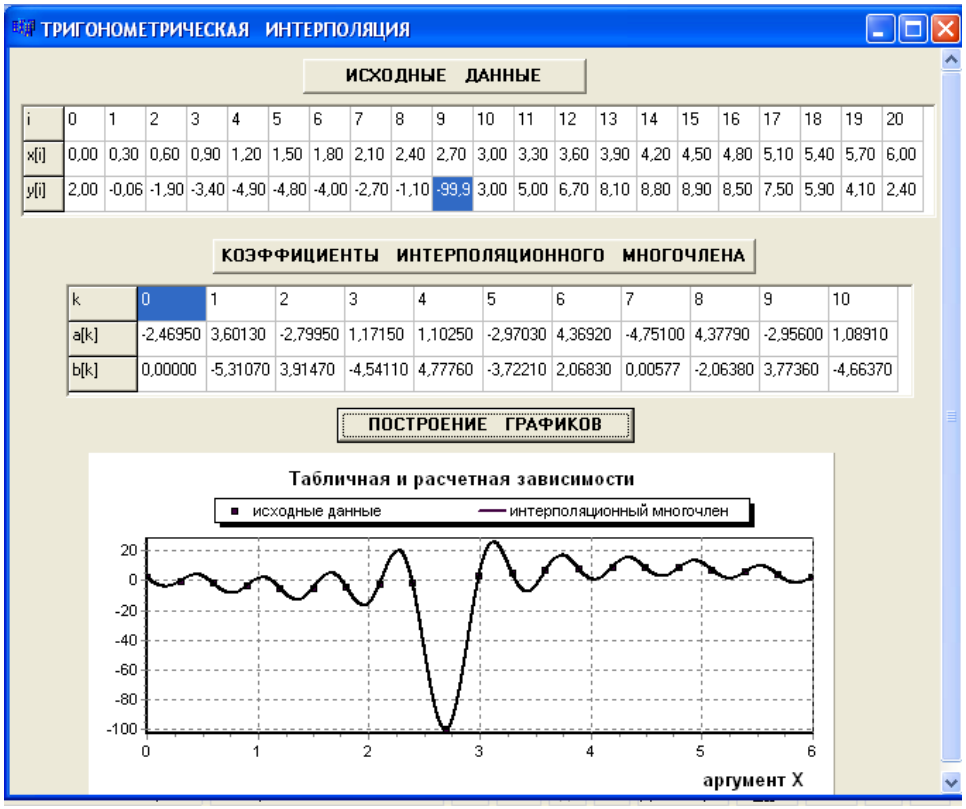


Рис. 2.7. Результаты после изменения исходных данных

9. В нижнюю часть формы поместите компонент **Chart1** (страница *Additional*). Зададим свойства компонента. Щелкните правой кнопкой мыши на компоненте **Chart1** и в появившемся меню выберите *Edit Chart*…. На экране появится окно *Редактора Диаграмм* (*Editing Chart1*) с открытой страницей *Chart,* которая имеет несколько закладок. На закладке *Panel*, нажав кнопку *Panel Color…*, выберите белый цвет. На закладке *Series* щелкните на кнопке *Add…* – добавить серию. В появившемся окне выберите тип графика – *Point* и выключите индикатор *3D*. После щелчка на *OK* снова появится окно *Editing Chart1*. Перейдите на закладку *Titles*. В окне редактирования, которое в данный момент соответствует *Title* – заголовку графика, сотрите *TChart* и напишите (шрифт *Font…* – черный, жирный, размер 10) *ТАБЛИЧНАЯ И РАСЧЕТНАЯ ЗАВИСИМОСТИ*. Цвет фона *Back Color.*. установите белый. В выпадающем списке от окна редактирования *Title* перейдите в окно редактирования *Foot* и напишите тем же шрифтом *аргумент X*. В группе кнопок *Alignment* нажмите кнопку *Right*. Цвет фона *Back Color.*. также установите белый. Перейдите на закладку *Axis* для задания координатных характеристик осей. Оставьте в группе *Axis* нажатой кнопку *Left* и включенным индикатор *Automatic*. Затем нажмите в группе *Axis* кнопку *Bottom* и выключите индикатор *Automatic*. Нажмите среднюю кнопку *Change…*и в окне *Value* редактора *Minimum Bottom Axis* впишите число *0*. Аналогичные действия выполните с верхней кнопкой *Change…*, но впишите число *6*. И, нажав нижнюю кнопку *Change…*, в окно *Increment:* занесите *0,6*. Перейдите на закладку *Series,* кнопкой *Add…*добавьте график *Line* при выключенном индикаторе *3D* . Выделив *Series 1,* кнопкой *Title…* вызовите *Change Series Title,* где дайте заголовок первому графику – *исходные данные*. Действуя подобным образом, т. е. выделив *Series 2,* дайте заголовок второму графику – *интерполяционный многочлен*. На вкладке *Legend* нажмите кнопку *Top*. Перейдите со страницы *Chart* на страницу *Series*. Здесь на закладке *Format* задают

параметры линий графиков. Требуемый график (*исходные данные*, *интерполяционный многочлен*) выбирается из выпадающего списка. Для графика *исходные данные* установите размеры прямоугольника: *Width*=2, *Height*=2, а нажав кнопку *Border…*, задайте толщину линии графика *Width*=1, и цвет линии графика *Color…* – черный. Для графика *интерполяционный многочлен* в группе *Line* через кнопку *Border…*устанавливается толщина (*Width*=2) линии графика, и через кнопку *Color…* – цвет линии графика – черный. Выключите индикатор *Color Each.*

10. Файл *LR2.cpp* с обработчиками щелчков на кнопках имеет вид:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "LR2.h"
#include<math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
double x[21],a[11],b[11];
double y[21]={2.0,-0.06,-1.9,-3.4,-4.9,-4.8,-4.0,-2.7,-1.1,0.95,
           3.0,5.0,6.7,8.1,8.8,8.9,8.5,7.47,5.94,4.06,2.43};
void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
  StringGrid1->Cells[0][0]="i";
  StringGrid1->Cells[0][1]="x[i]";
  StringGrid1->Cells[0][2]="y[i]";
  for(int i=0;i<21;i++){
      StringGrid1->Cells[i+1][0]=IntToStr(i);
      x[i]=2*M_PI*i/21;
      StringGrid1->Cells[i+1][1]=FloatToStrF(x[i],ffFixed,2,2);
      StringGrid1->Cells[i+1][2]=FloatToStrF(y[i],ffFixed,2,2);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    for(int k=0;k<=10;k++){
    StringGrid2->Cells[0][0]="k";
    StringGrid2->Cells[0][1]="a[k]";
    StringGrid2->Cells[0][2]="b[k]";
    StringGrid2->Cells[k+1][0]=IntToStr(k);
    a[k]=b[k]=0;
    for(int i=0;i<21;i++){
      a[k]+=StrToFloat(StringGrid1->Cells[i+1][2])*cos(k*x[i]);
      b[k]+=StrToFloat(StringGrid1->Cells[i+1][2])*sin(k*x[i]);}
    a[k]/=21;
    StringGrid2->Cells[k+1][1]=FloatToStrF(a[k],ffFixed,5,5);
    b[k]/=21;
    StringGrid2->Cells[k+1][2]=FloatToStrF(b[k],ffFixed,5,5);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
```

```
{
  Series1->Clear();
  Series2->Clear();
  for(int i=0;i<21;i++){
    Series1->AddXY(x[i],StrToFloat(StringGrid1->Cells[i+1][2]),"",clBlack); }
  for(float x=0;x<2*M_PI;x+=0.001){
     float p=a[0];
     for(int k=1;k<11;k++)
       p+=2*(a[k]*cos(k*x)+b[k]*sin(k*x));
     Series2->AddXY(x,p,"",clBlack);}
}
//---------------------------------------------------------------------------
```

11. Запустим приложение на выполнение, нажав быстрые кнопки *Сохранить все* и *Запуск*. После щелчков на кнопках *ИСХОДНЫЕ ДАННЫЕ, КОЭФФИЦИЕНТЫ ИНТЕРПОЛЯЦИОННОГО МНОГОЧЛЕНА, ПОСТРОЕНИЕ ГРАФИКОВ* получим результаты выполнения задания (рис. 2.6).

12. Воспользуемся тем, что таблица *ИСХОДНЫЕ ДАННЫЕ* позволяет редактировать значения *y[i]*. Изменив одно из значений, после щелчков на кнопках *КОЭФФИЦИЕНТЫ ИНТЕРПОЛЯЦИОННОГО МНОГОЧЛЕНА, ПОСТРОЕНИЕ ГРАФИКОВ,* получим результаты (рис. 2.7), которые также показывают выполнение основного требования интерполяции – в узлах интерполяции расчетные значения интерполяционного многочлена совпадают с исходными значениями таблично заданной функции.

13. Для возврата к первоначальным исходным данным достаточно щелкнуть на кнопке *ИСХОДНЫЕ ДАННЫЕ.*

14. Для завершения работы щелкните на кнопке формы «*Закрыть*» и выйдите из среды *Builder*.

Содержание отчета

1. Задание.
2. Формулы метода.
3. Результаты выполнения задания в *Mathcad'*е .
4. Блок-схема алгоритма и таблица идентификаторов.
5. Исходный код.
6. Результаты выполнения работы в виде значений коэффициентов интерполирующего полинома и графика полученного полинома с нанесенными на график исходными данными.
7. Библиографический список.

Контрольные вопросы

1. Что означают понятия интерполяции и экстраполяции?

2. Как решают задачу интерполяции и экстраполяции?

3. Каков результат решения задачи интерполяции и экстраполяции?

4. Какие методы интерполяции и почему применяют при небольшом числе узлов интерполяции?

5. Какие методы интерполяции и почему применяют при большом числе узлов интерполяции?

6. Сравните интерполяции по Лагранжу и сплайнами.

7. Какие недостатки имеют кусочно-линейная интерполяция и экстраполяция?

8. Расскажите о достоинствах и недостатках тригонометрической интерполяции.

9. Что нужно сделать с исходными данными, чтобы получить на графике существенное отличие сплайновой интерполяции от кусочно-линейной?

10. Как изменяется качество интерполяции для перечисленных методов при увеличении числа узлов интерполяции?

11. Сравните методы интерполяции по объему вычислений, т. е. по затратам машинного времени.

АЛГЕБРА, ФОРМИРОВАНИЕ, РАЗЛОЖЕНИЕ И ОБРАЩЕНИЕ МАТРИЦ

Краткие сведения

Алгебра матриц

Система $n \times m$ чисел (действительных или комплексных), расположенных в прямоугольной таблице из *n* строк и *m* столбцов,

$$A = \begin{bmatrix} a_{11} & a_{12} & .. & a_{1m} \\ a_{21} & a_{22} & .. & a_{2m} \\ .. & .. & .. & .. \\ a_{n1} & a_{n2} & .. & a_{nm} \end{bmatrix},$$

называется матрицей (*числовой*). Числа $a_{ij} (i = 1, 2, \ldots, n; j = 1, 2, \ldots, m)$ - элементы матрицы. Пользуются также сокращенными записями $A = \lfloor a_{ij} \rfloor, (i = 1, 2, .., n;\ j = 1, 2, .., m)$ и $A = \lfloor a_{ij} \rfloor_{nm}$, причем говорят, что матрица имеет размер $n \times m$.

Если $n \neq m$, то матрицу называют *прямоугольной*, а в случае $n = m$ - *квадратной*. Матрица размера $1 \times m$ называется *вектором-строкой*, а размера $n \times 1$ - *вектором-столбцом*. Число (скаляр) можно считать матрицей размера 1×1. Квадратная матрица, имеющая ненулевые элементы только на главной диагонали, называется *диагональной*. Если в диагональной матрице ненулевые элементы равны 1, то матрицу называют *единичной* и обозначают через E.

С квадратной матрицей связан *определитель* (число), определяемое по правилу $\det A = \sum_{(\alpha_1, \alpha_2, .., \alpha_n)} (-1)^{\chi} a_{1\alpha_1} a_{2\alpha_2} .. a_{n\alpha_n}$, где сумма распространена на всевозможные перестановки $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ элементов $1, 2, \ldots, n$ и,

следовательно, содержит $n!$ слагаемых, причем $\chi = 0$, если перестановка четная, и $\chi = 1$, если перестановка нечетная.

Ранг данной матрицы размера n есть такое число r, что по крайней мере один определитель r-го порядка, получаемый при удалении некоторых строк и (или) столбцов, отличен от нуля, а все определители $r+1$-го порядка равны нулю. Для *невырожденной* матрицы $r = n$.

След (*spur*) квадратной матрицы A равен сумме элементов на главной диагонали: $SpA = \sum_i a_{ii}, \quad i = 1, 2, \ldots, n.$

Для матриц произвольного типа и размера используют три нормы:

$$\|A\|_m = \max_i \sum_j |a_{ij}| \ (m\text{-}норма);$$

$$\|A\|_l = \max_j \sum_i |a_{ij}| \ (l\text{-}норма);$$

$$\|A\|_k = \sqrt{\sum_{i,j} |a_{ij}|^2} \quad (k\text{-}норма).$$

Матрицы равны, если они имеют один и тот же размер и равны их соответствующие элементы. Сумма и разность матриц определена для матриц одинакового типа. Правило умножения матриц: элемент a_{ij} матрицы-произведения представляет собой сумму произведений элементов *i*-й строки первой матрицы на элементы *j*- го столбца второй матрицы. Следовательно, если размер первой матрицы $n \times m,$ а второй - $p \times q,$ перемножение возможно при $m = p,$ а матрица-произведение имеет размер $n \times q.$

Определитель матрицы-произведения равен произведению определителей матриц-сомножителей.

Транспонированная матрица A^T получается из исходной A заменой строк столбцами.

Матрица называется *симметрической (симметричной)*, если она совпадает со своей транспонированной, т. е. если $a_{ij} = a_{ji}$. В случае $a_{ij} = -a_{ji}$ матрицу называют *кососимметрической*.

Обратной по отношению к данной A называется матрица A^{-1}, которая, будучи умноженной как справа, так и слева на данную матрицу, дает единичную матрицу E. Нахождение обратной матрицы для данной называется обращением данной матрицы.

При вычислении матрицы, обратной матрице A, точность оценивается с помощью модифицированной *k*-нормы матрицы ошибок D, равной $D = A * A^{-1} - E$,

$$\|D\| = \frac{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2}}{n}.$$

Квадратная матрица называется *неособенной*, если определитель ее не равен нулю. В противном случае матрица называется *особенной*. Обратную матрицу имеет всякая неособенная матрица.

Присоединенную (или *союзную*) матрицу получают транспонированием матрицы, составленной из алгебраических дополнений элементов исходной матрицы. Разделив присоединенную матрицу на определитель исходной матрицы, получают матрицу, обратную для исходной.

С помощью горизонтальных и вертикальных перегородок, идущих вдоль всей матрицы, последнюю разбивают на матрицы низших порядков (клетки или блоки). Матрицу, разбитую на клетки, называют *клеточной* или *блочной*.

Разбиение матрицы на клетки может быть осуществлено различными способами. Если все клетки являются квадратными матрицами и главные

диагонали всех клеток находятся на главной диагонали исходной матрицы, а вне клеток стоят нули, то такую клеточную матрицу называют *квазидиагональной*.

Окаймленные матрицы представляют собой другой важный частный случай клеточных матриц. Из исходной матрицы A окаймленную матрицу матрицу A_n получают следующим образом: $A_n = \begin{bmatrix} A_{n-1} & U_n \\ V_n & a_{nn} \end{bmatrix}$, где

$A_{n-1} = \begin{bmatrix} a_{11} & a_{12} & .. & a_{1,n-1} \\ a_{21} & a_{22} & .. & a_{2,n-1} \\ .. & .. & .. & .. \\ a_{n-1},_1 & a_{n-1,2} & .. & a_{n-1,n-1} \end{bmatrix}$ – матрица порядка *n-1*; $U_n = \begin{bmatrix} a_{1,n} \\ a_{2,n} \\ \dots \\ a_{n-1,n} \end{bmatrix}$ – матрица-столбец; $V_n = \lfloor a_{n,1} \quad a_{n,2} \quad \dots \quad a_{n,n-1} \rfloor$ – матрица-строка и a_{nn} – число.

Клеточные матрицы одинакового типа и с одинаковым разбиением называют *конформными*. Удобство клеточных матриц состоит в том, что действия над ними совершаются формально по тем же правилам, что и над обыкновенными матрицами.

Матрица, все элементы которой равны нулю, называется *нулевой* и обозначается через 0_{mn}.

Матрицы *общего вида* называют *плотными* или *заполненными*.

Практические задачи, например анализ протяженной электрической цепи с локальными связями между элементами, приводят к *разреженным* матрицам, в которых нулевых элементов значительно больше, чем ненулевых. Пример разреженной матрицы – *трехдиагональная* матрица, все ненулевые элементы которой расположены на главной и двух соседних с ней диагоналях.

Матрица называется *ленточной* с полушириной ленты, равной m, если $a_{ij} = 0$ для $|i - j| > m$. Все ненулевые элементы расположены на

$s = 2m + 1$ ближайших к главной диагоналях матрицы; число s называют шириной ленты. В случае $s << n$ ленточная матрица является разреженной.

Если в квадратной матрице элементы, стоящие выше (ниже) главной диагонали, равны нулю, то такую матрицу называют *нижней* (*верхней*) *треугольной* матрицей. Диагональная матрица является частным случаем как верхней, так и нижней треугольной матрицы. Определитель треугольной матрицы равен произведению ее диагональных элементов.

Всякую квадратную матрицу A, имеющую отличные от нуля главные диагональные миноры $\Delta_1 = a_{11} \neq 0; \quad \Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0; \ \ldots; \ \Delta_n = |A| \neq 0,$ можно представить в виде произведения двух треугольных матриц (верхней и нижней), причем это разложение будет единственным, если зафиксировать диагональные элементы одной из матриц (например, принять их равными 1).

Если главные диагональные миноры симметрической матрицы положительны, то она называется *положительно определенной*.

Действительная матрица A является *ортогональной*, если ее транспонированная матрица A^T совпадает с обратной A^{-1}, т. е. $A^T = A^{-1}$, или $A * A^T = A^T * A = E$. Ортогональная матрица имеет следующие свойства.

1. Строки (столбцы) попарно ортогональны, т. е.

$\sum_{k=1}^{n} a_{ik} a_{jk} = 0$ при $i \neq j$ и $\sum_{k=1}^{n} a_{ki} a_{kj} = 0$ при $i \neq j$.

2. Сумма квадратов элементов каждой строки (столбца) равна 1.

3. Определитель равен 1.

4. Транспонированная и обратная матрицы ортогональной матрицы также являются ортогональными матрицами.

Две матрицы называются *эквивалентными*, если одна получается из другой с помощью элементарных преобразований: перестановка двух строк или столбцов; умножение всех элементов какой-либо строки (столбца) на одно и то же число; прибавление к элементам какой-либо строки (столбца) соответствующих элементов другой строки (столбца), умноженных на одно и то же число.

Указание для изменения порядка следования строк (столбцов) матрицы задают в виде вектора транспозиции. Например, если для строк матрицы размера 5*6 задан вектор транспозиции $\{1,2,3,4,5\}$, то матрица не будет изменена, а в случае $\{5,3,2,4,1\}$ поменяются местами строки: первая – с пятой, третья – со второй.

Алгоритмы формирования матриц

Разработка и тестирование алгоритмов и программ для решения задач с матрицами требует формирования матриц различных видов. В качестве примеров на рис. 3.1 приведен алгоритм формирования нижней и верхней треугольных матриц с ненулевыми элементами на главных диагоналях и вычисления их определителей, а на рис. 3.2 – алгоритм формирования неособенной квадратной матрицы как произведение нижней и верхней треугольных матриц с ненулевыми элементами на главных диагоналях.

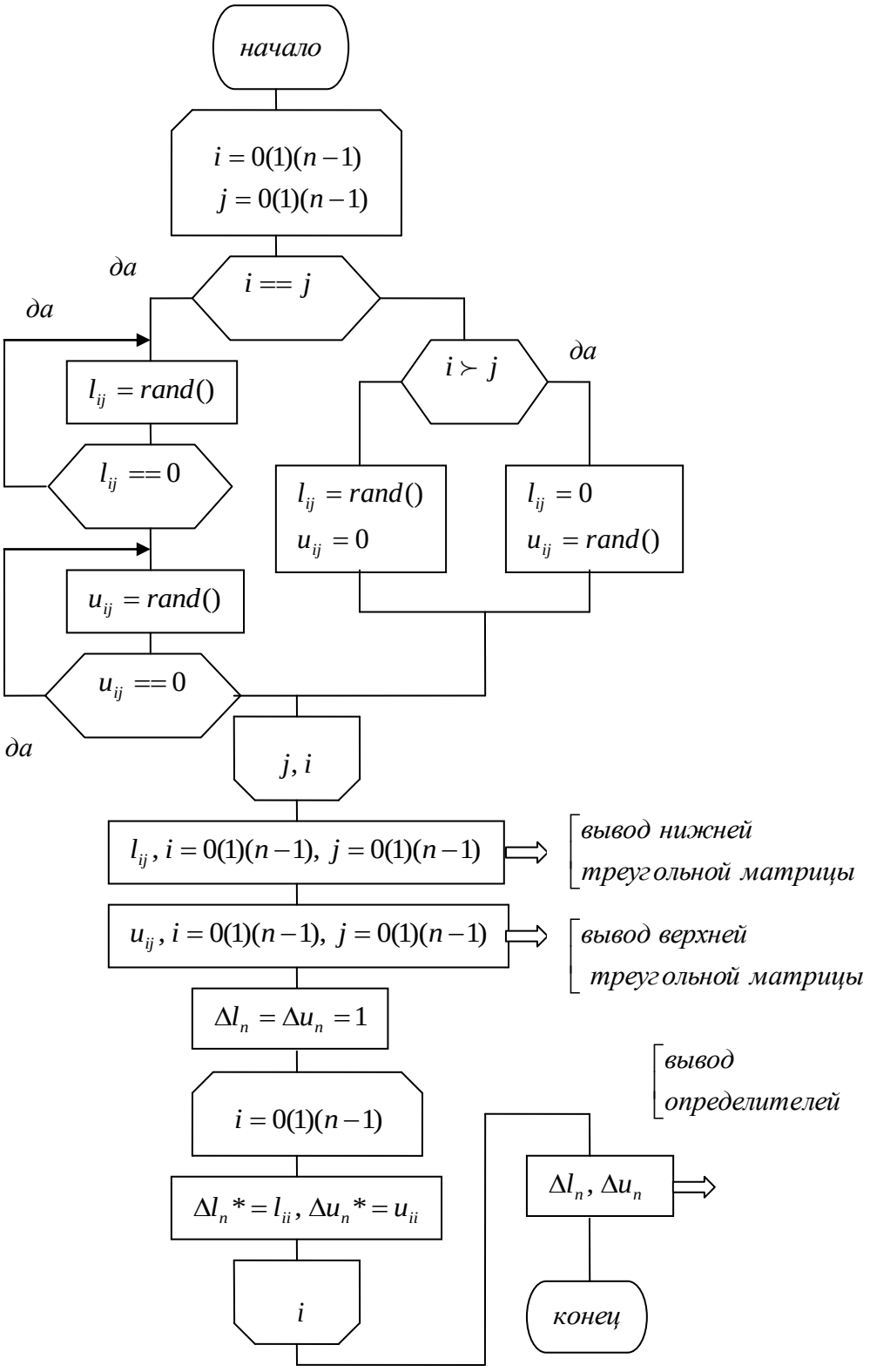


Рис. 3.1. Алгоритм формирования нижней и верхней треугольных матриц с ненулевыми элементами на главных диагоналях

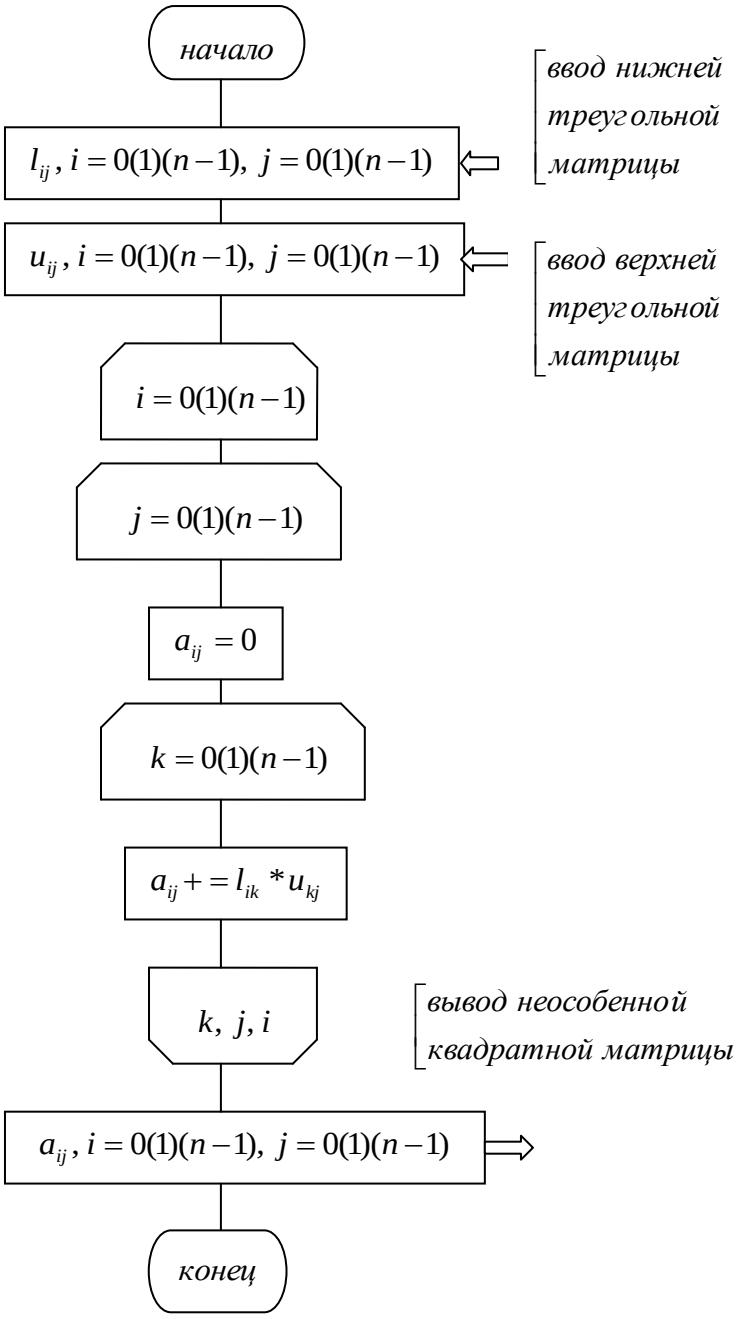


Рис. 3.2. Алгоритм формирования неособенной квадратной матрицы как произведение нижней и верхней треугольных матриц с ненулевыми элементами на главных диагоналях

Методы разложения матриц

Разложение неособенной квадратной матрицы в произведение двух треугольных матриц: верхней и нижней с единичной главной диагональю

Выше было указано, что всякую квадратную матрицу A, имеющую отличные от нуля главные диагональные миноры $\Delta_1 = a_{11} \neq 0;$ $\Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0;$ …; $\Delta_n = |A| \neq 0,$ можно представить в виде произведения двух треугольных матриц (верхней и нижней), причем это разложение будет единственным, если зафиксировать диагональные элементы одной из матриц (например, принять их равными 1). Следовательно, $A = L * U$, где L и U – нижняя и верхняя треугольные матрицы соответственно.

Для разработки алгоритма необходимо получить формулы, позволяющие вычислять элементы нижней и верхней треугольных матриц по известным значениям элементов исходной матрицы A.

При $n = 4$ из произведения матриц $L * U = A$ имеем:

$$l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} = a_{43},$$
$$l_{31}u_{13} + l_{32}u_{23} + l_{33}u_{33} = a_{33},$$
$$l_{31}u_{14} + l_{32}u_{24} + l_{33}u_{34} = a_{34}.$$

Распространив эти формулы на общий случай (n- произвольное), получим формулы для вычисления элементов матрицы L:

$$l_{ii} = 1, \quad i = 1, 2, \ldots, n.$$

$$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}) / u_{jj}, \quad i = 2, 3, \ldots, n; \quad j = 1, 2, \ldots, i-1.$$

и формулы для вычисления элементов матрицы U:

$$u_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk} u_{kj}, \quad j = 1, 2, \ldots, n.$$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}, \quad i = 1, 2, \ldots, n-1; \quad j = i+1, \ldots, n.$$

Полученные формулы позволяют построить алгоритм разложения неособенной квадратной матрицы в произведение двух треугольных матриц: верхней и нижней с единичной главной диагональю (рис. 3.3).

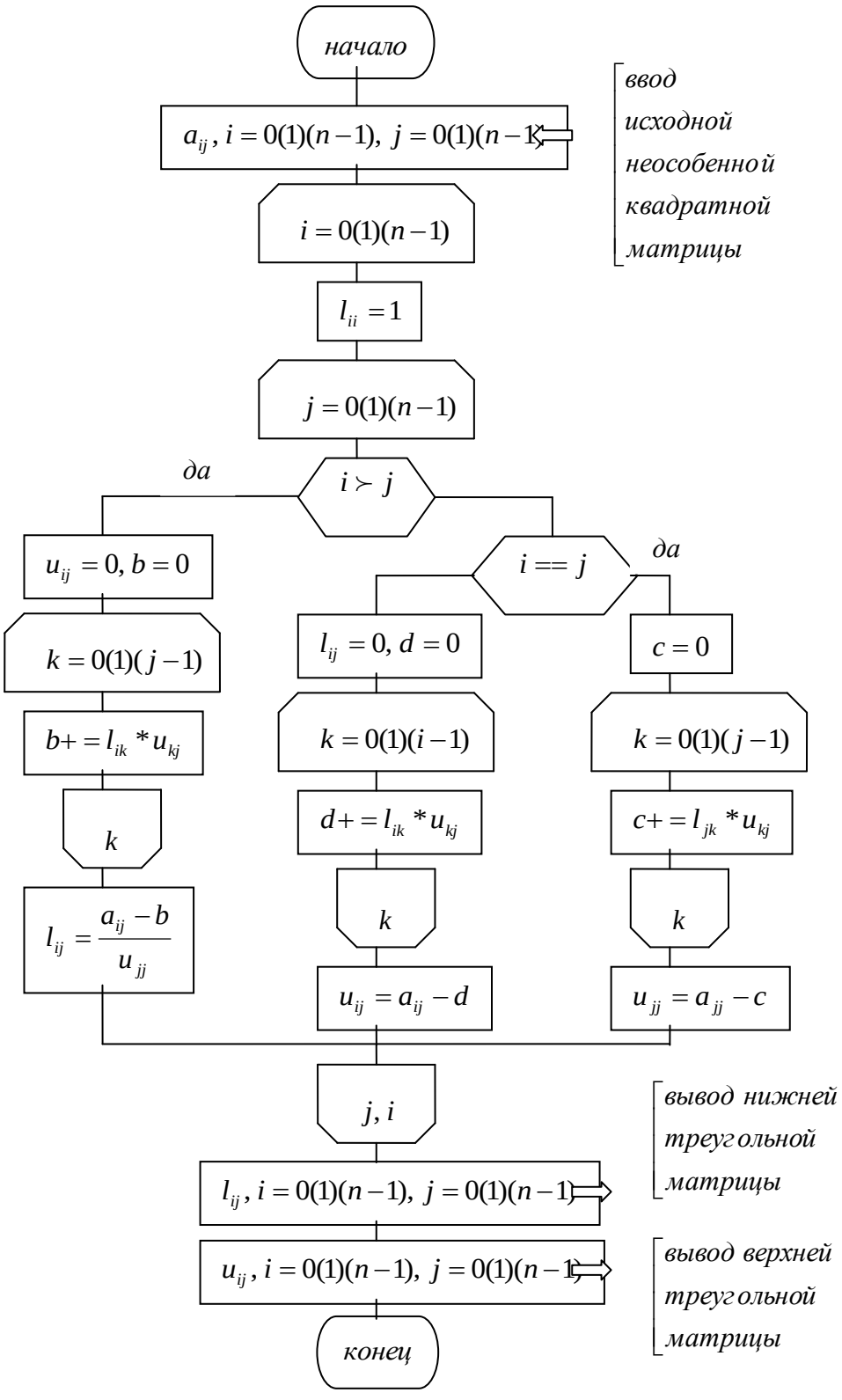


Рис. 3.3. Алгоритм разложения неособенной квадратной матрицы в произведение двух треугольных матриц: верхней и нижней с единичной главной диагональю

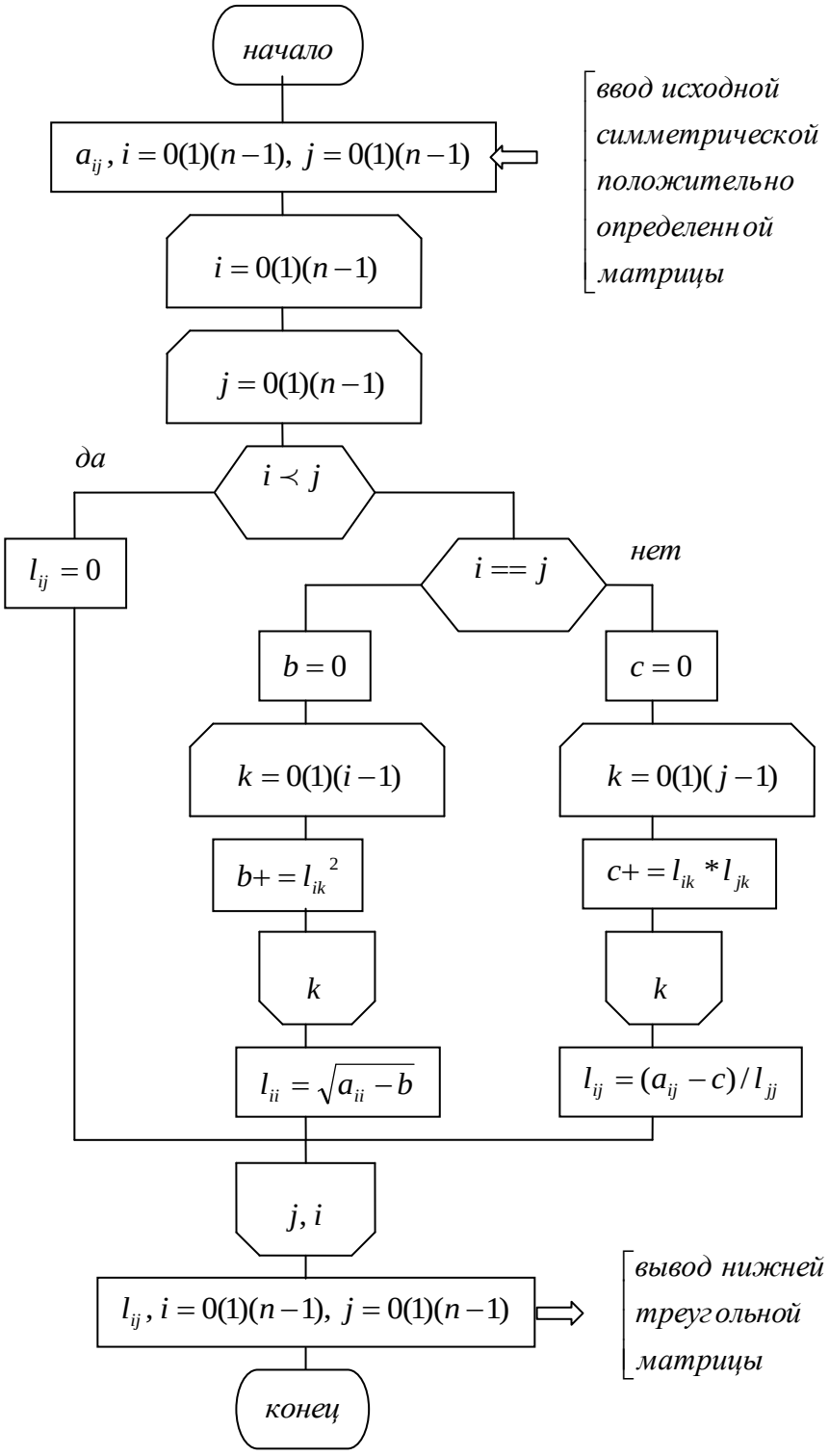


Рис. 3.4. Алгоритм разложения симметрической положительно определенной матрицы в произведение нижней треугольной и транспонированной ей матрицы

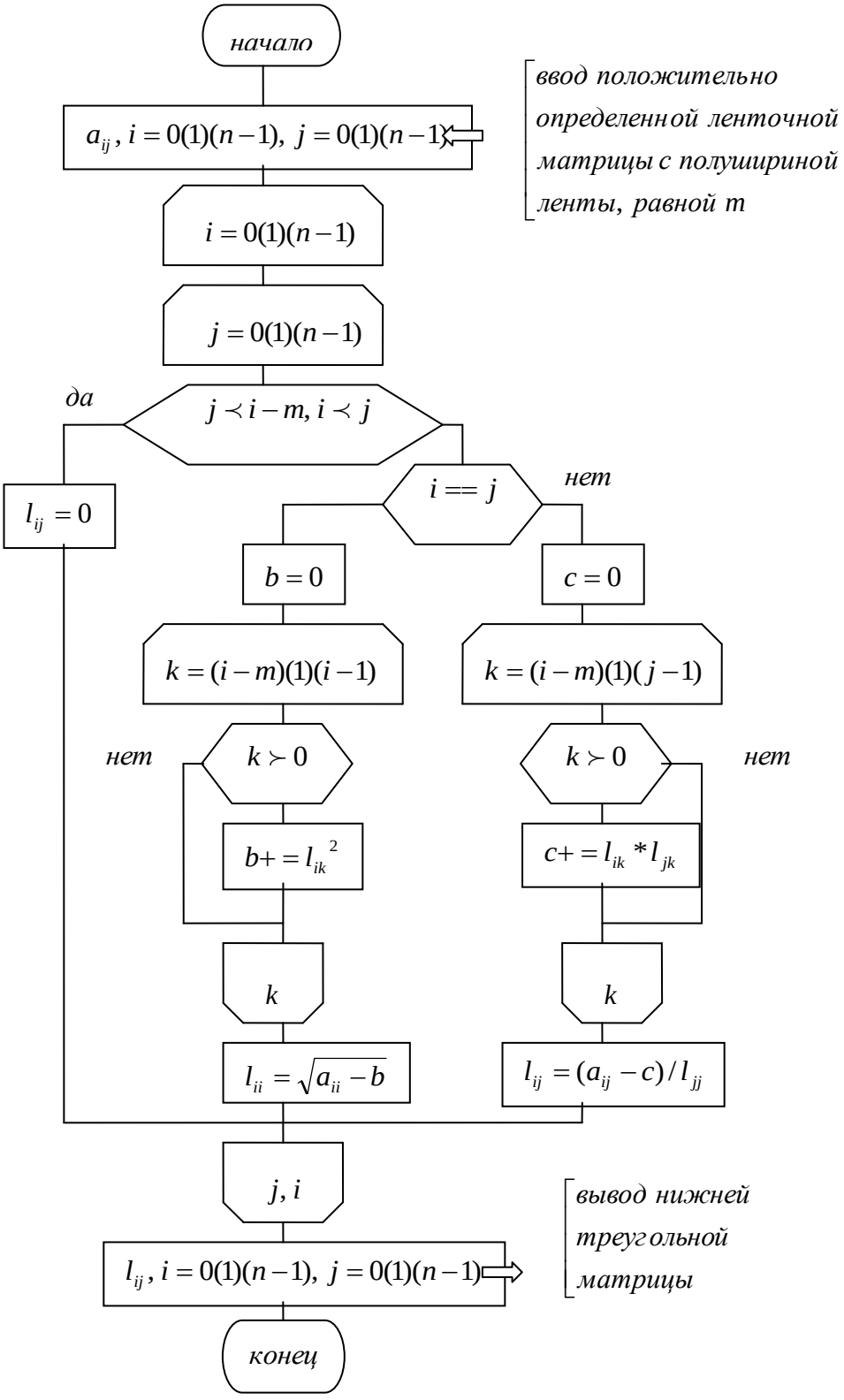


Рис. 3.5.алгоритм разложения положительно определенной ленточной матрицы в произведение нижней треугольной и транспонированной ей матрицы

Разложение неособенной симметрической матрицы в произведение двух взаимно транспонированных треугольных матриц

Подобным же образом получим формулы и алгоритм (рис. 3.4) для разложения симметрической положительно определенной матрицы в произведение двух треугольных, взаимно транспонированных матриц $A = L * L^T$:

$$l_{ij} = 0, \quad \textit{если } j > i;$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, \quad \textit{если } i = j;$$

$$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj}, \quad \textit{если } j < i.$$

Разложение положительно определенной ленточной матрицы в произведение двух взаимно транспонированных треугольных матриц

Приведем также формулы и алгоритм (рис. 3.5) для разложения положительно определенной ленточной матрицы с полушириной ленты, равной m, в произведение нижней треугольной и транспонированной ей матрицы:

$$l_{ij} = 0, \quad \textit{если } i - j > m, i < j;$$

$$l_{ij} = (a_{ij} - \sum_{k=i-m}^{j-1} l_{ik} l_{jk}) / l_{jj}, \quad j = i - m, \ldots, i - 1;$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=i-m}^{i-1} l_{ik}^2}, \quad j = i.$$

Разложение неособенной квадратной матрицы в произведение нижней треугольной матрицы с единичной диагональю и матрицы с ортогональными строками

Пусть дана действительная неособенная матрица

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

Из каждой i-й строки, начиная со второй, вычитают первую строку, умноженную на некоторое число $\lambda_{i1}(i=2,3,..,n)$, зависящее от номера преобразуемой строки. В результате получим преобразованную матрицу $A^{(1)}$. Множители λ_{i1} выбираются из условия ортогональности первой строки всем остальным строкам: $\lambda_{i1}=\left(\sum_{j=1}^{n} a_{1j}a_{ij}\right)/\sum_{j=1}^{n} a_{1j}^2$ Матрицу $A^{(1)}$ преобразуем аналогично: из каждой ее i-й строки $(i=3,4,..,n)$ вычитаем вторую строку $A^{(1)}$, умноженную на $\lambda_{i2}=\left(\sum_{j=1}^{n} a_{2j}^{(1)}a_{ij}^{(1)}\right)/\sum_{j=1}^{n}(a_{2j}^{(1)})^2$. Получим матрицу $A^{(2)}$ и т.д., пока не получится матрица $A^{(n-1)}$, все строки которой попарно ортогональны. Матрица $A^{(n-1)}=R$ с ортогональными строками получилась из матрицы A в результате цепи элементарных преобразований. Поэтому справедливо равенство $R=\lambda A$, где λ - нижняя треугольная матрица. Матрицу λ нетрудно получить, проделав над единичной матрицей все преобразования, совершенные над матрицей A. Затем находится λ^{-1} из условия $\lambda\lambda^{-1}=E$. Итак, окончательно $A=\lambda^{-1}R$.

Методы обращения матриц

Могут быть обращены только неособенные квадратные матрицы.

Метод окаймления (деления на клетки)

Исходную матрицу M размера $n \times n$ разобьем на четыре клетки $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, где $a, b, c, d -$ подматрицы размеров $p \times p, p \times q, q \times p, q \times q$.

Примем, что матрица M^{-1} существует и может быть разбита на клетки так же, как и матрица M, т.е. $M^{-1} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, где $A, B, C, D -$ подматрицы размеров $p \times p, p \times q, q \times p, q \times q$. Поскольку $MM^{-1} = E$,

то $\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} E_p & 0 \\ 0 & E_q \end{bmatrix}$, или $\begin{aligned} aA + bC &= E_p, \\ aB + bD &= 0, \\ cA + dC &= 0, \\ cB + dD &= E_q . \end{aligned}$

Пусть подматрица d имеет обратную d^{-1}, которая известна. Тогда после небольших преобразований получим формулы, которые могут быть последовательно решены относительно матриц A, B, C, D:

$$\begin{aligned} A &= \left(a - bd^{-1}c\right)^{-1}, \\ B &= -Abd^{-1}, \\ C &= -d^{-1}cA, \\ D &= d^{-1} - d^{-1}cB. \end{aligned} \qquad \textbf{(*)}$$

Вычисление обратной матрицы реализуется с помощью метода окаймления. Суть его заключается в следующем. Пусть дана матрица

$$M = \begin{bmatrix} m_{11} & \dots & m_{1n} \\ \dots & \dots & \dots \\ m_{n1} & \dots & m_{nn} \end{bmatrix}.$$

Образуем $M_1 = [m_{11}]$; $M_2 = \begin{bmatrix} M_1 & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$; $M_3 = \begin{bmatrix} M_2 & & m_{13} \\ & & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$ и т.д.

Каждая следующая матрица получена из предыдущей при помощи окаймления. Обратная к первой из этих матриц находится непосредственно: $M_1^{-1} = 1/m_{11}$. Зная M_1^{-1} и применив к M_2 схему вычислений (*), можно получить M_2^{-1}, а затем при помощи M_2^{-1} аналогично получить M_3^{-1} и т.д. Процесс заканчивается матрицей M_n^{-1}, т.к. $M_n^{-1} = M^{-1}$. Обращение можно начать и с правого нижнего угла матрицы M.

Метод Ершова (метод пополнения)

На основе исходной матрицы A и единичной матрицы E строится последовательность матриц

$$A^{(0)}, A^{(1)'}, A^{(1)}, \dots\dots, A^{(n)'}, A^{(n)}, \quad \textit{где} \quad A^{(0)} = A - E,$$

$$a_{ij}^{(m)'} = \begin{cases} a_{ij}^{(m-1)}, & \textit{если} \quad i \neq m, \\ 1, & \textit{если} \quad i = m \ \ u \ \ i = j, \\ 0, & \textit{если} \quad i = m \ \ u \ \ i \neq j; \end{cases}$$

$$a_{ij}^{(m)} = a_{ij}^{(m)'} - a_{im}^{(m)'} * a_{mj}^{(m-1)} / \left(1 + a_{mm}^{(m-1)}\right); \quad i, j, m = 1, 2, \dots, n.$$

Матрица $A^{(n)} = A^{(-1)}$. Матрицы $A^{(1)'}, A^{(2)'}, ..., A^{(n)'}$ являются вспомогательными.

Метод Фаддеева

Напомним, что следом (*spur*) матрицы A называется сумма ее элементов на главной диагонали: $SpA = \sum_i a_{ii}$, $i = 1, 2, ..., n$.

Вычисление обратной матрицы A^{-1} порядка n производится по следующим формулам:

$$A_1 = A, \qquad p_1 = SpA_1, \qquad B_1 = A_1 - p_1 E;$$

$$A_2 = AB_1, \qquad p_2 = \frac{1}{2} SpA_2, \qquad B_2 = A_2 - p_2 E;$$

……..……..……..……..……..……..…

$$A_{n-1} = AB_{n-2}, \qquad p_{n-1} = \frac{1}{n-1} SpA_{n-1}, \qquad B_{n-1} = A_{n-1} - p_{n-1} E;$$

$$A_n = AB_{n-1}, \qquad p_n = \frac{1}{n} SpA_n, \qquad A^{-1} = \frac{1}{p_n} B_{n-1}.$$

Операции с векторами и матрицами в системе Mathcad

Векторы и матрицы можно задавать путем ввода их элементов. Для указания индексов после имени переменной вводится открывающая квадратная скобка, например при вводе $M[1,2:$ увидим $M_{1,2} :=$. Для задания векторов и матриц можно либо воспользоваться командой *Матрица* в меню *Вставка*, либо нажать $Ctrl + M$, либо щелкнуть на кнопке с изображением шаблона матрицы, а затем в диалоговом окне указать размер матрицы.

Для работы с векторами и матрицами система поддерживает ряд операций (V – вектор, M – матрица, Z – скаляр), сведенных в таблицу.

Таблица

Операция	Клавиши	Описание
$V1+V2$	$V1+V2$	Сложение векторов $V1$ и $V2$
$-M$	$-M$	Смена знака у элементов матрицы
$Z*M, M*Z$	$Z*M, M*Z$	Умножение матрицы на скаляр
$M*V$	$M*V$	Умножение матрицы на вектор
$M1*M2$	$M1*M2$	Умножение двух матриц
$\frac{M}{Z}$	M/Z	Деление матрицы на скаляр
M^{-1}	$M\wedge -1$	Обращение матрицы
M^n	$M\wedge n$	Возведение матрицы в степень
$\lvert M\rvert$	$\lvert M$	Вычисление определителя матрицы
V^T	*VCtrl*!	Транспонирование вектора
M^T	*MCtrl*!	Транспонирование матрицы

Векторные и матричные функции в системе Mathcad

$length(V)$ – возвращает число элементов вектора;

$augment(M1, M2)$ – объединяет в одну две матрицы с одинаковым числом строк (объединение «бок о бок»);

$stack(M1, M2)$ – объединяет в одну две матрицы с одинаковым числом столбцов («сажая» $M1$ на $M2$);

$identity(n)$ – создаёт единичную квадратную матрицу размером $n \times n$;

$submatrix(A, ir, jr, ic, jc)$ – возвращает подматрицу, состоящую из всех элементов, содержащихся в строках от ir по jr и в столбцах с ic по jc ($ir \le jr, ic \le jc$);

$diag(V)$ – создаёт диагональную матрицу, элементы главной диагонали которой равны элементам вектора-столбца;

$cols(M), rows(M)$ – возвращает число столбцов, строк матрицы;

$rank(M)$ – возвращает ранг матрицы;

$tr(M)$ – возвращает след квадратной матрицы;

$norm1(M)$ – возвращает норму $L1$ (l –норму) матрицы;

$norm2(M)$ – возвращает норму $L2$ матрицы;

$norme(M)$ – возвращает евклидову норму (k – норму) матрицы;

$normi(M)$ – возвращает неопределенную норму (m – норму) матрицы.

Пример работы с матричными операциями и функциями

Задание матрицы A с размерностью 2×3

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \qquad \text{rows}(A) = 2 \qquad \text{cols}(A) = 3 \quad .$$

Транспонирование матрицы A

$$B := A^T \qquad B = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \quad .$$

Создание единичной матрицы и вычисление её следа

$$M := \text{identity}(2) \qquad M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \text{tr}(M) = 2 \quad .$$

Задание и обращение матрицы A

$$A := \begin{pmatrix} 7 & 8 \\ 4 & 5 \end{pmatrix} \qquad B := A^{-1} \qquad B = \begin{pmatrix} 1.667 & -2.667 \\ -1.333 & 2.333 \end{pmatrix} \quad .$$

Проверка правильности обращения матрицы A

$$C := B^{-1} \qquad C = \begin{pmatrix} 7 & 8 \\ 4 & 5 \end{pmatrix} \quad .$$

Умножение двух матриц

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad B := \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \qquad A \cdot B = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix} \quad .$$

Задание квадратной матрицы и вычисление её определителя

$$A := \begin{pmatrix} 2 & 5 & -3 \\ 1 & 4 & -1 \\ 1 & 3 & 2 \end{pmatrix} \qquad D := |A| \qquad D = 10 \quad .$$

Сортировка матрицы по первому столбцу

$$\mathrm{csort}(A,1)=\begin{pmatrix}1 & 3 & 2\\ 1 & 4 & -1\\ 2 & 5 & -3\end{pmatrix}.$$

Сортировка матрицы по первой строке

$$\mathrm{rsort}(A,1)=\begin{pmatrix}-3 & 2 & 5\\ -1 & 1 & 4\\ 2 & 1 & 3\end{pmatrix}.$$

ЛАБОРАТОРНАЯ РАБОТА 3

АЛГЕБРА И ФОРМИРОВАНИЕ МАТРИЦ

ЗАДАНИЕ

1. По предложенному из таблицы варианту задание выполнить в *Mathcad'*е, если там есть соответствующие операции и функции.

2. Разработать блок-схему алгоритма и код для выполнения задания.

Таблица

Вариант	Операция
1	транспонирование матрицы
2	перестановка столбцов матрицы согласно вектору транспозиции
3	умножение матрицы слева на ее транспонированную
4	формирование симметрической матрицы
5	формирование неособенной матрицы
6	формирование разреженной матрицы
7	перестановка строк матрицы согласно вектору транспозиции
8	формирование симметрической положительно определенной матрицы
9	вычислить m-норму матрицы
10	формирование ленточной матрицы
11	формирование кососимметрической матрицы
12	три способа получения эквивалентной матрицы
13	вычислить l – норму матрицы
14	формирование k – диагональной матрицы
15	формирование верхней треугольной матрицы и вычисление ее определителя

16	получение клеточной матрицы
17	формирование нижней треугольной матрицы и вычисление ее определителя
18	получение окаймленной матрицы
19	умножение матрицы справа на ее транспонированную
20	формирование квазидиагональной матрицы
21	сравнить две матрицы
22	вычислить след матрицы
23	сложение клеточных матриц
24	формирование особенной матрицы
25	умножение верхней треугольной матрицы на ее транспонированную
26	возведение матрицы в целую степень
27	умножение нижней треугольной матрицы на ее транспонированную
28	для прямоугольной матрицы вычислить *l*- норму
29	формирование положительно определенной симметрической матрицы
30	формирование неособенной матрицы

Пример выполнения задания по п. 2 (разработка кода)

Сформировать целочисленную матрицу и матрицу из вещественных чисел и найти их произведение.

1. Создайте новый проект командой *Файл/Новый/Приложение.*

2. Сохраните файлы модуля и проекта командой *Файл*/*Сохранить все* под именами LR3 и PR_LR3. Для этого удобно использовать

соответствующую быструю кнопку (*Сохранить все*). В последующих сеансах работы сохраненный проект можно открыть командой *Файл/Открыть проект* (или *Повторно открыть*). Теперь перейдем к проектированию приложения – переносам на форму необходимых компонентов и заданию их свойствам значений, а в обработчиках событий – размещению кодов соответствующих алгоритмов. (Рекомендуется нажимать кнопку *Сохранить все* по окончании работы с каждым компонентом.) В результате проектирования получим форму, представленную на рис. 3.6.

3. Выделите форму, щелкнув на ней левой кнопкой мыши, и в свойство **Caption** (надпись) впишите *АЛГЕБРА И ФОРМИРОВАНИЕ МАТРИЦ***.**

4. На форме размещены 12 меток **Label** (страница *Стандарт*), в свойство **Caption** (надпись) которых вписаны значения *размеры, диапазон чисел, число строк, число столбцов, макс значение, мин значение*; 4 кнопки **Button** (страница *Стандарт*), с надписями *целочисленная матрица, матрица из вещественных чисел, произведение матриц, конец.* Во всех перечисленных компонентах свойство **Font** имеет размер 10. Кроме того, на форме размещены три таблицы – компоненты **StringGrid1,2,3** (страница *Дополнительно*). Заданы следующие значения свойств всех компонентов **StringGrid**: **ColCount** – 5, **DefaultColWidth** -48, **FixedCols** – 0, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 5. Раскрыв свойство **Options** в компонентах **StringGrid1,2,** установите значение подсвойства **goEditing** – **true**, что даст возможность редактировать таблицы в **StringGrid1**,**2**. Далее, на форме размещены 8 компонентов ввода целых чисел – **CSpinEdit** (страница *Примеры*). Во всех компонентах шрифт – обычный, размер – 8. У компонентов, расположенных под метками *размеры*, занесите: в свойство **MaxValue** – 10, **MinValue** – 1, **Value** – 3. У компонентов, расположенных под метками *диапазон чисел* и напротив меток *макс значение* занесите в свойство **MaxValue** – 100, **MinValue** – 0, **Value** – 5,

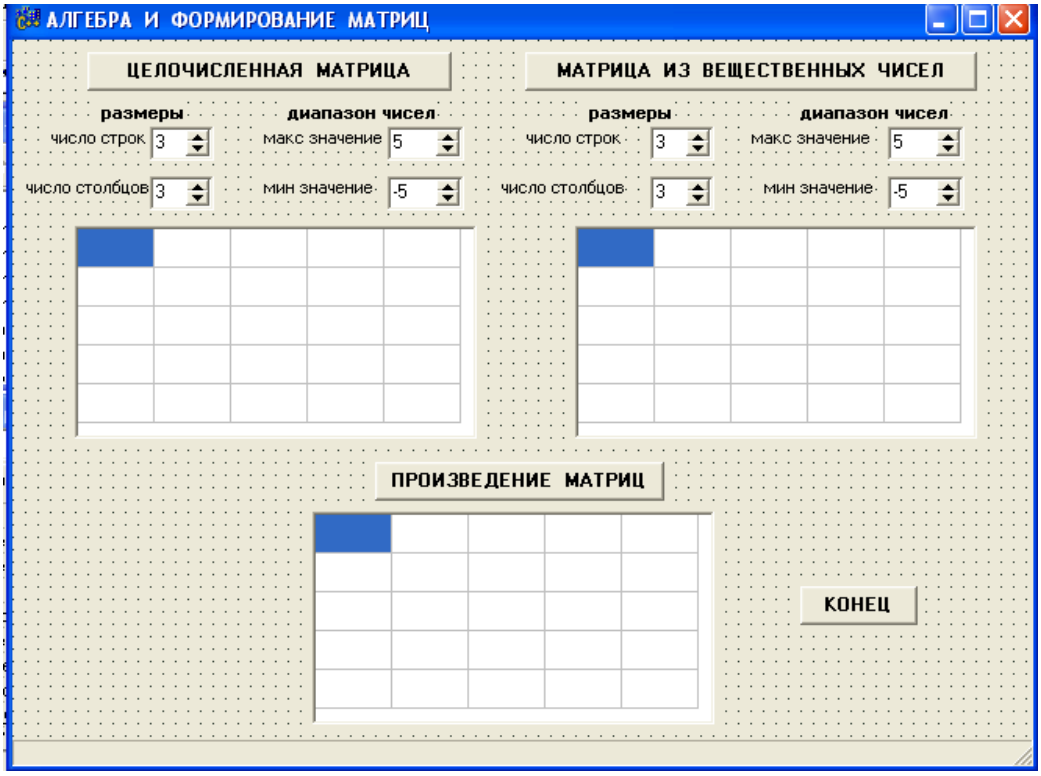


Рис. 3.6. Форма по окончании проектирования

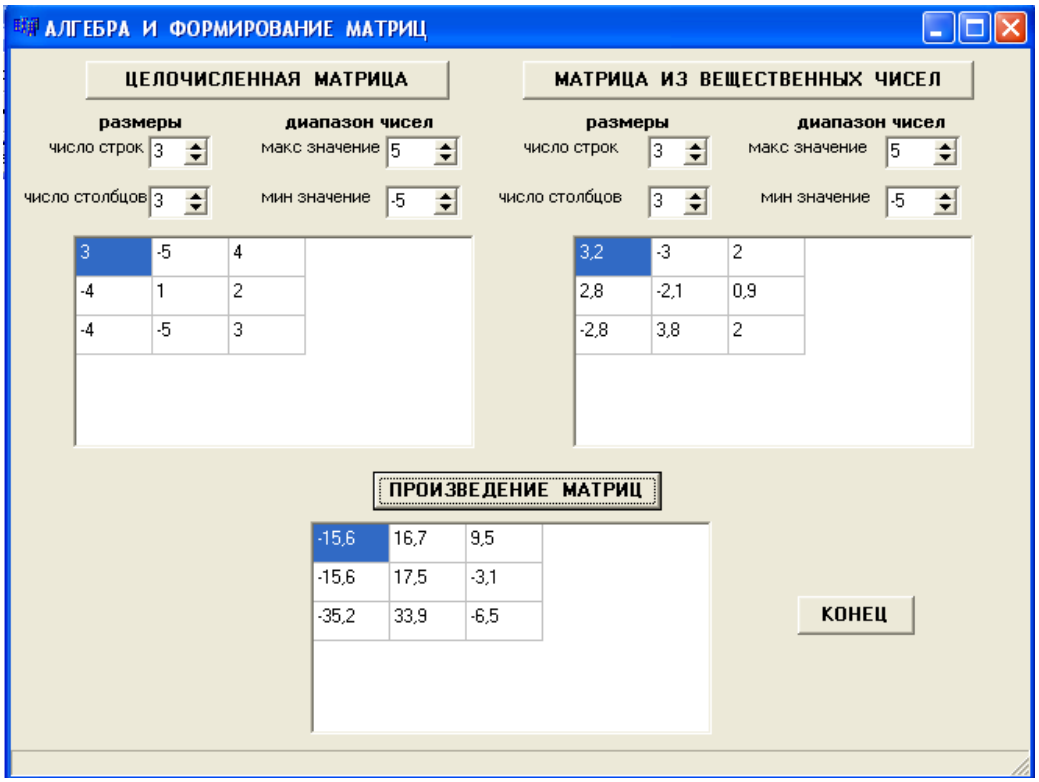


Рис. 3.7. Результат выполнения задания
с параметрами матриц по умолчанию

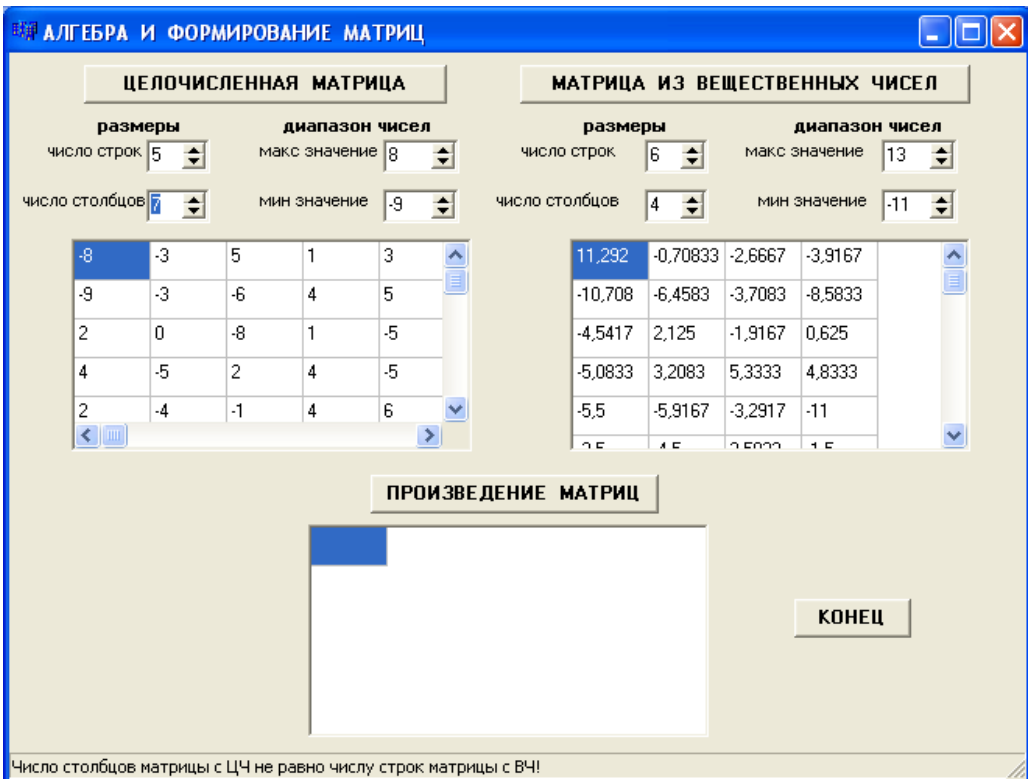


Рис. 3.8. Результат выполнения задания
с измененными параметрами матриц

а напротив меток *мин значение* – занесите в свойство **MaxValue** – 0, **MinValue** – -100, **Value** – -5. И, наконец, внизу формы размещен компонент **StatusBar1** (страница *Win32*) – представляет собой ряд панелей, отображающих полосу состояния. Поскольку в нашем случае достаточно одной панели (для одного сообщения), то установите свойство **SimplePanel** = **true**. Свойство **SimpleText** представляет собой текст, который задается во время проектирования или программно (в нашем случае программно).

5. Файл *LR3.cpp* содержит обработчики событий - щелчков на кнопках: **Button1** с надписью *целочисленная матрица,* **Button2** с надписью *матрица из вещественных чисел,* **Button3** с надписью *произведение матриц,* **Button4** с надписью *конец.*

/----------

```
#include <vcl.h>
#pragma hdrstop

#include "LR3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CSPIN"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 StatusBar1->SimpleText="";
 StringGrid3->RowCount=0;
 StringGrid3->ColCount=0;
 StringGrid3->Cells[0][0]="";
 StringGrid1->RowCount=CSpinEdit1->Value;
 StringGrid1->ColCount=CSpinEdit2->Value;
 for(int i=0;i<StringGrid1->RowCount;i++)
    for(int j=0;j<StringGrid1->ColCount;j++)
        StringGrid1->Cells[j][i]=IntToStr(
          random(CSpinEdit3->Value-CSpinEdit4->Value+1)
                +CSpinEdit4->Value);
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  StatusBar1->SimpleText="";
  StringGrid3->RowCount=0;
  StringGrid3->ColCount=0;
  StringGrid3->Cells[0][0]="";
  StringGrid2->RowCount=CSpinEdit5->Value;
  StringGrid2->ColCount=CSpinEdit6->Value;
  for(int i=0;i<StringGrid2->RowCount;i++)
     for(int j=0;j<StringGrid2->ColCount;j++){
        float z=random(CSpinEdit7->Value-CSpinEdit8->Value+1)
                          +CSpinEdit8->Value
              +(float)random(CSpinEdit7->Value-CSpinEdit8->Value+1)
                          /(CSpinEdit7->Value-CSpinEdit8->Value)-0.5;
        if(z>CSpinEdit7->Value)z=CSpinEdit7->Value;
        if(z<CSpinEdit8->Value)z=CSpinEdit8->Value;
        StringGrid2->Cells[j][i]=FloatToStrF(z,ffGeneral,5,5);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  if(StringGrid1->ColCount!=StringGrid2->RowCount){
       StatusBar1->SimpleText=
      "Число столбцов матрицы с ЦЧ не равно числу строк матрицы с ВЧ!";
      CSpinEdit2->SetFocus(); return;}
  StringGrid3->RowCount=CSpinEdit1->Value;
  StringGrid3->ColCount=CSpinEdit6->Value;
  for(int i=0;i<StringGrid1->RowCount;i++)
```

```
    for(int j=0;j<StringGrid2->ColCount;j++){
      float z=0;
      for(int k=0;k<StringGrid1->ColCount;k++)
         z+=StrToInt(StringGrid1->Cells[k][i])*
           StrToFloat(StringGrid2->Cells[j][k]);
      StringGrid3->Cells[j][i]=FloatToStrF(z,ffGeneral,5,5);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

6. Сохранив все, запустите приложение на выполнение с заданными во время проектирования параметрами матриц. Результат представлен на рис. 3.7. Чтобы убедиться в правильном перемножении матриц, сделайте одну из матриц единичной.

7. Измените параметры матриц подобно тому, как это сделано на рис. 3.8, и получите сообщение в строке состояния. Щелчком на кнопке *конец* перейдите из режима выполнения в режим проектирования приложения.

Содержание отчета

1. Задание.
2. Формулы с пояснениями.
3. Результат выполнения задания в *Mathcad'*е .
4. Тест для разработки алгоритма.
5. Блок-схема алгоритма с таблицей идентификаторов.
6. Исходный код.
7. Входные и выходные данные программы.
8. Библиографический список.

Контрольные вопросы

1. Как сформировать нижнюю и верхнюю треугольные матрицы?

2. Как сформировать неособенную и особенную квадратные матрицы?

3. Как сформировать неособенную квадратную матрицу с диагональным преобладанием для случаев:

1) $|a_{ii}| > |a_{i1}| + |a_{i2}| + ... + |a_{i,i-1}| + |a_{i,i+1}| + ... + |a_{in}|$ *для всех* i,

2) $|a_{ii}| > |a_{i1}| + |a_{i2}| + ... + |a_{i,i-1}| + |a_{i,i+1}| + ... + |a_{in}|$ *хотя бы для одного* i.

4. Приведите алгоритмы для вычисления $m-, l-, k-$норм матрицы.

5. Как определить ранг матрицы?

6. Приведите алгоритмы формирования вектора-строки и вектора-столбца.

7. Каким свойством обладает обратная матрица? Ответ подтвердите примером.

8. Как оцениваются точность и временные затраты обращения матрицы?

9. Как получить обратную матрицу?

10. Приведите алгоритм разбиения матрицы на клетки.

11. Как сформировать квазидиагональную матрицу?

12. Приведите алгоритм окаймления матрицы.

13. Приведите алгоритм формирования ленточной матрицы.

14. Как сформировать положительно определенную матрицу?

15. Как сформировать разреженную матрицу?

16. Как сформировать ортогональную матрицу?

17. Приведите алгоритм формирования вектора транспозиции.

18. Приведите примеры алгоритмов получения эквивалентных матриц.

19. Как разложить неособенную квадратную матрицу в произведение двух матриц? Каких матриц? Когда разложение будет единственно возможным?

20. Как можно разложить в произведение матриц положительно определенную матрицу?

21. Как можно разложить в произведение матриц ленточную положительно определенную матрицу?

22. Объясните обращение матриц методом окаймления.

23. Объясните обращение матриц методом Ершова.

24. Объясните обращение матриц методом Фаддеева.

ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ)

Краткие сведения

При решении системы уравнений n-го порядка $Ax = b$ характеристикой точности является *невязка*, равная

$$\delta = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(b_i - \overline{b_i}\right)^2}, \qquad \text{где } \overline{b} = A\overline{x} = \mathbf{A}\overline{x},\ \overline{x} \text{ - решение}$$

системы.

Прямые методы решений СЛАУ

Метод Гаусса

Методом Гаусса называют точный метод решения невырожденной системы линейных уравнений, состоящий в том, что последовательным исключением неизвестных систему

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, 2, \ldots, n,$$

приводят к эквивалентной системе с верхней треугольной матрицей

$$\begin{aligned} x_1 + c_{12}x_2 + \ldots\ldots\ldots + c_{1n}x_n &= d_1, \\ x_2 + \ldots\ldots\ldots + c_{2n}x_n &= d_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots &, \\ x_n &= d_n, \end{aligned}$$

решение которой находят по рекуррентным формулам

$$x_i = d_i - \sum_{k=i+1}^{n} c_{ik}x_k\ ,\ \ x_n = d_n, \quad i = n-1, n-2, \ldots,1\ .$$

Существует много вариантов этого метода. Рассмотрим схему единственного деления. Она эффективна, когда максимальные по модулю коэффициенты уравнений находятся на главной диагонали матрицы. Это положительно определенные матрицы и матрицы, обладающие свойством диагонального преобладания:

$$\sum_{j=1,\, j\neq i}^{n} |a_{ij}| < |a_{ii}|, \quad i = 1, 2, \ldots, n.$$

Пусть исходная система имеет вид

$$\begin{aligned} &a_{11}x_1 + \ldots\ldots\ldots + a_{1n}x_n = b_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \\ &a_{n1}x_1 + \ldots\ldots\ldots + a_{nn}x_n = b_n. \end{aligned} \qquad (1)$$

Предположим, что $a_{11} \neq 0$, и разделим обе части первого уравнения системы на a_{11}. В результате получим

$$x_1 + a_{12}^{(1)}x_2 + \ldots\ldots + a_{1n}^{(1)}x_n = b_1^{(1)}, \qquad (2)$$

где $a_{1j}^{(1)} = a_{1j}/a_{11}, \quad j = 2, 3, \ldots, n, \quad b_1^{(1)} = b_1/a_{11}$. С помощью уравнения (2) исключим во всех уравнениях системы (1), начиная со второго, слагаемые, содержащие x_1. Для этого умножаем обе части уравнения (2) последовательно на $a_{21}, a_{31}, \ldots, a_{n1}$ и вычитаем соответственно из второго, третьего, ..., n-го уравнения системы (1). В результате получаем систему, порядок которой на единицу меньше порядка исходной системы:

$$\begin{aligned} &a_{22}^{(1)}x_2 + \ldots\ldots\ldots + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \\ &a_{n2}^{(1)}x_2 + \ldots\ldots\ldots + a_{nn}^{(1)}x_n = b_n^{(1)}, \end{aligned}$$

где $a_{ij}^{(1)} = a_{ij} - a_{i1}a_{1j}^{(1)}, \quad i, j = 2, 3, \ldots, n, \quad b_i^{(1)} = b_i - a_{i1}b_1^{(1)}, \quad i = 2, 3, \ldots, n.$

Аналогично преобразуем полученную систему. В результате n-

кратного повторения этого преобразования получим систему с треугольной матрицей:

$$\begin{aligned} x_1 + c_{12}x_2 + c_{13}x_3 + \ldots\ldots\ldots + c_{1n}x_n &= d_1, \\ x_2 + c_{23}x_3 + \ldots\ldots\ldots + c_{2n}x_n &= d_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, & \\ x_n &= d_{n,} \end{aligned} \qquad (3)$$

которая эквивалентна системе (1) и легко решается. Найденное из последнего уравнения x_n подставляют в предпоследнее уравнение и находят x_{n-1}, затем x_{n-2}, и т.д. до x_1, которое находят из первого уравнения системы, когда уже известны $x_n, x_{n-1}, x_{n-2}, \ldots, x_2$.

Таким образом, метод Гаусса содержит прямой ход, на котором исходную систему преобразуют к треугольному виду, и обратный ход, на котором решают треугольную систему (3), эквивалентную исходной.

На рис. 4.1 и 4.2 представлены алгоритмы прямого хода и обратного хода при решении системы.

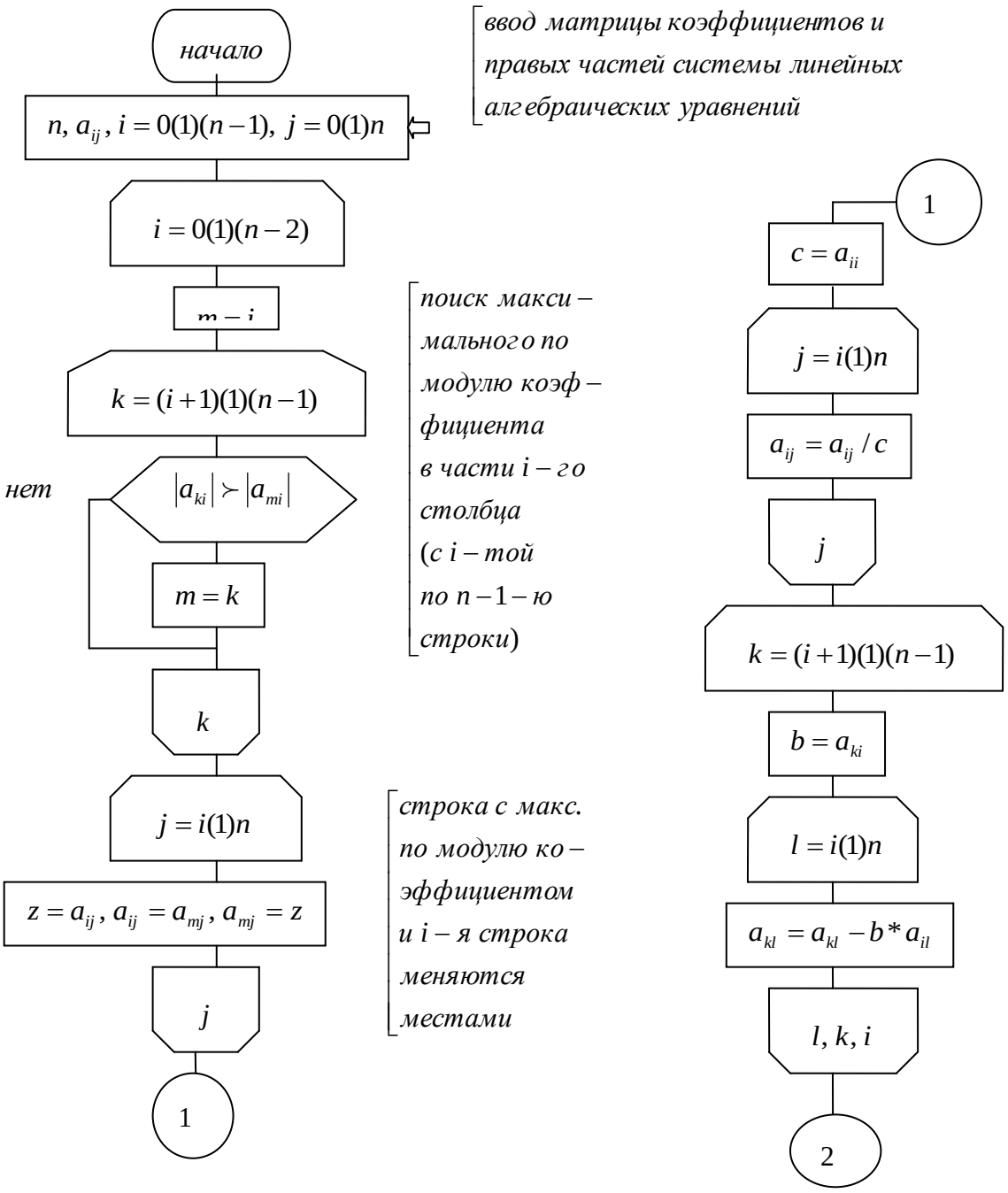


Рис. 4.1. Прямой ход алгоритма решения системы линейных алгебраических уравнений методом Гаусса по схеме с частичным выбором ведущего коэффициента по столбцу

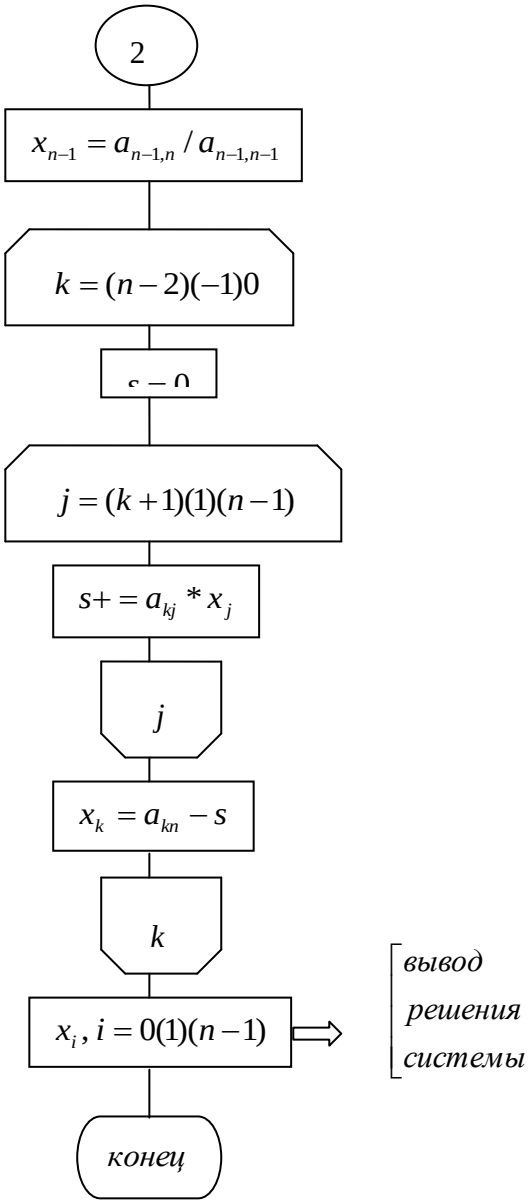


Рис. 4.2. Обратный ход алгоритма решения системы линейных алгебраических уравнений методом Гаусса по схеме с частичным выбором ведущего коэффициента по столбцу

Коэффициенты $a_{11}, a_{22}^{(1)}, a_{33}^{(2)}, \ldots$ называют ведущими (главными) элементами метода Гаусса. На каждом шаге предполагалось, что $a_{kk}^{(k-1)} \neq 0$. Если окажется, что это не так, то в качестве ведущего элемента можно использовать любой другой ненулевой элемент системы.

Однако, если коэффициент $a_{kk}^{(k-1)} \neq 0$ мал, то после деления на этот элемент и вычитания k-го уравнения из последующих возникают большие погрешности округления. Чтобы избежать этого, уравнения на каждом этапе переставляют так, чтобы на главной диагонали оказался наибольший по модулю элемент k-го столбца. При этом от схемы единственного деления переходят к схеме с частичным выбором (по столбцу) ведущего (главного) элемента, что и реализовано в алгоритме на рис. 4.1. Если матрица системы хорошо обусловлена (т.е. малые изменения ее элементов не приводят к существенным изменениям элементов ее обратной матрицы), то в методе Гаусса с выбором главного элемента погрешности округления невелики.

Одновременно с решением системы можно найти определитель матрицы системы, который равен произведению ведущих элементов, т. е. $a_{11}a_{22}^{(1)}a_{33}^{(2)}..a_{nn}^{(n-1)}$.

В схеме полного выбора (выбор главного элемента по всей матрице) допускается нарушение естественного порядка исключения неизвестных. На первом шаге среди a_{ij} определяют максимальный по модулю элемент $a_{i_1 j_1}$. Первое уравнение системы и уравнение с номером i_1 меняют местами. Далее исключают неизвестное x_{j_1} из всех уравнений, кроме первого. На k-м шаге среди $a_{ij}^{(k-1)}$ при неизвестных в уравнениях с номерами $i = k,..,n$ выбирают максимальный по модулю коэффициент $a_{i_k j_k}^{(k-1)}$. Затем k-е уравнение и уравнение, содержащее $a_{i_k j_k}^{(k-1)}$, меняют местами и исключают x_{j_k} из уравнений с номерами $i = k+1,...,n$. На этапе обратного хода неизвестные вычисляют в следующем порядке: $x_{j_n}, x_{j_{n-1}}, ..., x_{j_1}$.

Другой вариант метода Гаусса приводит к системе с обратной матрицей коэффициентов. Если $a_{11} \neq 0$, то можно разрешить первое из уравнений системы (1) относительно x_1 и подставить это выражение в остальные уравнения. В результате получим систему

$$\begin{aligned} x_1 &= a'_{11}b_1 + a'_{12}x_2 + \ldots\ldots\ldots + a'_{1n}x_n, \\ b_2 &= a'_{21}b_1 + a'_{22}x_2 + \ldots\ldots\ldots + a'_{2n}x_n, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \\ b_n &= a'_{n1}b_1 + a'_{n2}x_2 + \ldots\ldots\ldots + a'_{nn}x_n, \end{aligned} \qquad (4)$$

где $a'_{11} = 1/a_{11}, \quad a'_{1j} = -a_{1j}/a_{11}, \quad a'_{i1} = a_{i1}/a_{11}, \quad a'_{ij} = a_{ij} - a_{i1} * a_{1j}/a_{11},$ $i, j = 2, 3, \ldots, n$. На следующем шаге при условии, что $a'_{22} \neq 0$, исключают переменную x_2 из правых частей уравнений. Если возможно выполнить n таких шагов, то возникает система $x = A^{-1}b$, т.е. выполнено обращение исходной матрицы A.

Если на k-м шаге матрицу коэффициентов представить в виде блоков

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

то блок A_{11} размера $k * k$ определяет матрицу, обратную соответствующему блоку матрицы A, в то время как A_{22} есть матрица, обратная соответствующему блоку матрицы A^{-1}. Все это достаточно просто установить, если положить $x_{k+1} = x_{k+2} = \ldots. = x_n = 0$ в первых k уравнениях (1) и соответственно $b_1 = b_2 = \ldots = b_k = 0$ в последних $n - k$ уравнениях (4).

Метод Жордана-Гаусса

Если при реализации варианта с частичным выбором ведущего элемента позволить номеру строки i пробегать значения от 1 до $k - 1$ и от

$k+1$ до n, то все элементы k-го столбца, за исключением диагонального элемента, становятся нулями. Вместо верхней треугольной матрицы получают теперь в конечном результате единичную матрицу:

$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & 1 \end{bmatrix}.$$

Тем самым получение решения существенно упрощается: $x^T = (d_1', \dots, d_n')$. Тем не менее в совокупности этот метод требует больше операций и, следовательно, больших затрат машинного времени, чем метод Гаусса.

Метод ортогонализации строк

Пусть дана система линейных уравнений $Ax = b$ ($\det A \neq 0$). Преобразуем строки системы так, чтобы матрица A перешла в матрицу R с ортогональными строками. При этом вектор b перейдет в вектор β. В результате получим эквивалентную систему $Rx = \beta$, откуда $x = R^{-1}\beta$. Чтобы не вычислять обратную матрицу R^{-1}, воспользуемся свойством ортогональных матриц: $R^{-1} = R^T (RR^T)^{-1}$, *где* $RR^T = D; D$ – диагональная матрица. Поэтому $x = R^T D^{-1} \beta$.

Матрица D^{-1}, обратная диагональной, находится просто:

$$D = \begin{bmatrix} d_{11} & 0 & \dots & 0 \\ 0 & d_{22} & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & d_{nn} \end{bmatrix} \quad \Rightarrow \quad D^{-1} = \begin{bmatrix} d_{11}^{-1} & 0 & \dots & 0 \\ 0 & d_{22}^{-1} & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & d_{nn}^{-1} \end{bmatrix}$$

Следовательно, решение системы сводится в основном к нахождению матрицы R, которая может быть получена следующим образом.

Из каждой i-й строки системы $(i = 2, 3, \ldots, n)$ вычтем первую строку, умноженную на λ_{i1}. Получим матрицу $A^{(1)}$. Множители λ_{i1} должны быть такими, чтобы первая строка матрицы $A^{(1)}$ была ортогональна всем остальным строкам, т. е.

$$\lambda_{i1} = \left(\sum_{j=1}^{n} a_{1j} a_{ij} \right) / \sum_{j=1}^{n} a_{1j}{}^{2} .$$

Над матрицей $A^{(1)}$ проделываем аналогичную операцию: из каждой ее i-й строки $(i = 3, 4, \ldots, n)$ вычтем вторую строку $A^{(1)}$, умноженную на λ_{i2},

$$\lambda_{i2} = \left(\sum_{j=1}^{n} a_{2j}{}^{(1)} a_{ij}{}^{(1)} \right) / \sum_{j=1}^{n} (a_{2j}{}^{(1)})^{2} .$$

Получаем матрицу $A^{(2)}$ и т. д., пока не получится матрица $A^{(n-1)}$, все строки которой попарно ортогональны, т. е. матрицу R.

Систему $Ax = b$ можно решить и по-другому. Пусть она приведена к виду $Rx = \beta$, как было описано выше. Умножим каждое уравнение системы на

$$\mu_i = 1 / \sqrt{\sum_{j=1}^{n} r_{ij}{}^{2}} , \qquad i = 1, 2, \ldots, n .$$

Получим $\tilde{R}x = \tilde{\beta}$, где $\tilde{R}$ – ортогональная матрица. Поскольку у ортогональных матриц транспонированная матрица совпадает с обратной, то $x = \tilde{R}^{-1}\tilde{\beta} = \tilde{R}^{T}\tilde{\beta}$.

Метод решения системы с ленточными матрицами

Если A – положительно определенная ленточная матрица, такая, что $a_{ij} = 0$ при $|i - j| > m$, то существует действительная невырожденная треугольная матрица L, допускающая представление исходной матрицы в виде $LL^{T} = A$, где $l_{ij} = 0$, если $i - j > m$.

Элементы матрицы L можно определить по строкам, приравнивая элементы в обеих частях последнего уравнения. Если принять, что все элементы l_{pq} при $q \le 0$ и $q > p$ равны нулю, то элементы i-й строки удовлетворяют соотношениям

$$l_{ij} = (a_{ij} - \sum_{k=i-m}^{j-1} l_{ik} l_{jk}) / l_{jj} \ , \qquad j = i-m, \ldots, i-1;$$

$$l_{ii} = \sqrt{a_{ii-} \sum_{k=i-m}^{i-1} l_{ik}^{\ 2}} \ .$$

Решение системы уравнений $Ax = b$ осуществляется в два этапа:

$$Ly = b; \quad L^T x = y.$$

Учитывая ширину ленточной матрицы, получаем следующий алгоритм для решения системы уравнений:

$$y_i = (b_i - \sum_{k=i-m}^{i-1} l_{ik} y_k) / l_{ii}; \ i = 1, 2, \ldots, n;$$

$$x_i = (y_i - \sum_{k=i+1}^{i+m} l_{ki} x_k) / l_{ii}; \ i = n, n-1, \ldots, 1. \qquad (5)$$

Метод Холецкого

Дана СЛАУ $Ax = b$, где A – симметрическая положительно определенная матрица, для которой справедливо разложение $A = LDL^T$, где L – нижняя треугольная матрица с единичной диагональю; D – положительно определенная диагональная матрица.

Такое разложение может быть выполнено за n шагов, причем на i-м шаге определяют i-ю строку матрицы L и i-й элемент d_i матрицы D. Выражения для нахождения этих элементов имеют вид

$$l_{ij} d_j = a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_k l_{jk} \ , \ j = 1, 2, \ldots, i-1 \ ; \quad d_i = a_{ii} - \sum_{k=1}^{i-1} l_{ik} d_k l_{ik} \ .$$

После того, как матрицы L и D будут найдены, заменим исходную систему двумя эквивалентными ей системами $Ly = b, \quad L^T x = D^{-1} y.$

Эти уравнения можно решить, последовательно вычисляя величины

$$y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k \ , \ i = 1, 2, \ldots, n; \quad x_i = y_i / d_i - \sum_{k=i+1}^{n} l_{ki} x_k \ , \ i = n, n-1, \ldots, 1 \ .$$

Метод LU-разложения

В современных программах, реализующих метод Гаусса, вычисления разбивают на два основных этапа. Первый – вычисление LU-разложения матрицы системы, второй – обработка правых частей и вычисление решения.

Для проведения первого этапа не нужна информация о правой части СЛАУ, и поэтому он может быть выполнен независимо. Это этап предварительной подготовки к быстрому вычислению решения. Именно для получения LU-разложения производится основная масса вычислений ($\approx (2/3) n^3$ арифметических операций).

Итак, в результате выполнения первого этапа СЛАУ $Ax = b$ будет преобразована к виду $LUx = b.$

На втором этапе: 1) преобразуют b по формулам прямого хода, т. е. СЛАУ преобразуют к виду $Ux = L^{-1} b$; 2) с помощью обратной подстановки (обратный ход) решают полученную систему. Для непосредственного вычисления решения x на втором этапе требуется $\approx 2n^2$ арифметических операций.

Метод квадратного корня

Пусть требуется решить СЛАУ $Ax = b$ с симметрической положительно определенной матрицей A. Матрица A приводится к виду $A = LL^T$, где

$$L = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \quad \text{при } l_{ii} > 0, \quad i = 1, 2, \dots, n.$$

Найдем элементы матрицы L. Для этого вычислим элементы матрицы LL^T и приравняем их соответствующим элементам A. В результате получим систему уравнений

$$\begin{aligned}
& l_{11}^{\,2} = a_{11}, \\
& l_{i1} l_{11} = a_{i1}, \quad i = 2, 3, \dots, n, \\
& l_{21}^{\,2} + l_{22}^{\,2} = a_{22}, \\
& l_{i1} l_{21} + l_{i2} l_{22} = a_{i2}, \quad i = 3, 4, \dots, n, \\
& \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\
& l_{k1}^{\,2} + l_{k2}^{\,2} + \dots\dots + l_{kk}^{\,2} = a_{kk}, \\
& l_{i1} l_{k1} + l_{i2} l_{k2} + \dots\dots + l_{ik} l_{kk} = a_{ik}, \quad i = k+1, \dots, n, \\
& \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\
& l_{n1}^{\,2} + l_{n2}^{\,2} + \dots\dots\dots + l_{nn}^{\,2} = a_{nn}.
\end{aligned}$$

Решая эту систему, последовательно находим

$$l_{11} = \sqrt{a_{11}},$$
$$l_{i1} = a_{i1} / l_{11}, \quad i = 2, 3, \ldots, n,$$
$$l_{22} = \sqrt{a_{22} - l_{21}^{\ 2}},$$
$$l_{i2} = \left(a_{i2} - l_{i1} l_{21}\right) / l_{22}, \quad i = 3, 4, \ldots, n,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$l_{kk} = \sqrt{a_{kk} - l_{k1}^{\ 2} - l_{k2}^{\ 2} - \ldots - l_{k,k-1}^{\ 2}},$$
$$l_{ik} = \left(a_{ik} - l_{i1} l_{k1} - l_{i2} l_{k2} - \ldots - l_{i,k-1} l_{k,k-1}\right) / l_{kk}, \quad i = k+1, \ldots, n,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$l_{nn} = \sqrt{a_{nn} - l_{n1}^{\ 2} - l_{n2}^{\ 2} - \ldots - l_{n,n-1}^{\ 2}}.$$

После того, как матрица L будет найдена, заменим исходную систему двумя эквивалентными ей системами с треугольными матрицами $L^T y = b, \quad Lx = y$. Отсюда можно последовательно найти

$$y_i = \left(b_i - \sum_{k=1}^{i-1} l_{ki} y_k\right) / l_{ii}, \quad i = 1, 2, \ldots, n;$$
$$x_i = \left(y_i - \sum_{k=i+1}^{n} l_{ik} x_k\right) / l_{ii}, \quad i = n, n-1, \ldots, 1.$$

Метод прогонки

Это простой и эффективный алгоритм решения СЛАУ с трехдиагональными матрицами:

$$\begin{aligned}
b_1 x_1 + c_1 x_2 \qquad\qquad\qquad\qquad\qquad\qquad\qquad &= d_1, \\
a_2 x_1 + b_2 x_2 + c_2 x_3 \qquad\qquad\qquad\qquad\qquad &= d_2, \\
\ldots\ldots\ldots\ldots\ldots\ldots \qquad\qquad & \\
a_i x_{i-1} + b_i x_i + c_i x_{i+1} \qquad\qquad &= d_i, \qquad (6) \\
\ldots\ldots\ldots\ldots \qquad & \\
a_{n-1} x_{n-2} + b_{n-1} x_{n-1} + c_{n-1} x_n &= d_{n-1}, \\
a_n x_{n-1} + b_n x_n &= d_n.
\end{aligned}$$

Выведем расчетные формулы. Из первого уравнения системы (6) получим

$x_1 = \alpha_1 x_2 + \beta_1$, где $\alpha_1 = -c_1 / b_1$, $\beta_1 = d_1 / b_1$.

Подставим выражение для x_1 во второе уравнение системы (6) и получим

$$a_2(\alpha_1 x_2 + \beta_1) + b_2 x_2 + c_2 x_3 = d_2.$$

Преобразуем это уравнение к виду

$x_2 = \alpha_2 x_3 + \beta_2$, где $\alpha_2 = -c_2 / (b_2 + a_2\alpha_1)$, $\beta_2 = (d_2 - a_2\beta_1)/(b_2 + a_2\alpha_1)$.

Подставляем последнее выражение в третье уравнение и т.д. На i-м шаге этого процесса $(1 < i < n)$ уравнение системы преобразуется к виду

$$x_i = \alpha_i x_{i+1} + \beta_i, \qquad (7)$$

где $\alpha_i = -c_i / (b_i + a_i\alpha_{i-1})$, $\beta_i = (d_i - a_i\beta_{i-1})/(b_i + a_i\alpha_{i-1})$.

На n-м шаге подстановка в последнее уравнение выражения $x_{n-1} = \alpha_{n-1} x_n + \beta_{n-1}$ приведет к уравнению $a_n(\alpha_{n-1} x_n + \beta_{n-1}) + b_n x_n = d_n$. Отсюда $x_n = \beta_n = (d_n - a_n\beta_{n-1})/(b_n + a_n\alpha_{n-1})$. Значения остальных неизвестных x_i для $i = n-1, n-2, \ldots, 1$ вычисляются по формуле (7).

Алгоритм метода прогонки состоит из двух этапов. Прямой ход (прямая прогонка) состоит в вычислении прогоночных коэффициентов $\alpha_i (1 \le i \le n)$ и $\beta_i (1 \le i \le n)$. При $i = 1$ коэффициенты вычисляют по формулам: $\alpha_1 = -c_1 / \gamma_1$, $\beta_1 = d_1 / \gamma_1$, $\gamma_1 = b_1$, а при $i = 2, 3, \ldots, n-1$ - по рекуррентным формулам:

$$\alpha_i = -c_i / \gamma_i, \quad \beta_i = (d_i - a_i\beta_{i-1})/\gamma_i, \quad \gamma_i = b_i + a_i\alpha_{i-1}.$$

При $i = n$ прямая прогонка завершается вычислением

$$\beta_n = (d_n - a_n\beta_{n-1})/\gamma_n, \quad \gamma_n = b_n + a_n\alpha_{n-1}.$$

Обратный ход (обратная прогонка) дает значения неизвестных. Сначала полагают $x_n = \beta_n$. Затем значения остальных неизвестных вычисляют по формуле $x_i = \alpha_i x_{i-1} + \beta_i$, $i = n-1, n-2, \ldots, 1$.

Метод вращений

Прямой ход метода. На первом шаге исключают x_1 из всех уравнений СЛАУ (1), кроме первого. Для этого вычисляют $c_{12} = a_{11} / \sqrt{{a_{11}}^2 + {a_{21}}^2}$, $s_{12} = a_{21} / \sqrt{{a_{11}}^2 + {a_{21}}^2}$, имеющие свойства ${c_{12}}^2 + {s_{12}}^2 = 1,\ -s_{12}a_{11} + c_{12}a_{21} = 0$.

Затем первое уравнение системы заменяют линейной комбинацией первого и второго уравнений с коэффициентами c_{12} и s_{12}, а второе уравнение – аналогичной линейной комбинацией с коэффициентами $-s_{12}$ и c_{12}. В результате получаем систему

$$\begin{aligned}
&{a_{11}}^{(1)}x_1 + {a_{12}}^{(1)}x_2 + {a_{13}}^{(1)}x_3 + \ldots\ldots\ldots\ldots + {a_{1n}}^{(1)}x_n = {b_1}^{(1)},\\
&\qquad {a_{22}}^{(1)}x_2 + {a_{23}}^{(1)}x_3 + \ldots\ldots\ldots\ldots + {a_{2n}}^{(1)}x_n = {b_2}^{(1)},\\
&a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots\ldots\ldots\ldots\ldots\ldots + a_{3n}x_n = b_3,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots\ldots\ldots\ldots\ldots + a_{nn}x_n = b_n,
\end{aligned} \qquad (8)$$

в которой ${a_{1j}}^{(1)} = c_{12}a_{1j} + s_{12}a_{2j}$, ${a_{2j}}^{(1)} = -s_{12}a_{1j} + c_{12}a_{2j}$, $(1 \le j \le n)$, ${b_1}^{(1)} = c_{12}b_1 + s_{12}b_2$, ${b_2}^{(1)} = -s_{12}b_1 + c_{12}b_2$. Было ${a_{21}}^{(1)} = -s_{12}a_{11} + c_{12}a_{21} = 0$.

Если в исходной системе $a_{21} = 0$, то полагают $c_{12} = 1,\ s_{12} = 0$.

Выполненное преобразование эквивалентно повороту вектора x вокруг оси $0x_1x_2$ на угол φ_{12} такой, что $c_{12} = \cos\varphi_{12},\ s_{12} = \sin\varphi_{12}$.

Для исключения x_1 из третьего уравнения вычисляют

$$c_{13} = {a_{11}}^{(1)} / \sqrt{\left({a_{11}}^{(1)}\right)^2 + {a_{31}}^2},\ \ s_{13} = a_{31} / \sqrt{\left({a_{11}}^{(1)}\right)^2 + {a_{31}}^2},$$

причем ${c_{13}}^2 + {s_{13}}^2 = 1,\ -s_{13}{a_{11}}^{(1)} + c_{13}a_{31} = 0$.

Затем первое уравнение системы (8) заменяют линейной комбинацией первого и третьего уравнений с коэффициентами c_{13} и s_{13}, а третье уравнение – аналогичной комбинацией с коэффициентами $-s_{13}$ и c_{13}.

Таким же образом x_1 исключают из уравнений с номерами $i = 4, 5, \ldots, n$. В результате первого шага (состоит из $n-1$ малых шагов) система приводится к виду

$$\begin{aligned}
&a_{11}^{(n-1)}x_1 + a_{12}^{(n-1)}x_2 + a_{13}^{(n-1)}x_3 + \ldots\ldots + a_{1n}^{(n-1)}x_n = b_1^{(n-1)},\\
&a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots\ldots + a_{2n}^{(1)}x_n = b_2^{(1)},\\
&a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \ldots\ldots + a_{3n}^{(1)}x_n = b_3^{(1)}, \qquad (9)\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \ldots\ldots + a_{nn}^{(1)}x_n = b_n^{(1)}.
\end{aligned}$$

На втором шаге метода вращений, состоящем из $n-2$ малых шагов, из уравнений системы (9) с номерами $i = 3, 4, \ldots, n$ исключают x_2. Для этого каждое i-е уравнение комбинируют со вторым уравнением. В результате приходят к системе:

$$\begin{aligned}
&a_{11}^{(n-1)}x_1 + a_{12}^{(n-1)}x_2 + a_{13}^{(n-1)}x_3 + \ldots\ldots + a_{1n}^{(n-1)}x_n = b_1^{(n-1)},\\
&a_{22}^{(n-1)}x_2 + a_{23}^{(n-1)}x_3 + \ldots\ldots + a_{2n}^{(n-1)}x_n = b_2^{(n-1)},\\
&a_{33}^{(2)}x_3 + \ldots\ldots + a_{3n}^{(2)}x_n = b_3^{(2)},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n3}^{(2)}x_3 + \ldots\ldots + a_{nn}^{(2)}x_n = b_n^{(2)}.
\end{aligned}$$

После завершения $(n-1)$-го шага система примет вид

$$\begin{aligned}
&a_{11}^{(n-1)}x_1 + a_{12}^{(n-1)}x_2 + a_{13}^{(n-1)}x_3 + \ldots\ldots + a_{1n}^{(n-1)}x_n = b_1^{(n-1)},\\
&a_{22}^{(n-1)}x_2 + a_{23}^{(n-1)}x_3 + \ldots\ldots + a_{2n}^{(n-1)}x_n = b_2^{(n-1)},\\
&a_{33}^{(n-1)}x_3 + \ldots\ldots + a_{3n}^{(n-1)}x_n = b_3^{(n-1)},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{nn}^{(n-1)}x_n = b_n^{(n-1)}.
\end{aligned}$$

Обратный ход метода вращений – как в методе Гаусса.

Итерационные методы решений СЛАУ

Для систем средней размерности чаще используют прямые методы. Итерационные методы применяют для решения задач большой размерности, когда использование прямых затруднено из-за необходимости выполнения чрезмерно большого числа арифметических операций. Большие системы уравнений, как правило, бывают разреженными. Методы исключения приводят к тому, что большое число нулевых элементов превращаются в ненулевые, и матрица теряет свойство разреженности. А в ходе итерационного процесса матрица не меняется и остается разреженной. Большая эффективность итерационных методов по сравнению с прямыми методами связана с возможностью существенного использования разреженных матриц.

Метод простой итерации (метод Якоби)

СЛАУ $Ax = b$ необходимо предварительно преобразовать к виду $x = Bx + c$, где $B-$ квадратная матрица с элементами $b_{ij}(i, j = 1, 2, \ldots, n)$; $c-$ вектор-столбец с c_i $(i = 1, 2, .., n)$. Для приведения системы (1) к виду, удобному для итераций, исключим неизвестные из уравнений следующим образом:

$$
\begin{aligned}
x_1 &= \quad b_{12}x_2 + b_{13}x_3 + \ldots\ldots\ldots + b_{1,n-1}x_{n-1} + b_{1n}x_n + c_1,\\
x_2 &= b_{21}x_1 + \quad b_{23}x_3 + \ldots\ldots\ldots + b_{2,n-1}x_{n-1} + b_{2n}x_n + c_2,\\
x_3 &= b_{31}x_1 + b_{32}x_2 \quad + \ldots\ldots\ldots + b_{3,n-1}x_{n-1} + b_{3n}x_n + c_3,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n &= b_{n1}x_1 + b_{n2}x_2 + b_{n3}x_3 + \ldots\ldots\ldots + b_{n,n-1}x_{n-1} \quad + c_n.
\end{aligned}
\qquad (10)
$$

На главной диагонали матрицы B находятся нулевые элементы, а остальные – выражаются по формулам: $b_{ij} = -a_{ij}/a_{ii}$, $c_i = b_i/a_{ii}$, $(i, j = 1, 2, \ldots, n,\ j \neq i)$.

Суть метода Якоби состоит в следующем. Выберем начальное приближение $x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \ldots\ldots, x_n^{(0)}\right)^T$ и, подставив его в правую часть системы (10), найдем первое приближение $x^{(1)} = Bx^{(0)} + c$. Продолжая процесс далее, получим последовательность $x^{(0)}$, $x^{(1)}$, …, , $x^{(k)}$,… приближений, вычисляемых по формуле $x^{(k+1)} = Bx^{(k)} + c,\quad k = 0, 1, 2, \ldots,$ или в развернутом виде:

$$
\begin{aligned}
x_1^{(k+1)} &= b_{12}x_2^{(k)} + b_{13}x_3^{(k)} + \ldots\ldots\ldots + b_{1,n-1}x_{n-1}^{(k)} + b_{1n}x_n^{(k)} + c_1,\\
x_2^{(k+1)} &= b_{21}x_1^{(k)} + b_{23}x_3^{(k)} + \ldots\ldots\ldots + b_{2,n-1}x_{n-1}^{(k)} + b_{2n}x_n^{(k)} + c_2,\\
x_3^{(k+1)} &= b_{31}x_1^{(k)} + b_{32}x_2^{(k)} + \ldots\ldots\ldots + b_{3,n-1}x_{n-1}^{(k)} + b_{3n}x_n^{(k)} + c_3,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n^{(k+1)} &= b_{n1}x_1^{(k)} + b_{n2}x_2^{(k)} + b_{n3}x_3^{(k)} + \ldots\ldots\ldots + b_{n,n-1}x_{n-1}^{(k)} + c_n.
\end{aligned}
$$

Метод простой итерации сходится при условии диагонального преобладания:

$$\sum_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}| < |a_{jj}|, \quad j = 1,2,\ldots,n.$$

Метод Зейделя (метод Гаусса-Зейделя, процесс Либмана, метод последовательных замещений)

На k+1-й итерации компоненты приближения $x^{(k+1)}$ вычисляются по формулам:

$$
\begin{aligned}
x_1^{(k+1)} &= b_{12}x_2^{(k)} + b_{13}x_3^{(k)} + \ldots\ldots\ldots + b_{1,n-1}x_{n-1}^{(k)} + b_{1n}x_n^{(k)} + c_1,\\
x_2^{(k+1)} &= b_{21}x_1^{(k+1)} + b_{23}x_3^{(k)} + \ldots\ldots\ldots + b_{2,n-1}x_{n-1}^{(k)} + b_{2n}x_n^{(k)} + c_2,\\
x_3^{(k+1)} &= b_{31}x_1^{(k+1)} + b_{32}x_2^{(k+1)} + \ldots\ldots\ldots + b_{3,n-1}x_{n-1}^{(k)} + b_{3n}x_n^{(k)} + c_3,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n^{(k+1)} &= b_{n1}x_1^{(k+1)} + b_{n2}x_2^{(k+1)} + b_{n3}x_3^{(k+1)} + \ldots\ldots + b_{n,n-1}x_{n-1}^{(k+1} + c_n.
\end{aligned}
$$

Метод Якоби ориентирован на системы с матрицами, близкими к диагональным, а метод Зейделя – на системы с матрицами, близкими к нижним треугольным. Алгоритм метода Зейделя представлен на рис. 4.3.

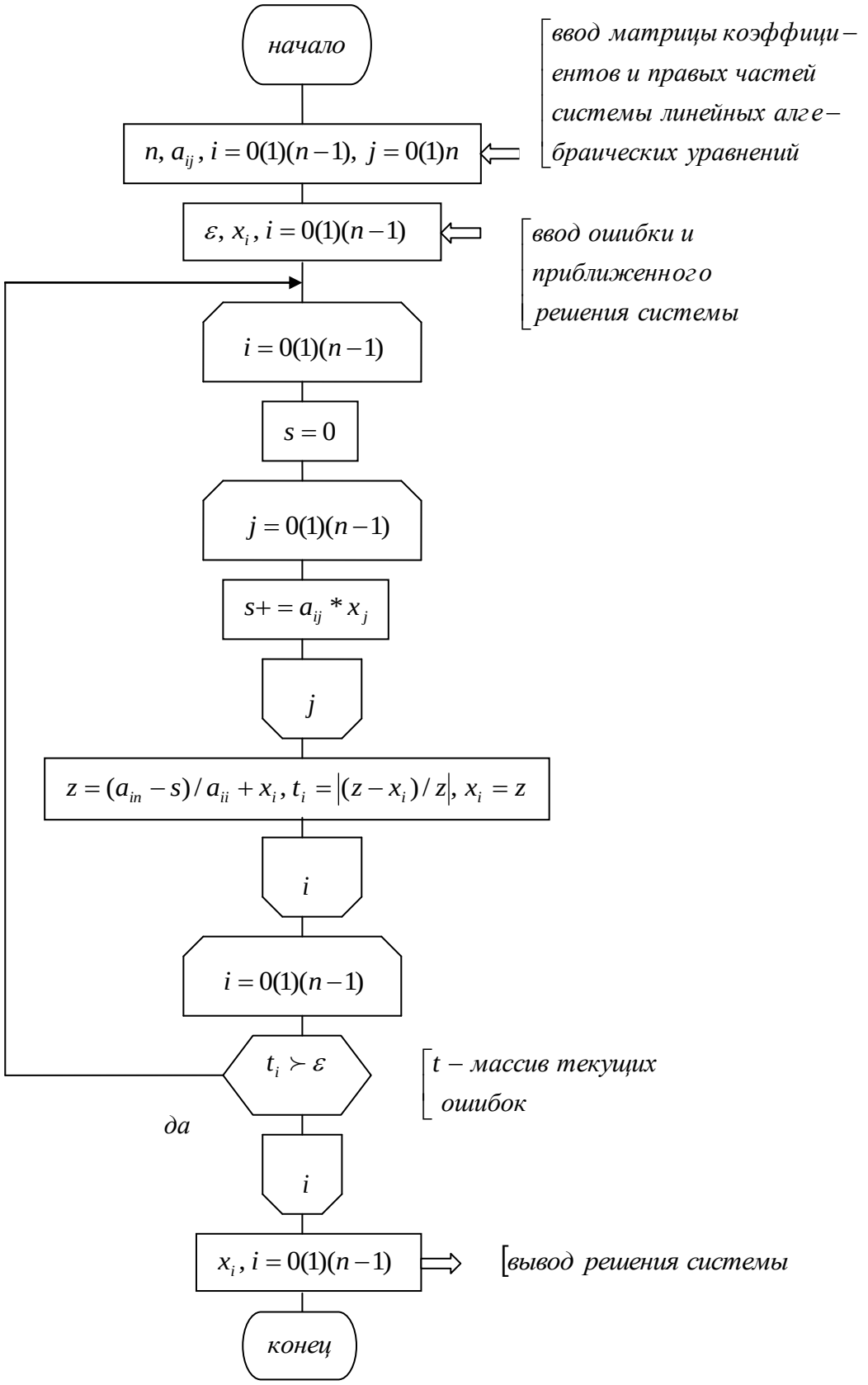


Рис. 4.3. Алгоритм итерационного метода Гаусса-Зейделя решения систем линейных алгебраических уравнений

Метод релаксации

Метод последовательной верхней релаксации является одним из наиболее эффективных и широко используемых итерационных методов для решения СЛАУ с симметрическими положительно определенными матрицами. После вычисления i-й компоненты $(k+1)$-го приближения по методу Гаусса-Зейделя

$$\tilde{x}_i^{(k+1)} = b_{i1}x_1^{(k+1)} + b_{i2}x_2^{(k+1)} + \ldots + b_{i,i-1}x_{i-1}^{(k+1)} + b_{i,i+1}x_{i+1}^{(k)} + \ldots + b_{in}x_n^{(k)} + c_i$$

производят дополнительное смещение этой компоненты на величину $(\omega-1)\left(\tilde{x}_i^{(k+1)} - x_i^{(k)}\right)$, где ω – параметр релаксации. Тогда

$$x_i^{(k+1)} = \tilde{x}_i^{(k+1)} + (\omega-1)\left(\tilde{x}_i^{(k+1)} - x_i^{(k)}\right) = \omega\tilde{x}_i^{(k+1)} + (1-\omega)x_i^{(k)}.$$

При $\omega = 1$ метод релаксации совпадает с методом Гаусса-Зейделя, при $\omega > 1$ - называют методом последовательной верхней релаксации, а при $\omega < 1$- нижней. Но часто для любых ω - методом последовательной верхней релаксации.

Если СЛАУ имеет симметрическую положительно определенную матрицу коэффициентов, то при любом ω $(0 < \omega < 2)$ метод релаксации сходится. Часто оказывается возможным выбрать (экспериментально) $\omega > 1$ так, чтобы метод релаксации сходился существенно быстрее, чем Якоби или Гаусса-Зейделя. Вариант метода релаксации – различные ω_i для вычисления различных компонент x_i очередного $(k+1)$-го приближения.

Вычисление матричных выражений

В матричных выражениях следует избегать вычисления обратных матриц, т. к. на это требуется большее время, чем на остальные вычисления.

Например, требуется вычислить $V = B^{-1}CA^{-1}WD^{-1}W$ при известных A, B, C, D, W.

Ошибочно вычислять сначала B^{-1}, A^{-1}, D^{-1}, а затем – по формуле.

Нужно вычислять с меньшими затратами машинного времени, в следующей последовательности.

Решая систему $Dx = W$, находят $x = D^{-1}W$, затем – $y = Wx$.

Решая систему $Az = y$, находят $z = A^{-1}y$, затем – $u = Cz$.

Решая систему $BV = u$, находят $V = B^{-1}u$.

Пример решения СЛАУ в системе Mathcad

$$A := \begin{pmatrix} 3.05 & 2.64 & 2.23 \\ 4.14 & 3.61 & 3.14 \\ 5.63 & 5.03 & 4.52 \end{pmatrix} \qquad B := \begin{pmatrix} 67.17 \\ 91.43 \\ 125.40 \end{pmatrix} \qquad X := A^{-1} \cdot B \qquad X = \begin{pmatrix} 14.783 \\ 8.056 \\ 0.365 \end{pmatrix}$$

ЛАБОРАТОРНАЯ РАБОТА 4

ИССЛЕДОВАНИЕ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ) МЕТОДОМ ГАУССА

ЗАДАНИЕ

1. Решить СЛАУ в *Mathcad'*е по заданному из таблицы варианту.

2. Составить алгоритм и написать код для решения СЛАУ методом Гаусса с частичным выбором ведущего коэффициента по столбцу. Заданный вариант использовать для тестирования.

3. Получить зависимости временных затрат от размера системы и усредненной точности решения от размера системы и типа представления (*float*, *double*, *long double*) коэффициентов.

Таблица

№ варианта	Матрица коэффициентов системы A			Столбец свободных членов b
1	1.84 2.32 1.83	2.25 2.60 2.06	2.53 2.82 2.24	-6.09 -6.98 -5.52
2	2.58 1.32 2.09	2.93 1.55 2.25	3.13 1.58 2.34	-6.66 -3.58 -5.01
3	2.18 2.17 3.15	2.44 2.31 3.22	2.49 2.49 3.17	-4.34 -3.91 -5.27
4	1.54 3.69 2.45	1.70 3.73 2.43	1.62 3.59 2.25	-1.97 -3.74 -2.26

5	1.53 2.35 3.83	1.61 2.31 3.73	1.43 2.07 3.45	-5.13 -3.69 -5.98
6	2.36 2.51 2.59	2.37 2.40 2.41	2.13 2.10 2.06	1.48 1.92 2.16
7	3.43 4.17 4.30	3.38 4.00 4.10	3.09 3.65 3.67	5.52 6.93 7.29
8	3.88 3.00 2.67	3.78 2.79 2.37	3.45 2.39 1.96	10.41 8.36 7.62
9	3.40 2.64 4.64	3.26 2.39 4.32	2.90 1.96 3.85	13.05 10.30 17.89
10	2.53 3.95 2.78	2.36 4.11 2.43	1.93 3.66 1.94	12.66 21.97 13.93
11	2.16 3.55 4.85	1.96 3.23 4.47	1.56 2.78 3.97	13.16 21.73 29.75
12	2.69 2.73 2.93	2.47 2.39 2.52	2.07 1.92 2.02	19.37 19.43 20.80
13	3.72 4.47 4.96	3.47 4.10 4.53	3.06 3.63 4.01	30.74 36.80 40.79

	4.35	4.39	3.67	40.15
14	4.04	3.65	3.17	36.82
	3.14	2.69	2.17	28.10
	4.07	3.79	3.37	40.77
15	2.84	2.44	1.95	27.68
	4.99	4.50	3.97	49.37
	3.19	2.89	2.47	33.91
16	4.43	4.02	3.53	47.21
	3.40	2.92	2.40	32.92
	2.57	2.26	1.84	28.66
17	4.47	4.03	3.57	50.27
	4.89	4.40	3.87	55.03
	2.83	2.50	2.08	33.28
18	3.00	2.55	2.07	33.59
	3.72	3.21	2.68	43.43
	3.78	3.44	3.02	46.81
19	4.33	3.88	3.39	53.43
	4.76	4.24	3.72	58.73
	4.59	4.24	3.82	59.54
20	4.83	4.36	3.88	62.33
	4.06	3.53	3.01	52.11
	4.56	4.20	3.78	61.86
21	3.21	2.73	2.25	42.98
	4.58	4.04	3.52	61.67
	3.75	3.39	2.97	53.38
22	4.18	3.70	3.22	59.28
	4.43	3.88	3.36	62.62

23	2.95 5.11 4.38	2.58 4.62 3.82	2.16 4.14 3.30	44.16 46.68 65.34
24	2.93 3.47 4.78	2.55 2.98 4.22	2.14 2.50 3.70	46.41 54.78 75.81
25	3.74 4.02 4.18	3.36 3.51 3.61	2.94 3.04 3.09	63.26 67.51 70.03
26	4.67 5.30 5.11	4.28 4.79 4.54	3.87 4.32 4.03	84.43 95.45 91.69
27	4.90 3.79 4.01	4.50 3.27 3.43	4.09 2.81 2.91	94.18 71.57 75.45
28	4.25 3.86 5.40	3.84 3.34 4.82	3.43 2.87 4.30	86.07 77.12 108.97
29	3.35 5.41 3.88	2.94 4.88 3.30	2.53 4.41 2.78	70.69 115.38 81.07
30	3.05 4.14 5.63	2.64 3.61 5.03	2.23 3.14 4.52	67.17 91.43 125.40

1. Для проектирования приложения логично использовать три формы согласно пунктам задания.

2. Создайте новый проект командой *Файл/Новый/Приложение.*

3. Сохраните файлы модуля и проекта командой *Файл*/*Сохранить все* под именами LR4_1 и PR_LR4. Командой *Файл/Новый/Форма* добавьте к проекту две формы, сохранив их модули под именами LR4_2 и LR4_3. Теперь перейдем к проектированию приложения. В результате проектирования получим формы, представленные на рис. 4.4, 4.5, 4.6. В свойство **Caption** форм впишите соответствующие заголовки.

4. Начните проектирование с первой формы (рис. 4.4). (Для перехода с одной формы на другую используйте быструю кнопку *Вид формы*, для перехода с модуля на модуль – *Вид модуля*, а с формы на модуль – *Переключатель Форма/Модуль*.) Перенесите на форму 3 кнопки (страница *Стандарт*) с надписями соответственно: **Button1 –** *ТЕСТИРУЕМАЯ СЛАУ 3-ГО ПОРЯДКА*, **Button2 -** *РЕШЕНИЕ*, **Button3 –** *ТЕСТИРОВАНИЕ СЛАУ n – ПОРЯДКА*. Для размещения системы перенесите на форму компонент **StringGrid1** (страница *Дополнительно*). Установите следующие значения свойств компонента **StringGrid1**: **ColCount** – 8, **DefaultColWidth** -40, **FixedCols** – 0, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 3. Для вывода на экран результатов тестирования перенесите на форму компонент **StringGrid2** и три метки: **LabeledEdit1, LabeledEdit2** и **LabeledEdit3** (страница *Дополнительно*). Установите следующие значения свойств компонента **StringGrid2**: **ColCount** – 2, **DefaultColWidth** -48, **FixedCols** – 1, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 3. Для всех трех меток свойству **LabelPosition** присвойте значение **lpLeft** (из выпадающего списка), а свойству **Font** – обычный, размер 8. Раскрывая свойство **EditLabel**, для всех трех меток установите подсвойство **Font** – черный, жирный, размер 8, а в подсвойстве **Caption** впишите соответственно **невязка**, **кол-во умн+дел**, **кол-во сл+выч**.

Рис. 4.4. Первая форма по окончании проектирования

5. Для выполнения п.1 задания файл *LR4_1.cpp* может быть таким:

```
#include <vcl.h>
#pragma hdrstop

#include "LR4_1.h"
#include "LR4_2.h"
#include <math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CSPIN"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
const int n=3;
```

```
const float a[][4]={{3.05,2.64,2.23,67.17},{4.14,3.61,3.14,91.43},
            {5.63,5.03,4.52,125.40}};
float x[3];
void __fastcall TForm1::Button1Click(TObject *Sender)
{AnsiString s;
 for(int j=0;j<3;j++){
   if(j<2)s=" + ";
     else s="";
   for(int i=0;i<3;i++){
   StringGrid1->Cells[2*j][i]=FloatToStrF(a[i][j],ffFixed,5,2)
                          +"  * ";
   StringGrid1->Cells[2*j+1][i]="  x["+IntToStr(j)+"]"+s;}
 for(int i=0;i<3;i++){
   StringGrid1->Cells[6][i]="    = ";
   StringGrid1->Cells[7][i]=FloatToStrF(a[i][3],ffFixed,5,2);}}
}
//---------------------------------------------------------------------------
 void __fastcall gauss(float& nev,int& mul,int& ad)
 { int i,j,k,l;
   float a1[3][4],c,b,s;
//копирование исходной матрицы во вспомогательную матрицу
   for(i=0;i<n;i++)
      for(j=0;j<n+1;j++) a1[i][j]=a[i][j];
//решение СЛАУ
   mul=ad=0;
   for(i=0;i<n-1;i++){
      c=a1[i][i];
      for(j=i;j<n+1;j++) {a1[i][j]/=c; mul++;}
      for(k=i+1;k<n;k++){
```

```
        b=a1[k][i];
        for(l=i;l<n+1;l++)
           {a1[k][l]-=b*a1[i][l]; ad++; mul++;}}}
   x[n-1]=a1[n-1][n]/a1[n-1][n-1]; mul++;
   for(k=n-2;k>=0;k--){
     s=0;
     for(j=k+1;j<n;j++)
         {s+=a1[k][j]*x[j]; ad++; mul++;}
     x[k]=a1[k][n]-s; ad++;}
//вычисление невязки
   c=0;
   for(int i=0;i<n;i++){
     b=0;
     for(int j=0;j<n;j++) b+=a[i][j]*x[j];
     c+=pow(b-a[i][n],2);}
   nev=sqrt(c)/n;
 }
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{int mul,ad;
 float nev;
 gauss(nev,mul,ad);
 for(int i=0;i<3;i++){
      StringGrid2->Cells[0][i]="  x["+IntToStr(i)+"]   = ";
      StringGrid2->Cells[1][i]=FloatToStrF(x[i],ffFixed,7,5);}
 LabeledEdit1->Text=FloatToStrF(nev,ffExponent,4,2);
 LabeledEdit2->Text=IntToStr(mul);
 LabeledEdit3->Text=IntToStr(ad);
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
 Form2->Show();
 Form2->StringGrid1->Cells[0][0]="   n";
 Form2->StringGrid1->Cells[1][0]="   k";
 Form2->StringGrid1->Cells[2][0]="  mul";
 Form2->StringGrid1->Cells[3][0]="   ad";
 Form2->StringGrid1->Cells[4][0]="  nev_f";
 Form2->StringGrid1->Cells[5][0]="  nev_d";
 Form2->StringGrid1->Cells[6][0]="  nev_ld";
}
//---------------------------------------------------------------------------
```

6. Перенесите на 2-ю форму (рис. 4.5) 11 меток **Label** (страница *Стандарт*), в свойство **Caption** (надпись) которых в порядке их номеров впишите значения *порядок СЛАУ, мин, макс, шаг, кол-во СЛАУ, диапазон коэфф-тов, макс, мин, диапазон правых частей, макс, мин*; 3 кнопки **Button** (страница *Стандарт*), с надписями *решение, график, сброс*. Во всех компонентах на форме шрифт имеет размер 8. Для задания параметров СЛАУ перенесите на форму 8 компонентов ввода целых чисел – **CSpinEdit** (страница *Примеры*). Свойства **MaxValue**, **MinValue**, **Value** установите соответственно: **CSpinEdit1** (*порядок СЛАУ, мин*) – 100,1,3; **CSpinEdit2** (*порядок СЛАУ, макс*) – 1000,2,10; **CSpinEdit3** (*порядок СЛАУ, шаг*) – 100,1,1; **CSpinEdit4** (*кол-во СЛАУ*) – 100,1,1; **CSpinEdit5** (*диапазон коэфф-тов, макс)* – 10,0,5; **CSpinEdit6** (*диапазон коэфф-тов, мин*) – 0,-10,-5; **CSpinEdit7** (*диапазон правых частей, макс*) – 100,1,20; **CSpinEdit8** (*диапазон правых частей, мин*) – 0,-100,-20. Для вывода результатов решения СЛАУ в таблицу перенесите на форму компонент **StringGrid1** (страница *Дополнительно*). Установите

следующие значения свойств компонента **StringGrid1**: **ColCount** – 7, **DefaultColWidth** -48, **FixedCols** – 0, **FixedRows** – 1, **Font** – черный, обычный, размер 8, **RowCount** – 100. Для отображения процесса решения СЛАУ перенесите в нижнюю часть формы компонент **ProgressBar1** (страница *Win32*).

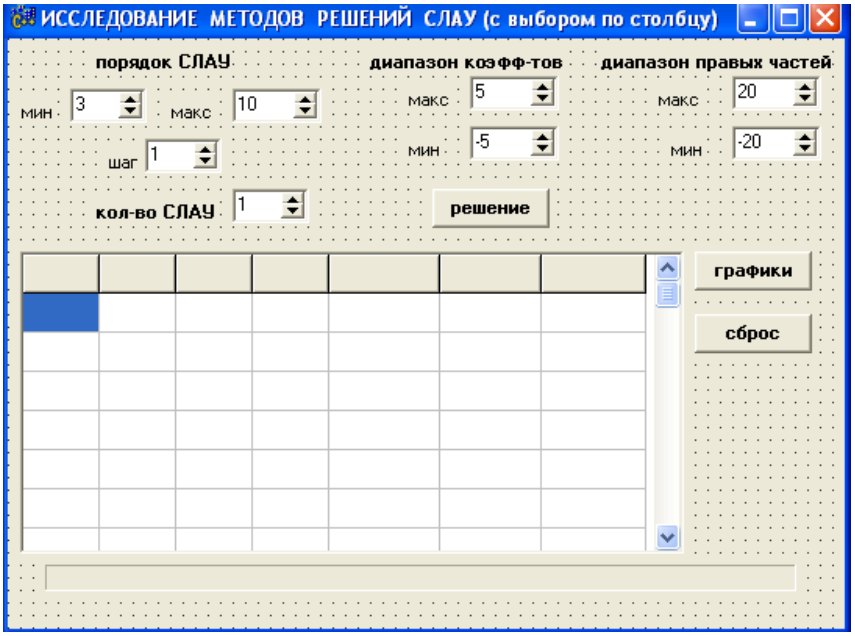


Рис. 4.5. Вторая форма по окончании проектирования

7. Для выполнения п.1 задания файл *LR4_2.cpp* может быть таким:

```
#include <vcl.h>
#pragma hdrstop

#include "LR4_2.h"
#include "LR4_3.h"
#include "LR4_1.h"
#include <math.h>
#include <time.h>
#include <stdlib.h>
//---------------------------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma link "CSPIN"
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
template<class T>
void __fastcall gauss_1(float**b,int n,T& nev,int& mul,int& ad)
{   T c,z,d,s;
    int max;
    T* x=new T[n];
    T** a=new T*[n];
    for(int i=0;i<n;i++) a[i]=new T[n+1];
    for(int i=0;i<n;i++)
       for(int j=0;j<n+1;j++)
          a[i][j]=b[i][j];
    mul=ad=0;
    for(int i=0;i<n-1;i++){
      max=i;
      for(int k=i+1;k<n;k++)
        if(fabs(a[k][i])>fabs(a[max][i])) max=k;
      for(int j=i;j<n+1;j++)
         {z=a[i][j]; a[i][j]=a[max][j]; a[max][j]=z;}
      c=a[i][i];
      for(int j=i;j<n+1;j++) {a[i][j]/=c; mul++;}
```

```
      for(int k=i+1;k<n;k++){
         d=a[k][i];
         for(int l=i;l<n+1;l++)
           {a[k][l]-=d*a[i][l]; mul++; ad++;}}}
    x[n-1]=a[n-1][n]/a[n-1][n-1]; mul++;
    for(int k=n-2;k>=0;k--){
      s=0;
      for(int j=k+1;j<n;j++) {s+=a[k][j]*x[j]; mul++; ad++;}
      x[k]=a[k][n]-s; ad++;}
//вычисление невязки
   c=0;
   for(int i=0;i<n;i++){
     d=0;
     for(int j=0;j<n;j++) d+=b[i][j]*x[j];
     c+=pow(d-b[i][n],2);}
   nev=sqrt(c)/n;
    for(int i=0;i<n;i++) delete[]a[i];
     delete[]a; a=0;
     delete[]x; x=0;
}
//---------------------------------------------------------------------------
int pk=1;
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
  Form3->Series1->Clear();
  Form3->Series2->Clear();
  Form3->Series3->Clear();
  Form3->Series4->Clear();
```

```
  Form3->Series5->Clear();
  int count=0, current=0;
  for(int n=CSpinEdit1->Value; n<=CSpinEdit2->Value;
                n+=CSpinEdit3->Value)
    count+=n; count*=CSpinEdit4->Value;
  float**b,w,z;
  int mul,smul,ad,sad;
  float nev_f,snev_f;
  double nev_d,snev_d;
  long double nev_ld,snev_ld;
  int k=CSpinEdit4->Value;
  for(int n=CSpinEdit1->Value;
      n<=CSpinEdit2->Value;n+=CSpinEdit3->Value){
    b=new float*[n];
    for(int i=0;i<n;i++)
      b[i]=new float[n+1];
    smul=0; sad=0; snev_f=0; snev_d=0; snev_ld=0;
    for(int kc=0;kc<k;kc++){
       srand(time(NULL));
      for(int i=0;i<n;i++){
        w=rand()%(CSpinEdit7->Value-CSpinEdit8->Value+1)+CSpinEdit8-
>Value
            +(float)random(CSpinEdit7->Value-CSpinEdit8->Value+1)
                        /(CSpinEdit7->Value-CSpinEdit8->Value)-0.5;
      if(w>CSpinEdit7->Value)w=CSpinEdit7->Value;
      if(w<CSpinEdit8->Value)w=CSpinEdit8->Value;
      b[i][n]=w;
      for(int j=0;j<n;j++){
```

```
      z=rand()%(CSpinEdit5->Value-CSpinEdit6->Value+1)+CSpinEdit6-
>Value
            +(float)random(CSpinEdit5->Value-CSpinEdit6->Value+1)
                  /(CSpinEdit5->Value-CSpinEdit6->Value)-0.5;
      if(z>CSpinEdit5->Value)z=CSpinEdit5->Value;
      if(z<CSpinEdit6->Value)z=CSpinEdit6->Value;
      b[i][j]=z;      }}
   gauss_1(b,n,nev_f,mul,ad);
   snev_f+=nev_f; smul+=mul; sad+=ad;
   gauss_1(b,n,nev_d,mul,ad);
   snev_d+=nev_d;
   gauss_1(b,n,nev_ld,mul,ad);
   snev_ld+=nev_ld;
   current+=n;
   ProgressBar1->Position=100*current/count;    }
   StringGrid1->Cells[0][pk]=IntToStr(n);
   StringGrid1->Cells[1][pk]=IntToStr(k);
   StringGrid1->Cells[2][pk]=IntToStr(smul);
   StringGrid1->Cells[3][pk]=IntToStr(sad);
   StringGrid1->Cells[4][pk]=FloatToStrF(snev_f/k,ffExponent,5,4);
   StringGrid1->Cells[5][pk]=FloatToStrF(snev_d/k,ffExponent,5,4);
   StringGrid1->Cells[6][pk]=FloatToStrF(snev_ld/k,ffExponent,5,4);
      Form3->Series1->AddXY(n,smul,"",clBlack);
      Form3->Series2->AddXY(n,sad,"",clBlack);
      Form3->Series3->AddXY(n,snev_f/k*1.0e6,"",clBlack);
      Form3->Series4->AddXY(n,snev_d/k*1.0e15,"",clBlack);
      Form3->Series5->AddXY(n,snev_ld/k*1.0e18,"",clBlack);
   pk++;
   for(int i=0;i<n;i++)delete[] b[i];
```

```
   delete[]b; b=0;    }
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button2Click(TObject *Sender)
{
 for(int i=1;i<StringGrid1->RowCount;i++)
   for(int j=0;j<StringGrid1->ColCount;j++)
     StringGrid1->Cells[j][i]="";
 ProgressBar1->Position=0;
 Form3->Series1->Clear();
 Form3->Series2->Clear();
 Form3->Series3->Clear();
 Form3->Series4->Clear();
 Form3->Series5->Clear();
 pk=1;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button3Click(TObject *Sender)
{
  Form3->Show();
}
//---------------------------------------------------
```

8. Для графического представления результатов исследования решения СЛАУ методом Гаусса с выбором ведущего коэффициента по столбцу на форму 3 перенесите (рис. 4.6) компоненты **Chart1** и **Chart2** (страница *Additional*). Зададим свойства компонента **Chart1**. Щелкните правой кнопкой мыши на компоненте **Chart1** и в появившемся меню выберите *Edit Chart*…. На экране появится окно *Редактора Диаграмм* (*Editing Chart1*) с открытой страницей *Chart*, которая имеет несколько закладок. В

данный момент открыта закладка *Series*. Щелкните на кнопке *Add…* – добавить серию. В появившемся окне выберите тип графика – *Line* и выключите индикатор *3D*. Кнопкой *Title…* вызовите *Change Series Title*, где дайте заголовок первому графику – *кол-во умн+дел*. Действуя подобным образом, добавьте *Series 2* и дайте заголовок второму графику – *кол-во сл+выч*. На вкладке *Legend* в группе кнопок *Position* нажмите кнопку *Top*. Цвет фона *Back Color*.. установите белый. На закладке *Panel*, нажав кнопку *Panel Color…*, выберите белый цвет. Перейдите на закладку *Titles*. В окне редактирования, которое в данный момент соответствует *Title* – заголовку графика, сотрите *TChart* и напишите (шрифт *Font…* - черный, жирный, размер 8) *ЗАВИСИМОСТЬ ВРЕМ ЗАТРАТ ОТ ПОРЯДКА СЛАУ*. Цвет фона *Back Color*.. установите белый. В выпадающем списке от окна редактирования *Title* перейдите в окно редактирования *Foot* и напишите тем же шрифтом *n* – *ПОРЯДОК СЛАУ*. В группе кнопок *Alignment* нажмите кнопку *Right*. Цвет фона *Back Color*.. также установите белый. Перейдите со страницы *Chart* на страницу *Series*. Здесь на закладке *Format* задайте параметры линий графиков. Требуемый график (*кол-во умн+дел, кол-во сл+выч*) выбирается из выпадающего списка. Для графика *кол-во сл+выч* в группе *Line* нажмите кнопку *Border…* и в группе кнопок *Style* включите кнопку *Dot*, а остальные значения параметров графиков оставьте заданными по умолчанию. Нажмите кнопку *Close* и выйдите из режима редактирования компонента **Chart1.**

9. Изложенную последовательность действий используйте и для компонента **Chart2**. Отличия: 3 графика – *невязка (float)*1.0e6* (толщина линии *Width* – 2), *невязка (double)*1.0e15* (толщина линии *Width* – 1, стиль *Style – solid*), *невязка (long double)*1.0e18* (толщина линии *Width* – 1, стиль *Style – Dot*). Кроме того, на этой форме следует разместить 2 кнопки **Button** (страница *Стандарт*), с надписями *таблица, конец*. Кнопка

таблица нужна для возврата на вторую форму, а кнопка *конец* – для перехода от выполнения приложения к его редактированию.

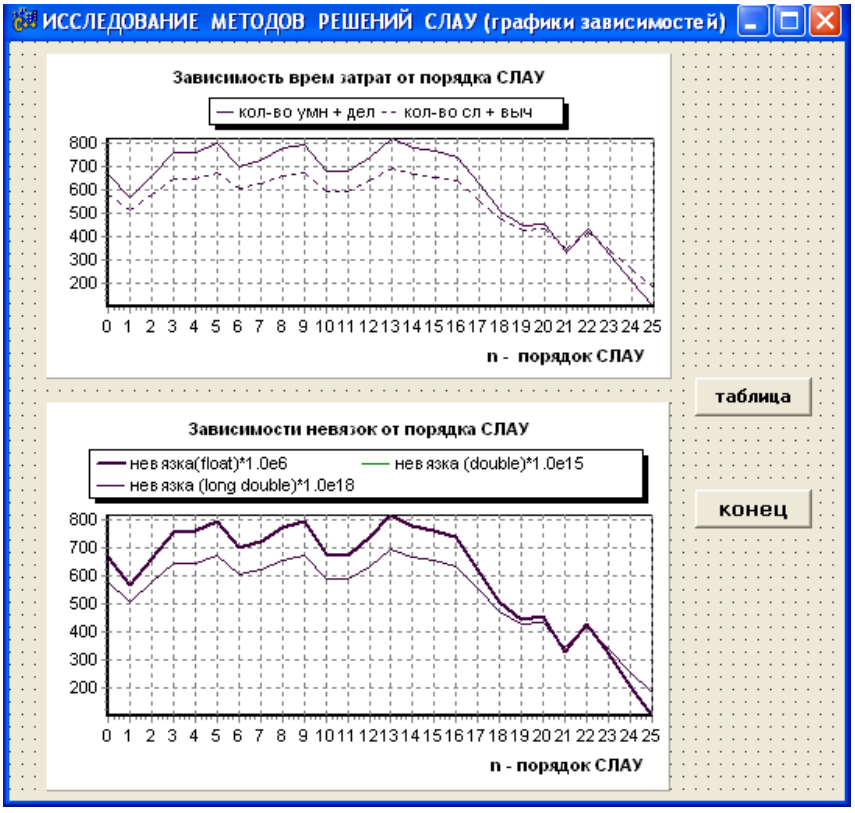


Рис. 4.6. Третья форма по окончании проектирования

10. Файл *LR4_3.cpp* имеет вид:

```
#include <vcl.h>
#pragma hdrstop

#include "LR4_3.h"
#include "LR4_2.h"
#include "LR4_1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
```

```
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button1Click(TObject *Sender)
{
  Form2->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button2Click(TObject *Sender)
{
  Close();
  Form2->Close();
  Form1->Close();
}
//---------------------------------------------------------------------------
```

Рис. 4.7. Результаты тестирования на первой форме

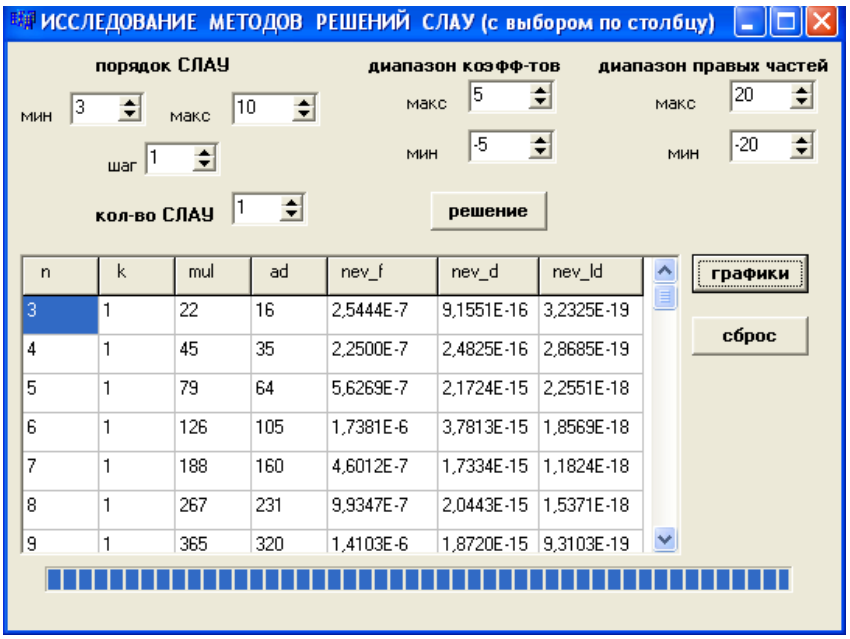


Рис. 4.8. Результаты тестирования на второй форме

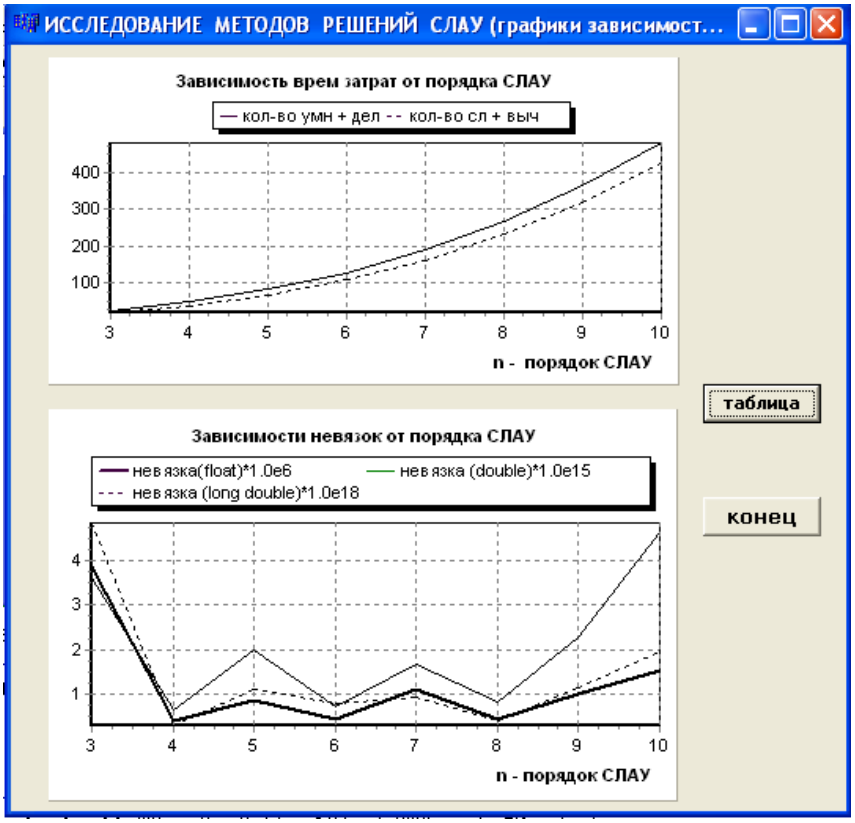


Рис. 4.9. Результаты тестирования на третьей форме

11. Для построения графиков зависимостей невязок, которые для разных типов коэффициентов СЛАУ отличаются друг от друга на несколько порядков, необходимо все невязки сблизить при помощи множителей. Величины множителей устанавливаются при решении СЛАУ. Запустим приложение на выполнение, нажав быстрые кнопки *Сохранить все* и *Запуск*. После щелчков на кнопках первой формы - *ТЕСТИРУЕМАЯ СЛАУ 3-ГО ПОРЯДКА*, *РЕШЕНИЕ*, *ТЕСТИРОВАНИЕ СЛАУ n – ПОРЯДКА,* и щелчка на кнопке *РЕШЕНИЕ* второй формы получим результаты выполнения задания (рис. 4.7, рис. 4.8).

12. Полученные результаты позволяют внести множители в соответствующие операторы:

```
Form3->Series3->AddXY(n,snev_f/k*1.0e6,"",clBlack);
Form3->Series4->AddXY(n,snev_d/k*1.0e15,"",clBlack);
Form3->Series5->AddXY(n,snev_ld/k*1.0e18,"",clBlack);
```

После этого графики зависимостей могут иметь вид (рис. 4.9).

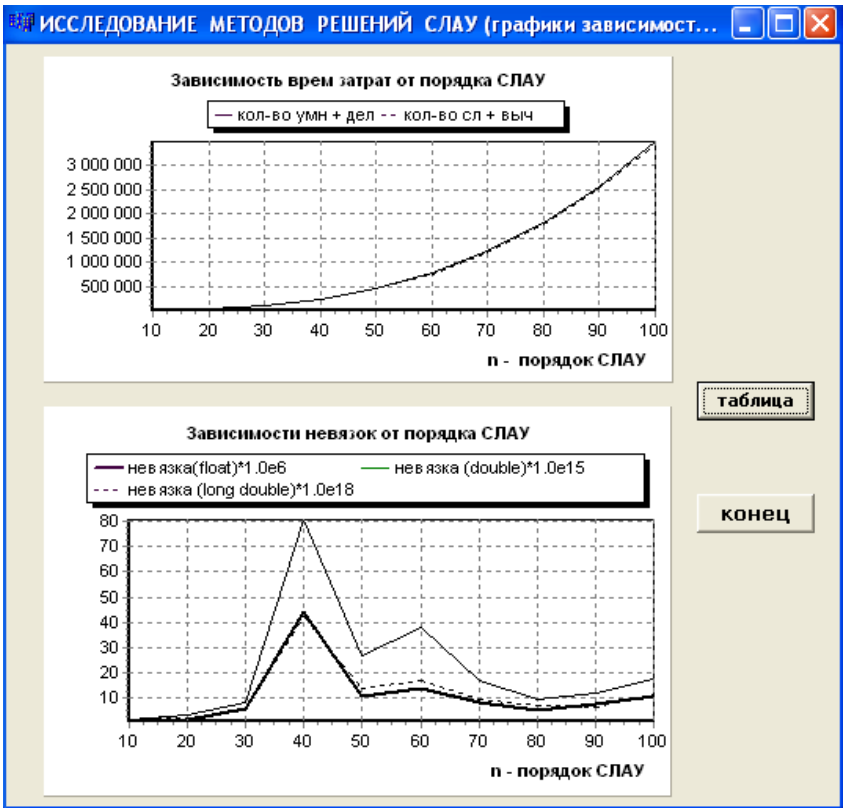


Рис. 4.10. результаты при 10 усредняемых СЛАУ

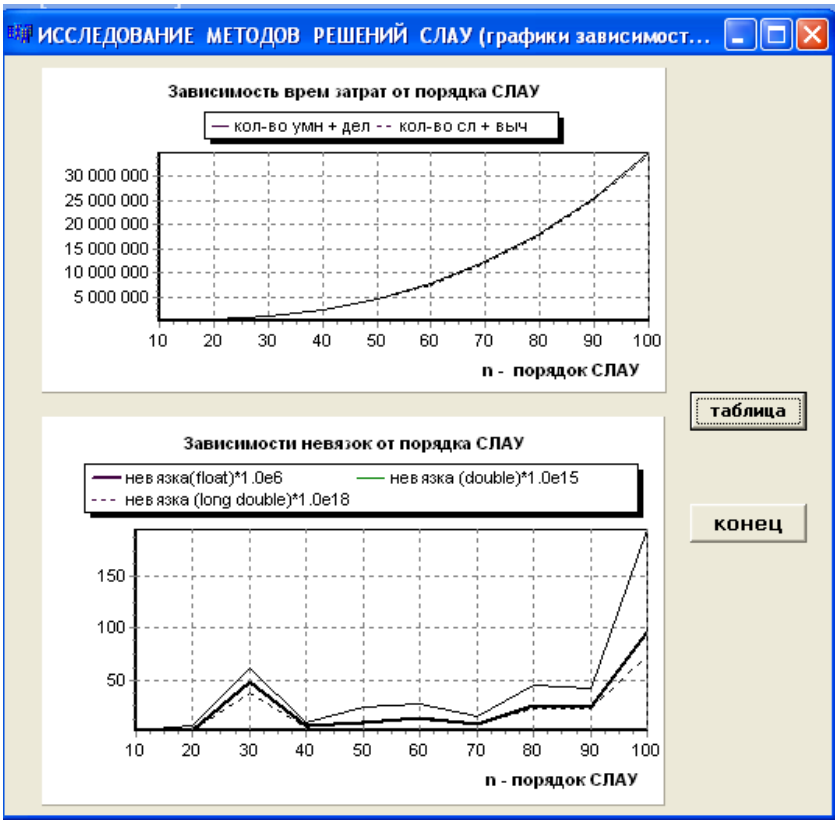


Рис. 4.11. результаты при 100 усредняемых СЛАУ

13. Последовательно нажимая кнопки *решение, графики, таблица, сброс, решение,* ….., убедитесь, что зависимости невязок от порядка системы случайны. Чтобы установить вид функциональных зависимостей, необходимо увеличить количество обрабатываемых СЛАУ заданного порядка для получения средних значений. Представление о характере зависимостей и вычислительных затратах при этом дают графики на рис. 4.10 и на рис. 4.11, полученные при 10 и 100 СЛАУ задаваемого порядка от 10 до 100 с шагом 10. Очевидно, что временные затраты для установления вида функциональных зависимостей значительны.

14. Щелчком на кнопке *конец* перейдите из режима выполнения в режим проектирования приложения.

Содержание отчета

1. Задание.
2. Формулы с пояснениями.
3. Решение СЛАУ в *Mathcad'*е.
4. Блок-схемы алгоритмов с комментариями.
5. Таблицы идентификаторов.
6. Код с комментариями.
7. Результаты выполнения работы в виде таблиц и графиков.
8. Библиографический список.

Контрольные вопросы

1. Когда целесообразно применять метод Гаусса?

2. Какова цель прямого хода в методе Гаусса?

3. Как выполняется обратный ход метода Гаусса?

4. На каком ходе, прямом или обратном, необходимо учитывать условия применения метода Гаусса?

5. Объясните алгоритм схемы единственного деления.

6. Объясните алгоритм схемы с частичным выбором ведущего коэффициента по столбцу.

7. Расскажите о достоинствах и недостатках схемы с полным выбором ведущего коэффициента.

8. Расскажите о методе Жордана-Гаусса.

9. Объясните зависимость временных затрат от размера системы.

10. Объясните зависимость ошибок от размера системы.

11. Представьте алгоритм получения зависимости усредненных ошибок от размера системы.

12. Объясните изменение хода зависимости усредненных ошибок от размера системы при увеличении количества усредняемых систем.

ЛАБОРАТОРНАЯ РАБОТА 5

ИССЛЕДОВАНИЕ МЕТОДОВ РЕШЕНИЯ СЛАУ И ОПЕРАЦИЙ С МАТРИЦАМИ

ЗАДАНИЕ

1.По заданному варианту разработать алгоритм, написать и отладить код.

2.Составить тест для проверки алгоритма, получить результаты в виде таблиц и графиков.

1. Обращение матриц методом окаймления. Исследовать точность и время обращения от размера матрицы.

2. Обращение матриц методом Ершова. Исследовать точность и время обращения от размера матрицы.

3. Обращение матриц методом Фаддеева. Исследовать точность и время обращения от размера матрицы.

4. Обращение матриц через единичную матрицу (по определению). Исследовать точность и время обращения от размера матрицы.

5. Обращение матриц через присоединенную матрицу. Исследовать точность и время обращения от размера матрицы.

6. Разложение неособенной квадратной матрицы в произведение двух треугольных матриц. Исследовать точность и время разложения от размера матрицы.

7. Разложение неособенной квадратной матрицы в произведение нижней треугольной матрицы с единичной диагональю и матрицы с ортогональными строками. Исследовать точность и время разложения от размера матрицы.

8. Разложение неособенной симметрической матрицы в произведение двух треугольных, транспонированных между собой матриц. Исследовать точность и время разложения от размера матрицы.

9. Решение СЛАУ методом Гаусса с приведением матрицы коэффициентов к обратной матрице. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

10. Решение СЛАУ методом ортогонализации строк матрицы коэффициентов. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

11. Решение СЛАУ методом Жордана-Гаусса . Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

12.Решение СЛАУ с ленточными матрицами коэффициентов. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

13. Решение СЛАУ методом Холецкого. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

14. Решение СЛАУ методом квадратного корня. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

15. Решение СЛАУ методом Гаусса с полным выбором ведущего коэффициента. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

16. Вычислить матричное выражение двумя методами: непосредственно по записи; методом, обеспечивающим минимальные временные затраты. Сравнить методы по зависимостям временных затрат от размера матриц. $\left((A+B)^T + E - A^T - B^T\right)^{-1}$**,** где A, B – матрицы размера $n \times n$.

17. Вычислить матричное выражение двумя методами: непосредственно по записи; методом, обеспечивающим минимальные временные затраты. Сравнить методы по зависимостям временных затрат от размера матриц. $(ABC)^{-1} + E - C^{-1}B^{-1}A^{-1}$, где A, B, C – матрицы размера $n \times n$.

18. Вычислить матричное выражение двумя методами: непосредственно по записи; методом, обеспечивающим минимальные временные затраты. Сравнить методы по зависимостям временных затрат от размера матриц. $PL^T C^{-1} LP$, где $C = LL^T$, P - симметрическая матрица размера $k \times k$, L - матрица общего вида размера $n \times k$.

19. Вычислить матричное выражение двумя методами: непосредственно по записи; методом, обеспечивающим минимальные временные затраты. Сравнить методы по зависимостям временных затрат от размера матриц. $(AB - BA)^{-1}$, где A, B - квадратные матрицы с элементами: $a_{ij} = 1$ при $j = i + 1$, $a_{ij} = 0$ при $j \neq i + 1$; $b_{ij} = d(i + 1)$ при $i = j + 1$, $b_{ij} = 0$ при $i \neq j + 1$.

20. Вычислить матричное выражение двумя методами: непосредственно по записи; методом, обеспечивающим минимальные временные затраты. Сравнить методы по зависимостям временных затрат от размера матриц. $((AB)^T + E - B^T A^T)^{-1}$, где A, B - матрицы общего вида размером $n \times n$.

21. Вычислить определитель симметрической положительно определенной матрицы путем ее разложения на две взаимно транспонированные треугольные матрицы. Получить зависимости затрат машинного времени от размера матрицы.

22. Вычислить определитель матрицы по методу Гаусса. Получить зависимости затрат машинного времени от размера матрицы.

23. Вычислить определитель матрицы через перестановки элементов матрицы. Получить зависимости затрат машинного времени от размера матрицы.

24. Вычислить определитель матрицы путем ее разложения на нижнюю и верхнюю треугольные матрицы. Получить зависимости затрат машинного времени от размера матрицы.

25. Решение СЛАУ методом простой итерации (метод Якоби). Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

26. Решение СЛАУ итерационным методом Гаусса-Зейделя. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

27. Решение СЛАУ методом LU- разложения. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

28. Решение СЛАУ методом прогонки. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

29. Решение СЛАУ с симметрической положительно определенной матрицей коэффициентов итерационным методом релаксации. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

30. Решение СЛАУ методом вращений. Сравнить по эффективности с методом Гаусса с частичным выбором ведущего коэффициента.

Содержание отчета

1. Задание.
2. Формулы с пояснениями.

3. Блок-схемы алгоритмов с комментариями.
4. Таблицы идентификаторов.
5. Код с комментариями.
6. Результаты выполнения работы в виде таблиц и графиков.
7. Библиографический список.

Контрольные вопросы

1. Расскажите о методе решения задачи.
2. Объясните алгоритм и докажите правильность выбранного способа тестирования.
3. Объясните полученные результаты.

ЧИСЛЕННЫЕ МЕТОДЫ АППРОКСИМАЦИИ ПО КРИТЕРИЮ НАИМЕНЬШИХ КВАДРАТОВ

Краткие сведения

При анализе эмпирических данных возникает необходимость найти в явном виде функциональную зависимость между величинами x и y, которые получены в результате измерений. Как правило, общий вид этой функциональной зависимости известен, а некоторые числовые параметры закона неизвестны.

Пусть, например, функция задана в виде (x_i, y_i) $i = 1, 2, \ldots, n$. Задача состоит в аппроксимации неизвестной функциональной зависимости между x и y многочленом заданной степени m:

$$\tilde{y}(x) = p_0 + p_1 x + \ldots + p_m x^m = \sum_{j=0}^{m} p_j x^j \ .$$

Для решения этой задачи применяют критерий наименьших квадратов. Согласно этому критерию, коэффициенты многочлена нужно

выбрать такими, чтобы сумма квадратов отклонений найденного многочлена от заданных значений функции была минимальной. Другими словами, коэффициенты $p_0, p_1,...,p_m$ должны минимизировать функцию

$$F(p_0, p_1,....,p_m) = \sum_{i=1}^{n}\left(p_0 + p_1 x_i + ... + p_m {x_i}^m - y_i\right)^2 .$$

В точке минимума функции F её производные $\partial F / \partial p_i$ обращаются в нуль. Дифференцируя F и приравнивая нулю производные, получим систему линейных алгебраических уравнений:

$$np_0 + \left(\sum_{i=1}^{n} x_i\right)p_1 + \left(\sum_{i=1}^{n} {x_i}^2\right)p_2 + ... + \left(\sum_{i=1}^{n} {x_i}^m\right)p_m = \sum_{i=1}^{n} y_i,$$

$$\left(\sum_{i=1}^{n} x_i\right)p_0 + \left(\sum_{i=1}^{n} {x_i}^2\right)p_1 + \left(\sum_{i=1}^{n} {x_i}^3\right)p_2 + ... + \left(\sum_{i=1}^{n} {x_i}^{m+1}\right)p_m = \sum_{i=1}^{n} x_i y_i,$$

……..……..……..……..……..……..……..……..……..……..…

$$\left(\sum_{i=1}^{n} {x_i}^m\right)p_0 + \left(\sum_{i=1}^{n} {x_i}^{m+1}\right)p_i + \left(\sum_{i=1}^{n} {x_i}^{m+2}\right)p_2 + ... + \left(\sum_{i=1}^{n} {x_i}^{2m}\right)p_m = \sum_{i=1}^{n} {x_i}^m y_i.$$

Отметим, что общее количество вычисляемых сумм равно $3m+1$.

Получена СЛАУ относительно неизвестных $p_0, p_1,...,p_m$. Определитель этой системы отличен от нуля. В алгоритме, представленном на рис. 6.1, для решения методом Гаусса СЛАУ приведена к виду:

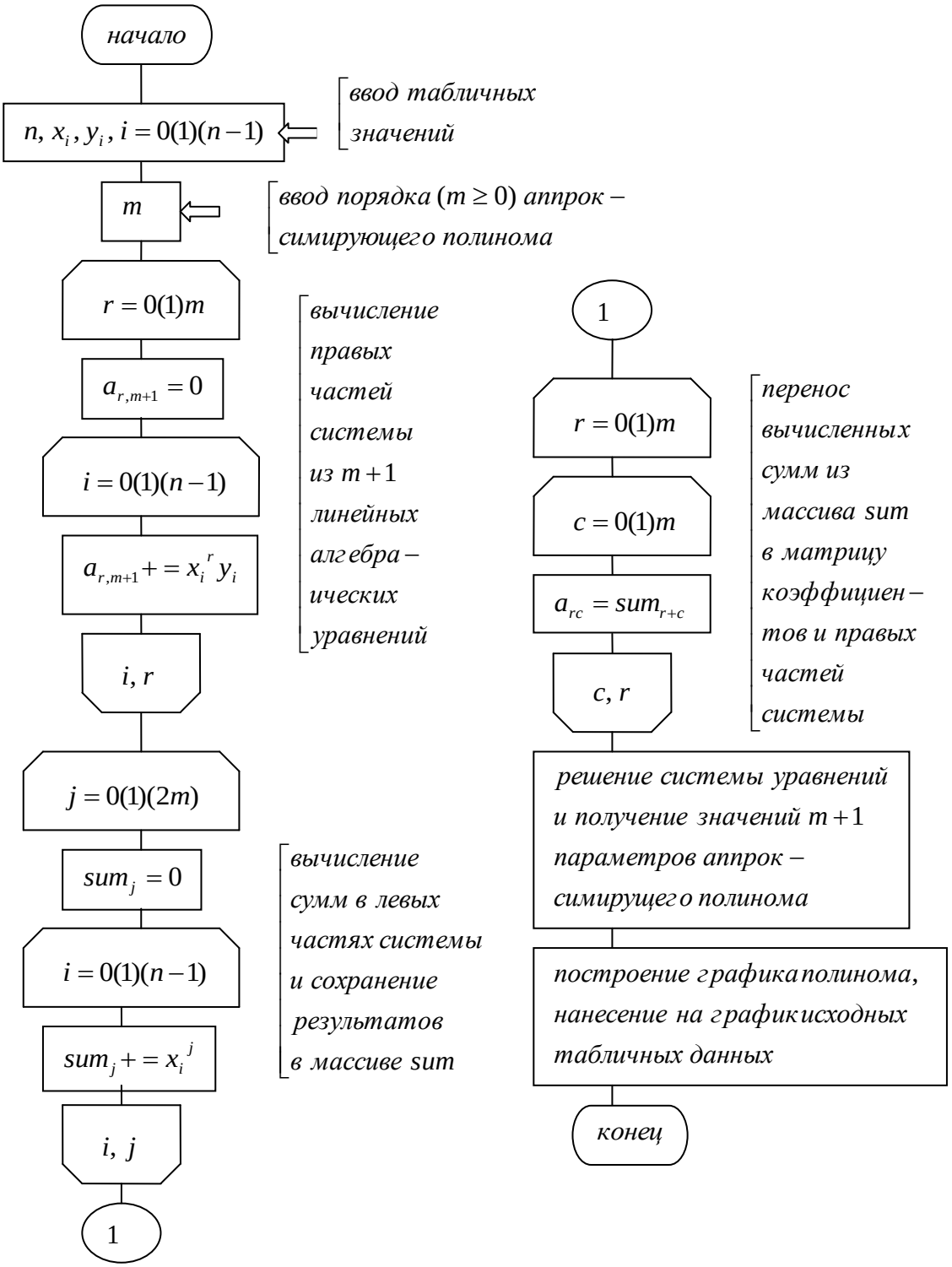


Рис. 6.1. Алгоритм аппроксимации степенным полиномом по критерию наименьших квадратов

$$a_{00}p_0 + a_{01}p_1 + \ldots\ldots\ldots + a_{0m}p_m = a_{0,m+1},$$
$$a_{10}p_0 + a_{11}p_1 + \ldots\ldots\ldots + a_{1m}p_m = a_{1,m+1},$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$
$$a_{m0}p_0 + a_{m1}p_1 + \ldots\ldots\ldots + a_{mm}p_m = a_{m,m+1}.$$

Выполнение аппроксимации (регрессии) в системе Mathcad

Чаще всего используется линейная регрессия, при которой функция $y(x)$ описывает отрезок прямой и имеет вид $y(x) = a + b \cdot x$. Для реализации линейной регрессии в систему встроен ряд функций: $corr(VX, VY)$ - возвращает коэффициент корреляции; *intercept(VX,VY)*- возвращает значение *a*; *slope(VX,VY)* – возвращает значение *b*. Координаты исходных точек хранятся в векторах *VX, VY.* Линейная регрессия выполняется с максимальным среднеквадратичным приближением к исходным данным. Чем ближе коэффициент корреляции к единице, тем точнее представленная исходными точками зависимость приближается к линейной.

В линейной регрессии общего вида заданная совокупность точек приближается к функции

$$F(x, K1, K2, \ldots, Kn) = K1 \cdot F1(x) + K2 \cdot F2(x) + \ldots + Kn \cdot Fn(x),$$

которая является линейной комбинацией функций $F1(x), F2(x), \ldots, Fn(x)$, причем сами эти функции могут быть нелинейными. Для реализации линейной регрессии общего вида используется функция *linfit(VX,VY,F),* возвращающая вектор коэффициентов *K,* при котором среднеквадратичная погрешность приближения совокупности исходных точек оказывается минимальной.

Пример проведения регрессий – линейной и линейной общего вида

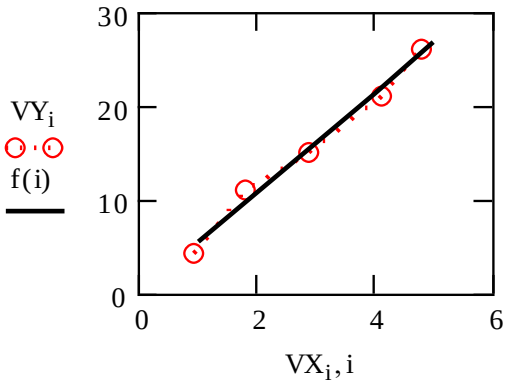
Линейная регрессия

$$VX := \begin{pmatrix} 0.95 \\ 1.8 \\ 2.9 \\ 4.1 \\ 4.8 \end{pmatrix} \qquad VY := \begin{pmatrix} 4.2 \\ 11 \\ 15 \\ 21 \\ 26 \end{pmatrix}$$

$a := \text{intercept}(VX, VY)$ $b := \text{slope}(VX, VY)$

$ORIGIN := 1$ $i := 1..5$ $f(x) := a + b \cdot x$ $a = -0.044$ $b = 5.321$

$\text{corr}(VX, VY) = 0.993$ $f(3) = 15.919$ $\text{linterp}(VX, VY, 3) = 15.5$

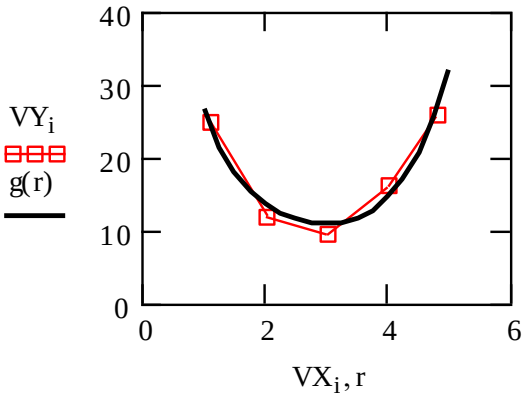


Линейная регрессия общего вида

$$VX := \begin{pmatrix} 1.1 \\ 2 \\ 3 \\ 4 \\ 4.8 \end{pmatrix} \qquad VY := \begin{pmatrix} 25 \\ 12 \\ 9.4 \\ 16.2 \\ 26 \end{pmatrix} \qquad F(x) := \begin{pmatrix} \frac{1}{x} \\ x^2 \\ \exp(x) \end{pmatrix}$$

$$K := \text{linfit}(VX, VY, F) \qquad K = \begin{pmatrix} 26.064 \\ -0.284 \\ 0.228 \end{pmatrix}$$

$$\text{ORIGIN} := 0 \qquad i := 0..4 \qquad g(t) := F(t) \cdot K \qquad r := 1, 1.25..5$$



ЛАБОРАТОРНАЯ РАБОТА 6

АППРОКСИМАЦИЯ ФУНКЦИИ ПО КРИТЕРИЮ НАИМЕНЬШИХ КВАДРАТОВ

ЗАДАНИЕ

1. Выполнить в *Mathcad*'е аппроксимацию (регрессию) двух вариантов – линейную и линейную общего вида – функции, заданной таблично ($y_i=y(x_i);\ x_i = i*0.1,\ i=1, 2, ..., 20$). Варианты заданий даны в таблице.

2. Составить алгоритм и написать код для аппроксимации заданной функции многочленом m-й степени (m вводится с клавиатуры) по методу наименьших квадратов. Вывести на экран параметры аппроксимирующего многочлена, график многочлена и нанести на график исходные данные.

Таблица

i	1	2	3	4	5	6	7	8	9	10
	11	12	13	14	15	16	17	18	19	20
1	2.05	1.94	1.92	1.87	1.77	1.88	1.71	1.60	1.56	1.40
	1.50	1.26	0.99	0.97	0.91	0.71	0.43	0.54	0.19	0.01
2	2.09	2.05	2.19	2.18	2.17	2.27	2.58	2.73	2.82	3.04
	3.03	3.45	3.62	3.85	4.19	4.45	4.89	5.06	5.63	5.91
3	2.02	1.98	1.67	1.65	1.57	1.42	1.37	1.07	0.85	0.48
	0.35	–0.30	–0.61	–1.20	–1.39	–1.76	–2.28	–2.81	–3.57	–4.06
4	1.99	2.03	2.20	2.39	2.19	2.61	2.35	2.60	2.55	2.49
	2.50	2.52	2.44	2.35	2.26	2.19	2.24	2.34	1.96	2.19
5	2.23	2.29	2.27	2.62	2.72	2.82	3.13	3.49	3.82	3.95
	4.22	4.48	5.06	5.50	5.68	6.19	6.42	7.04	7.57	8.10
6	2.07	2.17	2.21	2.31	2.10	2.09	2.12	1.63	1.78	1.52
	1.16	1.07	0.85	0.56	0.10	–0.25	–0.65	–1.06	–1.66	–2.01

7	2.18	2.43	2.40	2.43	2.65	2.75	2.67	2.66	2.63	2.75
	2.41	2.24	2.12	1.74	1.57	1.17	0.96	0.63	0.25	0.01
8	–0.10	–0.21	0.01	0.05	–0.13	–0.23	–0.21	–0.43	–0.57	–0.44
	–0.44	–0.83	–0.78	–0.81	–1.06	–1.41	–1.40	–1.70	–1.96	–1.91
9	–0.16	0.01	0.10	0.16	0.05	0.35	0.19	0.50	0.74	1.03
	1.06	1.49	1.79	2.03	2.22	2.50	2.88	3.21	3.63	3.90
10	2.09	2.31	2.72	2.77	2.78	2.97	3.00	3.54	3.43	3.58
	3.58	3.54	3.82	3.90	3.77	3.81	4.00	3.97	4.08	4.08
11	2.15	2.41	2.58	2.84	3.28	3.46	4.02	4.11	4.61	5.03
	5.34	5.86	6.33	6.81	7.21	7.67	8.23	8.68	9.35	9.93
12	0.10	–0.01	–0.19	–0.11	–0.31	–0.78	–0.64	–0.85	–1.18	–1.39
	–1.79	–2.02	–2.48	–2.93	–3.26	–3.91	–4.41	–4.91	–5.30	–6.00
13	0.17	0.07	0.17	0.05	0.12–	0.00	0.01	–0.05	–0.21	–0.50
	–0.50	–0.86	–1.24	–1.47	1.79	–2.25	–2.55	–3.18	–3.60	–3.93
14	0.80	0.29	0.52	0.77	0.93	1.20	1.20	1.35	1.39	1.48
	1.52	1.71	1.72	1.87	1.86	1.89	2.04	1.73	2.04	2.03
15	0.04	0.47	0.78	1.01	1.19	1.60	1.93	2.22	2.50	3.01
	3.22	3.71	4.23	4.78	5.27	5.75	6.16	6.76	7.30	8.00
16	0.08	0.14	0.37	0.36	0.44	0.48	0.27	0.39	0.50	0.48
	0.69	0.50	0.31	0.37	0.43	0.33	0.31	0.09	0.08	0.03
17	–0.02	0.44	0.51	0.67	0.69	1.04	1.14	1.37	1.77	2.00
	2.12	2.47	2.90	3.50	3.99	4.06	4.54	4.99	5.36	5.99
18	0.14	0.23	0.44	0.54	0.72	0.76	0.37	0.64	0.57	0.44
	0.41	0.30	–0.01	–0.03	–0.47	–0.68	–0.93	–1.28	–1.53	–1.93
19	–1.86	–1.95	–2.12	–2.06	–2.15	–2.00	–2.12	–2.31	–2.29	–2.57
	–2.56	–2.86	–2.85	–3.03	–3.25	–3.08	–3.29	–3.67	–3.70	–3.85
20	–1.65	–2.00	–1.87	–1.89	–1.75	–1.59	–1.44	–1.51	–1.00	–1.17
	–0.87	–0.47	–0.33	0.00	0.34	0.49	0.81	1.37	1.72	2.03

21	–1.89	–2.07	–2.30	–2.26	–2.34	–2.66	–2.88	–2.85	–3.16	–3.49
	–3.88	–4.22	–4.45	–4.99	–5.36	–5.71	–6.51	–6.76	–7.35	–8.02
22	–1.84	–1.98	–1.72	–1.58	–1.59	–1.59	–1.58	–1.64	–1.55	–1.35
	–1.33	–1.47	–1.50	–1.65	–1.62	–1.87	–1.61	–1.86	–1.84	–1.91
23	–1.92	–1.60	–1.57	–1.41	–1.36	–0.97	–0.59	–0.71	–0.15	0.01
	0.22	0.63	1.07	1.42	1.68	2.49	2.57	3.09	3.40	4.00
24	–1.90	–1.80	–1.82	–1.86	–1.83	–2.02	–2.01	–2.05	–2.46	–2.68
	–2.85	–2.98	–3.30	–3.40	–3.90	–4.37	–4.65	–5.00	–5.42	–6.13
25	–1.80	–1.66	–1.36	–1.41	–1.13	–0.82	–0.74	–0.76	–0.64	–0.46
	–0.30	–0.27	–0.22	–0.11	–0.02	–0.11	0.11	–0.02	0.03	0.01
26	–1.65	–1.54	–1.41	–0.91	–0.63	–0.34	–0.12	0.25	0.64	0.96
	1.50	1.77	2.24	2.93	3.17	3.77	4.42	4.79	5.50	6.01
27	–1.88	–1.69	–1.52	–1.55	–1.16	–1.27	–1.23	–1.36	–1.26	–1.47
	–1.72	–1.76	–2.00	–2.03	–2.35	–2.46	–2.88	–3.27	–3.68	–3.98
28	–4.01	–4.06	–3.88	–3.98	–4.36	–4.18	–4.16	–4.51	–4.53	–4.38
	–4.76	–4.66	–4.82	–4.77	–5.12	–5.23	–5.40	–5.84	–5.86	–6.01
29	–4.13	–4.11	–3.87	–3.74	–3.85	–3.71	–3.53	–3.56	–3.19	–3.04
	–2.98	–2.54	–2.41	–1.97	–1.78	–1.53	–1.04	–0.86	–0.48	0.09
30	–3.97	–4.07	–4.04	–4.30	–4.27	–4.54	–4.79	–5.07	–5.30	–5.51
	–5.83	–6.06	–6.40	–6.83	–7.54	–7.68	–8.36	–8.91	–9.39	–9.98

1. Создайте новый проект командой *Файл/Новый/Приложение.*

2. Сохраните файлы модуля и проекта командой *Файл*/*Сохранить все* под именами LR6 и PR_LR6. Для этого удобно использовать соответствующую быструю кнопку (*Сохранить все*). В последующих сеансах работы сохраненный проект можно открыть командой *Файл/Открыть проект* (или *Повторно открыть*). Теперь перейдем к проектированию приложения – переносам на форму необходимых

компонентов и заданию их свойствам значений, а в обработчиках событий – размещению кодов соответствующих алгоритмов. (Рекомендуется нажимать кнопку *Сохранить все* по окончании работы с каждым компонентом.) В результате проектирования получим форму, представленную на рис. 6.2.

3. Выделите форму, щелкнув на ней левой кнопкой мыши, и в свойство **Caption** (надпись) впишите *АППРОКСИМАЦИЯ ФУНКЦИИ ПО КРИТЕРИЮ НАИМЕНЬШИХ КВАДРАТОВ***.**

4. На форме разместите 4 кнопки (страница *Стандарт*) с надписями соответственно: **Button1 –** *ИСХОДНЫЕ ДАННЫЕ*, **Button2 –** *РАСЧЕТ КОЭФФИЦИЕНТОВ*, **Button3 –** *ГРАФИКИ*, **Button4 –** *КОНЕЦ.*

5. Для размещения исходных данных используйте в качестве таблицы компонент **StringGrid1** (страница *Дополнительно*). Установите следующие значения свойств компонента **StringGrid1**: **ColCount** – 21, **DefaultColWidth** – 32, **FixedCols** – 1, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 3. Раскрыв свойство **Options**, установите значение подсвойства **goEditing** – **true**, что даст возможность редактировать содержимое таблицы в **StringGrid1**.

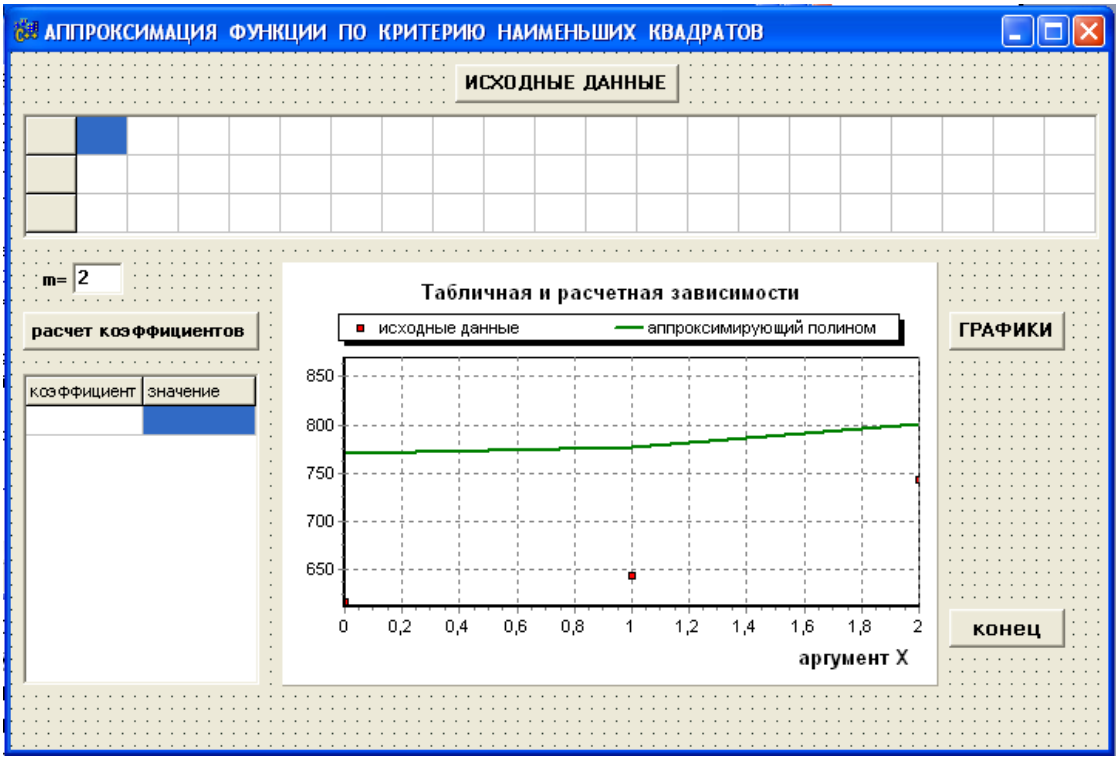


Рис. 6.2. Форма по окончании проектирования

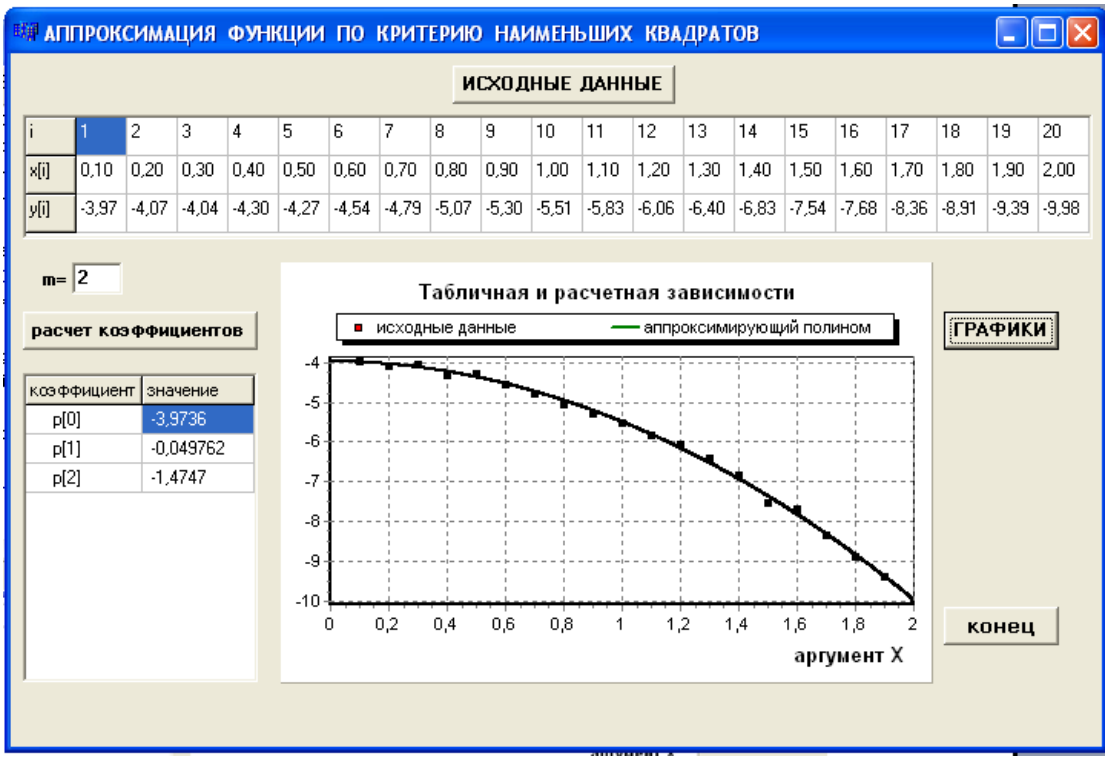


Рис. 6.3. Результаты выполнения задания

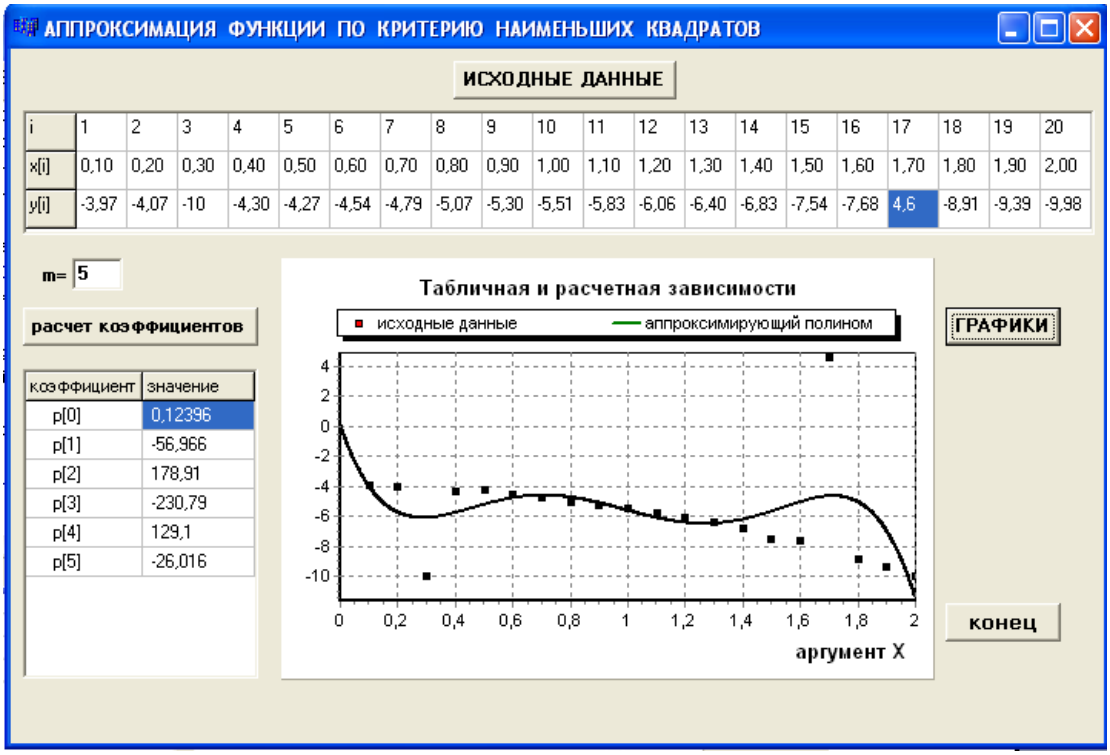


Рис. 6.4. ослабление выбросов в исходных данных

6. Для задания степени аппроксимирующего многочлена используйте метку **LabeledEdit1** (страница *Дополнительно*). Свойству **LabelPosition** присвойте значение **lpLeft** (из выпадающего списка), а свойству **Font** – жирный, размер 8, свойству **Text** – 2. Раскрыв свойство **EditLabel**, установите подсвойство **Font** – жирный, размер 8, а в подсвойстве **Caption** впишите соответственно **m=**.

7. Для размещения результатов расчетов коэффициентов аппроксимирующего многочлена используйте компонент **ValueListEditor1** (страница *Дополнительно*). Компонент содержит строки вида «*имя = значение*». Окно компонента имеет две колонки с заголовками «*Key*» для имен и «*Value*» для значений. Измените эти значения, используя свойство **TitleCaptions** типа **TStrings**. Нажатием на кнопку (три точки) свойства вызовите окно *Редактор строки списка*, в первой строке которого напишите *коэффициент,* во второй – *значение.* Значения других свойств установите такими: **Font** – черный, обычный, размер 8, **DefaultRowHeight** – 18, **DefaultColWidth** – 75.

8. Для графического представления исходных данных и аппроксимирующего многочлена используйте компонент **Chart1** (страница *Additional*). Зададим свойства компонента. Щелкните правой кнопкой мыши на компоненте **Chart1** и в появившемся меню выберите *Edit Chart….* На экране появится окно *Редактора Диаграмм* (*Editing Chart1*) с открытой страницей *Chart*, которая имеет несколько закладок. На закладке *Panel*, нажав кнопку *Panel Color…*, выберите белый цвет. На закладке *Series* щелкните на кнопке *Add…* - добавить серию. В появившемся окне выберите тип графика – *Point* и выключите индикатор *3D.* После щелчка на *OK* снова появится окно *Editing Chart1*. Перейдите на закладку *Titles*. В окне редактирования, которое в данный момент соответствует *Title* – заголовку графика, сотрите *TChart* и напишите (шрифт *Font…* - черный, жирный, размер 10) *ТАБЛИЧНАЯ И РАСЧЕТНАЯ*

ЗАВИСИМОСТИ. Цвет фона *Back Color*.. установите белый. В выпадающем списке от окна редактирования *Title* перейдите в окно редактирования *Foot* и напишите тем же шрифтом *аргумент X*. В группе кнопок *Alignment* нажмите кнопку *Right*. Цвет фона *Back Color*.. также установите белый. Перейдите на закладку *Axis* для задания координатных характеристик осей. Оставьте в группе *Axis* нажатой кнопку *Left* и включенным индикатор *Automatic*. Затем нажмите в группе *Axis* кнопку *Bottom* и выключите индикатор *Automatic*. Нажмите среднюю кнопку *Change…*и в окне *Value* редактора *Minimum Bottom Axis* впишите число 0. Аналогичные действия выполните с верхней кнопкой *Change…*, но впишите число 2. И, нажав нижнюю кнопку *Change…*, в окно *Increment:* занесите 0,2. Перейдите на закладку *Series*, кнопкой *Add…*добавьте график *Line* при выключенном индикаторе *3D* . Выделив *Series 1*, кнопкой *Title…* вызовите *Change Series Title*, где дайте заголовок первому графику – *исходные данные*. Действуя подобным образом, т. е. выделив *Series 2*, дайте заголовок второму графику – *аппроксимирующий полином*. На вкладке *Legend* нажмите кнопку *Top*. Перейдите со страницы *Chart* на страницу *Series*. Здесь на закладке *Format* задают параметры линий графиков. Требуемый график (*исходные данные*, *аппроксимирующий полином*) выбирается из выпадающего списка. Для графика *исходные данные* установите размеры прямоугольника: *Width*=2, *Height*=2, а нажав кнопку *Border…*, задайте толщину линии графика *Width*=1, и цвет линии графика *Color…* - черный. Для графика *аппроксимирующий полином* в группе *Line* через кнопку *Border…*устанавливается толщина (*Width*=2) линии графика, и через кнопку *Color…* - цвет линии графика – черный. Выключите индикатор *Color Each*.

9. Файл *LR6.cpp* с обработчиками щелчков на кнопках имеет вид:

```
//---------------------------------------------------------------------------
#include <vcl.h>
```

```
#pragma hdrstop

#include "LR6.h"
#include<math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
const int n=20;
double x[n+1]={0.0};
double y[n+1]={0.0,-3.97,-4.07,-4.04,-4.30,-4.27,-4.54,-4.79,-5.07,-5.30,
       -5.51,-5.83,-6.06,-6.40,-6.83,-7.54,-7.68,-8.36,-8.91,-9.39,-9.98};
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  ValueListEditor1->Strings->Clear();
  Series1->Clear();
  Series2->Clear();
  StringGrid1->Cells[0][0]="i";
  StringGrid1->Cells[0][1]="x[i]";
  StringGrid1->Cells[0][2]="y[i]";
  for(int i=1;i<n+1;i++){
     StringGrid1->Cells[i][0]=IntToStr(i);
     x[i]=i*0.1;
```

```
    StringGrid1->Cells[i][1]=FloatToStrF(x[i],ffFixed,2,2);
    StringGrid1->Cells[i][2]=FloatToStrF(y[i],ffFixed,3,2);}
  LabeledEdit1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{ ValueListEditor1->Strings->Clear();
  Series1->Clear();
  Series2->Clear();
  int m=StrToInt(LabeledEdit1->Text);
  float c,b,s,*p,**a;
  p=new float[m+1];
  a=new float*[m+1];
  for(int i=0;i<m+1;i++)
     a[i]=new float[m+2];
  for(int r=0;r<m+1;r++){
    a[r][m+1]=0;
    for(int i=1;i<n+1;i++)
       a[r][m+1]+=pow(x[i],r)*StrToFloat(StringGrid1->Cells[i][2]);
    for(int c=0;c<m+1;c++){
      a[r][c]=0;
      for(int i=1;i<n+1;i++)
        a[r][c]+=pow(x[i],r+c);}}
  for(int i=0;i<m;i++){
```

```
    c=a[i][i];
    for(int j=i;j<m+2;j++) a[i][j]/=c;
    for(int k=i+1;k<m+1;k++){
      b=a[k][i];
      for(int l=i;l<m+2;l++)
        a[k][l]=a[k][l]-b*a[i][l];}}
  p[m]=a[m][m+1]/a[m][m];
  for(int k=m-1;k>=0;k--){
    s=0;
    for(int j=k+1;j<m+1;j++) s+=a[k][j]*p[j];
    p[k]=a[k][m+1]-s;}
  AnsiString st,st1;;
  for(int i=0;i<=m;i++){
    st="     p["+IntToStr(i)+"]";
    st1=" "+FloatToStrF(p[i],ffGeneral,5,2);
    ValueListEditor1->Values[st]=st1;}
  delete []p; p=0;
  for(int i=0;i<=m;i++)delete []a[i];
  delete []a; a=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{float x1,p,pol;
 int m=StrToInt(LabeledEdit1->Text);
 Series1->Clear();
 Series2->Clear();
 for(int i=1;i<n+1;i++){
   Series1->AddXY(x[i],StrToFloat(StringGrid1->Cells[i][2]),"",clBlack); }
 for(x1=0;x1<=2;x1+=0.001){
```

```
    pol=0;
    for(int k=0;k<=m;k++){
      p=StrToFloat(ValueListEditor1->Cells[1][k+1]);
      if(k)pol+=p*pow(x1,k);
        else pol+=p;}
    Series2->AddXY(x1,pol,"",clBlack);}
}
//---------------------------------------------------------------------------
```

10. Запустим приложение на выполнение, нажав быстрые кнопки *Сохранить все* и *Запуск*. После щелчков на кнопках *ИСХОДНЫЕ ДАННЫЕ, РАСЧЕТ КОЭФФИЦИЕНТОВ, ГРАФИКИ,* получим результаты выполнения задания (рис. 6.3).

11. Объясните результаты при $m=0$ и при $m=1$.

12. Воспользуемся тем, что таблица *ИСХОДНЫЕ ДАННЫЕ* позволяет редактировать значения *y[i]*. Изменив два значения, а также порядок аппроксимирующего многочлена, после щелчков на кнопках *РАСЧЕТ КОЭФФИЦИЕНТОВ, ГРАФИКИ,* получим результаты (рис. 6.4), которые убедительно показывают, что аппроксимация эффективно ослабляет влияние выбросов в исходных данных, что полезно при обработке результатов экспериментов.

13. Для возврата к первоначальным исходным данным достаточно щелкнуть на кнопке *ИСХОДНЫЕ ДАННЫЕ*.

14. Щелчком на кнопке *конец* выйдите из режима выполнения и перейдите в режим проектирования приложения.

Содержание отчета

1. Задание.
2. Формулы с пояснениями.
3. Результаты выполнения задания в *Mathcad*'е.
4. Блок-схема алгоритма и таблица идентификаторов.
5. Код с комментариями.
6. Значения коэффициентов аппроксимирующего многочлена для m=0..8.
7. График аппроксимирующего многочлена для m=2 с нанесенными на график исходными данными.
8. Библиографический список.

Контрольные вопросы

1. В чем состоит цель задачи аппроксимации?

2. Расскажите о теоретических и практических достоинствах критерия наименьших квадратов.

3. Какие методы аппроксимации рассматриваются в лабораторной работе?

4. Когда используется линейная аппроксимация?

5. Когда используется аппроксимация степенным полиномом?

6. Когда используется линейная аппроксимация общего вида?

7. Как определяются параметры аппроксимирующей прямой?

8. Как определяются параметры аппроксимирующего полинома?

9. Как задаются и определяются параметры аппроксимирующей функции при линейной аппроксимации общего вида?

10. Объясните алгоритм аппроксимации степенным полиномом.

11. Объясните полученные результаты при $m=0$ и при $m=1$.

ЧИСЛЕННЫЕ МЕТОДЫ ВЫЧИСЛЕНИЯ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

Краткие сведения

Для вычисления определенного интеграла используется формула Ньютона-Лейбница, если первообразная $F(x)$ может быть найдена или не является слишком громоздкой:

$$\int_a^b f(x)dx = F(b) - F(a).$$

В противном случае вычисляют приближенное значение интеграла по квадратурной формуле

$$\int_a^b f(x)dx \cong \sum_{i=1}^{n} A_i f(x_i),$$

содержащей $2n+1$ параметров (n абсцисс x_i, n коэффициентов A_i и число узлов), которые нужно выбрать так, чтобы формула давала достаточно малую погрешность для широкого класса функций $f(x)$. Очевидно, чем больше n, тем выше точность. Указанному требованию удовлетворяют квадратурные формулы Ньютона-Котеса и квадратурная формула Гаусса.

Квадратурные формулы Ньютона-Котеса

Формулы Ньютона-Котеса относятся к случаю равноотстоящих узлов интегрирования. Они получены как результат интерполяции подынтегральной функции полиномом Лагранжа и в общем случае имеют вид

$$\int_a^b f(x)dx \cong (b-a)\sum_{i=0}^{n} H_i f(x_i), \text{ где } x_i = a + ih; \quad i = 0,1,..,n; \quad h = (b-a)/n;$$

$$H_i = \frac{1}{n}\frac{(-1)^{n-1}}{i!(n-i)!}\int_0^n \frac{q(q-1)(q-2).....(q-n)}{q-i}dq \text{ – коэффициенты Котеса.}$$

Очевидно, коэффициенты Котеса не зависят от подынтегральной функции.

Для n от 1 до 8 они имеют значения:

n=1: $H_0 = H_1 = 0.5$;

n=2: $H_0 = H_2 = 0.1666667$, $H_1 = 0.6666667$;

n=3: $H_0 = H_3 = 0.125$, $H_1 = H_2 = 0.375$;

n=4: $H_0 = H_4 = 0.0777778$, $H_1 = H_3 = 0.3555556$, $H_2 = 0.1333333$;

n=5: $H_0 = H_5 = 0.0659722$, $H_1 = H_4 = 0.2604166$, $H_2 = H_3 = 0.1736111$;

n=6: $H_0 = H_6 = 0.0488095$, $H_1 = H_5 = 0.2571428$, $H_2 = H_4 = 0.0321428$, $H_3 = 0.3238095$;

n=7: $H_0 = H_7 = 0.0434606$, $H_1 = H_6 = 0.2070023$, $H_2 = H_5 = 0.0765626$, $H_3 = H_4 = 0.1729745$;

n=8: $H_0 = H_8 = 0.0348853$, $H_1 = H_7 = 0.2076895$, $H_2 = H_6 = -0.0327336$, $H_3 = H_5 = 0.3702292$, $H_4 = –0.1601410$.

Правильность вычисления коэффициентов Котеса можно проверить по их свойствам:

$$1) \sum_{i=0}^{n} H_i = 1, \qquad 2) \; H_i = H_{n-i}.$$

Поскольку интервал интегрирования может быть большим, то его делят на k равных частей и к каждой из них применяют соответствующую формулу. Формула для всего интервала интегрирования становится составной. Обычно используют $n = 1, \quad n = 2, \quad n = 3$.

1. Формула трапеций (n=1)

$$\int_a^b f(x)dx \cong \frac{h}{2}\left(f_0 + 2f_1 + 2f_2 + \ldots + 2f_{k-1} + f_k\right),$$

где $h = \frac{b-a}{k}$ – шаг интегрирования; $f_i = f(x_i);\ x_i = x_0 + ih;\ x_0 = a;$ $i = 0, 1, \ldots, k.$

2. Формула Симпсона (n=2)

$$\int_a^b f(x)dx \cong \frac{h}{3}\left[\left(f_0 + f_{2k}\right) + 4\left(f_1 + f_3 + \ldots + f_{2k-1}\right) + 2\left(f_2 + f_4 + \ldots + f_{2k-2}\right)\right],$$

где $h = \frac{b-a}{2k}$; $2h$ - шаг интегрирования; $f_i = f(x_i);\ x_i = x_0 + ih;\ x_0 = a;$ $i = 0, 1, \ldots, 2k.$

3. Формула Ньютона (“трех восьмых”, n=3)

$$\int_a^b f(x)dx \cong \frac{3h}{8}\left[\left(f_0 + f_{3k}\right) + 2\left(f_3 + f_6 + \ldots + f_{3k-3}\right) + 3\left(f_1 + f_2 + \ldots + f_{3k-2} + f_{3k-1}\right)\right],$$

где $h = \frac{b-a}{3k}$; $3h$ – шаг интегрирования, $f_i = f(x_i);\ x_i = x_0 + ih;\ x_0 = a;$ $i = 0, 1, \ldots, 3k.$

В этих формулах шаг интегрирования постоянный, и поэтому они эффективны в методах с автоматическим выбором шага интегрирования.

Процедура автоматического выбора шага интегрирования основана на двукратном вычислении интеграла на каждом шаге: вначале с h (значение интеграла S_1), а затем – с $h/2$ (значение интеграла S_2). Если $|S_1 - S_2|$ меньше заданной наперед ошибки, то считается, что интегрирование на данном шаге выполнено правильно. В противном

случае h делится пополам, и повторяется описанная выше процедура интегрирования.

Примечание. Способ повышения точности путем двойного пересчета называется *правилом Рунге*.

Квадратурная формула Гаусса

Для увеличения точности здесь взяты неравноотстоящие узлы x_i, что оказалось возможным при интерполировании подынтегральной функции полиномом Лежандра:

$$\int_a^b f(x)dx \cong (b-a)\sum_{i=1}^{n} A_i f(x_i),$$

где $x_i = a + (b-a)X_i;\quad i = 1, 2, \ldots, n.$ Ниже приведены абсциссы X_i и коэффициенты A_i формулы Гаусса для значений пределов интегрирования $a = 0,\ b = 1,$ при которых формула Гаусса имеет вид

$$\int_0^1 f(x)dx \cong \sum_{i=1}^{n} A_i f(x_i).$$

Абсциссы и коэффициенты формулы Гаусса

n=1: $X_1 = 0.5$; $A_1 = 1.0$;

n=2: $X_1 = 0.2113249$, $X_2 = 0.7886751$; $A_1 = A_2 = 0.5$;

n=3: $X_1 = 0.1127017$, $X_2 = 0.5$, $X_3 = 0.8872983$;

$A_1 = A_3 = 0.2777778$, $A_2 = 0.4444444$;

n=4: $X_1 = 0.0694318$, $X_2 = 0.3300095$, $X_3 = 0.6699905$, $X_4 = 0.9305682$;

$A_1 = A_4 = 0.1739274$, $A_2 = A_3 = 0.3260726$;

n=5: $X_1 = 0.0469101$, $X_2 = 0.2307653$, $X_3 = 0.5$, $X_4 = 0.7692347$,

$X_5 = 0.9530899$;

$A_1 = A_5 = 0.1184634$, $A_2 = A_4 = 0.2393143$, $A_3 = 0.2844444$;

n=6: $X_1 = 0.0337652$, $X_2 = 0.1693953$, $X_3 = 0.3806904$, $X_4 = 0.6193096$,

$X_5 = 0.8306047$, $X_6 = 0.9662348$;

$A_1 = A_6 = 0.0856622$, $A_2 = A_5 = 0.1803808$, $A_3 = A_4 = 0.2339570$;

n=7: $X_1 = 0.0254460$, $X_2 = 0.1292344$, $X_3 = 0.2970774$, $X_4 = 0.5$,

$X_5 = 0.7029226$, $X_6 = 0.8707656$, $X_7 = 0.9745540$;

$A_1 = A_7 = 0.0647425$, $A_2 = A_6 = 0.1398527$,

$A_3 = A_5 = 0.1909150$, $A_4 = 0.2089796$;

n=8: $X_1 = 0.0198551$, $X_2 = 0.1016668$, $X_3 = 0.2372338$, $X_4 = 0.4082827$,

$X_5 = 0.5917173$, $X_6 = 0.7627662$, $X_7 = 0.8983332$, $X_8 = 0.9801449$;

$A_1 = A_8 = 0.0506143$, $A_2 = A_7 = 0.1111905$,

$A_3 = A_6 = 0.1568533$, $A_4 = A_5 = 0.1813419$.

Если отрезок интегрирования $b-a$ разбить на k равных частей и к каждой из них применить формулу Гаусса для фиксированного n, то

$$\int_a^b f(x)dx = \sum_{j=1}^{k} \int_{x_{j-1}}^{x_j} f(x)dx \cong h\sum_{j=1}^{k}\sum_{i=1}^{n} A_i f(x_i^{(j)}),$$

где $h=(b-a)/k$ – шаг интегрирования;

$x_i^{(j)}$ – i-я абсцисса j-го шага интегрирования;

$x_0 = a, \qquad x_j = x_{j-1} + h, \quad j = 1, 2, \ldots, k;$

$x_i^{(j)} = x_{j-1} + X_i h, \quad i = 1, 2, \ldots, n;$

X_i, A_i - абсциссы и коэффициенты формулы Гаусса.

При одинаковом числе ординат формула Гаусса дает большую точность, чем другие формулы, но она менее эффективна в методах с автоматическим выбором шага интегрирования.

Самой простой квадратурной формулой является формула прямоугольников ($n=1,\ X_1=0.5,\ A_1=1.0$)

$$\int_a^b f(x)dx \cong (b-a) f\left(\frac{a+b}{2}\right).$$

Если отрезок интегрирования большой и разбит на k равных частей, то составная формула прямоугольников имеет вид

$$\int_a^b f(x)dx \cong h\sum_{j=1}^{k} f(x_i),$$

где $h=(b-a)/k$; $x_1=a+h/2$; $x_j=x_{j-1}+h$; $j=2,3,\ldots,k$.

На практике используются также модификации общей формулы: формула «левых» прямоугольников ($x_1=a$) и формула «правых» прямоугольников ($x_1=a+h$).

Приведем две группы алгоритмов для выполнения заданий в лабораторной работе.

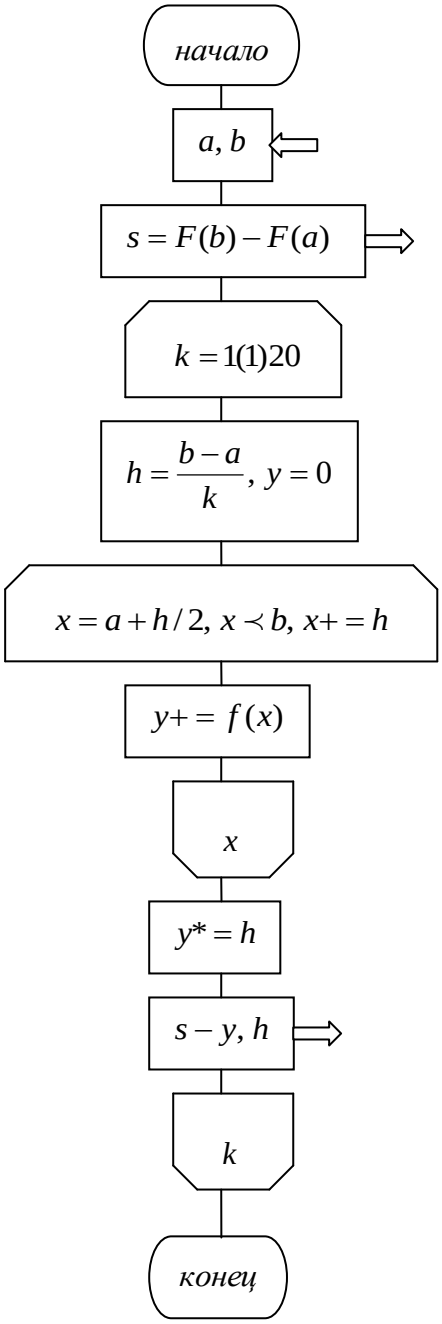


Рис. 7.1. Алгоритм получения зависимости фактической ошибки интегрирования функции по формуле прямоугольников от шага интегрирования

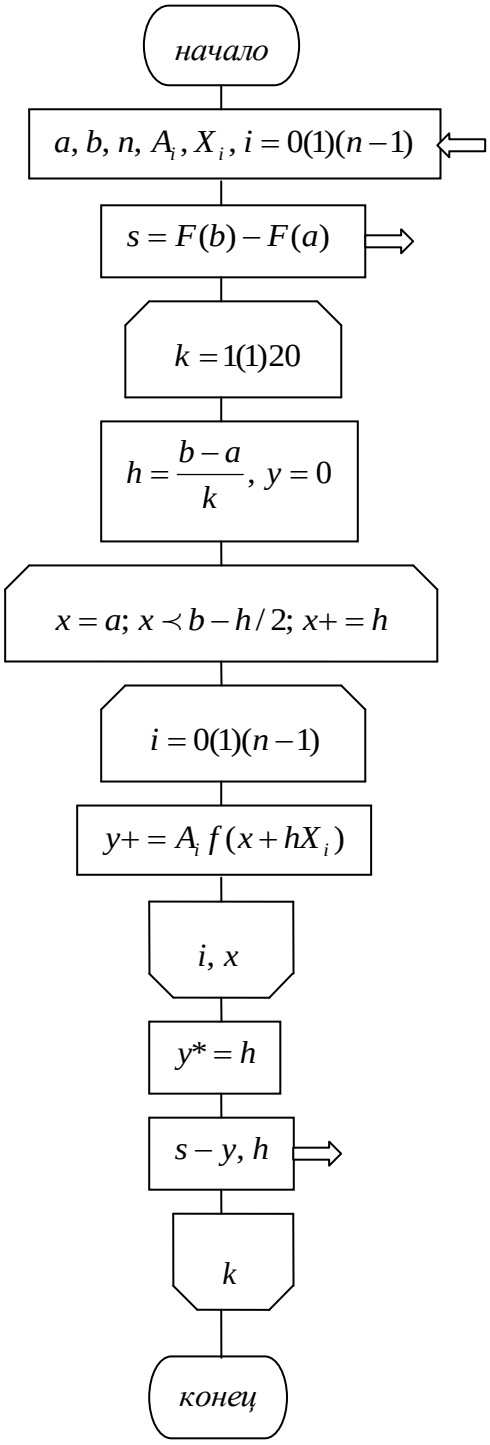


Рис. 7.2. Алгоритм получения зависимости фактической ошибки интегрирования функции методом Гаусса от шага интегрирования

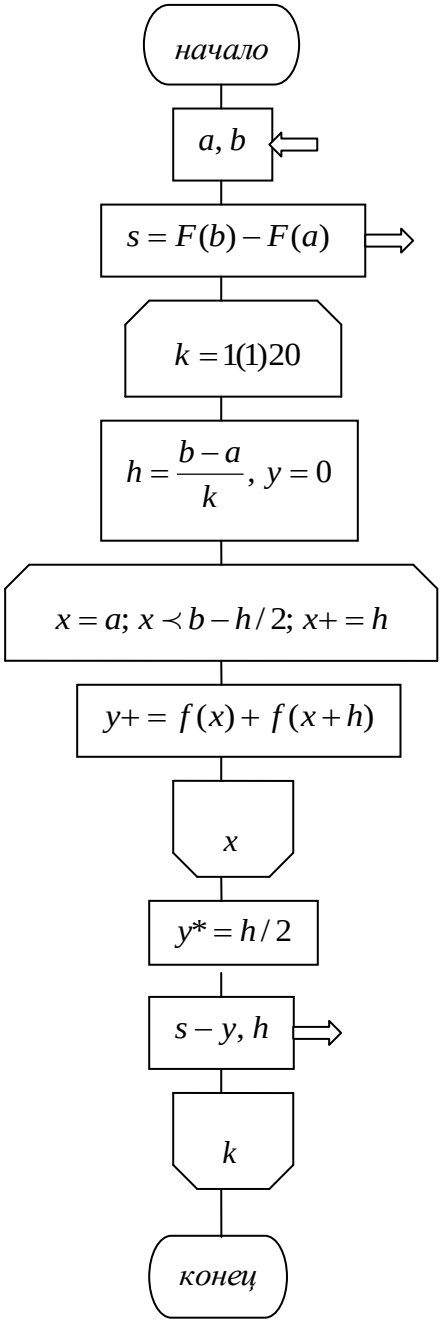


Рис. 7.3. Алгоритм получения зависимости фактической ошибки интегрирования функции по формуле трапеций от шага интегрирования

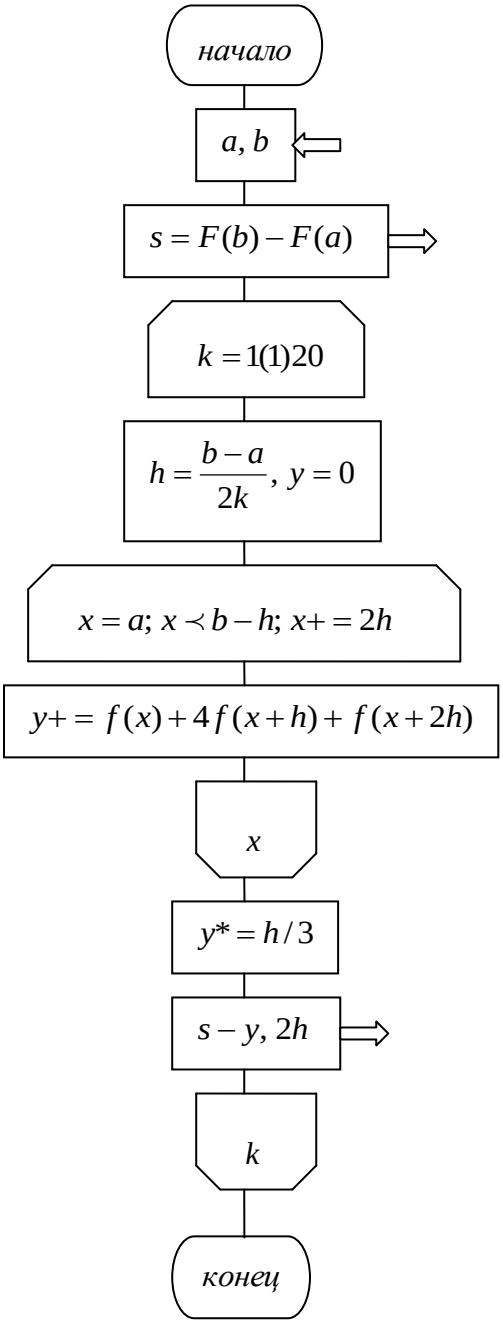


Рис. 7.4. Алгоритм получения зависимости фактической ошибки интегрирования функции по формуле Симпсона от шага интегрирования

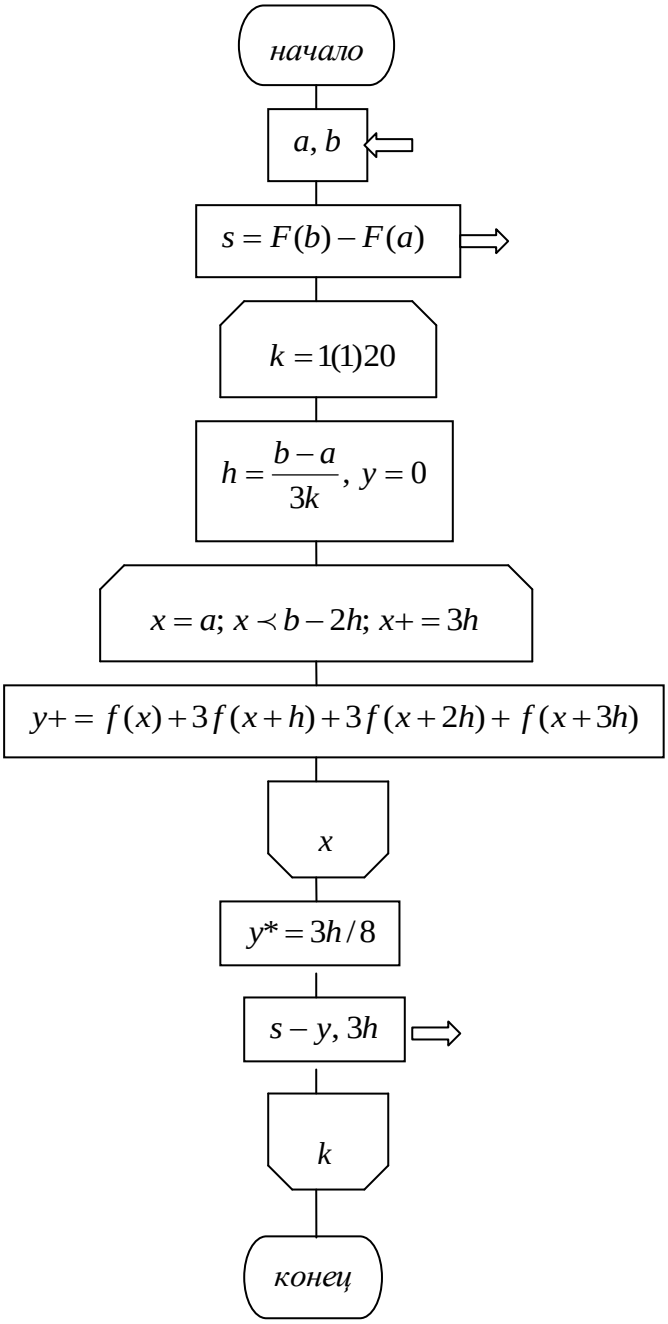


Рис. 7.5. Алгоритм получения зависимости фактической ошибки интегрирования функции по формуле Ньютона от шага интегрирования

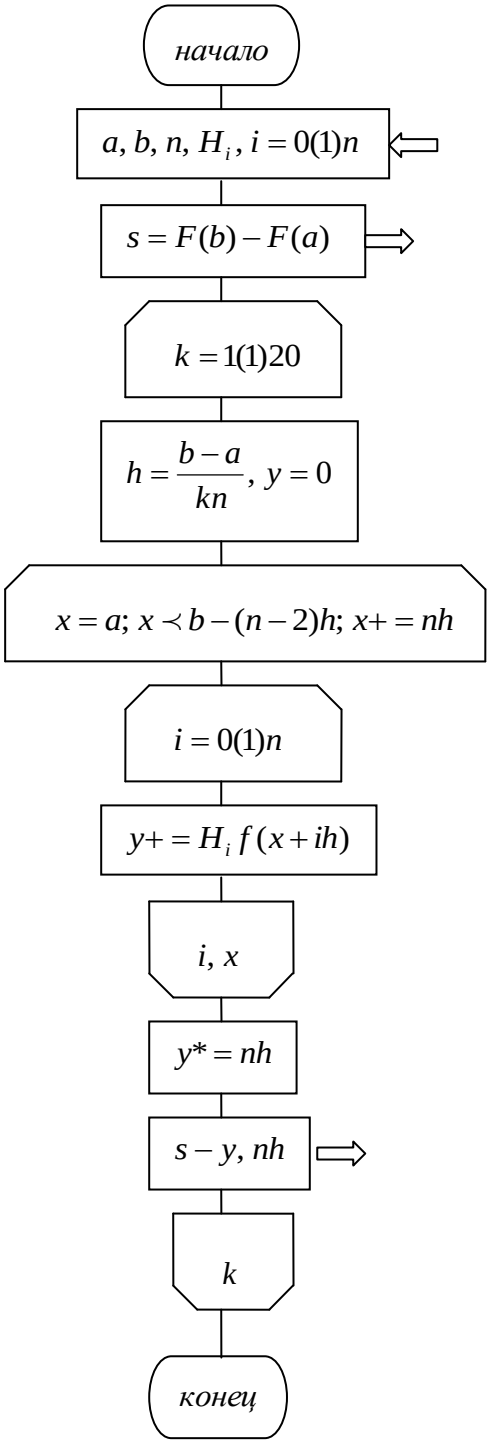


Рис. 7.6. Алгоритм получения зависимости фактической ошибки интегрирования функции методом Ньютона-Котеса от шага интегрирования

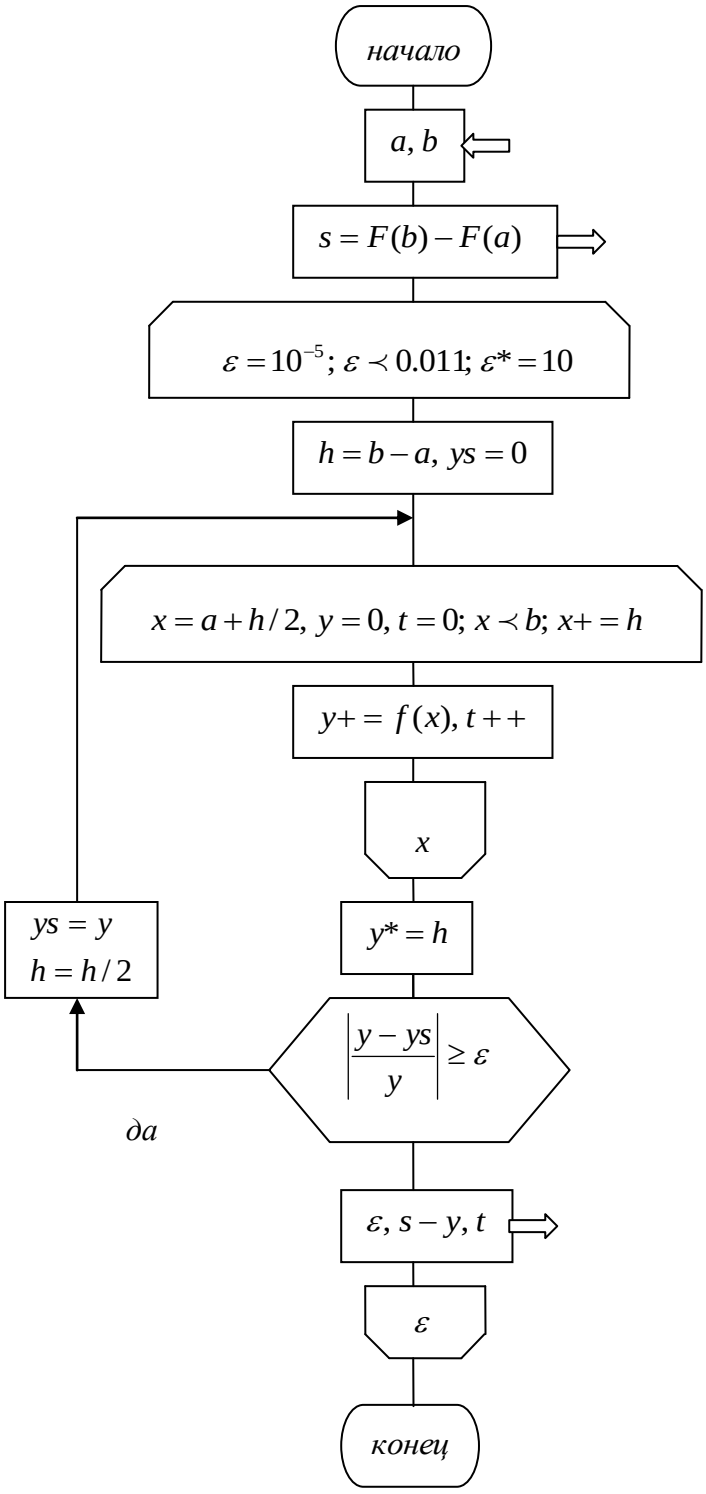


Рис. 7.7. Адаптивный алгоритм получения зависимостей фактической ошибки и временных затрат при интегрировании функции по формуле прямоугольников от задаваемой ошибки

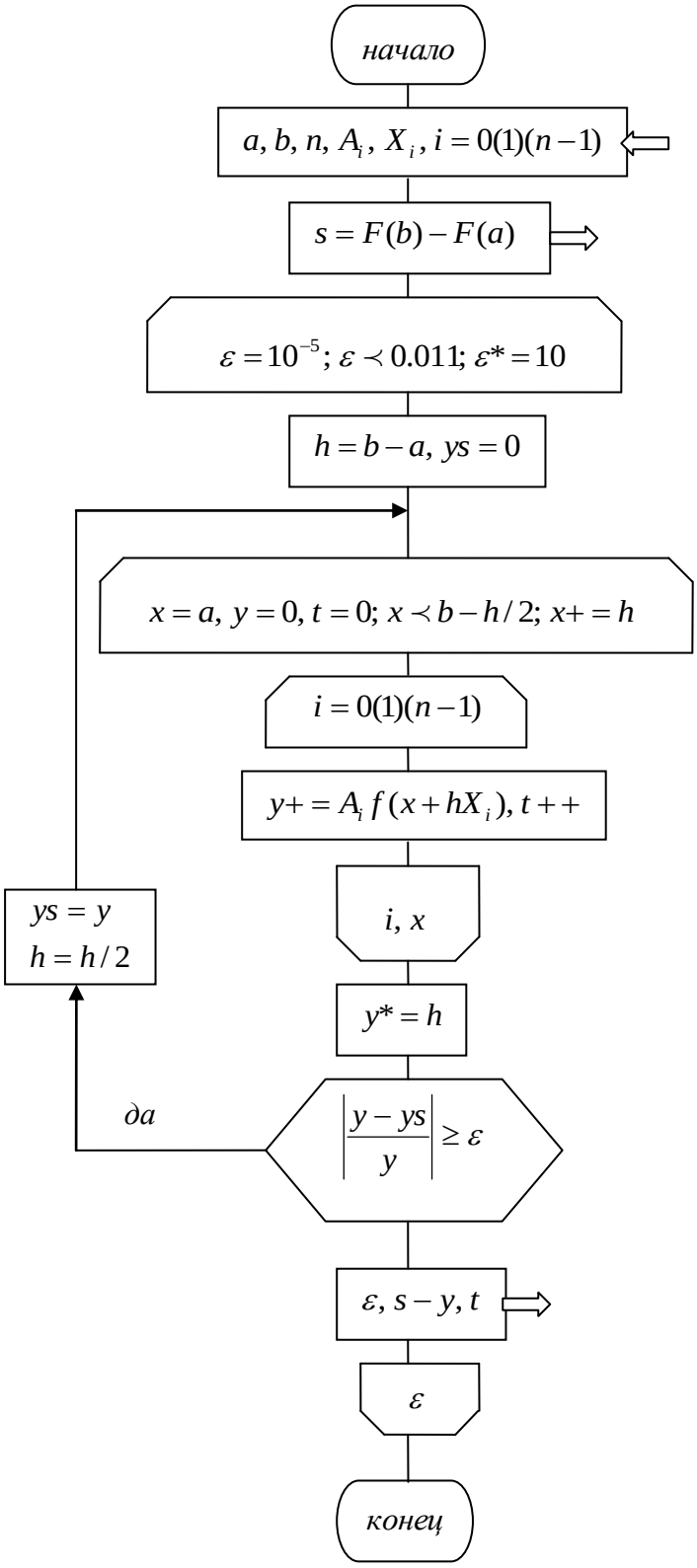


Рис. 7.8. Адаптивный алгоритм получения зависимостей фактической ошибки и временных затрат при интегрировании функции методом Гаусса от задаваемой ошибки

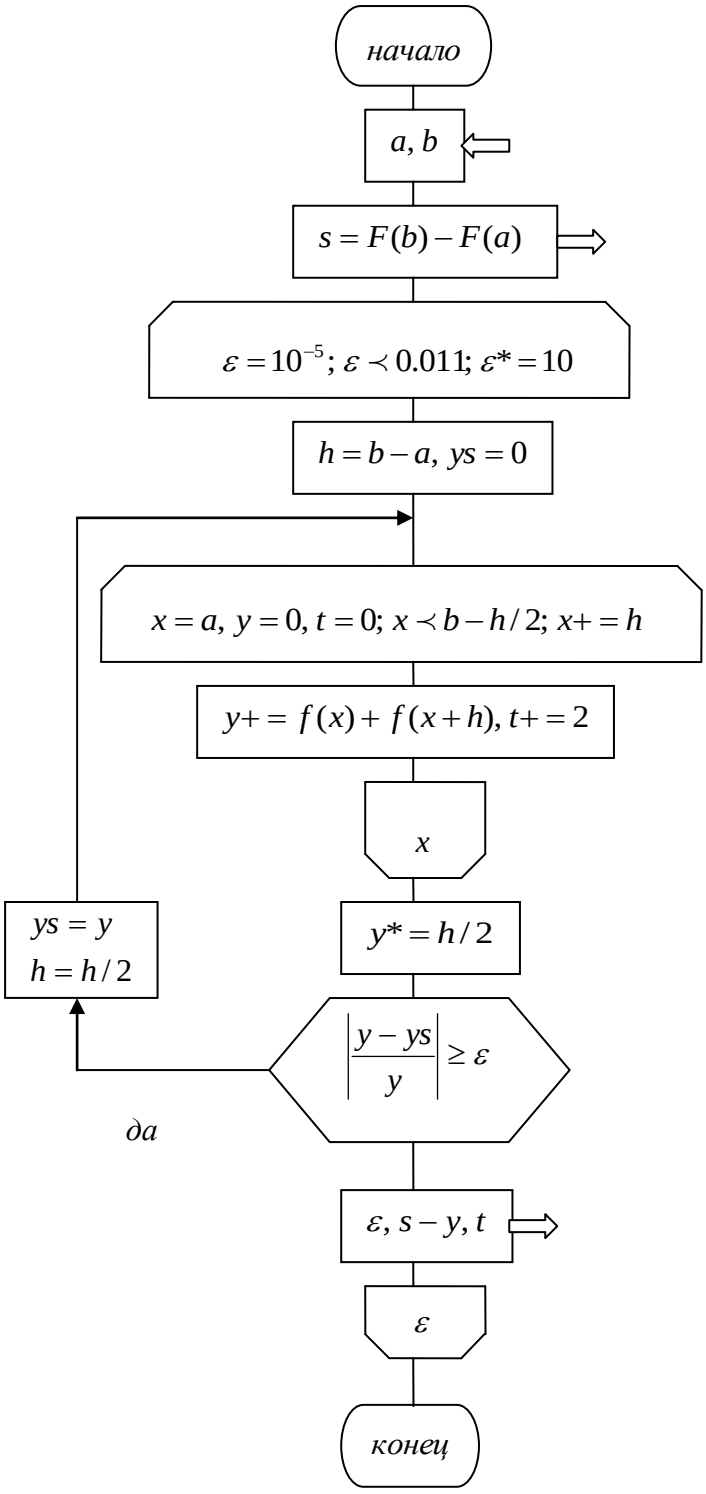


Рис. 7.9. Адаптивный алгоритм получения зависимостей фактической ошибки и временных затрат при интегрировании функции по формуле трапеций от задаваемой ошибки

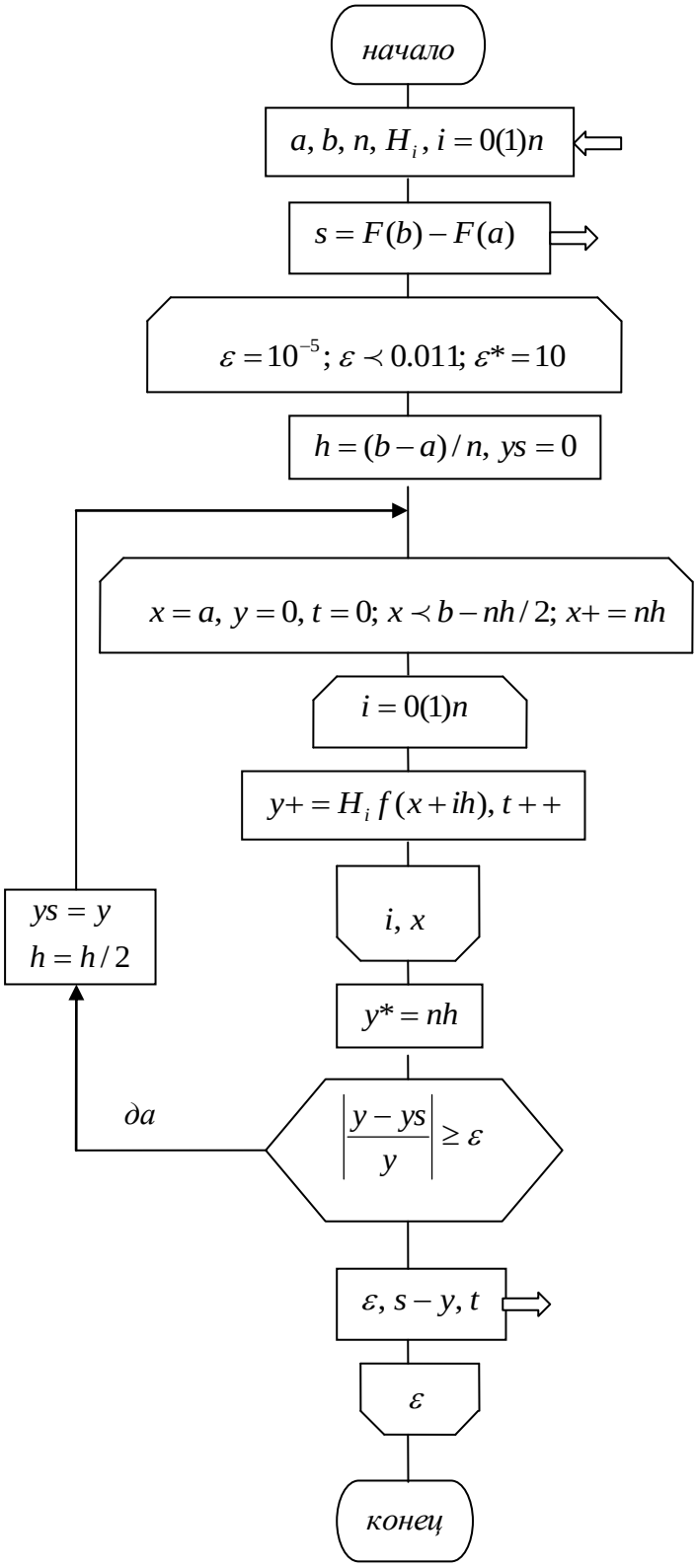


Рис. 7.10. Адаптивный алгоритм получения зависимостей фактической ошибки и временных затрат при интегрировании функции методом Ньютона-Котеса от задаваемой ошибки

Алгоритмы для получения зависимости фактической ошибки вычисления интеграла от шага интегрирования

Эта группа алгоритмов представлена на рис. 7.1…7.6. Алгоритмы на рис. 7.1, 7.3, 7.4, 7.5 соответствуют частным случаям формул Гаусса и Ньютона-Котеса. Алгоритмы на рис. 7.2, 7.6 построены по формулам Гаусса и Ньютона-Котеса в общем случае.

Исходными данными являются: подынтегральная $f(x)$ и первообразная $F(x)$ функции, пределы интегрирования a, b (вводятся). В алгоритме использованы переменные: s – точное значение интеграла, вычисленное по формуле Ньютона-Лейбница, k – число шагов интегрирования, h – шаг интегрирования, y – приближенное значение интеграла. Как результат, выводятся фактическая ошибка вычисления интеграла $s - y$ и шаг интегрирования h (или nh).

Алгоритмы для получения зависимостей фактической ошибки и временных затрат вычисления интеграла от задаваемой ошибки интегрирования

Эти алгоритмы (рис. 7.7…7.10) являются адаптивными, т. к. здесь шаг интегрирования автоматически устанавливается, начиная со значения $h = b - a$, таким, чтобы выполнилось условие: $\left|\frac{y - ys}{y}\right| < \varepsilon$ – относительная ошибка интегрирования меньше заданной ε. Для изменения величины шага используется правило Рунге. Через ys обозначено предыдущее значение интеграла; в начале вычислений оно может принято равным любому числу, здесь – нулю. Затраты машинного

времени в алгоритме оцениваются косвенно – по числу вызовов подынтегральной функции $f(x)$, для чего использована переменная t.

Адаптивные алгоритмы для других частных случаев можно легко составить по аналогии с представленными алгоритмами.

Вычисление первообразных и интегралов в системе Mathcad

Для получения первообразной (символического вычисления) по записи подынтегральной функции, например, по записи $1+\sin(x-y)-x^2-\cos(y)$, нужно поместить курсор под переменную, по которой интегрируют, например, под x (в любом месте), и ввести команду *Символика-Переменная-Интеграция*.

Для вычисления интегралов можно использовать в меню *Математика* ручной режим вычислений (команда *Калькуляция* либо клавиша *F9*, либо кнопка = на панели инструментов) или автоматический режим вычислений (команда *Автоматическая калькуляция*). Автоматический режим установлен по умолчанию. В меню символьных вычислений *Символика* для вычисления интегралов используется команда *Вычисление*.

Кроме того, командой *Упрощение* в меню *Символика* можно получить аналитическое выражение для интеграла, например при вычислении интеграла с переменным верхним пределом.

Примечание. Во всех случаях для записи интегралов используется шаблон определенного интеграла.

ЛАБОРАТОРНАЯ РАБОТА 7

ИССЛЕДОВАНИЕ МЕТОДОВ ВЫЧИСЛЕНИЯ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

ЗАДАНИЕ

1. Варианты вычисляемых интегралов и методов (формул) вычислений представлены в табл. 1 и 2.
2. В *Mathcad*'е найти значение интеграла и первообразную по записи подынтегральной функции.
3. Разработать алгоритм и код, позволяющий:

По формуле Ньютона-Лейбница вычислить точное значение интеграла.

Для нечетных вариантов получить зависимости фактической ошибки вычисления интеграла (равна разности между точным и приближенным значениями интеграла) от шага интегрирования. Диапазон изменения числа шагов интегрирования 1…20.

Для четных вариантов, используя автоматический выбор шага интегрирования, получить зависимости временных затрат и фактической ошибки вычисления интеграла от задаваемой ошибки интегрирования (диапазон изменения ошибки 0.01…0.0001).

Таблица 1

Вариант	Вычисляемый интеграл	Первообразная
1	$\int_{-0.5}^{2.0} \frac{x^2}{1+x}\,dx$	$\frac{x^2}{2} - x + \ln(1+x)$
2	$\int_{-1.2}^{1.5} \frac{x}{1+x^4}\,dx$	$\frac{1}{2} arctg\left(x^2\right)$
3	$\int_{-0.8}^{0.9} \frac{2x}{1-x^4}\,dx$	$\frac{1}{2}\ln\frac{1+x^2}{1-x^2}$

4	$\int_{0.1}^{3.0} \frac{dx}{x\left(1+x^2\right)}$	$\ln \frac{x}{\sqrt{1+x^2}}$
5	$\int_{0}^{2.5} \frac{1+2x}{(2+x)\sqrt{2+x}} dx$	$\frac{14+4x}{\sqrt{2+x}}$
6	$\int_{0}^{2.8} x^3 e^x dx$	$e^x\left(x^3-3x^2+6x-6\right)$
7	$\int_{-0.5}^{1.5} \frac{\sin^3 x}{\cos x} dx$	$\frac{1}{2}\cos^2 x - \ln(\cos x)$
8	$\int_{-\pi/4}^{\pi/4} \frac{\cos(3x)}{\cos^2 x} dx$	$4\sin x - 3\ln\left(tg\left(\frac{\pi}{4}+\frac{x}{2}\right)\right)$
9	$\int_{-0.5}^{1.2} \frac{tgx}{1+2tg^2 x} dx$	$\frac{1}{2}\ln\left(\cos^2 x + 2\sin^2 x\right)$
10	$\int_{-0.4}^{1.2} \frac{\sin x}{\sqrt{1+3\sin^2 x}} dx$	$-\frac{1}{\sqrt{3}}\arcsin\left(\frac{\sqrt{3}}{2}\cos x\right)$
11	$\int_{0}^{\pi/2} x^2 \sin^2 x dx$	$\frac{x^3}{6} - \frac{x}{4}\cos(2x) - \frac{1}{4}\left(x^2-\frac{1}{2}\right)* *\sin(2x)$
12	$\int_{0}^{\pi/2} \frac{x+\sin x}{1+\cos x} dx$	$xtg\frac{x}{2}$
13	$\int_{-\pi/2}^{\pi} x^3 \cos x^2 dx$	$\frac{x^2}{2}\sin x^2 + \frac{1}{2}\cos x^2$
14	$\int_{1.2}^{2.0} \frac{1}{x\ln x} dx$	$\ln(\ln x)$
15	$\int_{0.5}^{4.0} x^2 \ln x dx$	$x^3\left(\frac{\ln x}{3}-\frac{1}{9}\right)$
16	$\int_{0.2}^{3.3} x^3 \ln\left(x^2+1\right) dx$	$\frac{1}{4}\left(\left(x^4-1\right)\ln\left(x^2+1\right) - \frac{x^4}{2}+x^2\right)$
17	$\int_{0}^{4.0} xarctg\left(\frac{x}{2}\right) dx$	$\frac{1}{2}\left(x^2+4\right)arctg\left(\frac{x}{2}\right) - x$
18	$\int_{0}^{1.5} \frac{\sin x}{\sqrt{1+2\sin^2 x}} dx$	$-\frac{1}{\sqrt{2}}\arcsin\left(\sqrt{\frac{2}{3}}\cos x\right)$
19	$\int_{0.5}^{1.2} \frac{chx}{\sqrt{3+sh^2 x}} dx$	$\ln\left(shx + \sqrt{3+sh^2 x}\right)$

20	$\int_{0.2\pi}^{0.9\pi} \frac{\cos(20x)}{\sin x} dx$	$2\sum_{k=1}^{10} \frac{\cos(2k-1)x}{2k-1} + \ln\left(tg\left(\frac{x}{2}\right)\right)$
21	$\int_{0}^{3} \frac{x}{1+chx} dx$	$2(x - \frac{x}{1+e^x} - \ln(1+e^x))$
22	$\int_{0}^{\pi/4} \frac{dx}{3+tg^2 x}$	$\frac{1}{3}\left(x - \sqrt{\frac{1}{3}} arctg\left(\sqrt{\frac{1}{3}} tgx\right)\right)$
23	$\int_{0}^{\pi/3} x^2 \cos^2 x dx$	$\frac{x^3}{6} + \frac{x}{4}\cos(2x) + \frac{1}{4}\left(x^2 - \frac{1}{2}\right) * * \sin(2x)$
24	$\int_{-\pi/4}^{\pi/4} \frac{x\sin x}{\cos^2 x} dx$	$\frac{x}{\cos x} - \ln\left(tg\left(\frac{x}{2} + \frac{\pi}{4}\right)\right)$
25	$\int_{\pi/6}^{\pi/3} xctg^2 x dx$	$-xctgx + \ln(\sin x) - \frac{x^2}{2}$
26	$\int_{0}^{\pi/2} \frac{x\cos x}{(1+\sin x)^2} dx$	$-\frac{x}{1+\sin x} + tg\left(\frac{x}{2} - \frac{\pi}{4}\right)$
27	$\int_{0.5}^{1.5} e^{2x} \sin 3x dx$	$\frac{1}{13} e^{2x}(2\sin 3x - 3\cos 3x)$
28	$\int_{1}^{4} \ln(x^2+3) dx$	$x\ln(x^2+3) - 2x + 2\sqrt{3} arctg \frac{x}{\sqrt{3}}$
29	$\int_{1}^{5} \frac{\ln x}{(4+5x)^2} dx$	$-\frac{\ln x}{5(4+5x)} + \frac{1}{20}\ln\frac{x}{4+5x}$
30	$\int_{0.5}^{1.5} x \arcsin\frac{x}{2} dx$	$\left(\frac{x^2}{2} - 1\right)\arcsin\frac{x}{2} + \frac{x}{4}\sqrt{4-x^2}$

Таблица 2

Вариант	Метод (формула) вычислений
1,9,17,25	прямоугольников (Гаусса для n=1) , «левых» прямоугольников, «правых» прямоугольников, трапеций
2,10,18,26	Симпсона, Гаусса для n=2
3,11,19,27	«трех восьмых», Гаусса для n=3

4,12,20,28	Ньютона-Котеса для n=4, Гаусса для n=4
5,13,21,29	Ньютона-Котеса для n=5, Гаусса для n=5
6,14,22,30	Ньютона-Котеса для n=6, Гаусса для n=6
7,15,23,	Ньютона-Котеса для n=7, Гаусса для n=7
8,16,24,	Ньютона-Котеса для n=8, Гаусса для n=8

Пример выполнения задания в среде *C++Builder6*

1. Вычислить точное значение определенного интеграла

$$\int_{0.5}^{1.5} x \arcsin\frac{x}{2}\,dx$$

по формуле Ньютона-Лейбница. Первообразная функция имеет вид

$$\left(\frac{x^2}{2}-1\right)\arcsin\frac{x}{2}+\frac{x}{4}\sqrt{4-x^2} \quad .$$

2. Получить зависимость фактической ошибки вычисления интеграла (равна разности между точным и приближенным значениями интеграла) от шага интегрирования. Диапазон изменения числа шагов интегрирования 1…20.

3. Получить зависимости фактической ошибки вычисления интеграла и временных затрат от задаваемой ошибки интегрирования.

Примечания. 1. Для п. 2 и п. 3 задается *n* – количество узлов интегрирования. 2. Приближенные значения интеграла вычисляются по формулам Ньютона-Котеса и Гаусса. 3. Для проектирования приложения по п. 2 и п. 3 полезно иметь зависимости от *n* фактической ошибки вычисления интеграла по формулам Ньютона-Котеса и Гаусса.

1. Создайте новый проект командой *Файл/Новый/Приложение.*

2. Сохраните файлы модуля и проекта командой *Файл*/*Сохранить все* под именами LR7 и PR_LR7.

3. Для создания интерфейса воспользуйтесь компонентом **MainMenu** (страница *Стандарт*) – *главное меню*, и свойствами компонентов **Enabled** – *доступный* и **Visible –** *видимый***.**

4. Для ускорения проектирования контролируйте правильность своих действий по рис. 7.11, 7.12 и 7.13, на которых приведены результаты выполнения задания.

5. Для выполнения п.1 задания перенесем на форму компоненты: метку **Label1** (страница *Стандарт*), кнопки **Button1** и **Button2** (страница *Стандарт*) и три метки: **LabeledEdit1, LabeledEdit2** и **LabeledEdit3** (страница *Дополнительно*). В свойство **Caption** метки **Label1** впишите подынтегральную функцию, метки **LabeledEdit1** и **LabeledEdit2** служат для задания пределов интегрирования, **LabeledEdit3 –** для вывода точного значения интеграла. При щелчке на кнопке **Button1** (*расчет*) вычисляется интеграл по формуле Ньютона-Лейбница, щелчок на кнопке **Button2** (*конец*) означает перевод приложения из режима выполнения в режим проектирования. Для свойства **Enabled** кнопки **Button1** установите значение **false**.

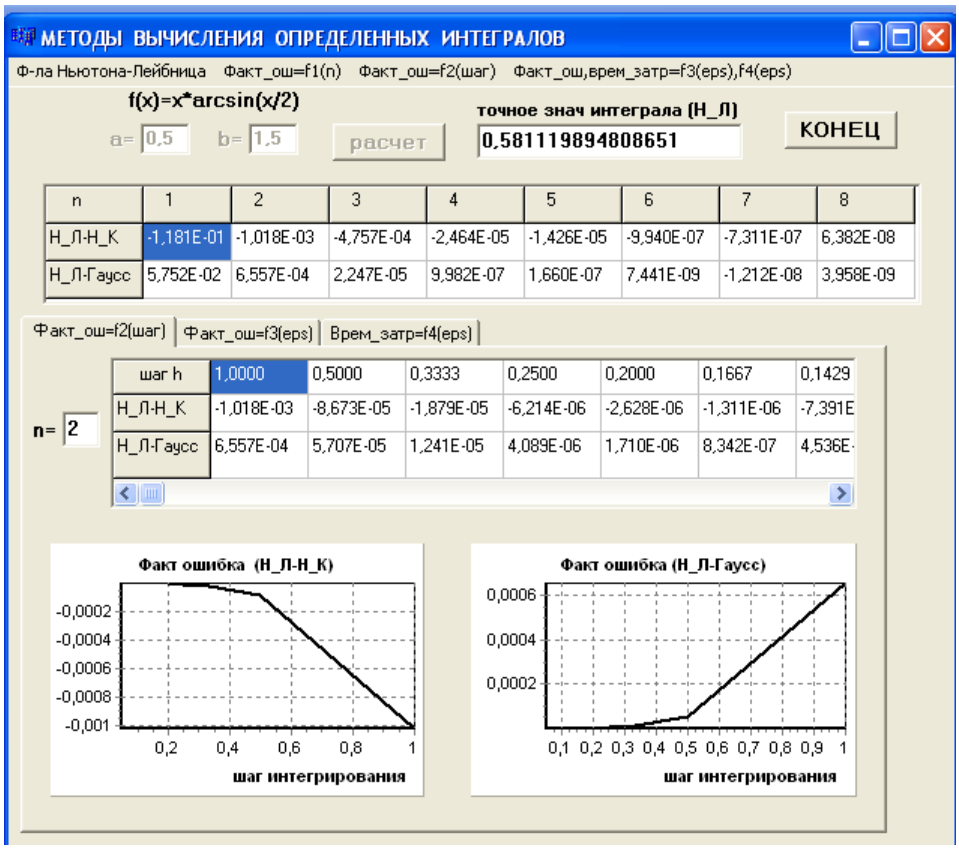


Рис. 7.11. Форма с результатами (вид 1)

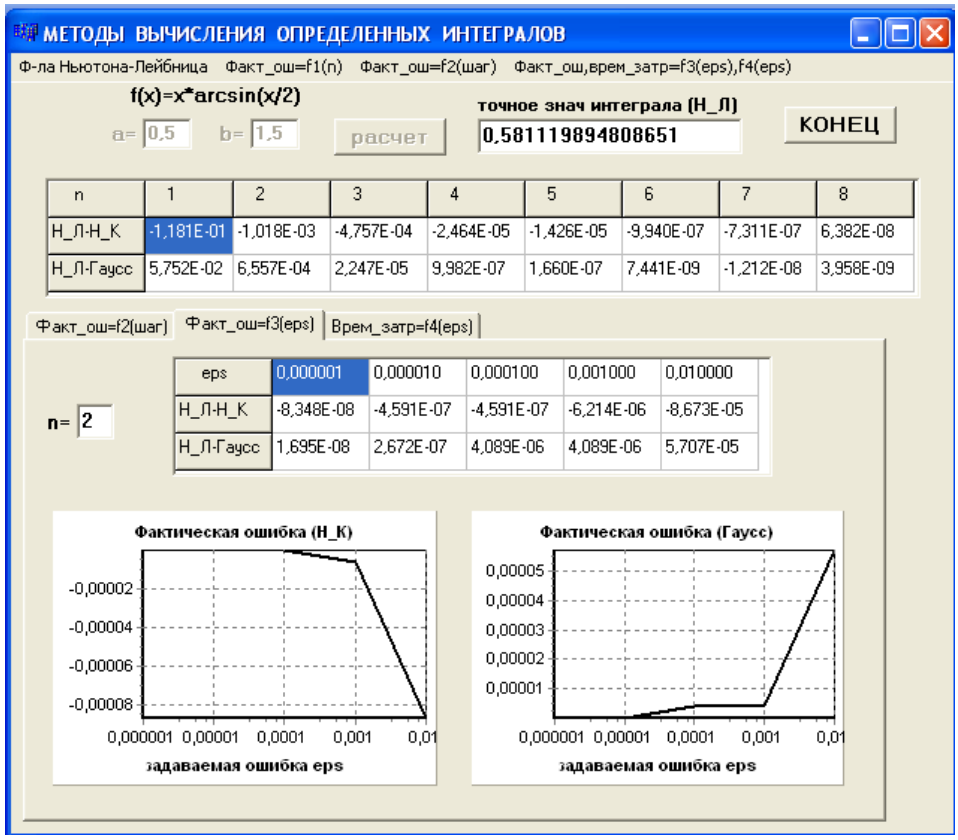


Рис. 7.12. Форма с результатами (вид 2)

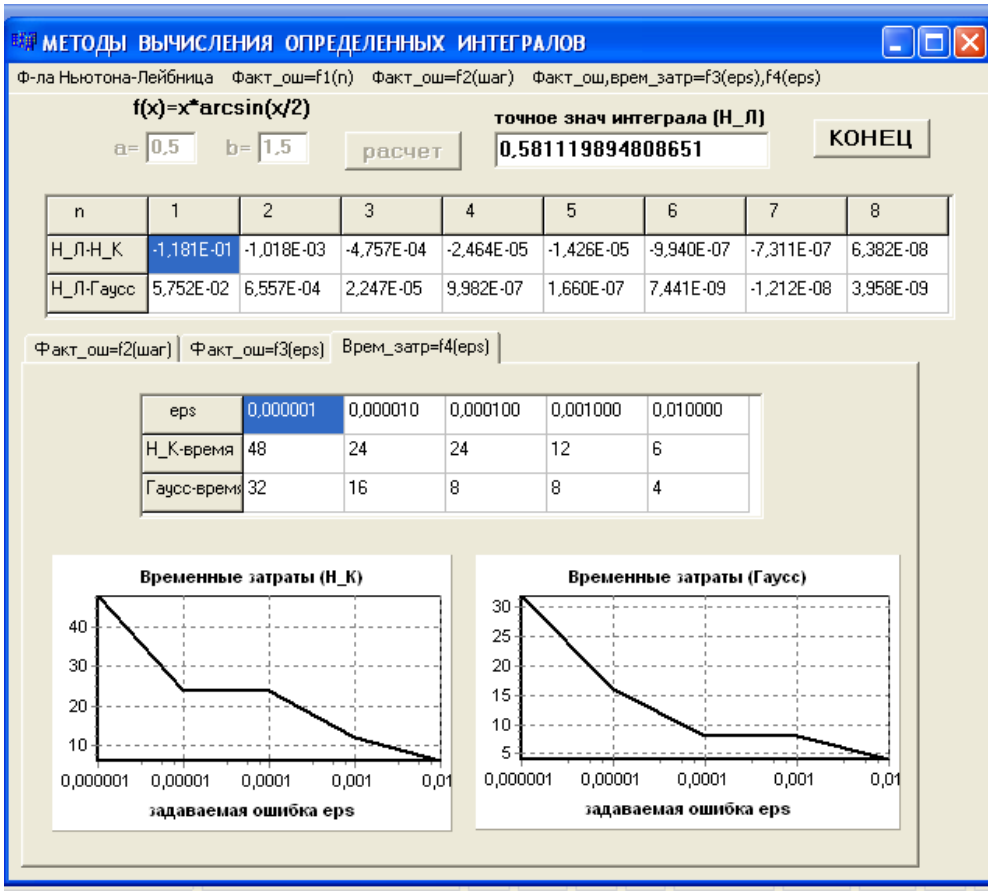


Рис. 7.13. Форма с результатами (вид 3)

6. Кроме того, для выполнения п. 1 задания необходимо в файл реализации *LR7.cpp* включить директиву *#include<math.h>*, функцию для вычисления первообразной

```
double fp(double x)
{return (x*x/2-1)*asin(x/2)+x/4*sqrt(4-x*x);}
```

и обработчик щелчка на кнопке **Button1** (*расчет*), где вычисляется точное значение интеграла (*tin*):

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 a=StrToFloat(LabeledEdit1->Text);
  b=StrToFloat(LabeledEdit2->Text);
  tin=fp(b)-fp(a);
```

LabeledEdit3->Text=FloatToStr(tin);

f1->Enabled=true;

f2->Enabled=true;

f3->Enabled=true;

LabeledEdit1->Enabled=false;

LabeledEdit2->Enabled=false;

Button1->Enabled=false;

}

После щелчка на кнопке **Button1** (*расчет*) компоненты **Button1** (*расчет*), **LabeledEdit1** (**a=**) и **LabeledEdit2** (**b=**) становятся недоступными.

7. Перенесите на форму компонент **MainMenu** (страница *Стандарт*), двойным щелчком на нем перейдите в *Проектировщик Меню* (окно **Form1->MainMenu1**) и сконструируйте меню с указанными разделами и подразделами; названия записываются в свойство **Caption** разделов и подразделов:

Ф-ла Ньютона-Лейбница *Факт_ош=f1(n)* *Факт_ош=f2(шаг)* *Факт_ош,врем_затр=f3(eps),f4(eps)*

Ньютон-Котес *Ньютон-Котес* *Ньютон-Котес*

Гаусс *Гаусс* *Гаусс*

8. Выделите раздел *Ф-ла Ньютона-Лейбница*. Значения свойств **Enabled** и **Visible** оставьте в **true**. Среда *Builder* в качестве значения свойства **Name** разделов и подразделов ставит номера: *N1*, *N2*, и т. д. При проектировании рекомендуется присваивать осмысленные имена, поэтому в свойство **Name** впишите *N_L*. Обработчик щелчка на этом разделе по окончании проектирования будет содержать (*курсив*):

void __fastcall TForm1::N_LClick(TObject *Sender)

{ *f1->Enabled=false;*

f2->Enabled=false;

f3->Enabled=false;

```
 StringGrid1->Visible=false;
 PageControl1->Visible=false;
 LabeledEdit1->Enabled=true;
 LabeledEdit2->Enabled=true;
 Button1->Enabled=true;
 LabeledEdit3->Text="";
 for(int i=0;i<StringGrid1->RowCount;i++)
  for(int j=0;j<StringGrid1->ColCount;j++)
    StringGrid1->Cells[j][i]="";
 for(int i=0;i<StringGrid2->RowCount;i++)
  for(int j=0;j<StringGrid2->ColCount;j++)
    StringGrid2->Cells[j][i]="";
 for(int i=0;i<StringGrid3->RowCount;i++)
  for(int j=0;j<StringGrid3->ColCount;j++)
    StringGrid3->Cells[j][i]="";
 for(int i=0;i<StringGrid4->RowCount;i++)
  for(int j=0;j<StringGrid4->ColCount;j++)
    StringGrid4->Cells[j][i]="";
 Series1->Clear();
 Series2->Clear();
 Series3->Clear();
 Series4->Clear();
 Series5->Clear();
 Series6->Clear();
}
```

9. Свойству **Enabled** остальных разделов присвойте значение **false**, свойству **Name** остальных разделов и их подразделов присвойте следующие значения соответственно: *f1, f1_N_K, f1_G; f2, f2_N_K, f2_G; f3, f3_N_K, f3_G.* Выполнение приложения может начаться только с

головного раздела меню *Ф-ла Ньютона-Лейбница*, поскольку остальные разделы меню и кнопка **Button1** (*расчет*) в этот момент недоступны. После щелчка на разделе *Ф-ла Ньютона-Лейбница* кнопка станет доступной.

10. Раздел меню *Факт_ош=f1(n)* предназначен для выполнения п. 3 примечания; для вывода результатов поместите на форму компонент **StringGrid1** (страница *Дополнительно*). Названия строк *n*, *Н_Л-Н_К*, *Н_Л-Гаусс* будут заданы во время выполнения приложения. Установите следующие значения свойств компонента **StringGrid1**: **ColCount** – 9, **FixedCols** – 1, **FixedRows** – 1, **Font** – черный, обычный, размер 8, **RowCount** – 3, **Visible** – **false**.

11. Разделы меню *Факт_ош=f2(шаг)* и *Факт_ош,врем_затр=f3(eps), f4(eps)* предназначены для выполнения п. 2 и п. 3 задания. Для выполнения п. 2 и п. 3 задания необходимо разместить на форме большое количество компонентов, имеющих значительные размеры. Использование многостраничной панели – компонента **PageControl1** (страница *Win32)* позволяет преодолеть это затруднение. Перенесите компонент **PageControl1** на форму. Щелкните на нем правой кнопкой мыши и во всплывшем меню трижды используйте команду *Новая страница*. В свойство **Caption** первой страницы впишите *Факт_ош=f2(шаг)*, второй – *Факт_ош=f3(eps)*, третьей – *Врем_затр=f4(eps)*. Установите свойства компонента **PageControl1**: **MultiLine** – **false**, **Style** – **tsTabs**, **TabPosition** – **tpTop**, **Visible** – **false**. Перенесите на первую страницу (*Факт_ош=f2(шаг)*) компоненты: **LabeledEdit4** (**n=)** – для указания числа узлов интегрирования, **StringGrid2 (**страница *Дополнительно*) – для таблицы с результатами расчетов и компоненты **Chart1** и **Chart2** (страница *Additional)* для представления в графическом виде результатов расчетов. В свойство **Text** компонента **LabeledEdit4** занесите число 2. Установите следующие

значения свойств компонента **StringGrid2**: **ColCount** – 21, **FixedCols** – 1, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 3.

12. Задайте свойства компонента **Chart1**. Щелкните правой кнопкой мыши на компоненте **Chart1** и в появившемся меню выберите *Edit Chart*…. На экране появится окно *Редактора Диаграмм* (*Editing Chart1*) с открытой страницей *Chart,* которая имеет несколько закладок. В данный момент открыта закладка *Series*. Щелкните на кнопке *Add*… – добавить серию. В появившемся окне выберите тип графика – *Line* и выключите индикатор *3D*. На закладке *Panel*, нажав кнопку *Panel Color*…, выберите белый цвет. На закладке *Legend* выключите индикатор *Visible*. Перейдите на закладку *Titles*. В окне редактирования, которое в данный момент соответствует *Title* – заголовку графика, сотрите *TChart* и напишите (шрифт *Font*… - черный, жирный, размер 8) ***Факт ошибка (Н_Л-Н_К).*** Цвет фона *Back Color*.. установите белый. В выпадающем списке от окна редактирования *Title* перейдите в окно редактирования *Foot* и напишите тем же шрифтом ***шаг интегрирования***. Цвет фона *Back Color*.. также установите белый. Перейдите на закладку *Axis*. В группе кнопок *Axis* нажата кнопка *Left* и открыта закладка *Scales*. Нажмите кнопку *Change*… и задайте *Increment* равным 1E-12. На закладке *Labels* в значение *Values Format* добавьте справа 9 символов #; оно станет равным # ##0,###########. Затем в группе кнопок *Axis* нажмите кнопку *Bottom* и задайте *Increment* равным 0,01. Перейдите со страницы *Chart* на страницу *Series*. Здесь на закладке *Format* в группе *Line* нажмите *Border*… и установите *Width* равным 2. Нажмите кнопку *Close* и выйдите из режима редактирования компонента **Chart1.** Свойства компонента **Chart2** задаются так же.

13. Перенесите на вторую страницу (*Факт_ош=f3(eps)*) компоненты: **LabeledEdit5** (**n=)** – для указания числа узлов интегрирования, **StringGrid3 (**страница *Дополнительно*) – для таблицы с результатами

расчетов и компоненты **Chart3** и **Chart4** (страница *Additional)* для представления в графическом виде результатов расчетов. В свойство **Text** компонента **LabeledEdit5** занесите число 2. Установите следующие значения свойств компонента **StringGrid3**: **ColCount** – 6, **FixedCols** – 1, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 3.

14. Свойства компонентов **Chart3** и **Chart4** задаются так же, как **Chart1** и **Chart2**. Отличие состоит в задании характеристик оси абсцисс. На закладке *Axis* в группе кнопок *Axis* нажмите кнопку *Bottom*, включите индикатор *Logarithmic* и установите *Minimum* – 0,000001; *Maximum* – 0,01. На закладке *Labels* в значение *Values Format* добавьте справа 7 символов #; оно станет равным # ##0,##########.

15. Перенесите на третью страницу (*Врем_зamp=f4(eps)*) компоненты **StringGrid4 (**страница *Дополнительно*) – для таблицы с результатами расчетов и компоненты **Chart5** и **Chart6** (страница *Additional)* для представления в графическом виде результатов расчетов. Установите значения свойств компонента **StringGrid4** равными значениям свойств компонента **StringGrid3**: **ColCount** – 6, **FixedCols** – 1, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 3. Свойства компонентов **Chart5** и **Chart6** устанавливаются проще, чем свойства компонентов **Chart3** и **Chart4** – характеристики оси ординат (нажата кнопка *Left*) сохраняются заданными по умолчанию.

16. По окончании проектирования файл *LR7.cpp* может выглядеть так:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "LR7.h"
#include<math.h>
//---------------------------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
double a,b,tin;
double nkh[][9]={{0.5,0.5},{0.1666667,0.6666667,0.1666667},
{0.125,0.375,0.375,0.125},{0.0777778,0.3555556,0.1333333,
0.3555556,0.0777778},
{0.0659722,0.2604166,0.1736111,0.1736111,0.2604166,0.0659722},
{0.0488095,0.2571428,0.0321428,0.3238095,0.0321428,0.2571428,
0.0488095},
{0.0434606,0.2070023,0.0765626,0.1729745,0.1729745,0.0765626,
0.2070023,0.0434606},{0.0348853,0.2076895,-0.0327336,
0.3702292,-0.1601410,0.3702292,-0.0327336,0.2076895,0.0348853}};
double gx[][8]={{0.5},{0.2113249,0.7886751},{0.1127017,0.5,0.8872983},
{0.0694318,0.3300095,0.6699905,0.9305682},{0.0469101,0.2307653,0.5,
0.7692347,0.9530899},{0.0337652,0.1693953,0.3806904,0.6193096,
0.8306047,0.9662348},{0.0254460,0.1292344,0.2970774,0.5,0.7029226,
0.8707656,0.9745540},{0.0198551,0.1016668,0.2372338,0.4082827,
0.5917173,0.7627662,0.8983332,0.9801449}};
double
ga[][8]={{1.0},{0.5,0.5},{0.2777778,0.4444444,0.2777778},{0.1739274,
0.3260726,0.3260726,0.1739274},{0.1184634,0.2393143,0.2844444,
0.2393143,0.1184634},{0.0856622,0.1803808,0.2339570,0.2339570,
```

```
0.1803808,0.0856622},{0.0647425,0.1398527,0.1909150,0.2089796,
0.1909150,0.1398527,0.0647425},{0.0506143,0.1111905,0.1568533,
0.1813419,0.1813419,0.1568533,0.1111905,0.0506143}};
//---------------------------------------------------------------------------
double fp(double x)
{return (x*x/2-1)*asin(x/2)+x/4*sqrt(4-x*x);}
//---------------------------------------------------------------------------
double f(double x)
{return x*asin(x/2);}
//---------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  a=StrToFloat(LabeledEdit1->Text);
  b=StrToFloat(LabeledEdit2->Text);
  tin=fp(b)-fp(a);
  LabeledEdit3->Text=FloatToStr(tin);
  f1->Enabled=true;
  f2->Enabled=true;
  f3->Enabled=true;
  LabeledEdit1->Enabled=false;
  LabeledEdit2->Enabled=false;
  Button1->Enabled=false;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N_LClick(TObject *Sender)
{
 f1->Enabled=false;
 f2->Enabled=false;
 f3->Enabled=false;
 StringGrid1->Visible=false;
 PageControl1->Visible=false;
 LabeledEdit1->Enabled=true;
 LabeledEdit2->Enabled=true;
 Button1->Enabled=true;
 LabeledEdit3->Text="";
  for(int i=0;i<StringGrid1->RowCount;i++)
    for(int j=0;j<StringGrid1->ColCount;j++)
       StringGrid1->Cells[j][i]="";
  for(int i=0;i<StringGrid2->RowCount;i++)
    for(int j=0;j<StringGrid2->ColCount;j++)
       StringGrid2->Cells[j][i]="";
  for(int i=0;i<StringGrid3->RowCount;i++)
    for(int j=0;j<StringGrid3->ColCount;j++)
       StringGrid3->Cells[j][i]="";
  for(int i=0;i<StringGrid4->RowCount;i++)
    for(int j=0;j<StringGrid4->ColCount;j++)
       StringGrid4->Cells[j][i]="";
  Series1->Clear();
  Series2->Clear();
  Series3->Clear();
  Series4->Clear();
  Series5->Clear();
  Series6->Clear();
```

```
}
//---------------------------------------------------------------------------

void __fastcall TForm1::f1Click(TObject *Sender)
{
 StringGrid1->Visible=true;
 StringGrid1->Cells[0][0]="     n";
 StringGrid1->Cells[0][1]="H_Л-H_К";
 StringGrid1->Cells[0][2]="H_Л-Гаусс";
 for(int n=1;n<9;n++)
      StringGrid1->Cells[n][0]="    "+IntToStr(n);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::f1_N_KClick(TObject *Sender)
{
  double h,y;
  for(int n=1;n<9;n++){
    h=(b-a)/n;
    y=0;
    for(int i=0;i<n+1;i++)
       y+=nkh[n-1][i]*f(a+i*h);
    y*=b-a;
    StringGrid1->Cells[n][1]=FloatToStrF(tin-y,ffExponent,4,2);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::f1_GClick(TObject *Sender)
{
  double y;
  for(int n=1;n<9;n++){
```

```
    y=0;
    for(int i=0;i<n;i++)
      y+=ga[n-1][i]*f(a+(b-a)*gx[n-1][i]);
    y*=b-a;
    StringGrid1->Cells[n][2]=FloatToStrF(tin-y,ffExponent,4,2);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::f2_N_KClick(TObject *Sender)
{
  double h1,h,y,x;
  Series1->Clear();
  int n=StrToInt(LabeledEdit4->Text);
  for(int k=1;k<21;k++){
     h1=(b-a)/k;
     h=h1/n;
     y=0;
     for(x=a;x<b-h/2;x+=n*h)
        for(int i=0;i<n+1;i++)
          y+=nkh[n-1][i]*f(x+i*h);
     y*=h1;
     StringGrid2->Cells[k][0]=FloatToStrF(h1,ffFixed,5,4);
     StringGrid2->Cells[k][1]=FloatToStrF(tin-y,ffExponent,4,2);
     Series1->AddXY(h1,tin-y,"",clBlack);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::f2_GClick(TObject *Sender)
{
 double h,y,x;
 Series2->Clear();
```

```
 int n=StrToInt(LabeledEdit4->Text);
 for(int k=1;k<21;k++){
     h=(b-a)/k;
     y=0;
     for(x=a;x<b-h/2;x+=h)
        for(int i=0;i<n;i++)
          y+=ga[n-1][i]*f(x+h*gx[n-1][i]);
     y*=h;
     StringGrid2->Cells[k][0]=FloatToStrF(h,ffFixed,5,4);
     StringGrid2->Cells[k][2]=FloatToStrF(tin-y,ffExponent,4,2);
     Series2->AddXY(h,tin-y,"",clBlack);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::f2Click(TObject *Sender)
{
 PageControl1->Visible=true;
 StringGrid2->Cells[0][0]="     шаг h";
 StringGrid2->Cells[0][1]="H_Л-H_K";
 StringGrid2->Cells[0][2]="H_Л-Гаусс";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::f3Click(TObject *Sender)
{ PageControl1->Visible=true;
  StringGrid3->Cells[0][0]="     eps";
  StringGrid4->Cells[0][0]="     eps";
  StringGrid3->Cells[0][1]="H_Л-H_K";
  StringGrid4->Cells[0][1]="H_K-время";
  StringGrid3->Cells[0][2]="H_Л-Гаусс";
  StringGrid4->Cells[0][2]="Гаусс-время";
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::f3_N_KClick(TObject *Sender)
{
  double h1,h,y,ys,x,eps;
  Series3->Clear();
  Series5->Clear();
  int t,k,n=StrToInt(LabeledEdit5->Text);
  for( eps=1E-6,k=1;eps<0.011;eps*=10,k++){
      h=(b-a)/n;
      ys=0;
M:      for(x=a,t=0,y=0;x<b-n*h/2;x+=n*h)
          for(int i=0;i<n+1;i++)
             {y+=nkh[n-1][i]*f(x+i*h);t++;}
     y*=n*h;
     if(fabs((y-ys)/y)>=eps) {ys=y; h=h/2; goto M;}
     StringGrid3->Cells[k][0]=FloatToStrF(eps,ffFixed,7,6);
     StringGrid4->Cells[k][0]=FloatToStrF(eps,ffFixed,7,6);
     StringGrid3->Cells[k][1]=FloatToStrF(tin-y,ffExponent,4,2);
     StringGrid4->Cells[k][1]=IntToStr(t);
     Series3->AddXY(eps,tin-y,"",clBlack);
     Series5->AddXY(eps,t,"",clBlack);}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::f3_GClick(TObject *Sender)
{
  double h,y,x,eps,ys;
  Series4->Clear();
  Series6->Clear();
```

```
  int t,k,n=StrToInt(LabeledEdit5->Text);
  for( eps=1E-6,k=1;eps<0.011;eps*=10,k++){
      h=(b-a);
      ys=0;
M:      for(x=a,t=0,y=0;x<b-h/2;x+=h)
          for(int i=0;i<n;i++)
            {y+=ga[n-1][i]*f(x+h*gx[n-1][i]);t++;}
     y*=h;
     if(fabs((y-ys)/y)>=eps) {ys=y; h=h/2; goto M;}
     StringGrid3->Cells[k][2]=FloatToStrF(tin-y,ffExponent,4,2);
     StringGrid4->Cells[k][2]=IntToStr(t);
     Series4->AddXY(eps,tin-y,"",clBlack);
     Series6->AddXY(eps,t,"",clBlack);}
}
//---------------------------------------------------------------------------
```

17. Запустите приложение на выполнение, нажав быстрые кнопки *Сохранить все* и *Запуск*. В начале выполнения доступны раздел меню *Ф-ла Ньютона-Лейбница* и окна редактирования, где находятся нижний и верхний пределы интегрирования. После щелчка на разделе меню *Ф-ла Ньютона-Лейбница* станет доступной кнопка *расчет*. Щелчок на ней дает точное значение интеграла и делает ее недоступной. Недоступными становятся и окна с пределами интегрирования, а доступными – остальные разделы меню. Теперь пользователь может получить все зависимости согласно заданию. Требуемое значение *n* можно установить в окне редактирования на первой и второй страницах компонента **PageControl1**. Щелчок на разделе меню *Ф-ла Ньютона-Лейбница* приводит к сбросу всех результатов расчетов и позволяет найти точное значение интеграла и получить зависимости для других пределов интегрирования.

18. Щелчком на кнопке *конец* завершите выполнение задания.

Содержание отчета

1. Задание.
2. Формулы с пояснениями .
3. Результаты выполнения задания в *Mathcad*’е.
4. Блок-схемы алгоритмов.
5. Таблица идентификаторов.
6. Код.
7. Результаты выполнения работы в виде таблиц и графиков.
8. Библиографический список.

Контрольные вопросы

1. С какой целью и как интерполируют подынтегральные функции?

2. Как вычисляются и какими свойствами обладают коэффициенты Котеса?

3. Как используются коэффициенты Котеса?

4. Как используются коэффициенты и абсциссы Гаусса?

5. Расскажите о частных случаях формулы Ньютона-Котеса.

6. Приведите иллюстрации вычисления интегралов по частным случаям формулы Ньютона-Котеса.

7. Приведете иллюстрации использования формулы прямоугольников (срединных, левых, правых).

8. Объясните алгоритм вычисления приближенного значения интеграла в вашем задании.

9. Как получить составные формулы в частных случаях методов Ньютона-Котеса и Гаусса?

10. Как сократить затраты машинного времени в алгоритмах на рис. 7.1, 7.3, 7.4, 7.5?

11. Как сократить затраты машинного времени в алгоритмах на рис. 7.7, 7.9?

12. Сравните по затратам машинного времени частные случаи формулы Ньютона-Котеса.

13. Сравните по ошибке интегрирования частные случаи формулы Ньютона-Котеса.

14. Сравните по ошибке интегрирования формулы прямоугольников (срединных, левых, правых).

15. Где, зачем и как используется правило Рунге?

16. Объясните адаптивный алгоритм по методу Ньютона-Котеса.

17. Объясните адаптивный алгоритм по методу Гаусса.

ЧИСЛЕННЫЕ МЕТОДЫ ИНТЕГРИРОВАНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ (ОДУ) И СИСТЕМ ОДУ

Краткие сведения

Решением обыкновенного дифференциального уравнения (ОДУ)

$$y' = f(x, y) \tag{1}$$

называется функция $y = y(x)$ (в графическом варианте её называют также интегральной кривой), определенная в некоторой области D плоскости (x, y), дифференцируемая на некотором интервале Δx и, следовательно, удовлетворяющая условиям $y(x),\ x \in D,\ y'(x) = f(x, y(x)),\ x \in \Delta x$. Задача отыскания решения ОДУ, имеющего начальные значения x_0, $y_0 = y(x_0)$, называется задачей Коши.

В численных методах решение ОДУ $y' = f(x, y)$ с начальным условием $y_0 = y(x_0)$ вычисляется для значений аргумента

$x_m = x_0 + m*h,\ m = 0,1,...,M,\ h = \Delta x / M.$ Значения x_m называются узлами, а величина h - шагом интегрирования; через y_m обозначают значение приближенного решения в узле x_m.

Метод интегрирования ОДУ с помощью ряда Тейлора

Этот метод теоретически годится для решения любых ДУ, но практического интереса не представляет. Однако ценность его в том, что он может быть использован в качестве эталона, по которому сравниваются практически удобные методы Эйлера и Рунге-Кутта.

Разложение функции $y(x)$ в ряд Тейлора в окрестности точки $x = x_m$ имеет вид

$$y(x) = y_m + (x - x_m)y'_m + \frac{(x - x_m)^2}{2!}y''_m + \frac{(x - x_m)^3}{3!}y'''_m + ... = \sum_{k=0}^{\infty}\frac{(x - x_m)^k}{k!}y_m^{(k)}.$$

Если значения x расположены на расстоянии h друг от друга, то

$$y_{m+1} = y_m + hy'_m + \frac{h^2}{2!}y''_m + \frac{h^3}{3!}y'''_m + ... = \sum_{k=0}^{\infty}\frac{h^k}{k!}y_m^{(k)}. \qquad (2)$$

Ряд Тейлора позволяет получить решение в узле x_1 по начальным условиям задачи Коши: $y_1 = y_0 + hy'_0 = y_0 + hf(x_0, y_0)$, а по решению в произвольном узле x_m - решение в узле x_{m+1}:

$$y_{m+1} = y_m + hy'_m = y_m + hf(x_m, y_m). \qquad (3)$$

Сравнение (2) и (3) приводит к выводу, что решение получено при отбрасывании членов ряда, начиная с члена, содержащего h^2. Следовательно, ошибка ограничения равна $O(h^2)$. Для уменьшения ошибки ограничения необходимо вычислять значения первой и последующих производных функции $f(x, y)$ в узле x_m, что требует выполнения практически неприемлемого объема вычислений.

Метод Эйлера

Метод Эйлера основан на формуле (3). Иллюстрация метода Эйлера представлена на рис. 8.1; l_0 – касательная к $y(x)$ в точке (x_0, y_0) и, следовательно, её наклон определяется значением производной $f(x_0, y_0)$. Погрешность метода пропорциональна h^2 и, следовательно, слишком велика при допустимой величине h. Для повышения точности метод Эйлера модифицируют путем использования значений функции $f(x, y)$ в дополнительных точках.

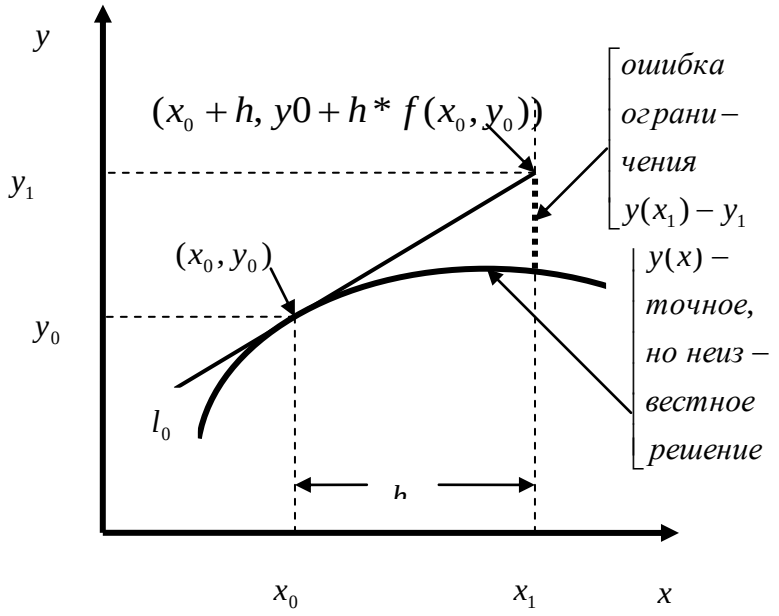


Рис. 8.1. Иллюстрация метода Эйлера

Первый модифицированный метод Эйлера («с пересчетом»)

Этот метод основан на формулах:

$$y_{m+1} = y_m + h(k_1 + k_2)/2,$$
$$\text{где} \quad k_1 = f(x_m, y_m), \quad k_2 = f(x_m + h,\ y_m + hk_1). \tag{4}$$

Иллюстрация метода представлена на рис. 8.2, где l_0, l_1 – касательные, наклон которых определяется значениями k_1, k_2 соответственно, наклон прямой l_2 и параллельной ей l^* определяется значением $(k_1 + k_2)/2$. Отметим, что прямая l_2 является биссектрисой угла, образованного прямыми l_0, l_1.

Иллюстрация метода наглядно доказывает уменьшение ошибки ограничения по сравнению с методом Эйлера.

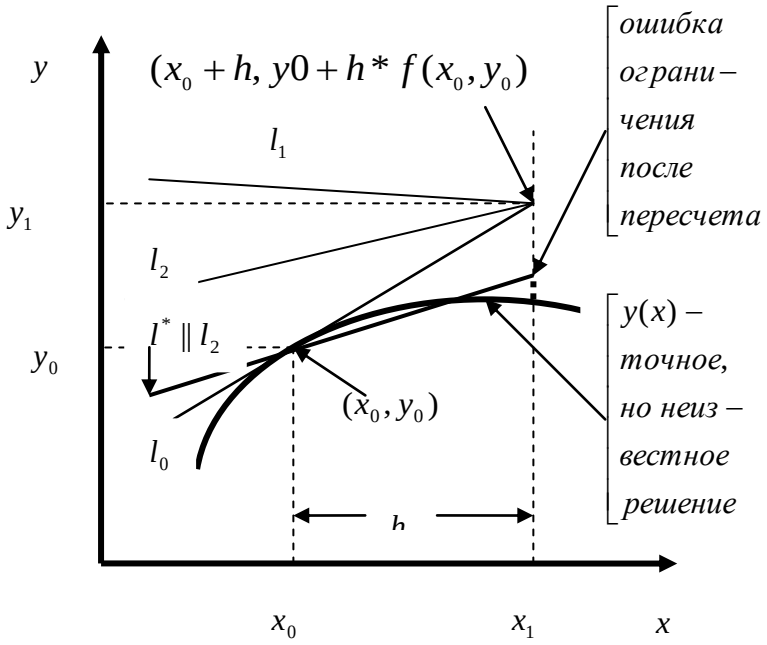


Рис. 8.2. **И**ллюстрация первого модифицированного метода Эйлера (метода «с пересчетом»)

Теперь докажем факт уменьшения ошибки ограничения, показав, как этот метод согласуется с разложением в ряд Тейлора. Итак, разложим $f(x, y)$ в окрестности точки (x_m, y_m) в ряд Тейлора:

$$f(x, y) = f(x_m, y_m) + (x - x_m) f_x + (y - y_m) f_y + \ldots\ ,$$

где $f_x = \dfrac{\partial f(x,y)}{\partial x}$, $f_y = \dfrac{\partial f(x,y)}{\partial y}$ вычислены в точке (x_m, y_m). Обозначив $f(x_m, y_m)$ через f, получим

$$f(x_m + h, y_m + hy'_m) = f + hf_x + hff_y + O(h^2),$$

где $O(h^2)$ – сумма остальных членов ряда. Подставим разложение в (4) и получим

$$y_{m+1} = y_m + hf + \frac{h^2}{2}(f_x + ff_y) + O(h^3).$$

Полученный результат позволяет сделать следующие выводы.

1) Метод «с пересчетом» согласуется с разложением в ряд Тейлора вплоть до членов степени h^2.
2) Порядок метода равен максимальной степени h в его разложении. По сравнению с методом Эйлера порядок увеличился на 1 и стал равным 2. Поэтому метод «с пересчетом» называют методом Рунге-Кутта 2-го порядка, или РК2.
3) Увеличение порядка на 1 явилось следствием вычисления производной $f(x,y)$ в дополнительной точке, т. е. вычисления $k_2 = f(x_m + h,\ y_m + hk_1)$.

Второй модифицированный метод Эйлера (метод «ломаных»)

Метод «ломаных» использует формулы:

$$y_{m+1} = y_m + hf(x_m + h/2,\ y_m + hk/2),\quad k = f(x_m, y_m). \qquad (5)$$

Иллюстрация метода представлена на рис. 8.3, где l' – касательная к $y(x)$ в точке $(x_0 + h/2, y_0 + hk/2)$ и, следовательно, её наклон определяется значением производной $f(x_0 + h/2, y_0 + hk/2)$.

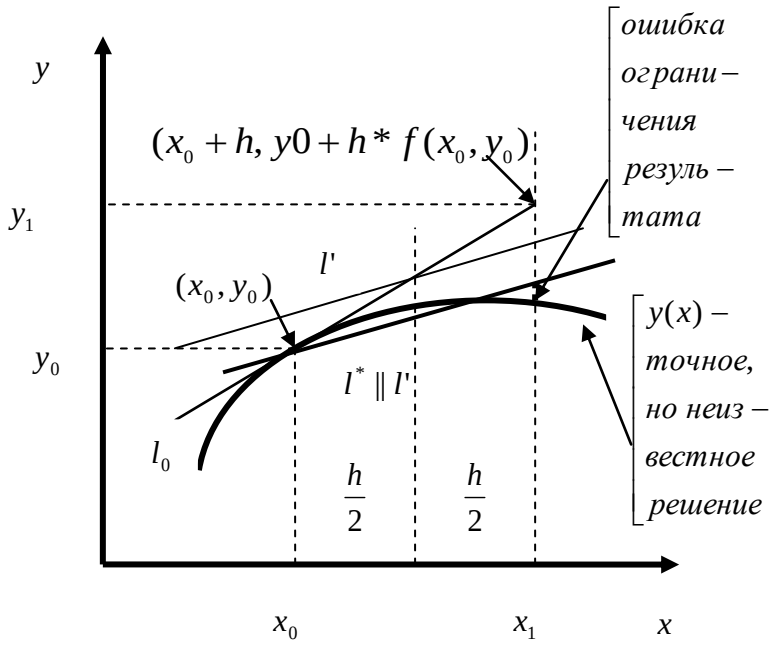


Рис. 8.3. Иллюстрация второго модифицированного метода Эйлера (метод «ломаных»)

Следовательно, в методе «ломаных» производная вычисляется в дополнительной точке $(x_0 + h/2, y_0 + hk/2)$, которая была получена после вычисления производной в точке (x_0, y_0), как $k = f(x_0, y_0)$. Можно показать, что этот метод также является методом РК2.

Итак, модифицированные методы Эйлера имеют погрешность порядка h^3. Они относятся к семейству методов Рунге-Кутта второго порядка (РК2).

Метод Рунге-Кутта третьего порядка РК3

Порядок погрешности метода – h^4. Формулы метода:

$$\begin{aligned} y_{m+1} &= y_m + (h/6)*(k_1 + 4k_2 + k_3), \\ k_1 &= f(x_m, y_m), \\ k_2 &= f(x_m + h/2, y_m + hk_1/2), \\ k_3 &= f(x_m + h, y_m - hk_1 + 2hk_2\ ). \end{aligned} \qquad (6)$$

Метод Рунге-Кутта четвертого порядка РК4

Наиболее распространен и имеет порядок погрешности h^5. Формулы метода:

$$\begin{aligned} &y_{m+1} = y_m + (h/6)*(k_1 + 2k_2 + 2k_3 + k_4), \\ &k_1 = f(x_m, y_m), \\ &k_2 = f(x_m + h/2, y_m + hk_1/2), \\ &k_3 = f(x_m + h/2, y_m + hk_2/2), \\ &k_4 = f(x_m + h, y_m + hk_3). \end{aligned} \qquad (7)$$

Все вышеприведенные методы называются одношаговыми, так как для вычисления y_{m+1} достаточно знать лишь y_m – значение решения на предыдущем шаге. Это позволяет использовать их при переменном шаге интегрирования.

Ошибки методов

При выборе шага интегрирования исходят из того, что численные методы интегрирования ОДУ имеют два источника ошибок: метод и вычисления. Ошибка метода тем меньше, чем меньше шаг интегрирования и оценивается величиной $\delta_м = \Delta x * c * h^p$, где $\Delta x = x_k - x_0$ – интервал интегрирования; c – константа; h^p – погрешность метода.

Ошибка вычислений, которую называют ещё ошибкой округления, не зависит от шага интегрирования. Если на одном шаге она равна ε, то накопившаяся от округлений ошибка удовлетворяет неравенству $\varepsilon\sqrt{M} \le \delta_в \le \varepsilon * M$, где M – количество шагов интегрирования.

Если компьютер работает с округлением, то более вероятна левая граница, а при отбрасывании разрядов – правая. Приближенно суммарная ошибка интегрирования может быть определена по формуле

$$\delta_\Sigma \cong \Delta x\left(c * h^p + \varepsilon / h\right). \qquad (8)$$

Из формулы (8) следует, что по мере уменьшения h ошибка метода убывает и численное решение сходится к точному. Но когда шаг интегрирования слишком мал, то при больших отрезках интегрирования численное решение начинает расходиться вследствие накопления ошибок округлений. Кроме того, для всех методов интегрирования затраты машинного времени обратно пропорциональны шагу интегрирования.

Интегрирование систем ОДУ и ОДУ высших порядков

Прикладные задачи часто приводят к системам ОДУ и к ОДУ n-го порядка.. В нормальной форме система ОДУ n-го порядка имеет вид

$$y_i^{'} = f_i(x, y_1, y_2, ..., y_n), \quad i = 1, 2, ..., n, \tag{9}$$

где $y_1, y_2, ..., y_n$ – неизвестные функции от переменного x, а f_i - заданные функции от $n+1$ переменных.

Задача Коши для системы (9) состоит в отыскании решения, удовлетворяющего начальным условиям $y_i(t_0),\ i = 1, 2, ..., n$.

ОДУ n-го порядка разрешают относительно старшей производной

$$y^{(n)} = f(x, y, y', y'', ..., y^{(n-1)}) \tag{10}$$

и введением новых переменных по правилу $y^{(k)} = y_k,\ \ k = 0, 1, ..., n-1$ приводят к нормальной системе ОДУ первого порядка

$$\begin{cases} y_0^{'} = y_1, \\ y_1^{'} = y_2, \\ \ldots\ldots\ldots \\ y_{n-2}^{'} = y_{n-1}, \\ y_{n-1}^{'} = f(x, y_0, y_1 \ldots, y_{n-1}) \end{cases} \tag{11}$$

с начальными условиями

$$y_0(x_0) = y(x_0) = y_0,\ y_1(x_0) = y'(x_0) = y_0', ..., y_{n-1}(x_0) = y^{(n-1)}(x_0) = y_0^{(n-1)}. \tag{12}$$

Решением ОДУ(10) является n раз дифференцируемая функция $y(x), x \in \Delta x,$ которая обращает уравнение (10) в тождество и удовлетворяет начальным условиям (12).

Например, для решения системы ОДУ

$$\begin{cases} y' = f(x, y, z), \\ z' = g(x, y, z) \end{cases}$$

с начальными условиями $x = x_0, y(x_0) = y_0, z(x_0) = z_0$ формулы метода РК4 запишутся в виде

$$\begin{cases} y_{m+1} = y_m + (h/6) * (k_1 + 2k_2 + 2k_3 + k_4), \\ z_{m+1} = z_m + (h/6) * (l_1 + 2l_2 + 2l_3 + l_4), \end{cases}$$

$$\begin{cases} k_1 = f(x_m, y_m, z_m), \\ l_1 = g(x_m, y_m, z_m), \\ k_2 = f(x_m + h/2, y_m + hk_1/2, z_m + hl_1/2), \\ l_2 = g(x_m + h/2, y_m + hk_1/2, z_m + hl_1/2), \\ k_3 = f(x_m + h/2, y_m + hk_2/2, z_m + hl_2/2), \\ l_3 = g(x_m + h/2, y_m + hk_2/2, z_m + hl_2/2), \\ k_4 = f(x_m + h, y_m + hk_3, z_m + hl_3), \\ l_4 = g(x_m + h, y_m + hk_3, z_m + hl_3). \end{cases} \quad (13)$$

Алгоритмы одношаговых методов Рунге-Кутта

Поскольку алгоритмы одношаговых методов однотипны, то достаточно рассмотреть один пример, чтобы построить алгоритм для любого другого задания.

Методом РК3 решить систему ОДУ

$$\begin{cases} y' = \dfrac{y}{y+z}, \\ z' = \dfrac{z}{y+z} \end{cases} \quad \textit{в интервале } x_0 = 0, x_k = 8, \quad \textit{при } y(x_0) = 2, z(x_0) = 8.$$

Введем новые переменные $y_0 = y, y_1 = z.$ Тогда система примет вид:

$$\begin{cases} y_0' = y_0/(y_0 + y_1), \\ y_1' = y_1/(y_0 + y_1). \end{cases}$$

Используем формулы (6) для метода РК3:

$$\begin{cases} k_1 = y_{0m}/(y_{0m} + y_{1m}), \\ l_1 = y_{1m}/(y_{0m} + y_{1m}), \\ k_2 = (y_{0m} + \frac{h}{2}k_1)/(y_{0m} + \frac{h}{2}k_1 + y_{1m} + \frac{h}{2}l_1), \\ l_2 = (y_{1m} + \frac{h}{2}l_1)/(y_{0m} + \frac{h}{2}k_1 + y_{1m} + \frac{h}{2}l_1), \\ k_3 = (y_{0m} - hk_1 + 2hk_2)/(y_{0m} - hk_1 + 2hk_2 + y_{1m} - hl_1 + 2hl_2), \\ l_3 = (y_{1m} - hl_1 + 2hl_2)/(y_{0m} - hk_1 + 2hk_2 + y_{1m} - hl_1 + 2hl_2). \end{cases}$$

$$\begin{cases} y_{0,m=1} = y_{0m} + \frac{h}{6}(k_1 + 4k_2 + k_3), \\ y_{1,m+1} = y_{1m} + \frac{h}{6}(l_1 + 4l_2 + l_3). \end{cases}$$

Алгоритм численного интегрирования системы ОДУ представлен на рис. 8.5.

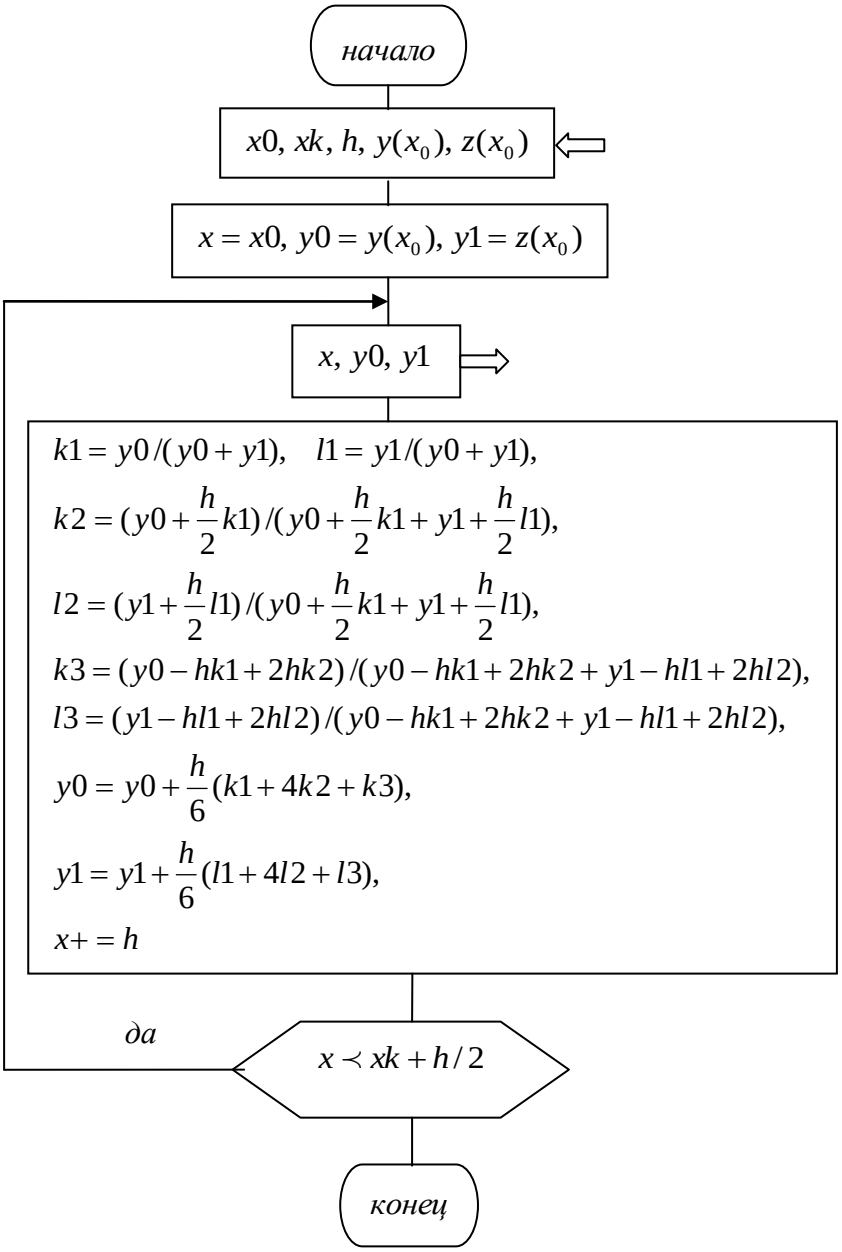


Рис. 8.5. Алгоритм численного интегрирования системы ДУ методом РКЗ

Методы прогноза и коррекции

Для численного решения ОДУ $y' = f(x, y)$ используют также многошаговые методы, в которых вычисление y_{m+1} ведется не только по y_m, $f(x_m, y_m)$, но и по значениям y_{m-i}, $f(x_{m-i}, y_{m-i})$ в нескольких предыдущих узлах . Формулы k – шагового метода имеют вид

$$y_{m+1} = \sum_{q=1}^{k} a_{-q} y_{m+1-q} + h \sum_{q=0}^{k} b_{-q} f\left(x_{m+1-q}, y_{m+1-q}\right),$$

где a_{-q}, b_{-q} – постоянные коэффициенты. Если $b_0 = 0$, соответствующий метод называется экстраполяционным или явным, если $b_0 \neq 0$ – интерполяционным или неявным методом.

Частным случаем многошаговых методов является метод Адамса:

$$y_{m+1} = y_m + h \sum_{q=0}^{k} b_{-q} f\left(x_{m+1-q}, y_{m+1-q}\right).$$

Обычно вычисления ведут по паре формул, одна из которых явная, а другая – неявная. Такие пары формул называются методами прогноза и коррекции. Прогноз, выполняемый один раз на шаге, служит цели получения хорошего начального приближения для последующей коррекции. Последняя может выполняться на каждом шаге заданное число раз (часто только один раз) или повторяться до сходимости. Применение многошаговых методов возможно лишь в том случае, если известны решения в k первых узлах. Для нахождения этих значений обычно пользуются одношаговыми методами, что увеличивает объём программы. Поэтому в настоящее время многошаговые методы употребляются значительно реже, чем четырехточечный метод Рунге-Кутта РК4.

Рассмотрим простейший вариант метода прогноза и коррекции.

Формулы метода:

$$\begin{aligned} y_{m+1}^{(0)} &= y_{m-1} + 2hf\left(x_m, y_m\right), \\ y_{m+1}^{(k)} &= y_m + (h/2) * \left(f\left(x_m, y_m\right) + f\left(x_{m+1}, y_{m+1}^{(k-1)}\right)\right). \end{aligned} \qquad (14)$$

По первой формуле выполняется прогноз, по второй – k-я коррекция. На рис. 8.4 представлена иллюстрация этого метода при $k = 1$, т. е. для первой коррекции.

Коррекцию можно выполнять сколько угодно раз; для получения решения итерационный процесс должен быть сходящимся. Условие

сходимости: $h < 2/Z$, *где* $Z \geq \left|\frac{\partial f}{\partial y}\right|$. Вычисления прекращаются, когда $\left|y_{m+1}^{(i)} - y_{m+1}^{i-1}\right| < \varepsilon$, где ε – заданная ошибка.

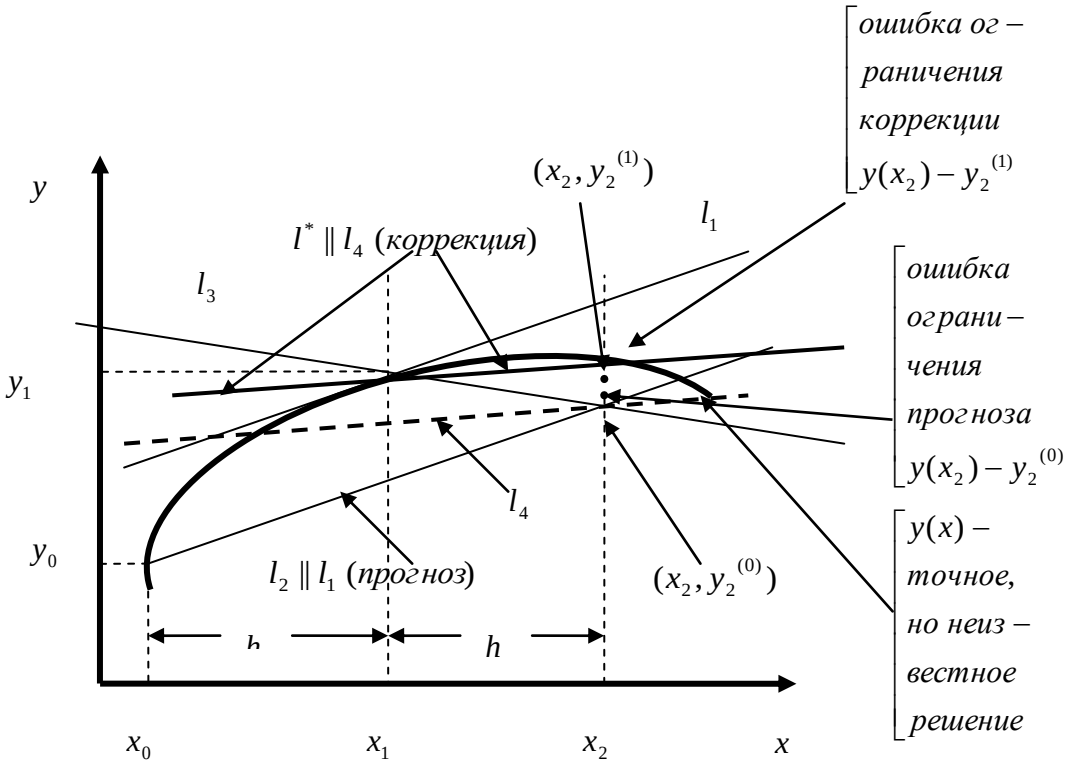


Рис. 8.4. Иллюстрация метода прогноза и коррекции

Ошибки ограничения прогноза и коррекции

Представим решение в виде разложения в ряд:

$$y(x) = y_m + (x - x_m)y_m' + \frac{(x - x_m)^2}{2} y_m'' + \frac{(x - x_m)^3}{6} y'''(\xi), \quad \xi \in [x, x_m].$$

При $x = x_{m+1} = x_m + h$:

$$y_{m+1} = y_m + hy_m' + \frac{h^2}{2} y_m'' + \frac{h^3}{6} y'''(\xi_1), \quad \xi_1 \in [x_m, x_{m+1}].$$

При $x = x_{m-1} = x_m - h$:

$$y_{m-1} = y_m - hy_m' + \frac{h^2}{2} y_m'' - \frac{h^3}{6} y'''(\xi_2), \quad \xi_2 \in [x_{m-1}, x_m].$$

Из разности $y_{m+1} - y_{m-1}$ получим

$$y_{m+1} = y_{m-1} + 2hy'_m + \frac{h^3}{6} y'''(\xi_3), \quad \xi_3 \in [x_{m-1}, x_{m+1}].$$

Отсюда ошибка ограничения прогноза: $E_T^{(n)} = \frac{h^3}{3} y'''(\xi), \quad \xi \in [x_{m-1}, x_{m+1}].$

Подобным же образом можно получить ошибку ограничения коррекции:

$$E_T^{(к)} = -\frac{h^3}{12} y'''(\eta), \quad \eta \in [x_{m-1}, x_{m+1}].$$

Получение оценки ошибки ограничения в процессе получения решения

Пусть Y_m – точное значение решения при $x = x_m$. Тогда

$$Y_m = y_m^{(0)} + \frac{h^3}{3} y'''(\xi),$$

$$Y_m = y_m^{(i)} - \frac{h^3}{12} y'''(\eta).$$

Вычтя из одного выражения другое и учитывая, что в промежутке $[x_{m-1}, x_{m+1}]$ третью производную можно считать практически постоянной, получим $y_m^{(0)} - y_m^{(i)} = -\frac{5}{12} h^3 y'''$, откуда $E_T^{(к)} = -\frac{h^3}{12} y''' = \frac{1}{5}(y_m^{(0)} - y_m^{(i)})$.

Решение на шаге получается в результате i-той коррекции, и здесь же как побочный продукт вычислений можно получить оценку ошибки ограничения коррекции по выражению $E_T^{(к)} = \frac{1}{5}(y_m^{(0)} - y_m^{(i)})$, которая позволяет уточнить решение: $y_m = y_m^{(i)} + \frac{1}{5}(y_m^{(0)} - y_m^{(i)})$.

В заключение приведем формулы для ряда методов прогноза и коррекции (здесь x – функция, t – аргумент).

Первый вариант метода Адамса

$$x_{m+1}^{(0)} = x_m + (h/2)*(3f(x_m, t_m) - f(x_{m-1}, t_{m-1})),$$
$$x_{m+1}^{(k)} = x_m + (h/2)*\left(f\left(x_{m+1}^{(k-1)}, t_{m+1}\right) + f(x_m, t_m)\right). \quad (15)$$

Второй вариант метода Адамса

$$x_{m+1} = x_m + (h/24)* \\ *(55f(x_m, t_m) - 59f(x_{m-1}, t_{m-1}) + 37f(x_{m-2}, t_{m-2}) - 9f(x_{m-3}, t_{m-3})). \quad (16)$$

Метод на основе методов Милна и Адамса-Башфорта

Формула для прогноза:

$$p_{m+1} = (x_m + x_{m-1})/2 + (h/48)* \\ *(119f(x_m, t_m) - 99f(x_{m-1}, t_{m-1}) + 69f(x_{m-2}, t_{m-2}) - 17f(x_{m-3}, t_{m-3})), \quad (17)$$

формула для поправки:

$$d_{m+1} = p_{m+1} - (161/170)*(p_m - c_m), \quad (18)$$

формула для коррекции:

$$c_{m+1} = (x_m + x_{m-1})/2 + (h/48)* \\ *(17f(d_{m+1}, t_{m+1}) + 51f(x_m, t_m) + 3f(x_{m-1}, t_{m-1}) + f(x_{m-2}, t_{m-2})), \quad (19)$$

формула для приближенного решения:

$$x_{m+1} = c_{m+1} + (9/170)*(p_{m+1} - c_{m+1}). \quad (20)$$

На первом шаге, когда отсутствует $p_m - c_m$, положим $p_m - c_m = 0$.

Метод Хемминга

Формула для прогноза:

$$p_{m+1} = (2x_{m-1} + x_{m-2})/3 + (h/72)* \\ *(191f(x_m, t_m) - 107f(x_{m-1}, t_{m-1}) + 109f(x_{m-2}, t_{m-2}) - 25f(x_{m-3}, t_{m-3})), \quad (21)$$

формула для поправки:

$$d_{m+1} = p_{m+1} - (707/750)*(p_m - c_m), \quad (22)$$

формула для коррекции:

$$c_{m+1} = (2x_{m-1} + x_{m-2})/3 + (h/72)* \\ *(25f(d_{m+1}, t_{m+1}) + 91f(x_m, t_m) + 43f(x_{m-1}, t_{m-1}) - 9f(x_{m-2}, t_{m-2})), \quad (23)$$

формула для приближенного решения:

$$x_{m+1} = c_{m+1} + (43/750)*(p_{m+1} - c_{m+1}). \quad (24)$$

На первом шаге, когда отсутствует $p_m - c_m$, положим $p_m - c_m = 0$.

Два последних метода имеют погрешность на шаге порядка h^6.

Алгоритмы методов прогноза и коррекции

Порядок построения алгоритма численного интегрирования ОДУ методом прогноза и коррекции с разгоном одношаговым методом рассмотрим на примере.

Первым вариантом метода Адамса (формулы (15)) с разгоном методом РК4 (формулы (7)) найти решение ОДУ

$$y'' - y' + 16y - e^{-x} = 0$$

в интервале $x_0 = 0, x_к = 4$ с начальными условиями $y(x_0) = 0, y'(x_0) = 1.$

Разрешаем уравнение относительно старшей производной

$$y'' = y' - 16y + e^{-x},$$

вводим новые переменные $y_0 = y, y_1 = y'$ и исходное уравнение сводим к нормальной системе ДУ первого порядка:

$$\begin{cases} y_0' = y_1, \\ y_1' = y_1 - 16y_0 + e^{-x}. \end{cases}$$

Разгон выполняется однократным использованием формул:

$$\begin{cases} y_{0,m+1} = y_{0m} + \dfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ y_{1,m+1} = y_{1m} + \dfrac{h}{6}(l_1 + 2l_2 + 2l_3 + l_4). \end{cases}$$

$$
\begin{cases}
k_1 = y_{1m}, \\
l_1 = y_{1m} - 16y_{0m} + \exp(-x_m), \\
k_2 = y_{1m} + \dfrac{h}{2} l_1, \\
l_2 = (y_{1m} + \dfrac{h}{2} l_1) - 16(y_{0m} + \dfrac{h}{2} k_1) + \exp(-(x_m + \dfrac{h}{2})), \\
k_3 = y_{1m} + \dfrac{h}{2} l_2, \\
l_3 = (y_{1m} + \dfrac{h}{2} l_2) - 16(y_{0m} + \dfrac{h}{2} k_2) + \exp(-(x_m + \dfrac{h}{2})), \\
k_4 = y_{1m} + h l_3, \\
l_4 = (y_{1m} + h l_3) - 16(y_{01m} + h k_3) + \exp(-(x_m + h)).
\end{cases}
$$

Теперь необходимо перейти к методу прогноза и коррекции, где в прогнозе используются начальные условия и узел с решением, полученным в результате разгона. Для этого нужно соответствующим образом переименовать переменные, что позволит связать разгон с методом прогноза и коррекции и построить алгоритм.

В начальных условиях имена переменных y_{0m}, y_{1m} заменим на y_0, y_1. Результаты разгона $y_{0,m+1}, y_{1,m+1}$ переименуем в y_{00}, y_{10}. Связь полученных через переименование имен и имен переменных, которые будут использованы для прогноза и коррекции, представлена на рис. 8.6.

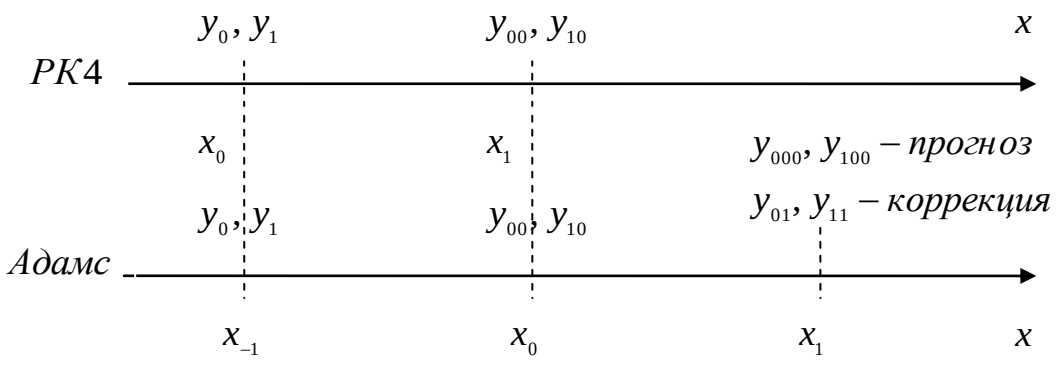
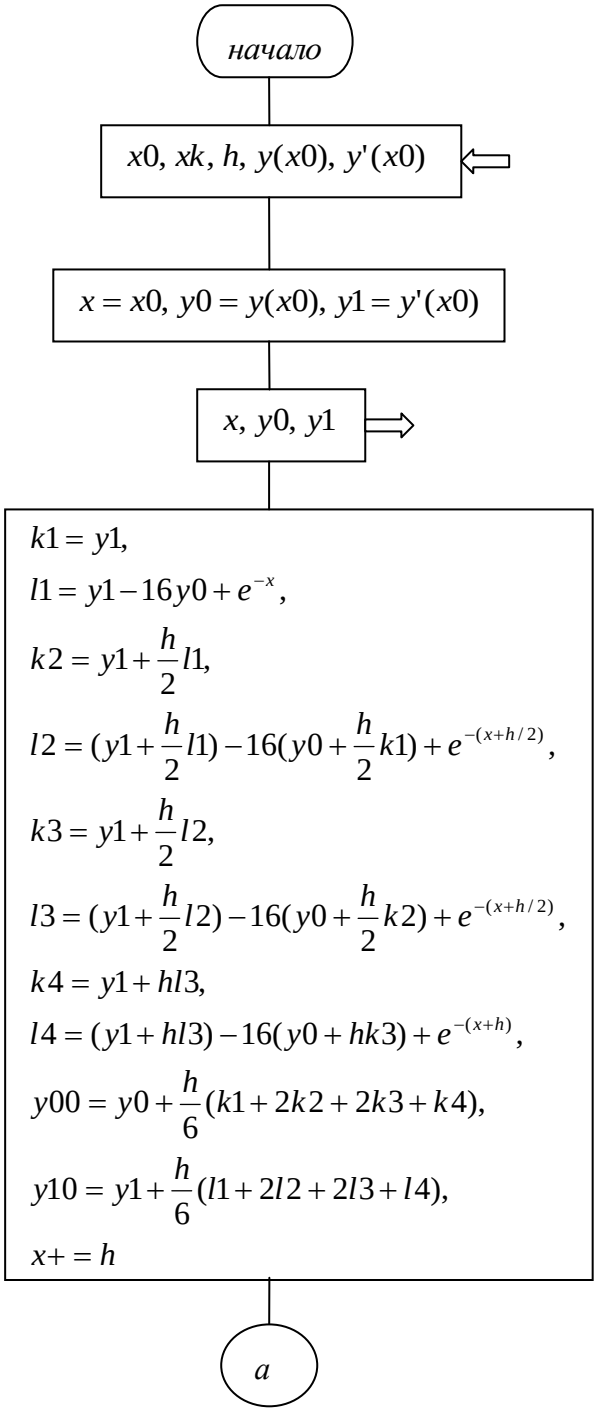


Рис. 8.6. Связь начальных условий, разгона, прогноза и коррекции


начало
x0, xk, h, y(x0), y'(x0)
x = x0, y0 = y(x0), y1 = y'(x0)
x, y0, y1
k1 = y1,
l1 = y1 − 16y0 + e^(−x),
k2 = y1 + (h/2)l1,
l2 = (y1 + (h/2)l1) − 16(y0 + (h/2)k1) + e^(−(x+h/2)),
k3 = y1 + (h/2)l2,
l3 = (y1 + (h/2)l2) − 16(y0 + (h/2)k2) + e^(−(x+h/2)),
k4 = y1 + hl3,
l4 = (y1 + hl3) − 16(y0 + hk3) + e^(−(x+h)),
y00 = y0 + (h/6)(k1 + 2k2 + 2k3 + k4),
y10 = y1 + (h/6)(l1 + 2l2 + 2l3 + l4),
x+ = h
a

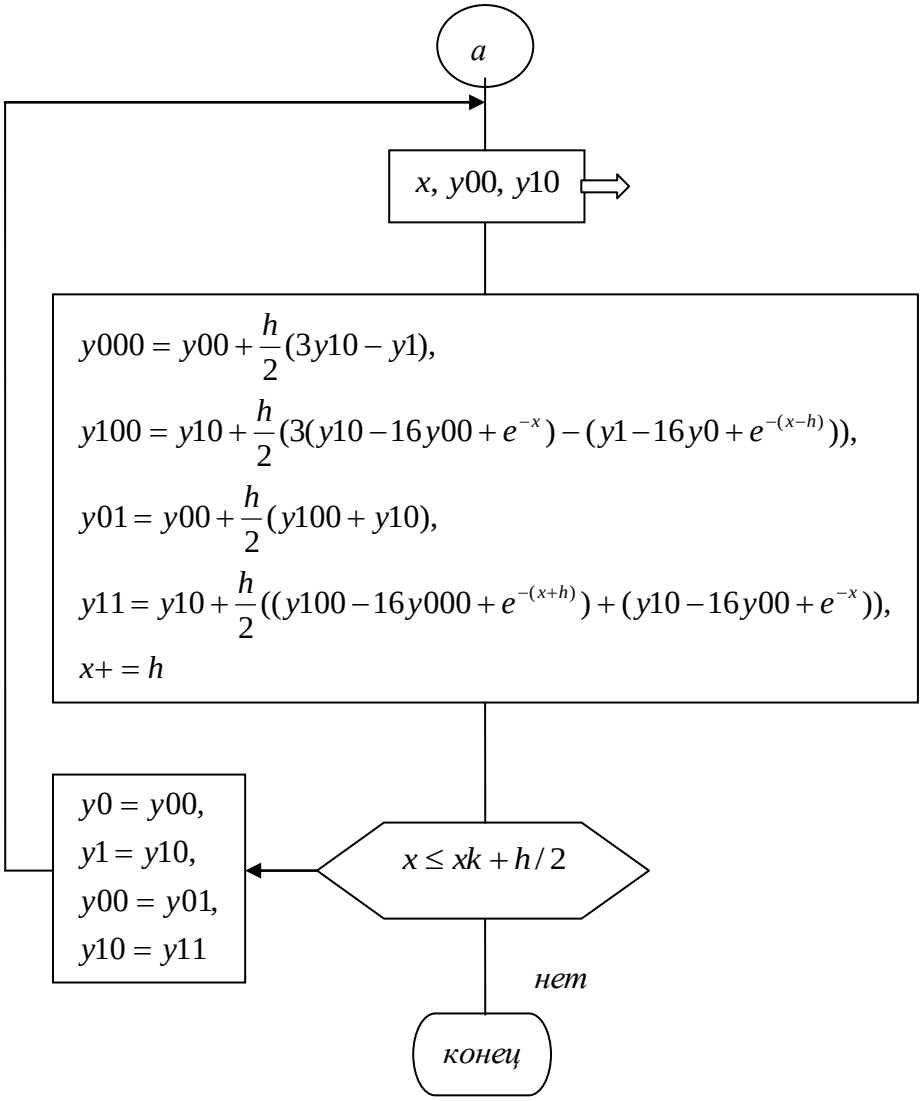


Рис. 8.7. Алгоритм численного интегрирования
ДУ второго порядка методом прогноза и коррекции
(первый вариант метода Адамса) с разгоном методом РК4

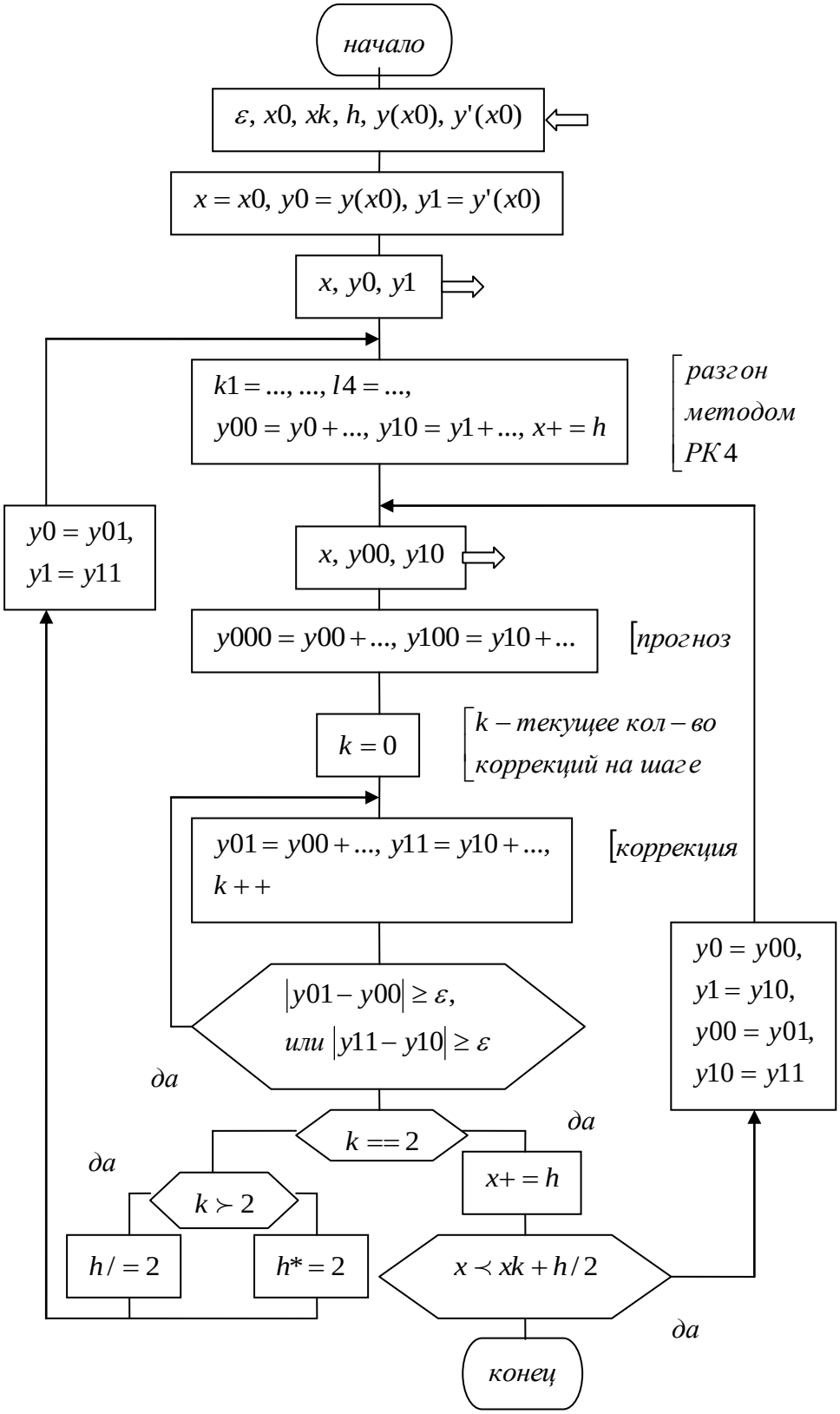


Рис. 8.8. Адаптивный алгоритм численного интегрирования ДУ второго порядка методом прогноза и коррекции (первый вариант метода Адамса) с разгоном методом РК4

Теперь формулы для прогноза и коррекции будут иметь вид:

$$\begin{cases} y_{000} = y_{00} + \dfrac{h}{2}(3y_{10} - y_1), \\ y_{100} = y_{10} + \dfrac{h}{2}(3(y_{10} - 16y_{00} + \exp(-x)) - (y_1 - 16y_0 + \exp(-(x-h)))). \end{cases}$$

$$\begin{cases} y_{01} = y_{00} + \dfrac{h}{2}(y_{100} + y_{10}), \\ y_{11} = y_{10} + \dfrac{h}{2}((y_{100} - 16y_{000} + \exp(-(x+h))) + (y_{10} - 16y_{00} + \exp(-x))). \end{cases}$$

Полученные выше формулы позволяют построить алгоритм (рис. 8.7) для выполнения задания.

Адаптивный алгоритм численного интегрирования ОДУ методом прогноза и коррекции с разгоном методом РК

Допустим теперь, что в рассмотренном задании есть требования – интегрировать с заданной ошибкой и обеспечить минимальные затраты машинного времени.

Известно, что минимальные затраты машинного времени достигаются при таком шаге интегрирования, когда количество итераций на шаге равно двум. Это обстоятельство позволяет достаточно просто строить адаптивные алгоритмы, в которых величина шага автоматически устанавливается по числу итераций на шаге. Пример адаптивного алгоритма для рассмотренного выше задания приведен на рис. 8.8.

При построении адаптивного алгоритма необходимо учитывать, что при изменении величины шага нарушаются условия применения метода прогноза и коррекции. Последняя вычисленная точка (узел) перед

изменением шага становится точкой начальных условий, и поэтому продолжать интегрирование нужно с разгона решения, включив метод РК4.

Интегрирование систем ОДУ в системе Mathcad

Для интегрирования систем ОДУ в *Mathcad* введён ряд функций. Приведём только некоторые функции, дающие решения для систем ОДУ, представленных в обычной форме Коши:

Rkadapt(y, x1, x2, n, D) – возвращает матрицу решений адаптивным методом Рунге-Кутта на интервале от *x1* до *x2* с переменным шагом, при минимальном числе шагов *n*, причем правые части уравнений в символьной форме задаются в векторе *D*, а начальные условия – в векторе *y*;

rkfixed(y, x1, x2, n, D) – возвращает матрицу решений методом Рунге-Кутта на интервале от *x1* до *x2* при фиксированном числе шагов *n*, причем правые части уравнений записаны в символьном векторе *D*, а начальные условия – в векторе *y*.

Рассмотрим примеры интегрирования ОДУ в *Mathcad*'е.

Пример 1.

$$\begin{cases} y' = z + \sqrt{x} * \sin(x), \\ z' = y - x^2. \end{cases}$$

на интервале $x_0 = 0, x_к = 4$ с начальными условиями $y(x_0) = 1, z(x_0) = 0.$

Заданную систему ОДУ следует подготовить для решения в *Mathcad'*е. Для этого переходят к новым индексированным переменным: $y_0 = y, y_1 = z$. Исходная система примет вид

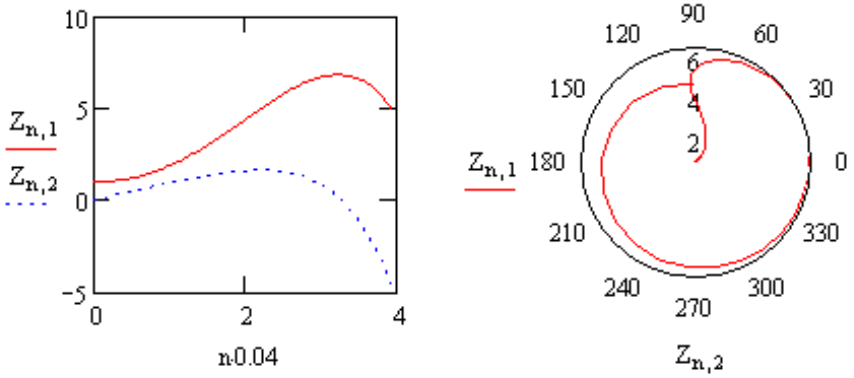
$$\begin{cases} y_0' = y_1 + \sqrt{x} * \sin(x), \\ y_1' = y_0 - x^2 \end{cases}$$

с начальными условиями $y_0(x_0) = 1,\ y_1(x_0) = 0.$

Решение системы на экране представлено декартовым графиком и полярным графиком (фазовым портретом):

$$y := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad D(x,y) := \begin{pmatrix} y_1 + \sqrt{x} \cdot \sin(x) \\ y_0 - x^2 \end{pmatrix}$$

$Z := Rkadapt(y,0,4,100,D) \qquad n := 0..99$



Пример 2

$$2y'' - 3y' + 1.5y = -1$$

на интервале $x_0 = 0, x_к = 2$ с начальными условиями $y(x_0) = 0, y'(x_0) = 1.$

После замены переменных $y_0 = y, y_1 = y'$ и преобразований получим систему

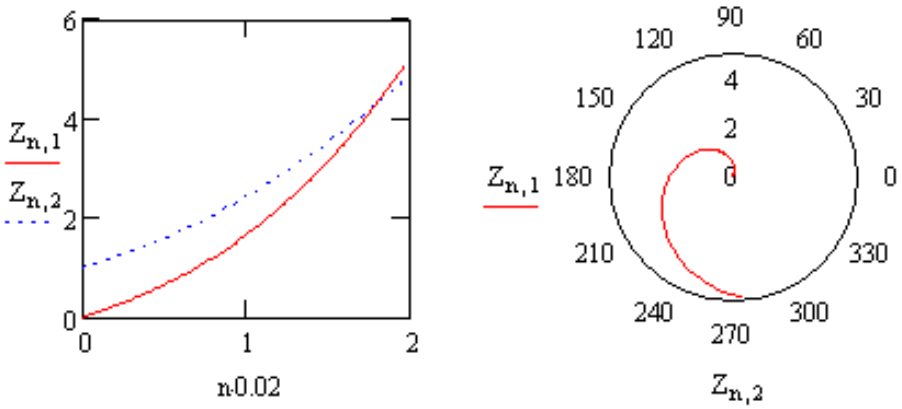
$$\begin{cases} y_0' = y_1, \\ y_1' = 1.5y_1 - 0.75y_0 - 0.5 \end{cases}$$

при $y_0(x_0) = 0, y_1(x_0) = 1.$

Результаты решения уравнения в *Mathcad'e*:

$$y := \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad D(x,y) := \begin{pmatrix} y_1 \\ 1.5 \cdot y_1 - 0.75 \cdot y_0 - 0.5 \end{pmatrix}$$

$$Z := \text{Rkadapt}(y,0,2,100,D) \qquad n := 0..99$$



Пример 3.

$$\begin{cases} x'' = x + y + (x')^t, \\ y' = -2xx' + tx. \end{cases}$$

на интервале $t_0 = 0,\ t_к = 1.5$ с начальными условиями $x(0) = 0.5,\ x'(0) = 0.5,\ y(0) = 0.$

После замены переменных $y_0 = x, y_1 = x', y_2 = y$ и преобразований получим систему

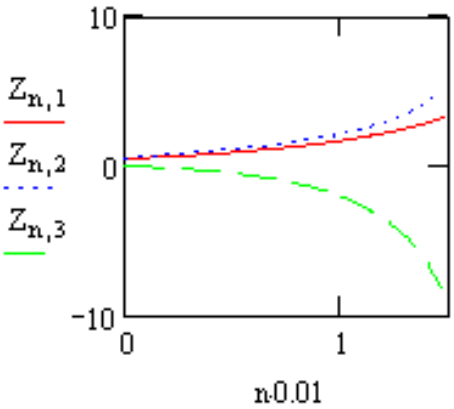
$$\begin{cases} y_0' = y_1, \\ y_1' = y_0^2 + y_2 + (y_1)^t, \\ y_2' = -2y_0y_1 + ty_0. \end{cases}$$

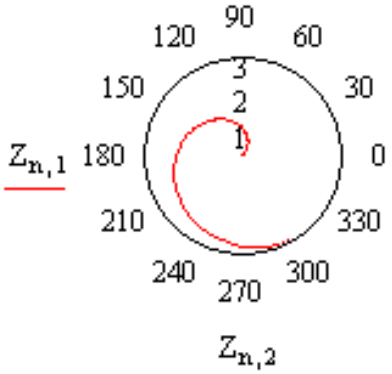
при $y_0(0) = 0.5, y_1(0) = 0.5, y_2(0) = 0.$

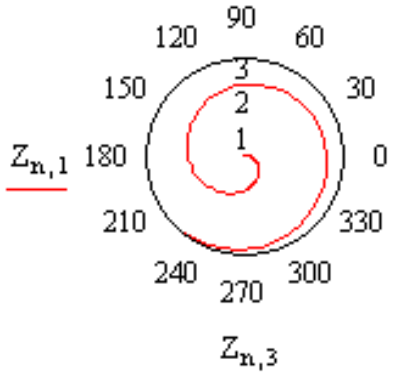
Результаты решения системы уравнений в *Mathcad'e*:

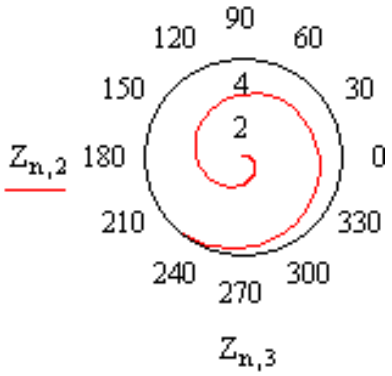
$$y := \begin{pmatrix} 0.5 \\ 0.5 \\ 0 \end{pmatrix} \qquad D(t,y) := \begin{bmatrix} y_1 \\ (y_0)^2 + y_2 + (y_1)^t \\ -2 \cdot y_0 \cdot y_1 + t \cdot y_0 \end{bmatrix} +$$

$$Z := \text{Rkadapt}(y,0,1.5,150,D) \qquad n := 0..149$$









ЛАБОРАТОРНАЯ РАБОТА 8

ИССЛЕДОВАНИЕ МЕТОДОВ ИНТЕГРИРОВАНИЯ ОДУ И СИСТЕМ ОДУ

ЗАДАНИЕ

1. Ознакомиться с заданием (варианты представлены ниже). Исходные данные взять из таблиц 1, 2, 3 (номер). Обозначения в таблицах: t_0, x_0 – начало интервала интегрирования; t_k, x_k – конец интервала интегрирования; $\Delta t, \Delta x$ – шаг выдачи результатов.
2. Выполнить задание в *Mathcad*'е.
3. Составить алгоритм и написать код для численного интегрирования указанным методом ОДУ или системы ОДУ. Шаг интегрирования h выбирается самостоятельно. Результаты интегрирования сводятся в таблицу решений и представляются также в виде графиков функций.

Варианты заданий

1. Метод Эйлера «с пересчетом», табл. 3(1), таблица решений – $x, y(x), z(x)$, графики функций – $y(x), z(x)$.
2. Метод «ломаных», табл. 3(7), таблица решений – $x, y(x), z(x)$, графики функций – $y(x), z(x)$.
3. Метод РК4, табл. 3(5), таблица решений – $x, y(x), z(x)$, графики функций – $y(x), z(x)$.
4. Метод РК4, табл. 2(1), таблица решений – $x, y(x), y'(x), y''(x), y'''(x)$, графики функций – $y(x), y'(x), y''(x), y'''(x)$.
5. Метод РК4, табл. 3(6), таблица решений – $x, y(x), z(x)$, графики функций – $y(x), z(x)$.

6. Метод прогноза и коррекции – формула (14), табл.1(1), таблица решений – $t, x(t), x'(t)$, графики функций – $x(t), x'(t)$. Для разгона – метод РК4.
7. Первый вариант метода Адамса, табл. 1(2), таблица решений – $t, x(t), x'(t)$, графики функций – $x(t), x'(t)$. Для разгона – метод РК4.
8. Второй вариант метода Адамса, табл.1(3), таблица решений – $t, x(t), x'(t)$, графики функций – $x(t), x'(t)$. Для разгона – метод РК3.
9. Метод РК3, табл. 1(4), таблица решений – $t, x(t), x'(t)$, графики функций – $x(t), x'(t)$.
10. Метод РК4, табл. 1(5), таблица решений – $t, x(t), x'(t)$, графики функций – $x(t), x'(t)$.
11. Метод прогноза и коррекции – формулы (17)...(20), табл. 1(6), таблица решений – $t, x(t), x'(t)$, графики функций – $x(t), x'(t)$. Для разгона – метод РК4.
12. Метод «ломаных», табл. 1(7), таблица решений – $t, x(t), x'(t)$, графики функций – $x(t), x'(t)$.
13. Метод Эйлера «с пересчетом», табл. 1(8), таблица решений – $t, x(t), x'(t), x''(t)$, графики функций – $x(t), x'(t), x''(t)$.
14. Метод прогноза и коррекции – формулы (21)...(24), табл. 1(9), таблица решений – $t, x(t), x'(t), x''(t)$, графики функций – $x(t), x'(t), x''(t)$. Для разгона – метод РК4.
15. Метод РК3, табл.1(10), таблица решений – $t, x(t), x'(t), x''(t)$, графики функций – $x(t), x'(t), x''(t)$.
16. Метод РК4, табл.1(11), таблица решений – $t, x(t), x'(t), x''(t)$, графики функций – $x(t), x'(t), x''(t)$.

17. Метод прогноза и коррекции – формула (14), табл. 1(12), таблица решений – $t, x(t), x'(t), x''(t)$, графики функций – $x(t), x'(t), x''(t)$. Для разгона – РК4.
18. Первый вариант метода Адамса, табл.3(4), таблица решений – $x, y(x), z(x)$, графики функций – $y(x), z(x)$. Для разгона – метод РК3.
19. Второй вариант метода Адамса, табл.3(2), таблица решений – $x, y(x), y'(x), z(x)$, графики функций – $y(x), y'(x), z(x)$. Для разгона – РК4.
20. Метод прогноза и коррекции – формулы (17)...(20), табл.3(3), таблица решений – $x, y(x), z(x)$, графики функций – $y(x), z(x)$. Для разгона – РК3.
21. Метод РК4, табл. 2(2), таблица решений – $x, y(x), y'(x), y''(x), y'''(x)$, графики функций – $y(x), y'(x), y''(x), y'''(x)$.
22. Метод РК3, табл. 2(3), таблица решений – $x, y(x), y'(x)$, графики функций – $y(x), y'(x)$.
23. Метод Эйлера «с пересчетом», табл. 2(4), таблица решений – $x, y(x), y'(x), y''(x)$, графики функций – $y(x), y'(x), y''(x)$.
24. Метод «ломаных», табл. 2(5), таблица решений – $x, y(x), y'(x)$, графики функций – $y(x), y'(x)$.
25. Метод РК4, табл. 2(6), таблица решений – $x, y(x), y'(x)$, графики функций – $y(x), y'(x)$.
26. Метод прогноза и коррекции – формула (14), табл. 2(7), таблица решений – $x, y(x), y'(x)$, графики функций – $y(x), y'(x)$. Для разгона – метод РК4.

27. Метод прогноза и коррекции – формула (15), табл. 2(8), таблица решений – $x, y(x), y'(x)$, графики функций – $y(x), y'(x)$. Для разгона – метод РК4.
28. Метод прогноза и коррекции – формула (16), табл. 2(9), таблица решений – $x, y(x), y'(x)$, графики функций – $y(x), y'(x)$. Для разгона – метод РК4.
29. Метод прогноза и коррекции – формулы (17)...(20), табл. 2(10), таблица решений – $x, y(x), y'(x)$, графики функций – $y(x), y'(x)$. Для разгона – РК4.
30. Метод прогноза и коррекции – формулы (21)...(24), табл. 2(11), таблица решений – $x, y(x), y'(x)$, графики функций – $y(x), y'(x)$. Для разгона – РК4.

Таблица 1

№	Уравнение	t_0	$x(t_0)$	$x'(t_0)$	$x''(t_0)$	t_k	Δt
1	$x'' - 6x^2 - t = 0$	0	0	0		2.0	0.1
2	$x'' + 4\sin x = 0$	0	0	1.0		4.0	0.2
3	$x'' + 0.5\sin x - 0.5\sin t = 0$	0	0	0		9.0	0.3
4	$x'' + 0.1(x')^2 + 2x = 0$	0	0.5	1.0		6.0	0.25
5	$x'' + x - \cos 3t = 0$	π/2	4	1		5π/2	0.2
6	$x'' - x - 2sht = 0$	0	0	0		4.0	0.2
7	$x''\sqrt{t} - \sqrt[3]{x} = 0$	1.0	0.5	0		10.0	0.25
8	$x''' + x''x - (x')^2 + 1 = 0$	0	0	0	0	5.0	0.25
9	$x''' - x''x + (x')^2 = 0$	0	1.0	–1.0	0.5	5.0	0.25
10	$x''' + 4x''x' = 0$	0	1.0	1.0	–0.5	4.0	0.2
11	$x''' - x = 0$	0	3.0	–1.0	1.0	3.0	0.1
12	$x''' - 3x' - 2x - 9e^{2t} = 0$	0	0	–3.0	3.0	2.5	0.1

Таблица 2

№	Уравнение	x_0	$y(x_0)$	$y'(x_0)$	$y''(x_0)$	$y'''(x_0)$	x_k	Δx
1	$y^{(4)} - y = 0$	0	1.0	1.0	1.0	1.0	5.0	0.2
2	$y^{(4)} + y'' - 2\cos x = 0$	0	–2.0	1.0	0	0	3π	$\frac{\pi}{10}$
3	$y'' = e^{-x}$	0	0	0			2.5	0.1
4	$y''' = e^x / x$	1.0	0	0	0		4.0	0.2
5	$y'' + 9y - \frac{9}{\cos 3x} = 0$	0	1.0	0			4.0	0.2
6	$y'' + 3\frac{y'}{x} + \frac{y}{x^2} - \frac{1}{x^3} = 0$	1	1	0			5.0	0.2
7	$y'' + 4y - \sin 2x - 1 = 0$	0	0.25	0			5.0	0.2
8	$y'' - 8y' + 16y - e^{4x} = 0$	0	0	1			2.0	0.1
9	$y'' + y - \frac{1}{\sin x} = 0$	$\frac{\pi}{2}$	1	0			$\frac{5\pi}{2}$	$\frac{\pi}{8}$
10	$y'' + \pi^2 y - \frac{\pi^2}{\cos \pi x} = 0$	0	3	0			4.0	0.2
11	$y'' + 4y - 4ctg\,2x = 0$	$\frac{\pi}{4}$	3	2			$\frac{5\pi}{4}$	$\frac{\pi}{20}$

Таблица 3

№	Система уравнений	x_0	$y(x_0)$	$y'(x_0)$	$z(x_0)$	x_k	Δx
1	$y' = y + z,$ $z' = \left(-\frac{2}{x^2} + \frac{2}{x} - 1\right)y$ $+\left(\frac{2}{x} - 1\right)z.$	1.0	3.0		2.0	8.0	0.2
2	$y'' = y^2 + z,$ $z' = -2yy' + y.$	0	1.0	1.0	0	4.0	0.2
3	$y' = z,$ $z' = z^2 / y.$	0	2.0		4.0	2.5	0.1
4	$y' = z,$ $z' = 0.09xz - x^2 y.$	0	0		0.1	3	0.1
5	$y' = z,$ $z' = -0.1e^{-0.5x}$	0	0		0.1	3	0.1
6	$y' = y/(y+z),$ $z' = z/(y+z).$	0	2		4	8.0	0.4
7	$y' = z + x,$ $z' = y + e^x.$	0	1		0	4.0	0.2

Пример выполнения задания в среде *C++Builder 6*

Выполнить интегрирование ОДУ

$$y''-y'+16y+e^{-x}=0$$

на интервале $x_0 = 0, \quad x_k = 2$ с начальными условиями $y(x_0) = 0, \quad y'(x_0) = 1$ и шагом выдачи результатов $\Delta x = 0{,}1$ методом РК3 и методом Адамса

с разгоном методом РК4. Результаты интегрирования $y(x)$, $y'(x)$ вывести в таблицу и представить в виде графиков.

1. Создайте новый проект командой *Файл/Новый/Приложение.*

Сохраните файлы модуля и проекта командой *Файл*/*Сохранить все* под именами LR8 и PR_LR8. Для этого удобно использовать соответствующую быструю кнопку (*Сохранить все*). В последующих сеансах работы сохраненный проект можно открыть командой *Файл/Открыть проект* (или *Повторно открыть*). Теперь перейдите к проектированию приложения – переносам на форму необходимых компонентов и заданию их свойствам значений, а в обработчиках событий – размещению кодов соответствующих алгоритмов. (Рекомендуется нажимать кнопку *Сохранить все* по окончании работы с каждым компонентом.) В результате проектирования получим форму, представленную на рис. 8.9 и на рис. 8.10.

2. Выделите форму, щелкнув на ней левой кнопкой мыши, и в свойство **Caption** (надпись) впишите *МЕТОДЫ ИНТЕГРИРОВАНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ***.**

3. В правый верхний угол формы поместите метку **Label1** (страница *Стандарт*). Выделив метку щелчком на ней, установите свойство **Font** – жирный, размер 10. В свойство **Caption** метки впишите уравнение**.**

4. Ниже метки **Label1** расположите на форме метку **LabeledEdit1** (страница *Дополнительно*). В свойство **Text** впишите значение 0,01. Раскрыв свойство **EditLabel**, установите подсвойство **Font** – жирный, размер 8, а в подсвойстве **Caption** впишите соответственно **шаг интегрирования**.

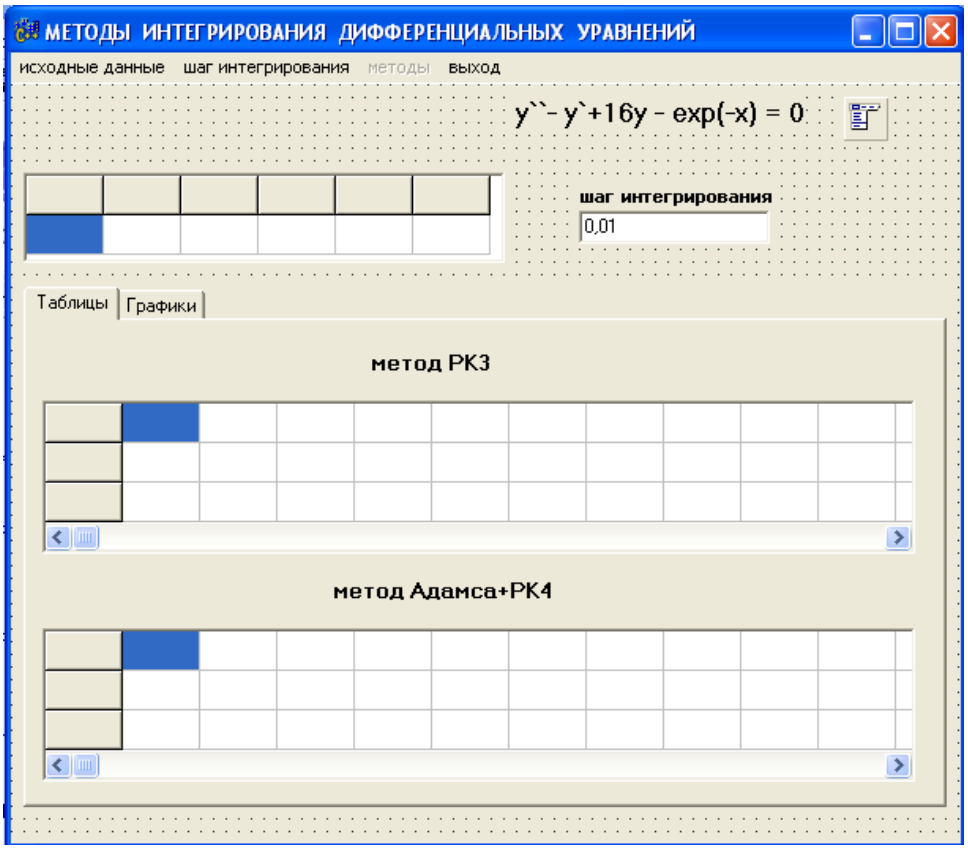


Рис. 8.9. Форма по окончании проектирования (вид 1)_

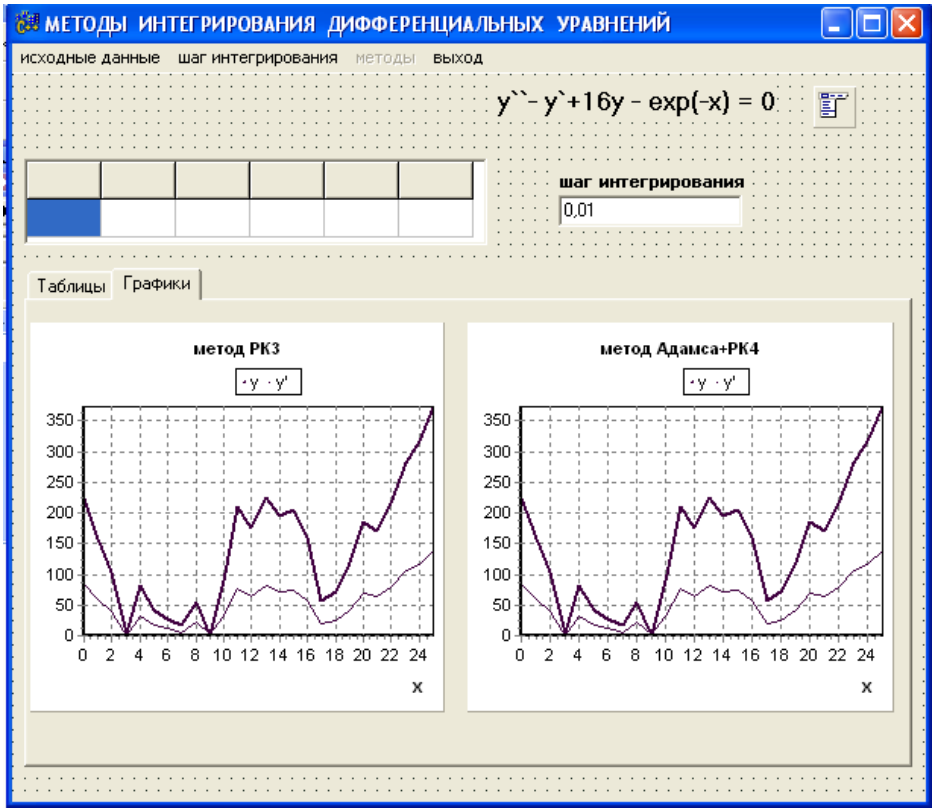


Рис. 8.10. Форма по окончании проектирования (вид 2)

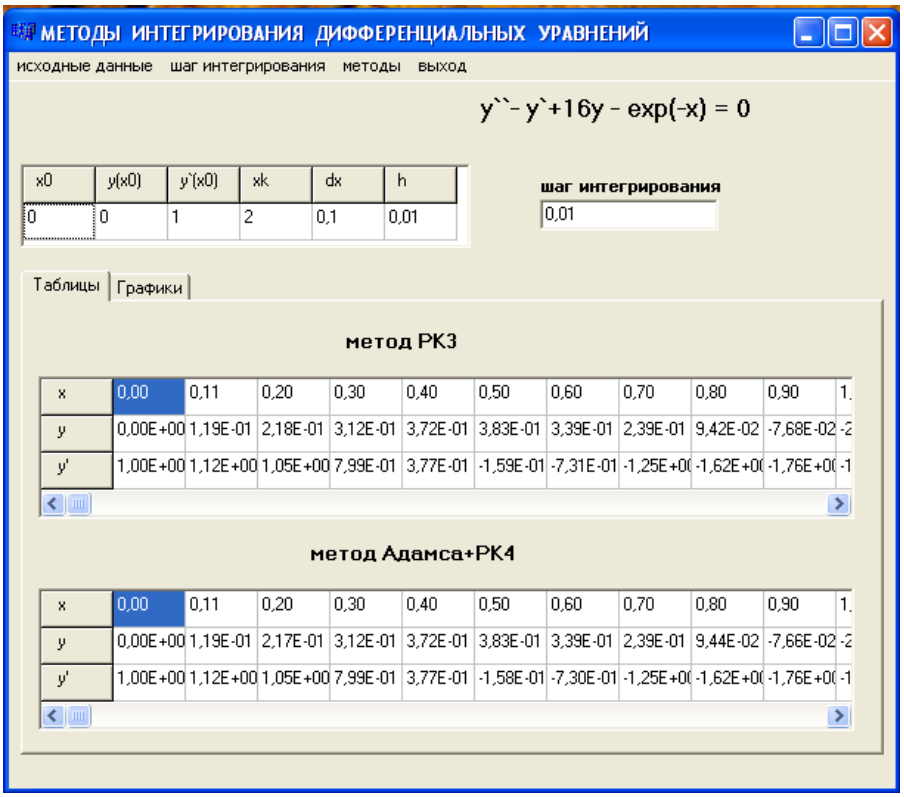


Рис. 8.11. Результаты в виде таблиц

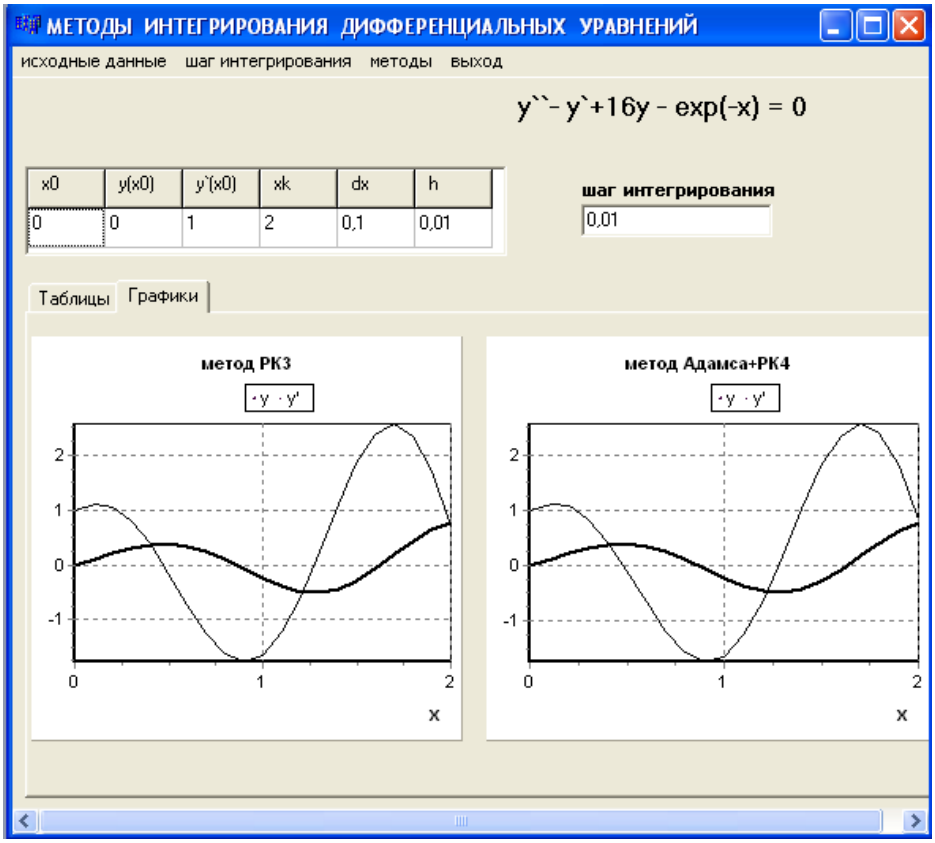


Рис. 8.12. Результаты в виде графиков

5. Для размещения исходных данных используйте в качестве таблицы компонент **StringGrid1** (страница *Дополнительно*). Установите следующие значения свойств компонента **StringGrid1**: **ColCount** – 6, **DefaultColWidth** – 48, **FixedCols** – 0, **FixedRows** – 1, **Font** – черный, обычный, размер 8, **RowCount** – 2. Раскрыв свойство **Options**, установите значение подсвойства **goEditing** – **true**, что даст возможность редактировать содержимое таблицы в **StringGrid1**.

6. Для вывода результатов в таблицы и графики перенесите на форму многостраничную панель – компонент **PageControl1** (страница *Win32)*. Щелкните на нем правой кнопкой мыши и во всплывшем меню дважды используйте команду *Новая страница*. В свойство **Caption** первой страницы впишите *Таблицы*, второй – *Графики*. Установите свойства компонента **PageControl1**: **MultiLine** – **false**, **Style** – **tsTabs**, **TabPosition** – **tpTop**. Перенесите на первую страницу (*Таблицы*) метки **Label1** и **Label2**, в свойство **Caption** которых впишите соответственно **метод РК3** и **метод Адамса+РК4,** и компоненты **StringGrid2** и **StringGrid3** (страница *Дополнительно*), в которых установите: **ColCount** – 22, **DefaultColWidth** - 48, **FixedCols** – 1, **FixedRows** – 0, **Font** – черный, обычный, размер 8, **RowCount** – 3. Перенесите на вторую страницу (*Графики*) компоненты **Chart1** и **Chart2** (страница *Additional)*. Свойства компонентов задайте согласно рис. 8.10.

7. Перенесите на форму (в правый верхний угол) компонент **MainMenu** (страница *Стандарт*), двойным щелчком на нем перейдите в *Проектировщик Меню* (окно **Form1->MainMenu1**) и сконструируйте меню с указанными головными разделами и подразделами; названия записываются в свойство **Caption** разделов и подразделов:

исходные данные	*шаг интегрирования*	*методы*	*выход*
методы	*h/2*	*РК3*	
	*2*h*	*Адамс+РК4*	
	исходный шаг		

8. Не выходя из *Проектировщика Меню* (окно **Form1->MainMenu1**), выделите головной раздел *исходные данные* и в свойство **Name** впишите *icx_dan*. В обработчик щелчка на этом разделе впишите (*курсив*):

```
void __fastcall TForm1::icx_danClick(TObject *Sender)
{
Series1->Clear();
Series2->Clear();
Series3->Clear();
Series4->Clear();
StringGrid1->Cells[0][0]="  x0";
StringGrid1->Cells[1][0]="  y(x0)";
StringGrid1->Cells[2][0]="  y`(x0)";
StringGrid1->Cells[3][0]="  xk";
StringGrid1->Cells[4][0]="  dx";
StringGrid1->Cells[5][0]="  h";
StringGrid1->Cells[0][1]=FloatToStr(0);
StringGrid1->Cells[1][1]=FloatToStr(0);
StringGrid1->Cells[2][1]=FloatToStr(1);
StringGrid1->Cells[3][1]=FloatToStr(2.0);
StringGrid1->Cells[4][1]=FloatToStr(0.1);
StringGrid1->Cells[5][1]=FloatToStr(0.01);
LabeledEdit1->Text=StringGrid1->Cells[5][1];
metod->Enabled=false;
for(int j=0;j<StringGrid2->ColCount;j++)
   for(int i=0;i<StringGrid2->RowCount;i++){
      StringGrid2->Cells[j][i]="";
      StringGrid3->Cells[j][i]="";}
}
```

Выделите подраздел *методы* и в свойство **Name** впишите *na_metod*. В обработчик щелчка на этом разделе впишите (*курсив*):

```
void __fastcall TForm1::na_metodClick(TObject *Sender)
{
  metod->Enabled=true;
  StringGrid2->Cells[0][0]="  x";
  StringGrid2->Cells[0][1]="  y";
  StringGrid2->Cells[0][2]="  y'";
  StringGrid3->Cells[0][0]="  x";
  StringGrid3->Cells[0][1]="  y";
  StringGrid3->Cells[0][2]="  y'";
}
```

9. Заданные имена остальных разделов меню и обработчики щелчков на них нетрудно видеть из приведенного ниже файла *LR8.cpp*; но следует отметить, что раздел *методы* имеет имя *metod*, его свойству **Enabled** присвоено значение **false**, а обработчик щелчка на нем отсутствует.

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "LR8.h"
#include<math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
```

```
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::outClick(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
double h;
void __fastcall TForm1::icx_danClick(TObject *Sender)
{
 Series1->Clear();
 Series2->Clear();
 Series3->Clear();
 Series4->Clear();
 StringGrid1->Cells[0][0]="  x0";
 StringGrid1->Cells[1][0]="  y(x0)";
 StringGrid1->Cells[2][0]="  y`(x0)";
 StringGrid1->Cells[3][0]="  xk";
 StringGrid1->Cells[4][0]="  dx";
 StringGrid1->Cells[5][0]="  h";
 StringGrid1->Cells[0][1]=FloatToStr(0);
 StringGrid1->Cells[1][1]=FloatToStr(0);
 StringGrid1->Cells[2][1]=FloatToStr(1);
 StringGrid1->Cells[3][1]=FloatToStr(2.0);
 StringGrid1->Cells[4][1]=FloatToStr(0.1);
 StringGrid1->Cells[5][1]=FloatToStr(0.01);
 LabeledEdit1->Text=StringGrid1->Cells[5][1];
```

```
 metod->Enabled=false;
 for(int j=0;j<StringGrid2->ColCount;j++)
    for(int i=0;i<StringGrid2->RowCount;i++){
       StringGrid2->Cells[j][i]="";
       StringGrid3->Cells[j][i]="";}
}
//---------------------------------------------------------------------------

void __fastcall TForm1::na_metodClick(TObject *Sender)
{
  metod->Enabled=true;
  StringGrid2->Cells[0][0]="   x";
  StringGrid2->Cells[0][1]="   y";
  StringGrid2->Cells[0][2]="   y'";
  StringGrid3->Cells[0][0]="   x";
  StringGrid3->Cells[0][1]="   y";
  StringGrid3->Cells[0][2]="   y'";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::dva_hClick(TObject *Sender)
{
 LabeledEdit1->Text=FloatToStr(2*StrToFloat(LabeledEdit1->Text));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::pol_hClick(TObject *Sender)
{
 LabeledEdit1->Text=FloatToStr(0.5*StrToFloat(LabeledEdit1->Text));
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::ich_hClick(TObject *Sender)
{
LabeledEdit1->Text=StringGrid1->Cells[5][1];
}
//---------------------------------------------------------------------
void __fastcall TForm1::RK3Click(TObject *Sender)
{ double y0,y1,x,xk,dx,k1,k2,k3,l1,l2,l3;
  int i=1;
  Series1->Clear();
  Series2->Clear();
  h=StrToFloat(LabeledEdit1->Text);
  y0=StrToFloat(StringGrid1->Cells[1][1]);
  y1=StrToFloat(StringGrid1->Cells[2][1]);
  x=StrToFloat(StringGrid1->Cells[0][1]);
  dx=StrToFloat(StringGrid1->Cells[4][1]);
  xk=StrToFloat(StringGrid1->Cells[3][1]);
  do{
    if(fmod(x,dx)<h){
      StringGrid2->Cells[i][0]=FloatToStrF(x,ffFixed,3,2);
      StringGrid2->Cells[i][1]=FloatToStrF(y0,ffExponent,3,2);
      StringGrid2->Cells[i][2]=FloatToStrF(y1,ffExponent,3,2);
      Series1->AddXY(x,y0,"",clBlack);
      Series2->AddXY(x,y1,"",clBlack);
      i++;
       }
    k1=y1;
    l1=y1-16*y0+exp(-x);
    k2=y1+h/2*l1;
    l2=(y1+h/2*l1)-16*(y0+h/2*k1)+exp(-(x+h/2));
```

```
    k3=y1-h*l1+2*h*l2;
    l3=(y1-h*l1+2*h*l2)-16*(y0-h*k1+2*h*k2)+exp(-(x+h));
    y0=y0+h/6*(k1+4*k2+k3);
    y1=y1+h/6*(l1+4*l2+l3);
    x+=h;}
    while(x<=xk+h);

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Ad_RK4Click(TObject *Sender)
{
  double y0,y00,y000,y01,y1,y10,y100,y11,x,xk,dx,k1,k2,k3,k4,l1,l2,l3,l4;
  int i=1,f=0;
  Series3->Clear();
  Series4->Clear();
  h=StrToFloat(LabeledEdit1->Text);
  y0=StrToFloat(StringGrid1->Cells[1][1]);
  y1=StrToFloat(StringGrid1->Cells[2][1]);
  x=StrToFloat(StringGrid1->Cells[0][1]);
  dx=StrToFloat(StringGrid1->Cells[4][1]);
  xk=StrToFloat(StringGrid1->Cells[3][1]);
  StringGrid3->Cells[i][0]=FloatToStrF(x,ffFixed,3,2);
  StringGrid3->Cells[i][1]=FloatToStrF(y0,ffExponent,3,2);
  StringGrid3->Cells[i][2]=FloatToStrF(y1,ffExponent,3,2);
  Series3->AddXY(x,y0,"",clBlack);
  Series4->AddXY(x,y1,"",clBlack);
  i++;
  k1=y1;
```

```
 l1=y1-16*y0+exp(-x);
 k2=y1+h/2*l1;
 l2=(y1+h/2*l1)-16*(y0+h/2*k1)+exp(-(x+h/2));
 k3=y1+h/2*l2;
 l3=(y1+h/2*l2)-16*(y0+h/2*k2)+exp(-(x+h/2));
 k4=y1+h*l3;
 l4=(y1+h*l3)-16*(y0+h*k3)+exp(-(x+h));
 y00=y0+h/6*(k1+2*k2+2*k3+k4);
 y10=y1+h/6*(l1+2*l2+2*l3+l4);
 x+=h;
 do{
  if(f){y0=y00;y1=y10;y00=y01;y10=y11;}
  f=1;
  if(fmod(x,dx)<h){
  StringGrid3->Cells[i][0]=FloatToStrF(x,ffFixed,3,2);
  StringGrid3->Cells[i][1]=FloatToStrF(y00,ffExponent,3,2);
  StringGrid3->Cells[i][2]=FloatToStrF(y10,ffExponent,3,2);
  Series3->AddXY(x,y0,"",clBlack);
  Series4->AddXY(x,y1,"",clBlack);
  i++;}
   y000=y00+h/2*(3*y10-y1);
   y100=y10+h/2*(3*(y10-16*y00+exp(-x))-(y1-16*y0+exp(-(x-h))));
   y01=y00+h/2*(y100+y10);
   y11=y10+h/2*((y100-16*y000+exp(-(x+h)))+(y10-16*y00+exp(-x)));
   x+=h;}
   while(x<=xk+h);
}
```

10. Запустите приложение на выполнение, нажав быстрые кнопки *Сохранить все* и *Запуск*. После щелчка на разделе меню *исходные данные*

станет доступным раздел *методы* и подразделы *РК3* и *Адамс+РК4*. После щелчков на них получим результаты в виде таблиц и графиков (рис. 8.11, рис. 8.12).

11. Исследуйте влияние шага интегрирования на результаты, изменяя шаг с помощью меню.

12. Исследуйте влияние начальных условий и соотношения шага выдачи результатов к шагу интегрирования на результаты.

13. Щелчком на пункте меню *выход* завершите выполнение задания.

Содержание отчета

1. Задание.
2. Формулы с пояснениями.
3. Результаты выполнения задания в *Mathcad*'е.
4. Блок-схема алгоритма.
5. Таблица идентификаторов.
6. Код.
7. Результаты выполнения работы в виде таблиц и графиков.
9. Библиографический список.

Контрольные вопросы

1. Как сравниваются методы интегрирования ДУ?

2. Приведите иллюстрации метода Эйлера и модифицированных методов Эйлера.

3. Приведите пример алгоритма метода РК3.

4. Приведите пример алгоритма метода РК4.

5. Приведите иллюстрацию метода прогноза и коррекции.

6. Как получить оценку ошибки ограничения в методе прогноза и коррекции?

7. Как скорректировать решение ДУ?

8. Приведите пример алгоритма метода прогноза и коррекции.

9. Сравните одношаговые и многошаговые методы интегрирования ДУ.

10. Объясните адаптивный алгоритм интегрирования ДУ методом прогноза и коррекции.

11. Как практически сравнить методы интегрирования ДУ? Что для этого нужно иметь?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Амосов А.А., Дубинский Ю.А., Копченова Н.В. Вычислительные методы для инженеров: Учеб. пособие.-2-е изд., доп. – М.: Издательство МЭИ, 2003.-596 с.

2. Вержбицкий В.М. Основы численных методов: Учебник для вузов. / В.М. Вержбицкий. - 2-е изд., перераб. - М.: Высш. шк., 2005. - 840 с.

3. Киреев В.И. Численные методы в примерах и задачах: Учеб. пособие / В.И. Киреев, А.В. Пантелеев. - М.: Высш. шк., 2004. - 480 с.

4. Турчак Л.И., Плотников Л.В. Основы численных методов: Учеб. пособие. - М.: ФИЗМАТЛИТ, 2003.- 304 с.

5. Чуркин В.В. Сборник заданий по вычислительным методам / В.В. Чуркин. - Киров: Изд-во ВятГУ, 2009. - 64 с.

6. Чуркин В.В. Лабораторные работы по вычислительным методам в среде C++ Builder 6 / В.В.Чуркин. – Киров: Изд-во ВятГУ, 2010. – 66 с.

7. Гурский Д.А. Вычисления в MathCAD / Д.А. Гурский. - Минск.: Новое знание, 2003. - 814 с.

8. Кирьянов Д.В. Самоучитель Mathcad 11. - СПб.:БХВ-Петербург, 2004. - 560 с.

9. Плис А.И., Сливина Н.А. Mathcad. Математический практикум для инженеров и экономистов: Учеб. пособие. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2003. – 656 с.

10.Макаров Е. Инженерные расчеты в Mathcad 14 (+CD) – СПб.:Питер, 2007. – 592 с.

Учебное издание

Чуркин Владимир Васильевич

ЧИСЛЕННЫЕ МЕТОДЫ

(с алгоритмами и программами в среде C++Builder)

Учебно-методическое пособие

Подписано к использованию 03.03.2014. Заказ № 1510.

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Вятский государственный университет»

610000, г. Киров, ул. Московская, 36, тел.: (8332) 64-23-56, http://vyatsu.ru