Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра высшей математики

# ЧИСЛЕННЫЕ МЕТОДЫ В КОМПЬЮТЕРНОЙ МАТЕМАТИКЕ

*Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальностей 1-36 04 01 «Программно-управляемые электронно-оптические системы», 1-39 02 02 «Проектирование и производство программно-управляемых электронных средств», 1-39 02 03 «Медицинская электроника», 1-40 01 01 «Программное обеспечение информационных технологий», 1-40 03 01 «Искусственный интеллект», 1-58 01 01 «Инженерно-психологическое обеспечение информационных технологий»*

Минск БГУИР 2025

УДК 519.61:004.42(076)
ББК 22.19я73+32.973я73
Ч67

А в т о р ы:

Т. С. Степанова, Е. А. Баркова, Л. П. Князева, П. А. Самсонов

Р е ц е н з е н т ы:

кафедра математических методов в строительстве
Белорусского национального технического университета
(протокол № 6 от 17.02.2023);

доцент кафедры дифференциальных уравнений и системного
анализа Белорусского государственного университета
кандидат физико-математических наук, доцент Н. Л. Щеглова

**Численные** методы в компьютерной математике : пособие /
Ч67 Т. С. Степанова, Е. А. Баркова, Л. П. Князева, П. А. Самсонов. – Минск :
БГУИР, 2025. – 128 с. : ил.
ISBN 978-985-543-763-6.

Предназначено для студентов очной и заочной форм обучения, выполняющих практические задания, лабораторные работы и типовые расчеты по курсу «Численные методы» с использованием системы компьютерной алгебры (СКА) Mathematica. Приведены краткие теоретические сведения по шести основным разделам курса: численным методам решения нелинейных уравнений, систем линейных и нелинейных уравнений, аппроксимации функций, численного дифференцирования и интегрирования и численного решения дифференциальных уравнений и их систем. Материал по каждому разделу содержит описание основных встроенных функций СКА Mathematica, позволяющих просто и наглядно реализовать вычислительные схемы численных методов, примеры решения типовых задач как непосредственным вычислением, так и средствами СКА, задания для самостоятельного выполнения.

**УДК 519.61:004.42(076)**
**ББК 22.19я73+32.973я73**

**ISBN 978-985-543-763-6**

# Введение

Учебная дисциплина «Численные методы» посвящена изучению вопросов реализации современной методологии – математическому моделированию, состоящему в замене объекта исследования его математической моделью и изучении ее с помощью вычислительного эксперимента, реализуемых на компьютерах вычислительных алгоритмов. Система компьютерной математики (СКА) Mathematica является одним из наиболее распространенных программных средств математического моделирования и обеспечивает проведение аналитических, численных и графических вычислений, позволяя создавать и анализировать модели, изменяя параметры в ходе моделирования. Ускоренные темпы продвижения информационной технологии математического моделирования делают важным формирование у выпускников следующих базовых профессиональных компетенций: умение выбирать эффективные алгоритмы вычислительной математики для решения поставленной профессиональной задачи, интерпретировать и анализировать результаты ее решения, что и является целью изучения дисциплины «Численные методы».

Настоящее пособие составлено в соответствии с учебной программой по курсу «Численные методы» для специальностей 1-36 04 01 «Программно-управляемые электронно-оптические системы», 1-39 02 02 «Проектирование и производство программно-управляемых электронных средств», 1-39 02 03 «Медицинская электроника», 1-40 01 01 «Программное обеспечение информационных технологий», 1-40 03 01 «Искусственный интеллект» и 1-58 01 01 «Инженерно-психологическое обеспечение информационных технологий».

Пособие предназначено для студентов очной и заочной форм обучения, выполняющих практические задания, лабораторные работы и типовые расчеты по курсу «Численные методы» с использованием СКА Mathematica. В нем приведены краткие теоретические сведения по основным разделам курса – численным методам решения нелинейных уравнений, систем линейных и нелинейных уравнений, аппроксимации функций, численного интегрирования и дифференцирования и численного решения дифференциальных уравнений. Материал по каждому разделу содержит описание основных встроенных функций СКА Mathematica, позволяющих просто и наглядно реализовать вычислительные схемы численных методов, примеры решения типовых задач как непосредственным вычислением, так и средствами СКА и задания для самостоятельного выполнения.

Авторы выражают признательность профессору кафедры компьютерных технологий и систем Белорусского государственного университета доктору физико-математических наук Таранчуку Валерию Борисовичу, оказавшему помощь в процессе работы над пособием и сделавшему ряд полезных замечаний и предложений по использованию возможностей СКА Mathematica.

# 1. Элементы теории погрешностей

## Точные и приближенные числа

При численном решении задач приходится оперировать двумя видами чисел – ***точными*** и ***приближенными***. К точным относятся числа, которые дают истинное значение исследуемой величины.

К приближенным относятся числа, близкие к истинному значению, причем степень близости определяется погрешностью вычислений. Например, в утверждениях «в аудитории 102 слушателя», «в январе 31 день» числа 102 и 31 – точные. В высказываниях «сто метров спортсмен пробежал за 10,1 секунды», «масса кирпича 2 килограмма» числа 10,1 и 2 – приближенные.

При измерениях появление приближенных чисел может быть связано с несовершенством измерительных приборов или с ошибкой наблюдателя. При счете ошибка может возникнуть, если количество предметов в системе изменяется. Кроме того, часто нет необходимости знать точное значение исследуемой величины.

## Абсолютная и относительная погрешности

Пусть $x$ – точное значение числа, а $\hat{x}$ – его приближение, или приближенное значение. Говорят, что число $\hat{x}$ есть приближенное значение числа $x$ ***с недостатком***, если $\hat{x} \le x$, и приближенное значение ***с избытком***, если $\hat{x} \ge x$.

***Погрешностью*** (или ***ошибкой***) приближенного числа $\hat{x}$ называется разность $\Delta_{\hat{x}}$ между точным значением числа $x$ и его приближенным значением $\hat{x}$: $\Delta_{\hat{x}} = x - \hat{x}$.

***Абсолютной погрешностью*** $\Delta(\hat{x})$ приближенного числа $\hat{x}$ называется модуль (или абсолютное значение) его погрешности $\Delta_{\hat{x}}$: $\Delta(\hat{x}) = |x - \hat{x}|$.

Отсюда следует, что $x = \hat{x} \pm \Delta(\hat{x})$.

Абсолютная погрешность $\Delta(\hat{x})$ – это всегда неотрицательная величина, измеряемая в тех же единицах, что и $x$.

***Относительной погрешностью*** $\delta(\hat{x})$ приближенного числа $\hat{x}$ называется отношение его абсолютной погрешности $\Delta(\hat{x})$ к модулю соответствующего точного числа $x$ или, чаще, к модулю приближенного числа $\hat{x}$, т. к. точное значение $x$ обычно неизвестно:

$$\delta(\hat{x}) = \frac{\Delta(\hat{x})}{|x|} = \frac{|x - \hat{x}|}{|x|} \quad \text{или} \quad \delta(\hat{x}) = \frac{\Delta(\hat{x})}{|\hat{x}|} = \frac{|x - \hat{x}|}{|\hat{x}|}.$$

Отсюда следует равенство $x = \hat{x} \cdot (1 \pm \delta(\hat{x}))$.

Относительная погрешность $\delta(\hat{x})$ – это всегда неотрицательная безразмерная величина, которую часто выражают в процентах:

$$\delta(\hat{x})\ \% = \frac{\Delta(\hat{x})}{|\hat{x}|} \cdot 100\ \%.$$

Поскольку точное число *x* чаще всего неизвестно, то невозможно определить абсолютную и относительную погрешности $\Delta(\hat{x})$ и $\delta(\hat{x})$. Поэтому вводят их оценки сверху – так называемые предельную абсолютную и предельную относительную погрешности.

***Предельной абсолютной погрешностью*** $\Delta$ приближенного числа $\hat{x}$ называется любое число, не меньшее абсолютной погрешности $\Delta(\hat{x})$ этого числа, т. е.

$$\Delta(\hat{x}) = |x - \hat{x}| \le \Delta.$$

Отсюда следует, что

$$\hat{x} - \Delta \le x \le \hat{x} + \Delta.$$

При этом говорят, что число $\hat{x}$ является *приближением* числа *x с точностью до* $\Delta$ и пишут

$$x = \hat{x} \pm \Delta.$$

Интервал $[x - \Delta,\ x + \Delta]$ называется ***интервалом приближения*** (или ***доверительным интервалом***) величины $x$.

***Предельной относительной погрешностью*** $\delta$ приближенного числа $\hat{x}$ называется любое число, не меньшее относительной погрешности $\delta(\hat{x})$ этого числа, т. е.

$$\delta(\hat{x}) = \frac{|x - \hat{x}|}{|\hat{x}|} \le \delta.$$

Тогда

$$\hat{x}(1 - \delta) \le x \le \hat{x}(1 + \delta), \text{ если } \hat{x} > 0,$$

$$\hat{x}(1 + \delta) \le x \le \hat{x}(1 - \delta), \text{ если } \hat{x} < 0.$$

При этом пишут

$$x = \hat{x}(1 \pm \delta).$$

Если известна предельная относительная погрешность $\delta$ приближенного числа $\hat{x}$, то в качестве его предельной абсолютной погрешности можно взять $\Delta = |\hat{x}| \cdot \delta$.

## Запись приближенных чисел. Верные значащие цифры

Запись приближенных чисел должна подчиняться правилам, связанным с понятиями о верных значащих цифрах.

***Значащей цифрой*** приближенного числа называется любая отличная от нуля цифра, а также цифра 0, если она является сохраненным десятичным знаком точного числа. Нули, расположенные слева, если они есть, значащими не считаются.

Например, число 0,00028745 имеет пять значащих цифр (2, 8, 7, 4 и 5), а число 200,37500, являющееся приближением точного числа 200,375012, имеет семь значащих цифр (2, 0, 0,3, 7, 5 и 0). Последняя цифра 0 не является сохраненным знаком точного числа, поэтому значащей не считается.

Пусть число $\hat{x}$ в десятичной системе счисления имеет вид

$$\hat{x} = \overline{a_n a_{n-1} \ldots a_0 , a_{-1} a_{-2} \ldots} \ .$$

Цифра $a_i$ числа $\hat{x}$ называется ***верной в широком смысле***, если $\Delta(\hat{x}) \le 10^i$, т. е. если абсолютная погрешность числа $\hat{x}$ не превосходит одной единицы соответствующего разряда десятичного числа. Цифра $a_i$ числа $\hat{x}$ называется ***верной в узком*** (или ***строгом***) ***смысле***, если $\Delta(\hat{x}) \le \frac{1}{2} \cdot 10^i$, т. е. если абсолютная погрешность числа $\hat{x}$ не превосходит половины единицы соответствующего разряда. Цифры, не являющиеся верными, называются ***сомнительными***.

Далее будем рассматривать только верные в широком смысле цифры.

Если верная цифра $a_i$ – значащая, то она называется ***верной значащей цифрой***.

***Замечание.*** Приближенные числа принято записывать таким образом, чтобы все цифры числа, кроме нулей слева, если они есть, были значащими и верными цифрами. Приближенное число, в отличие от точного, дает не только числовое значение какой-либо величины, но и точность, с которой оно задано. Например, числа 3,41; 3,410 и 3,4100 – различные приближенные числа. Абсолютная погрешность числа 3,41 не превосходит 0,01, числа 3,410 – 0,001, а абсолютная погрешность числа 3,4100 не превосходит 0,0001.

Если цифра $a_i$ является верной значащей цифрой приближенного числа, то это не значит, что она обязательно совпадает с соответствующей цифрой (десятичным знаком) точного числа. Например, число 2853 с одной, двумя или тремя верными значащими цифрами записывается в виде 3000, 2900 или 2850 соответственно.

**Пример 1.1.** Числа 27,0563 и 27,0563000, как приближенные, различны. Абсолютная погрешность первого числа не превосходит $10^{-4}$ $(i = -4)$, а второго – $10^{-7}$ $(i = -7)$. ▲

**Пример 1.2.** Числа $x_1 = 34{,}174562$ и $x_2 = 375{,}16342$ содержат пять верных цифр. Определить их абсолютную погрешность.

$\Delta$ В числе $x_1$ последняя верная значащая цифра 4, считая слева, находится на третьем месте после запятой, т. е. $i = -3$. Следовательно, по определению $\Delta(\hat{x}_1) = 10^{-3} = 0{,}001$.

Во втором числе $x_2$ последняя значащая цифра 6 имеет индекс $i = -2$. Следовательно, $\Delta(\hat{x}_2) = 10^{-2} = 0{,}01$. ▲

**Пример 1.3.** Абсолютная погрешность чисел $x_1 = 28{,}9356$ и $x_2 = 183{,}45123$ составляет $\Delta = 0{,}03$. Определить, какие цифры являются верными, и округлить числа, оставив только верные цифры.

$\Delta$ При заданной максимальной абсолютной погрешности $\Delta = 0{,}03$ верной значащей цифрой по определению должна быть та $a_i$, для которой

$$\Delta(x) = 0{,}03 \le 10^i,$$

т. е. $i = -1$. Значит, в заданных числах верными являются все цифры до разряда десятых, считая слева. Для числа $x_1$ это цифры 2, 8 и 9, для числа $x_2$ – цифры 1, 8, 3 и 4. Округлив эти числа до указанного разряда, получим $\hat{x}_1 = 28{,}9$, $\hat{x}_2 = 183{,}5$.

Найдем абсолютные погрешности чисел $\hat{x}_1$ и $\hat{x}_2$. Они равны сумме погрешности заданного числа $\Delta$ и погрешности округления:

$$\Delta(\hat{x}_1) = \Delta + \Delta_{окр} = 0{,}03 + |28{,}9356 - 28{,}9| = 0{,}0656,$$

$$\Delta(\hat{x}_2) = \Delta + \Delta_{окр} = 0{,}03 + |183{,}45123 - 183{,}5| = 0{,}07877.$$

Так как найденные погрешности не превышают $10^{-1}$, то все цифры чисел $\hat{x}_1$ и $\hat{x}_2$ являются верными. ▲

### Связь между числом верных знаков и погрешностью числа

Пусть число $\hat{x}$ является приближенным значением точного числа $x$ и имеет вид

$$\hat{x} = a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + a_i \cdot 10^i + \ldots + a_{-m} \cdot 10^{-m}, \quad a_n \neq 0,$$

где цифры $a_n$, $a_{n-1}$, ..., $a_i$ – верные.

По определению число верных знаков числа $\hat{x}$ определяется неравенством $\Delta(\hat{x}_1) = |x - \hat{x}| < 10^i$. Разделив обе части этого неравенства на $|\hat{x}|$, получим

$$\delta(\hat{x})=\frac{|x-\hat{x}|}{|\hat{x}|}\le\frac{10^{i}}{\left|a_n\cdot10^{n}+a_{n-1}\cdot10^{n-1}+\ldots+a_i\cdot10^{i}+\ldots+a_{-m}\cdot10^{-m}\right|}.$$

Это неравенство еще больше усилится, если число $\hat{x}$ заменить заведомо меньшим числом $a_n\cdot10^{n}$. Тогда будем иметь

$$\delta(\hat{x})\le\frac{10^{i}}{a_n\cdot10^{n}}=\frac{1}{a_n}\cdot10^{i-n}.$$

Таким образом, относительная погрешность числа характеризуется количеством верных цифр числа, отсчитываемых от первой значащей цифры числа ($a_n$) до последней значащей цифры этого числа ($a_i$). В качестве ***предельной относительной погрешности*** $\delta$ приближенного числа $\hat{x}$ можно принять величину

$$\delta_{\hat{x}}=\frac{1}{a_n}\cdot10^{i-n}.$$

**Пример 1.4.** Какова предельная относительная погрешность в процентах чисел $x_1=358{,}563$ и $x_2=0{,}047$, если все их цифры верные?

$\Delta$ Для числа $x_1$ имеем $n=2$, $i=-3$. По формуле для $\delta_{\hat{x}}$ получаем

$$\delta_1=\frac{1}{3}\cdot10^{-3-2}\cdot100\,\%\approx0{,}0003\,\%.$$

Для числа $x_2$ при $n=-2$, $i=-3$ будем иметь

$$\delta_2=\frac{1}{4}\cdot10^{-3-(-2)}\cdot100\,\%=2{,}5\,\%.\ \blacktriangle$$

**Пример 1.5.** С каким минимальным количеством верных цифр нужно взять число $\sqrt[3]{28}$, чтобы погрешность не превышала 0,25 %?

$\Delta$ Так как $\sqrt[n]{1+\alpha}\approx1+\frac{\alpha}{n}$, при достаточно малых $\alpha$, то

$$\sqrt[3]{28}=\sqrt[3]{27+1}=3\cdot\sqrt[3]{1+\frac{1}{27}}\approx3\cdot\left(1+\frac{1}{81}\right)\approx3{,}037037.$$

По условию $\delta\le0{,}25\,\%$, т. е. $\delta\le0{,}0025$. Для числа $\sqrt[3]{28}$ имеем $a_n=3,\ n=0$.

Согласно формуле для предельной относительной погрешности $\delta_{\hat{x}}$ получим

$$\frac{1}{3}\cdot 10^{i-0} \le 0{,}0025 \;\Leftrightarrow\; 10^{i} \le 0{,}0075 \;\Rightarrow\; i=-3.$$

Таким образом, число $\sqrt[3]{28}\approx 3{,}037037$ нужно округлить до разряда тысячных, оставив четыре верные значащие цифры, т. е. $\sqrt[3]{28}\approx 3{,}037$. ▲

### Оценки погрешности при арифметических действиях

Оценим теперь погрешности, возникающие при арифметических операциях с приближенными числами.

Пусть $x$ и $y-$ точные числа, $\hat{x}$ и $\hat{y}-$ их приближения, $\Delta(\hat{x})$ и $\Delta(\hat{y})-$ их абсолютные погрешности, $\delta(\hat{x})$ и $\delta(\hat{y})$ – их относительные погрешности.

**1.** ***Погрешность суммы и разности.*** Согласно определению имеем

$$\Delta(\hat{x}+\hat{y})=\left|(x+y)-(\hat{x}+\hat{y})\right|=\left|(x-\hat{x})+(y-\hat{y})\right|\le\left|x-\hat{x}\right|+\left|y-\hat{y}\right|=\Delta(\hat{x})+\Delta(\hat{y}).$$

Таким образом,

$$\Delta(\hat{x}+\hat{y})\le\Delta(\hat{x})+\Delta(\hat{y}),$$

т. е. ***абсолютная погрешность суммы*** *не превосходит суммы их абсолютных погрешностей*.

Аналогично получаем оценку погрешности для ***разности*** двух чисел:

$$\Delta(\hat{x}-\hat{y})=\left|(x-y)-(\hat{x}-\hat{y})\right|=\left|(x-\hat{x})-(y-\hat{y})\right|\le\left|x-\hat{x}\right|+\left|y-\hat{y}\right|=\Delta(\hat{x})+\Delta(\hat{y}),$$

т. е.

$$\Delta(\hat{x}-\hat{y})\le\Delta(\hat{x})+\Delta(\hat{y}).$$

Пусть теперь $x_1, x_2, ..., x_n$ – точные числа, а $\hat{x}_1, \hat{x}_2, ..., \hat{x}_n$ – соответствующие приближения. Тогда, обобщая изложенное выше, будем иметь

$$\Delta\left(\hat{x}_1\pm\hat{x}_2\pm...\pm\hat{x}_n\right)\le\Delta(\hat{x}_1)+\Delta(\hat{x}_2)+...+\Delta(\hat{x}_n).$$

Отсюда следует, что ***предельная абсолютная погрешность алгебраической суммы*** приближенных чисел равна сумме их предельных абсолютных погрешностей.

Из полученных результатов вытекает следующее ***правило сложения приближенных чисел различной абсолютной точности***:

а) выделяются числа, имеющие наибольшую абсолютную погрешность (числа наименьшей точности);

б) более точные числа округляются таким образом, чтобы сохранить в них на один знак больше, чем в выделенном числе (запасной знак);

в) производится сложение или вычитание всех чисел с учетом сохраненных знаков;

г) полученный результат округляется на один знак.

**Пример 1.6.** Вычислить $a=0{,}1732+17{,}45-0{,}00354$, где все цифры чисел верные.

$\Delta$ Поскольку цифры всех чисел верные, то погрешность вычисления первого слагаемого не превышает 0,0001, второго – 0,01, третьего – 0,00001. Наибольшую погрешность имеет второе слагаемое 17,45, поэтому числа 0,1732 и 0,00354 округляются до трех знаков после запятой. В результате получим приближенное число

$$\hat{a}=0{,}173+17{,}45-0{,}004=17{,}619.$$

Допускаемая при этом погрешность $\Delta(\hat{a})$ не превышает величины

$$\Delta=0{,}01+0{,}001+0{,}00001=0{,}01101.$$

Округлив $\hat{a}$ до двух знаков после запятой, окончательно получим $\hat{a}=17{,}62$. ▲

***Относительная погрешность суммы и разности*** приближенных чисел определяются соотношениями:

$$\delta(\hat{x}+\hat{y})=\frac{\Delta(\hat{x}+\hat{y})}{|\hat{x}+\hat{y}|}\le\frac{\Delta(\hat{x})+\Delta(\hat{y})}{|\hat{x}+\hat{y}|},\qquad \delta(\hat{x}-\hat{y})=\frac{\Delta(\hat{x}-\hat{y})}{|\hat{x}-\hat{y}|}\le\frac{\Delta(\hat{x})+\Delta(\hat{y})}{|\hat{x}-\hat{y}|}.$$

**2.** ***Погрешность произведения***. Согласно определению $x=\hat{x}\pm\Delta(\hat{x})$, $y=\hat{y}\pm\Delta(\hat{y})$. Тогда

$$\Delta(\hat{x}\cdot\hat{y})=\left|(\hat{x}+\Delta(\hat{x}))(\hat{y}+\Delta(\hat{y}))-\hat{x}\hat{y}\right|=\left|\hat{x}\Delta(\hat{y})+\hat{y}\Delta(\hat{x})+\Delta(\hat{x})\Delta(\hat{y})\right|\le$$

$$\le|\hat{x}|\Delta(\hat{y})+|\hat{y}|\Delta(\hat{x})+\Delta(\hat{x})\Delta(\hat{y}).$$

Таким образом, справедливы формулы

$$\Delta(\hat{x}\cdot\hat{y})\le|\hat{x}|\Delta(\hat{y})+|\hat{y}|\Delta(\hat{x})+\Delta(\hat{x})\Delta(\hat{y}),$$

$$\delta(\hat{x}\cdot\hat{y})=\frac{\Delta(\hat{x}\cdot\hat{y})}{|\hat{x}\cdot\hat{y}|}=\frac{|\hat{x}|\Delta(\hat{y})+|\hat{y}|\Delta(\hat{x})+\Delta(\hat{x})\Delta(\hat{y})}{|\hat{x}|\cdot|\hat{y}|}=\delta(\hat{x})+\delta(\hat{y})+\delta(\hat{x})\cdot\delta(\hat{y}).$$

Отсюда следует ***правило умножения приближенных чисел различной абсолютной точности***:

а) выделяется число с наименьшим количеством верных значащих цифр;

б) оставшиеся сомножители округляются таким образом, чтобы они содержали на одну значащую цифру больше, чем количество верных значащих цифр в выделенном числе;

в) в произведении сохраняется столько значащих цифр, сколько верных значащих цифр имеет выделенное число.

**Пример 1.7.** Найти произведение приближенных чисел $x_1 = 3{,}75$ и $x_2 = 0{,}73788$, все цифры которых верные.

$\Delta$ Число $x_1$ содержит три верные значащие цифры, а число $x_2$ – пять. Поэтому второе число $x_2$ округляем до четырех значащих цифр: $x_2 = 0{,}7379$. Тогда $\hat{a} = x_1 \cdot x_2 = 3{,}75 \cdot 0{,}7379 = 2{,}767125$.

В этом произведении сохраняем три значащие цифры, так что $\hat{a} = 2{,}77$.

Заметим, что если считать числа $x_1$ и $x_2$ точными, то $x_1 \cdot x_2 = 3{,}75 \cdot 0{,}73788 = 2{,}76705$. ▲

**3.** ***Погрешность частного***. При $y \neq 0$ и $\hat{y} \neq 0$ имеем

$$\Delta\left(\frac{\hat{x}}{\hat{y}}\right) = \left|\frac{\hat{x}+\Delta(\hat{x})}{\hat{y}+\Delta(\hat{y})} - \frac{\hat{x}}{\hat{y}}\right| = \left|\frac{\hat{y}\cdot\Delta(\hat{x}) - \hat{x}\cdot\Delta(\hat{y})}{\hat{y}^2\left(1+\dfrac{\Delta(\hat{y})}{\hat{y}}\right)}\right| \le \frac{|\hat{x}|\cdot\Delta(\hat{y}) + |\hat{y}|\cdot\Delta(\hat{x})}{\hat{y}^2\left(1+\dfrac{\Delta(\hat{y})}{\hat{y}}\right)}.$$

Так как $|a+b| \ge \big||a|-|b|\big|$, то $\left|1+\dfrac{\Delta(\hat{y})}{\hat{y}}\right| \ge \left|1-\left|\dfrac{\Delta(\hat{y})}{\hat{y}}\right|\right| = |1-\delta(\hat{y})|$. Отсюда получаем оценки погрешностей частного:

$$\Delta\left(\frac{\hat{x}}{\hat{y}}\right) \le \frac{|\hat{x}|\cdot\Delta(\hat{y}) + |\hat{y}|\cdot\Delta(\hat{x})}{\hat{y}^2\cdot|1-\delta(\hat{y})|}, \qquad \delta\left(\frac{\hat{x}}{\hat{y}}\right) = \Delta\left(\frac{\hat{x}}{\hat{y}}\right)\Big/\left|\frac{\hat{x}}{\hat{y}}\right| \le \frac{\delta(\hat{x})+\delta(\hat{y})}{|1-\delta(\hat{y})|}.$$

Таким образом, получаем следующее ***правило деления приближенных чисел различной абсолютной погрешности***:

а) выделяется число с наименьшим количеством верных значащих цифр;

б) оставшиеся числа округляются таким образом, чтобы они содержали на однузначащую цифру больше, чем выделенное число;

в) в полученном результате деления сохраняется столько значащих цифр, сколько верных значащих цифр имеет выделенное число.

На практике при работе с числами достаточно хорошей точности обычно считают, что $\Delta(\hat{x})\cdot\Delta(\hat{y}) \approx 0$, $\delta(\hat{x})\cdot\delta(\hat{y}) \approx 0$. Поэтому вместо полученных ранее формул пользуются **упрощенными формулами**:

$$\Delta(\hat{x}\cdot\hat{y}) \le |\hat{x}|\,\Delta(\hat{y}) + |\hat{y}|\,\Delta(\hat{x}), \quad \delta(\hat{x}\cdot\hat{y}) \le \delta(\hat{x}) + \delta(\hat{y});$$

$$\Delta\left(\frac{\hat{x}}{\hat{y}}\right) \le \frac{|\hat{x}|\cdot\Delta(\hat{y}) + |\hat{y}|\cdot\Delta(\hat{x})}{\hat{y}^2}, \quad \delta\left(\frac{\hat{x}}{\hat{y}}\right) \le \delta(\hat{x}) + \delta(\hat{y}).$$

Отсюда следует, что ***предельная относительная погрешность произведения и частного*** приближенных чисел, отличных от нуля, равна сумме их предельных относительных погрешностей.

Можно также показать, что:

– ***предельная относительная погрешность степени*** приближенного числа $\hat{x}^m$ равна произведению степени $m$ на предельную относительную погрешность числа $\hat{x}$: $\delta\left(\hat{x}^m\right) = m \cdot \delta\left(\hat{x}\right)$;

– ***предельная относительная погрешность корня с натуральным показателем*** $\sqrt[n]{\hat{x}}$ равна предельной относительной погрешности числа $\hat{x}$, деленной на показатель степени $n$: $\delta(\sqrt[n]{\hat{x}}) = \frac{1}{n} \cdot \delta(\hat{x})$.

## Задания

**1.** Числа $a$ и $b$ содержат $n$ верных цифр. Определить их абсолютную погрешность:

а) $a = 43{,}25601,\ b = 184{,}59982,\ n = 6$;

б) $a = 1565{,}7543,\ b = 2{,}33294,\ n = 4$;

в) $a = 2{,}85463,\ b = 0{,}375124,\ n = 5$.

**2.** Абсолютная погрешность чисел $a$ и $b$ равна $\Delta$. Определить, какие цифры чисел $a$ и $b$ являются верными, и округлить их, оставив только верные цифры:

а) $a = 43{,}25601,\ b = 184{,}59982,\ \Delta = 0{,}02$;

б) $a = 1565{,}7543,\ b = 2{,}33294,\ \Delta = 0{,}6$;

в) $a = 2{,}85463,\ b = 0{,}375124,\ \Delta = 0{,}0042$.

**3.** Найти предельные относительные погрешности в процентах чисел $a$ и $b$, если все их цифры верные:

а) $a = 0{,}0148201,\ b = 32{,}9058$;

б) $a = 4531{,}17,\ b = 0{,}0094020$;

в) $a = 7490{,}020300,\ b = 0{,}63204$.

**4.** Вычислить $x$ и найти его предельные абсолютную и относительную погрешности:

а) $x = a + b - c$, $a = 23{,}723 \pm 0{,}005$, $b = 54{,}8 \pm 0{,}04$, $c = 7{,}2 \pm 0{,}03$;

б) $x = (a - b)/c^2$, $a = 17{,}21 \pm 0{,}02$, $b = 12{,}32 \pm 0{,}03$, $c = 25{,}16 \pm 0{,}01$;

в) $x = \sqrt{ab}/c$, $a = 4{,}385 \pm 0{,}003$, $b = 15{,}16 \pm 0{,}005$, $c = 9{,}2 \pm 0{,}1$.

**5.** Найти сумму и произведение приближенных чисел, все цифры которых верные:

а) $a = 0{,}023182$, $b = 46{,}80$;

б) $a = 7{,}542108$, $b = 3{,}79$.

# 2. Основы работы в системе Wolfram Mathematica

## Запуск программы. Главное меню программы

Для запуска программы **Mathematica** необходимо щелкнуть иконку **Mathematica** в меню «Программы» или ярлык программы в месте его расположения.

При запуске программы на экране появляется главное окно (рис. 2.1).

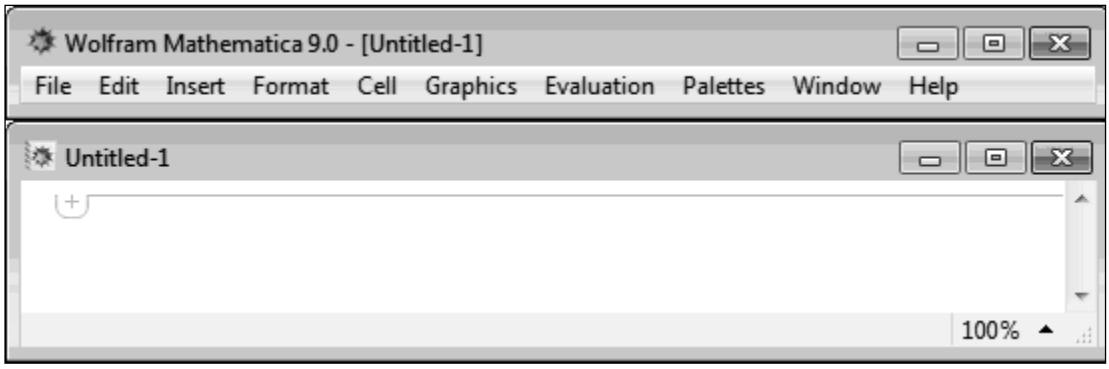



Рис. 2.1

Главное окно системы содержит строку заголовка, главное меню (File, Edit, Insert, Format, Cell, Graphics, Evaluation, Palettes, Window, Help) и большой экран редактирования (окно ввода).

### *Краткое описание пунктов главного меню системы*

**1. File** – действия с файлами программы:

- ***New*** – создание нового файла;
- ***Open*** – открытие из каталога уже созданного файла;
- ***Close*** – закрытие файла без сохранения;
- ***Save*** – сохранение файла с прежним именем;
- ***Save As*** – сохранение файла с новым именем;
- ***Printing Settings*** – управление параметрами представления данных на экране;
- ***Print*** – печать документа;
- ***Exit*** – завершение работы всей программы.

**2. Edit** – операции редактирования:

- ***Undo*** – отмена операции;
- ***Cut*** – удаление выделенного фрагмента документа и помещение его в буфер обмена;
- ***Copy*** – копирование;

– ***Paste*** – вставка фрагментов из буфера обмена в заданную область документа;

– ***Clear*** – удаление фрагментов документа без его сохранения в буфере;

– ***Copy As*** – копирование в заданном формате;

– ***Select All*** – выделение всего документа;

– ***Check Spelling*** – проверка орфографии;

– ***Find*** – задание шаблона для поиска;

– ***Find Next*** – переход к следующему фрагменту, совпадающему с шаблоном;

– ***Find Previous*** – переход к предыдущему фрагменту, совпадающему с шаблоном;

– ***Preferences*** – вызов окна настроек системы.

**3. Insert** – введение данных в окно редактирования. Например, в подпункте ***Typesetting***:

– ***Superscript*** – верхний индекс;

– ***Subscript*** – нижний индекс;

– ***Matching*** [ ] – текст в скобках [ ];

– ***Matching*** ( ) – текст в скобках ( );

– ***Matching*** { } – текст в скобках { } и т. д.

Можно вставлять в текст графику, формулы и т. д.

**4. Format** – изменение формата текста на экране и при печати, установка стилей, управление окном редактирования, стиль ячеек, их содержание, размер, управление шрифтами и т. д.:

– ***Style*** – установка параметров текста (шрифт, размер символов и т. д.);

– ***Screen Environment*** – изменение формата текста на экране; имеет следующие установки:

– ***Working*** – стиль типичный;

– ***Presentation*** – презентационный стиль с увеличением размера символов;

– ***Condensed*** – сжатый размер символов;

– ***Printout*** – оптимальный стиль для печати.

**5. Cell** – работа с ячейками:

– ***Convert To*** – преобразование формата содержимого ячеек:

– ***InputForm*** – формат ввода;

– ***OutputForm*** – формат вывода;

– ***StandartForm*** – стандартный формат;

– ***TradicionalForm*** – традиционный формат;

– ***Bitmap*** – растровый формат изображений.

При работе с большим числом математических знаков целесообразно использовать стандартный формат.

- ***Cell Properties*** – установление формата ячеек:
  - ***Open*** – устанавливает ячейку открытой или закрытой;
  - ***Editable*** – устанавливает ячейку редактируемой или нередактируемой;
  - ***Evaluatable*** – устанавливает ячейку оцениваемой или неоцениваемой;
  - ***Initialization Cell*** – делает ячейку инициализационной или неинициализационной.
- ***Grouping*** – группировка ячеек:
  - ***Automatic Grouping*** – объединение ячеек в соответствии с их стилем;
  - ***Manual Grouping*** – объединение и разъединение ячеек.

По умолчанию выбран режим ***Automatic Grouping*** .

- ***Divide Cell*** – разделение сгруппированных ячеек;
- ***Merge Cells*** – объединение выделенных ячеек.

**6. Graphics** – работа с графическими данными:

- ***New Graphics*** – вывод окна для построения графика;
- ***Drawing Tools*** – вывод окна графического редактора;
- ***Rendering*** – вывод подменю операций рендеринга:
  - ***Animate Selected Graphics*** – анимация с графиком выделенной ячейки;
  - ***Align Selected Graphics*** – выравнивание графиков;
  - ***Make Standart Size*** – установка стандартного размера ячейки;
  - ***Rerended Graphics*** – построение графиков заново.

**7. Evaluation** – управление процессом вычислений:

- ***Evaluate Cells*** – вычисление выделенных ячеек;
- ***Evaluate in Place*** – вычисление выделенных выражений в строке ввода;
- ***Evaluate Initialization Cells*** – вычисление инициализированных ячеек без их выделения;
- ***Evaluate Notebook*** – вычисление всех ячеек документа;
- ***Interrupt Evaluation*** – прерывание текущего вычисления;
- ***Abort Evaluation*** – сбрасывание текущего вычисления;
- ***Remove from Evaluation Queue*** – отмена вычисления ячеек, стоящих на очереди.

Следующие опции связаны с возможностью использования ядра ( ***Kernel*** ) другого компьютера:

- ***Default Kernel*** – выбор ядра, используемого по умолчанию;
- ***Notebook's Kernel*** – выбор ядра для вычислений в текущем документе;

- ***Start Kernel*** – запуск выбранного ядра;
- ***Quit Kernel*** – завершение работы ядра.

**8. Palettes** – управление вводом данных.

**Mathematica** позволяет осуществлять ввод данных в окно ввода двумя способами: вручную с клавиатуры и с использованием так называемых палитр (***Palettes*** – панели с кнопками быстрого управления). Они представляют собой окна, содержащие набор кнопок, за которыми закреплены определенные действия, и выпадающих списков (рис. 2.2). Можно выводить палитры на экран и убирать с экрана, создавать собственные палитры с требуемым набором функций.

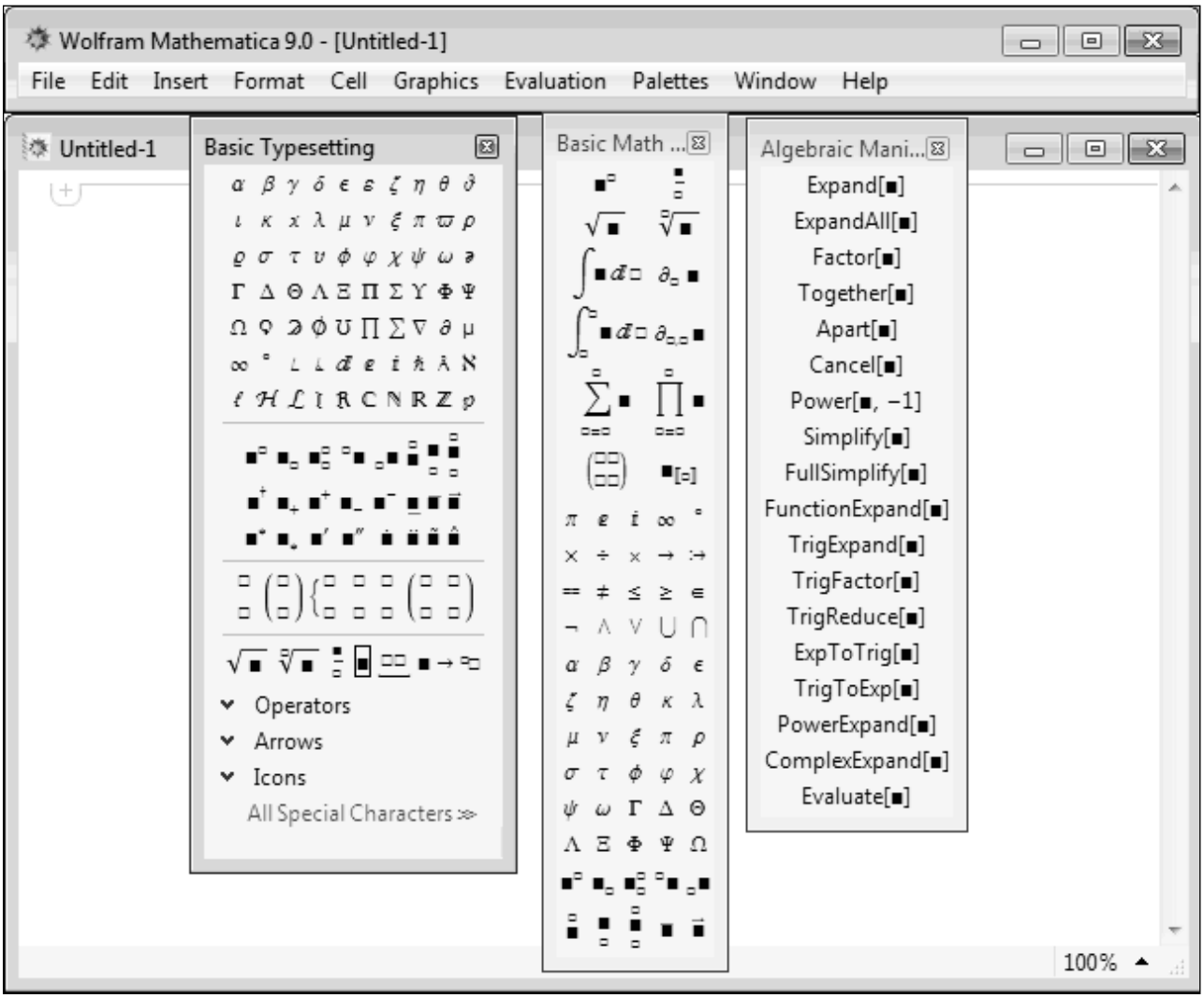



Рис. 2.2

**9. Window** – управление внешним видом окон:

- ***Stack Windows*** – каскадное расположение окон;
- ***Tile Windows Wide*** – расположить на экране окна открытых документов одно над другим вытянутыми по ширине;

– ***Tile Windows Tall*** – расположить на экране окна открытых документов одно рядом с другим вытянутыми по длине;

– ***Messages*** – вывод окна сообщений об ошибках.

**10. Help** – управление справочной системой.

В любом из пунктов главного меню некоторые команды могут быть выделены светло-серым шрифтом. Это означает, что команды не могут быть выполнены в данный момент. Например, если выражение не выделено, то его значение вычислить нельзя.

### *Окно ввода (экран редактирования)*

На экране появляется активное окно документа. По умолчанию создаваемый документ носит название **Untitled-1**. При сохранении можно присвоить ему нужное имя. Система автоматически присваивает файлам расширение **.nb**.

В программе **Mathematica** все введенные в окно ввода данные содержатся в отдельных, определенным образом выделенных областях экрана, называемых *ячейками*. Введенные данные автоматически объединяются во входную ячейку, которая обозначается квадратной скобкой – **]** в правой части окна ввода. Например, наберем **3 + 7** (рис. 2.3).

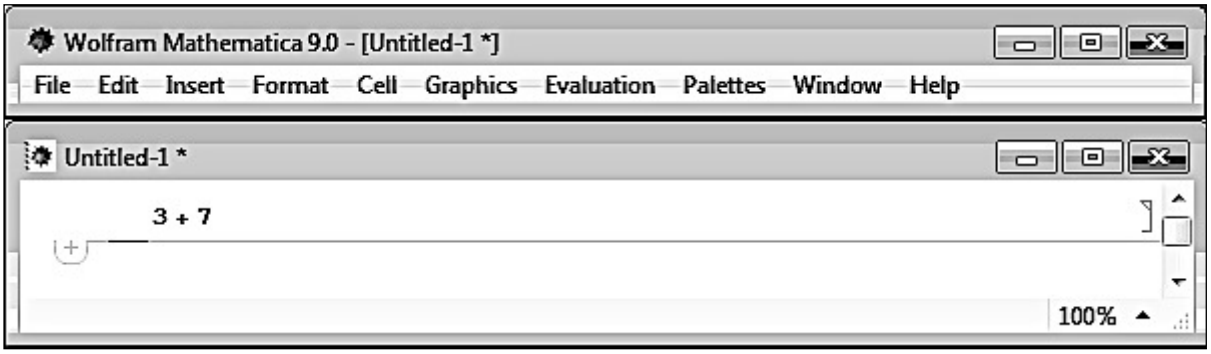



Рис. 2.3

**Для получения результата** нужно поместить курсор в любой части ячейки и нажать **Shift + Enter** (удерживая **Shift**, нажать **Enter**). Нажатие одной клавиши **Enter** приводит к созданию новой строки в той же ячейке.

Если введенные данные являются логически завершенными и не содержат синтаксических ошибок, программа **Mathematica** обрабатывает их и выдает результат. В противном случае указывается тип ошибки.

Результат **3 + 7** на рис. 2.4.

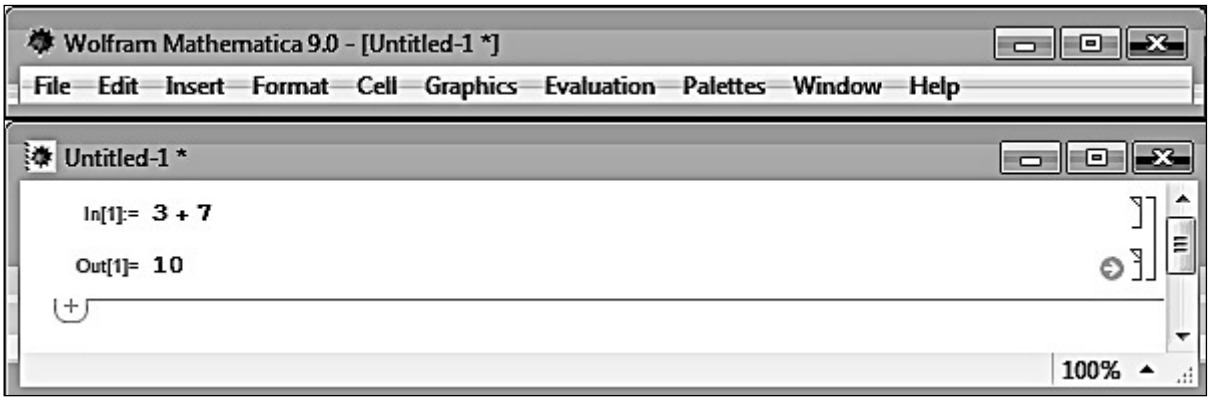



Рис. 2.4

На рис. 2.4 **Mathematica** добавляет к данным на экране метки:

1) **In**[*n*]**:=** – вводимые пользователем данные;

2) **Out**[*n*]**:=** – результат, выводимый программой **Mathematica**, где **n = 1, 2, 3, ...** – номер проводимого вычисления. Номер **n** может быть использован ***для ссылки на результат выполнения n-й операции*** с помощью записи **%n**.

Входную и выходную ячейки окаймляют квадратные скобки, а вместе они ограничены общей квадратной скобкой – это значит, что сформирована группа ячеек.

Для окончания работы с пакетом **Mathematica** необходимо выбрать команду ***Exit*** в разделе **File** главного меню. Если требуется сохранить введенные данные, то появляется дополнительное окно, в котором можно определить имя сохраняемого документа. Этот файл можно открыть при следующем сеансе работы с программой **Mathematica**, выбрав в разделе **File** команду ***Open***.

### Некоторые встроенные функции системы Mathematica

Названия всех встроенных функций системы **Mathematica** записываются с большой буквы. Если название функции состоит нескольких «склеенных» слов, то, как правило, каждое из них записывается с большой буквы. Аргументы всех функций, как встроенных, так и задаваемых пользователем, всегда заключаются в **квадратные скобки**. Уравнения в системе **Mathematica** формируются двойным знаком равно, т. е. «==».

***Abs***[*x*] – абсолютная величина (модуль) *x*.

***And***[*a,b,c,…*] – логическое умножение или логическое «И». Дает значение *True*, если логические значения *a, b, c, …* равны, в противном случае дает значение *False*.

***Append***[*lst*,*x*] – создает новый список, добавляя элемент *x* в конец списка *lst*.

***Array*** [*a*,*n*,*k*] – символьный список {*a*[*k*], *a*[*k* + 1], …, *a*[*k* + *n* – 1]}, состоящий из *n* элементов. Если $n = 0$, то функция дает пустой список {}. Аргумент *k* (начальное значение индекса) может быть нулевым или отрицательным. *Array*[*a*,*n*] эквивалентно *Array*[*a*,*n*,1], т. е. дает список {*a*[1], *a*[2], …, *a*[*n*]}.

***Array***[*a*,{*n*,*m*}] – дает сложный список {{*a*[1,1], *a*[1,2], …, *a*[1,*m*]}, {*a*[2,1], *a*[2,2], …, *a*[2,*m*]}, … ,{*a*[*n*,1], …, *a*[*n*,*m*]}}, состоящий из *n* списков длиной *m*.

***Clear***[$s_1$,$s_2$,…] – стирает любые значения, присвоенные указанным символам $s_1$, $s_2$, …

***Collect***[*expr*,*x*] – группирует члены выражения *expr* с одной и той же степенью переменной *x*.

***ColumnForm*** [*lst*,$w_1$,$w_2$] – выводит список *lst* на экран в виде колонки. Аргументы $w_1$ и $w_2$ необязательны. Первый ($w_1$) задает способ выравнивания элементов списка в колонке по горизонтали: *Center* – по центру, *Left* – по левому краю, *Right* – по правому краю. Второй ($w_2$) определяет выравнивание в ячейке вывода по вертикали: *Center* – по центру, *Below* – по верхнему краю, *Above* – по нижнему краю.

***ColumnForm***[*lst*] эквивалентно ***ColumnForm***[*lst*,*Left*,*Below*].

***Exp***[*x*] – $e^x$, где $e$ – основание натурального логарифма.

***expr*/.x→*value*** – выполняет подстановку значения *value* вместо *x* в выражение *expr* , которое зависит от *x*.

***Floor***[*x*] – наибольшее целое число, не превосходящее *x*.

***For***[*start*,*test*,*incr*,*body*] – цикл с параметром. Выражения *start* и *body* могут состоять из нескольких операторов, которые разделяются знаком «;» (точка с запятой). Сначала один раз вычисляется выражение *start* (начальные действия). Если справедливо условие *test*, зависящее от параметра цикла, то вычисляются выражения *body* (тело цикла) и *incr* (приращение параметра), и все повторяется. Тело цикла может ни разу не выполниться. В цикле *For* нельзя задать локальные переменные.

***FractionalPart***[*x*] – дробная часть числа *x*.

***If***[*test*, *expr*1, *expr*2] – определяет выбор и вычисление выражения. Сначала вычисляется логическое выражение *test*. Если получено значение *True*, то вычисляется выражение *expr*1, в противном случае вычисляется выражение *expr*2.

***IntegerPart***[*x*] – целая часть числа *x*.

***Graphics***[{*drvs*,*prvs*},*optns*] – формирует двумерные графические объекты-примитивы *prvs* (круги, диски, линии, прямоугольники, многоугольники, текст), которые с помощью функции *Show* можно показать на одном графике. Директивы *drvs* определяют размеры, цвет, стиль оформления рисуемых линий и текста. Они ставятся перед примитивами и могут действовать на несколько из них.

Для функций *Graphics* и *Plot* можно задавать одни и те же опции *optns*.

***GraphicsArray***[*lst*] – позволяет разместить в одном графическом окне на экране сразу несколько графиков, возможно разных системах координат. Графики можно расположить в одну строку или в виде таблицы. Здесь *lst* – линейный или двухуровневый список, который группирует графические объекты.

***Length*** [*lst*] – длина (количество элементов) списка *lst*.

***List***[*a*,*b*,*c*,…] – формирует список {*a*,*b*,*c*,…}. Элементы *a*,*b*,*c*,… в свою очередь могут быть списками, а также любыми другими выражениями. *List*[ ] дает пустой список, т. е. {}.

***ListPlot*** [*lst*,*optns*] – предназначена для построения графика по точкам. В общем случае аргумент *lst* представлен списком {{$x_1$,$y_1$},{$x_2$,$y_2$},…}, где ($x_1$,$y_1$),($x_2$,$y_2$),… – координаты отмечаемых точек. Список {{1,$y_1$}, {2,$y_2$},…} можно задать как {$y_1$,$y_2$,…}. Список также можно создать какой-либо функцией, например, *Range* или *Table*.

Для функции *ListPlot* используются в основном те же опции *optns*, что и для функции *Plot*. Если требуется соединять точки на графике отрезками прямых, то нужно установить дополнительную опцию *PlotJoined→True*. Для управления размером точек используется директива *PointSize*, а не *Thickness*, как для линий.

***Log***[*a*, *x*] – логарифм *x* по основанию *a*. *Log*[*x*] – натуральный логарифм *x*.

***MatrixForm***[*lst*] – выводит на экран двухуровневый список *lst* в виде матрицы, имеющей ячейки одного и того же размера. Линейный список выводится в виде столбца.

***Max*** [*a*, *b*, *c*,…] – максимальное значение из *a*, *b*, *c*, … Любой из аргументов может быть списком.

***Min*** [*a*,*b*,*c*,…] – минимальное значение из *a*, *b*, *c*, … Любой из аргументов может быть списком.

***Mod*** [*m*,*n*] – остаток от деления целого *m* на целое *n*.

***N***[*expr*,*n*] – вещественное значение выражения *expr*, *n* – разрядность, используемая при вычислении результата. Если второй аргумент не задан, то разрядность выбирается автоматически, а результат выводится с шестью значащими цифрами.

***NestList*** [*f*, *x*, *n*] – формирует список результатов *n*-кратного применения функции *f* к аргументу *x*, т. е. список {*x*, *f*(*x*), *f*(*f*(*x*)), …}.

***PaddedForm***[*expr*,{*m*,*n*}] – задает размер *m* (количество цифр) десятичного представления вещественного значения выражения *expr* при выводе его на экран. Здесь *n* – количество цифр после десятичной точки.

***Plot***[{$f_1$,$f_2$,…},{*x*,*x*min,*x*max},*optns*] – предназначена для построения графиков функций $y = f_1(x)$, $y = f_2(x)$, … при изменении независимой переменной *x* в пределах от *x*min до *x*max. При этом используется прямоугольная (декартова) система координат. Необязательные аргументы *optns* (опции), общие и для других графических функций, служат для настройки вида графиков. Если опции не указаны, то их стандартные значения устанавливаются автоматически.

***Prepend***[*lst*,*x*] – создает новый список, добавляя *x* в начало списка *lst*.

***Print***[$expr_1$,$expr_2$,…] – выводит на экран значения указанных выражений $expr_1$,$expr_2$,…. Следующий вывод переносится в начало новой строки.

***Random***[] – при последовательных вызовах дает случайные вещественные числа, равномерно распределенные на интервале от 0 до 1.

***Range***[*m*,*n*,*d*] – формирует числовой список, значения которого равномерно распределены в интервале от *m* до *n* с шагом *d*. Значения *m*, *n*, *d* – необязательно целые.

***Rationalize***[*x*] и ***Rationalize***[*x*, точность числа *x*] – представляет *x* в виде рационального числа $\frac{m}{n}$ (с заданной точностью).

***Return***[*expr*] – вызывает завершение выполнения подпрограммы-функции. Выражение *expr* определяет значение, возвращаемое как результат. Часто рассматривается как дополнительная, возможно, для исключительных ситуаций, точка завершения подпрограмм.

***Show***[*plot,opns*] – выводит на экран уже сформированный (например, функцией *plot*) график. С помощью второго параметра можно изменить значения опций в той графической функции, при помощи которой был получен рисунок.

***Show***[{$plot_1$, $plot_2$,…},*opns*] – совмещает в одном графическом окне несколько графиков. Функция полезна в тех случаях, когда желательно, не вычисляя заново исходные графики $plot_1$, $plot_2$,… просмотреть их при иных настройках опций *opns* или опоставить.

***Simplify***[*expr*,*optns*] – приводит выражение *expr* к простейшей форме. При этом тригонометрические тождества не используются, если указана опция *Trig→False*.

***Table***[*expr*,*n*] – список из *n* значений одного и того же выражения *expr*.

***Table***[*expr*,{*i*,*m*,*n*,*d*}] – список значений выражения *expr*, зависящего от параметра *i*, для *i* от *m* до *n* с шагом *d*.

***Table***[*expr*,{*i*,*m*,*n*}] эквивалентно ***Table***[*expr*,{*i*,*m*,*n*,1}].

***Table***[*expr*,{*i*,*n*}] эквивалентно ***Table***[*expr*,{*i*,1,*n*,1}].

***Table***[*expr*,{*i*,$m_1$,$n_1$},{*j*,$m_2$,$n_2$},…] – порождает многоуровневые списки, используется для создания числовых таблиц.

***TableForm***[*lst*,*opns*] – выводит на экран двухуровневый список *lst* в виде таблицы, высота строк и ширина столбцов которой определяются максимальными размерами элементов списка. Линейный список представляется строкой или колонкой в зависимости от значения (*Row* или *Column*) опции *TableDirections*. Если установить опцию *TableHeadings*, то можно вывести названия для строк и столбцов.

***Timing***[*expr*] – значение выражения *expr* и время, затраченное на его вычисление.

***While***[*test*,*body*] – цикл с условием. Выражение *body* (тело цикла) записывается в виде последовательности операторов, разделенных знаком «;» (точка с запятой). Сначала вычисляется логическое выражение *test*. Если получено значение *True*, то вычисляется тело цикла, и все повторяется. Если получено значение *False*, то цикл завершается. Цикл может оказаться бесконечным, если условие *test* никогда не выполняется. Тело цикла может ни разу не вычисляться, т. к. все начинается с проверки условия. В цикле *While* нельзя задать локальные переменные.

### *Основные опции графических функций*

В функциях, формирующих графики, в числе других аргументов могут указываться опции, которые записываются в виде подстановок, например, *AspectRatio→Automatic*. Они могут иметь числовые или символьные значения, а также значения в виде списков. При желании опции могут быть изменены при вызове функции *Show*. Наиболее распространенные символьные значения опций: *Automatic* – используется автоматический выбор стандартного значения, *None, False* – опция не используется, *All* – используется в любом случае.

### *Директивы, используемые в графических функциях*

Директивы передают значения многим графическим опциям и являются важным средством настройки графиков.

***AbsoluteDashing***[{$d_1$,$d_2$,…}] – устанавливает стиль рисования линии графика пунктиром, т. е. в виде последовательности отрезков длины $d_1$,$d_2$,…, которые повторяются циклически. Здесь $d_1$ – длина рисуемой линии, $d_2$ – пропуск и т. д. Значения задаются в пунктах ($\text{pt} = 1/72\ \text{inch} \approx 0{,}035\ \text{см}$).

***AbsolutePointSize***[$d$] – задает диаметр $d$ (в пунктах (pt)) отдельных точек графиков, построенных по точкам.

***AbsoluteThickness***[$d$] – задает толщину $d$ линии графика в пунктах (pt).

***GrayLevel***[$d$] – задает уровень интенсивности оттенка серого цвета для линий графиков, $0 \le d \le 1$ (0 – черный цвет, 1 – белый).

***RGBColor***[$d_1, d_2, d_3$] – задает цвет линии графика в цветовой модели *RGB* (красный, зеленый, голубой). Здесь $0 \le d_1 \le 1$, $0 \le d_2 \le 1$, $0 \le d_3 \le 1$ – интенсивности базовых цветов (1, 0, 0 – красный цвет; 0, 1, 0 – зеленый; 0, 0, 1 – голубой). Директива используется при выводе графиков на экран дисплея.

***Epilog→Point***[*tab*] – изображает на графике точки, координаты которых указаны в списке *tab*.

## Простейшие операции со списками в системе Mathematica

В **Mathematica** ***списки*** (***lists***) являются основным способом объединения объектов. Элементы списков всегда заключаются в ***фигурные скобки***. Например, {1, 2, 3} – список чисел. Списки не выполняют никаких действий над объектами, это лишь способ хранить их как единое целое. Списки $\{a, b, c\}$ и $\{b, c, a\}$ являются различными из-за порядка элементов.

Для выделения элементов списка *list* используются ***двойные квадратные скобки***: *list*[[*i*]] – возвращает *i*-й элемент списка, *list*[[*i*,*j*]] – возвращает элемент двумерного списка (т. е. списка списков), находящийся на позиции (*i*, *j*).

***Count***[*list*, *pattern*] – количество элементов списка *list*, соответствующих образцу *pattern*.

***Length***[*list*] – число элементов одномерного списка *list* или число размерностей многомерного списка.

***Drop***[*list*, *n*] – удаляет первые *n* элементов списка *list*.

***Drop***[*list*, *-n*] – удаляет последние *n* элементов списка *list*.

***Drop***[*list*, {*n*}] – удаляет *n*-й элемент списка *list*.

***Drop***[*list*, {*n*,*m*}] – удаляет элементы списка *list* с *n*-го по *m*-й.

***Delete***[*list*, *n*] – удаляет *n*-й элемент списка *list*.

***First***[*list*] – первый элемент списка *list*.

***Last***[*list*] – последний элемент списка *list*.

***Take***[*list*, *n*] – возвращает первые *n* элементов списка *list*.

***Take***[*list*, *-n*] – возвращает последние *n* элементов списка *list*.

***Take***[*list*, {*n*,*m*}] – возвращает элементы списка *list* с порядковыми номерами от *n* до *m*.

***Rest***[*list*] – возвращает список *list* без первого элемента.

***Append***[*list*, *element*] – добавляет элемент *element* в конец списка *list*.

***Prepend***[*list*, *element*] – добавляет элемент *element* в начало списка *list*.

***Insert***[*list*, *element*, *n*] – вставляет элемент *element* в позицию *n* списка *list*.

***Flatten***[*list*] – превращает список *list* в одномерный (выравнивает список).

***Sort***[*list*] – сортирует элементы списка *list* в каноническом порядке.

***Reverse***[*list*] – изменяет порядок элементов списка *list* на обратный.

***Join***[*list*1, *list*2,…] – объединяет списки *list*1, *list*2, … в единую цепочку.

***Intersection***[*list*1, *list*2,…] – возвращает упорядоченный список элементов, общих для всех списков *list*1, *list*2, … (т. е. находит пересечение множеств).

***Union***[*list*] – сортирует список *list*, удаляя из него все повторяющиеся элементы.

***Union*** [*list*1, *list*2, …] – возвращает отсортированный список из всех элементов списков *list*1, *list*2, …, в котором удалены все повторяющиеся элементы (т. е. находит объединение множеств).

## Примеры работы со встроенными функциями системы Mathematica, построение таблицы значений и вывод графика функции

**Пример 2.1.** Найти:

а) значение выражения $(32+48)/11$, выделить его целую и дробную части;

б) значение выражения $\ln(6^5-3^8)$ и рациональное приближение к нему с точностью $10^{-4}$.

Δ а) введем выражение, используя палитру инструментов (**Palettes**–>**Basic Math Assistant**), и найдем его значение (**Shift + Enter**).

```
In[1]:= (32 + 48)/11

Out[1]= 80/11
```

Как видно, результат представлен в виде дроби (рационального числа). Получим численное приближение к значению в виде десятичной дроби с количеством цифр, принятым по умолчанию, а также равным 9.

```
In[2]:= N[(32 + 48)/11]
        численное прибли

        N[(32 + 48)/11, 9]
        численное прибли

Out[2]= 7.27273

Out[3]= 7.27272727
```

Выделим целую и дробную части числа, сославшись на него как на результат выполнения первой операции.

```
In[4]:= IntegerPart[%1]
        целая часть

Out[4]= 7

In[5]:= FractionalPart[%1]
        дробная часть

Out[5]= 3/11

In[6]:= FractionalPart[%1] // N
        дробная часть          численное приближение

Out[6]= 0.272727
```

Найдем рациональное приближение к числу, являющемуся результатом третьей операции.

```
In[7]:= Rationalize[%3]
        найти рациональное приближение

Out[7]= 80/11
```

б) найдем числовое значение выражения $\ln\left(6^5-3^8\right)$ и рациональное приближение к нему с заданной точностью.

```
In[8]:= Log[6^5 - 3^8] // N
        натуральный ло··· численное приближение

Out[8]= 7.1025

In[9]:= Rationalize[Log[6^5 - 3^8], 10^-4]
        найти рацио··· натуральный логарифм

Out[9]= 277/39
```

**Пример 2.2.** Задать функцию $f(x)=(x+1)(x^2-4x+4)$. Раскрыть скобки в определяющем ее выражении, а затем разложить полученный многочлен на множители. Найти значения функции в точках $1{,}8;\ -e;\ \sqrt[3]{14}$. Найти $f(2a-b)$ и упростить полученное выражение. Составить таблицу из координат точек графика функции на отрезке $[-3,4]$ с шагом 1 и 0,5 по $x$. Построить график функции на отрезке $[-3,4]$.

Δ Зададим функцию $f(x)$. Знак «_» после переменной $x$ применяется для того, чтобы задать $x$ как локальную переменную внутри функции.

```
In[1]:= f[x_] = (x + 1) (x^2 - 4 x + 4)

Out[1]= (1 + x) (4 - 4 x + x^2)
```

Раскроем скобки, а затем разложим на множители данную функцию.

```
In[2]:= Expand[f[x]]
        раскрыть скобки

Out[2]= 4 - 3 x^2 + x^3

In[3]:= Factor[f[x]]
        факторизовать

Out[3]= (-2 + x)^2 (1 + x)
```

Вычислим значения функции в заданных точках. Обратите внимание, как **Mathematica** выполняет подстановку иррациональных значений.

```
In[4]:= f[1.8]

Out[4]= 0.112

In[5]:= f[-E]
        основание натурального логарифма

Out[5]= (1 - e) (4 + 4 e + e^2)

In[6]:= f[-E] // N
        осн··· численное приближение

Out[6]= -38.2527

In[7]:= f[∛14]

Out[7]= (1 + 14^(1/3)) (4 - 4 14^(1/3) + 14^(2/3))

In[8]:= f[∛14] // N
                численное приближение

Out[8]= 0.573643
```

Очистим переменные $a$ и $b$, найдем $f(2a-b)$, упростим и раскроем скобки в полученном выражении.

```
In[9]:= Clear[a, b]; f[2 a - b]
        очистить

Out[9]= (4 - 4 (2 a - b) + (2 a - b)^2) (1 + 2 a - b)

In[10]:= Simplify[f[2 a - b]]
         упростить

Out[10]= (1 + 2 a - b) (2 - 2 a + b)^2

In[11]:= Expand[f[2 a - b]]
         раскрыть скобки

Out[11]= 4 - 12 a^2 + 8 a^3 + 12 a b - 12 a^2 b - 3 b^2 + 6 a b^2 - b^3
```

Составим с помощью функции **Table** таблицу, содержащую координаты точек графика функции $f(x)$ на заданном отрезке с шагом 1 (используется по умолчанию). Выведем результат в табличной форме.

```
In[12]:= t1 = Table[{x, f[x]}, {x, -3, 4}]
          таблица значений

Out[12]= {{-3, -50}, {-2, -16}, {-1, 0}, {0, 4}, {1, 2}, {2, 0}, {3, 4}, {4, 20}}

In[13]:= t1 // TableForm
          табличная форма

Out[13]//TableForm=
-3   -50
-2   -16
-1   0
0    4
1    2
2    0
3    4
4    20
```

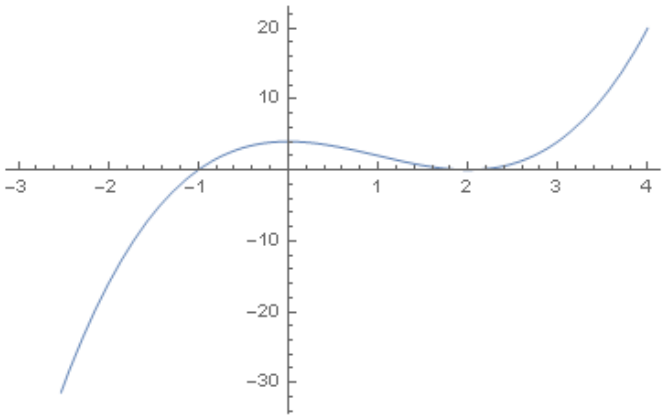
Составим таблицу значений функции $f(x)$ с шагом $0{,}5$ по $x$.

```
In[14]:= t2 = Table[{x, f[x]}, {x, -3, 4, 0.5}]
          таблица значений

Out[14]= {{-3., -50.}, {-2.5, -30.375}, {-2., -16.},
          {-1.5, -6.125}, {-1., 0.}, {-0.5, 3.125}, {0., 4.},
          {0.5, 3.375}, {1., 2.}, {1.5, 0.625}, {2., 0.},
          {2.5, 0.875}, {3., 4.}, {3.5, 10.125}, {4., 20.}}
```

Постоим график функции $f(x)$ на заданном отрезке с помощью функции **Plot**.

```
In[15]:= gr1 = Plot[f[x], {x, -3, 4}]
          график функции
```

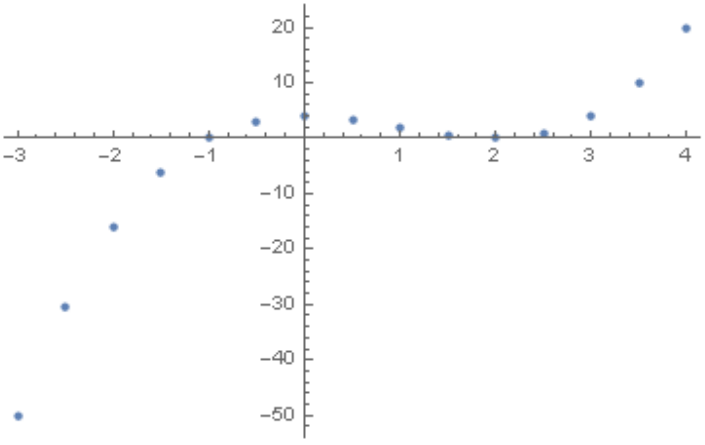
Out[15]=




Применим функцию **ListPlot** для построения точек графика функции $f(x)$, соответствующие табличным значениям.

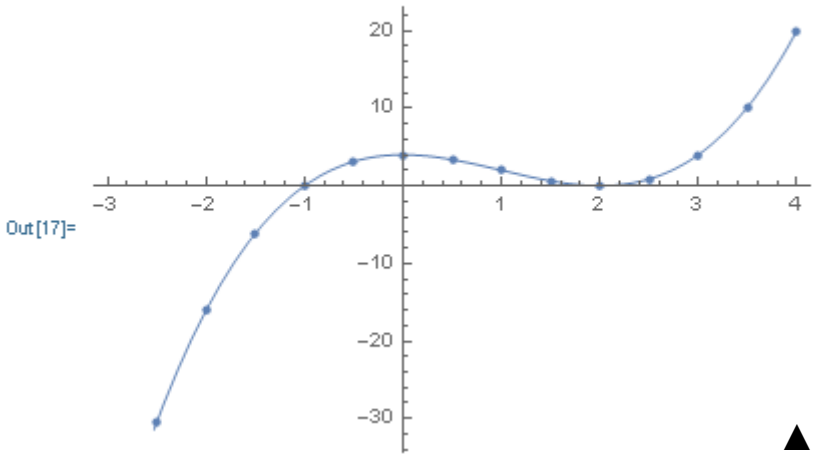
```
In[16]:= gr2 = ListPlot[t2]
          диаграмма разбр
```

Out[16]=




Объединим оба рисунка в одном графическом окне с помощью функции **Show**. Заметим, что такое же изображение можно получить с помощью одной функции **Plot**, в которой указана дополнительная опция **Epilog→Point**[t2].

```
In[17]:= Show[gr1, gr2]
         показать
```




▲

**Пример 2.3.** Задать функцию двух переменных $g(x, y) = 2x^2 + 3xy - 5y^2 + 7$. Найти значения функции в точках $(-2{,}7; -1{,}4)$, $(2/3; \ln 5)$, $(\sqrt{17}, \sin 2)$. Составить таблицу из координат точек графика функции $g(x, y)$ в прямоугольнике $[-3, 3]\times[-4, 4]$ с шагом 1 по $x$ и шагом 2 по $y$. Построить график функции $g(x, y)$ в прямоугольнике $[-10, 10]\times[-15, 15]$. Построить линии уровня (контурный график) функции $g(x, y)$ в прямоугольнике $[-5, 5]\times[-5, 5]$. Построить кривую $g(x, y) = 0$ в прямоугольнике $[-8, 8]\times[-8, 8]$.

Δ Очистим все переменные и зададим функцию $g(x, y)$.

```
In[1]:= ClearAll;
        очистить всё

In[2]:= g[x_, y_] = 2 x^2 + 3 x y - 5 y^2 + 7
Out[2]= 7 + 2 x^2 + 3 x y - 5 y^2
```

Обратите внимание, что произведение переменных $x$ и $y$ должно записываться с помощью пробела между ними либо знака умножения в виде точки. В противном случае запись «$xy$» **Mathematica** воспринимает как одну новую переменную.

Вычислим значения функции $g(x, y)$ в заданных точках.

```
In[3]:= g[-2.7, -1.4]
Out[3]= 23.12
```

```
In[4]:= g[2/3, Log[5]]
        натуральн
Out[4]= 71/9 + 2 Log[5] - 5 Log[5]^2

In[5]:= g[2/3, Log[5]] // N
        натуральн··· численное приближение
Out[5]= -1.84369

In[6]:= g[Sqrt[17], Sin[2]]
        синус
Out[6]= 41 + 3 Sqrt[17] Sin[2] - 5 Sin[2]^2

In[7]:= g[Sqrt[17], Sin[2]] // N
        синус  численное приближение
Out[7]= 48.1133
```

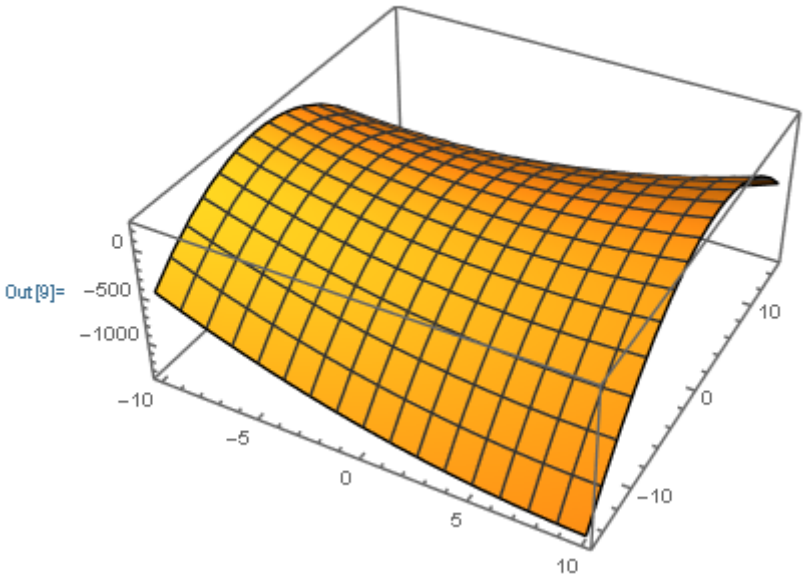
Составим таблицу значений функции в заданном прямоугольнике.

```
In[8]:= tab = Table[{x, y, g[x, y]}, {x, -3, 3}, {y, -4, 4, 2}]
          таблица значений
Out[8]= {{{-3, -4, -19}, {-3, -2, 23}, {-3, 0, 25}, {-3, 2, -13}, {-3, 4, -91}},
         {{-2, -4, -41}, {-2, -2, 7}, {-2, 0, 15}, {-2, 2, -17}, {-2, 4, -89}},
         {{-1, -4, -59}, {-1, -2, -5}, {-1, 0, 9}, {-1, 2, -17}, {-1, 4, -83}},
         {{0, -4, -73}, {0, -2, -13}, {0, 0, 7}, {0, 2, -13}, {0, 4, -73}},
         {{1, -4, -83}, {1, -2, -17}, {1, 0, 9}, {1, 2, -5}, {1, 4, -59}},
         {{2, -4, -89}, {2, -2, -17}, {2, 0, 15}, {2, 2, 7}, {2, 4, -41}},
         {{3, -4, -91}, {3, -2, -13}, {3, 0, 25}, {3, 2, 23}, {3, 4, -19}}}
```

Построим график функции $g(x, y)$ на заданном множестве (графиком функции двух переменных является поверхность в пространстве).

```
In[9]:= Plot3D[g[x, y], {x, -10, 10}, {y, -15, 15}]
        график функции 2-х переменных
```

Out[9]=

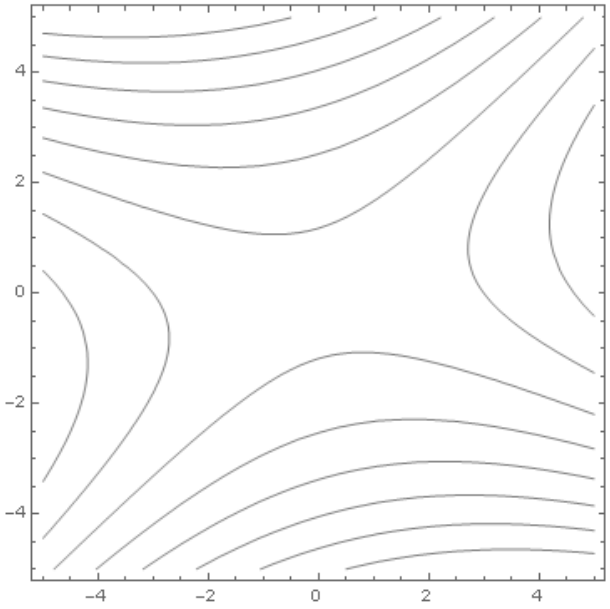



Графиком данной функции является гиперболический параболоид («седло»).

Построим линии уровня функции $g(x, y)$, т. е. множество кривых на плоскости $OXY$, определяемых уравнениями $g(x, y) = \text{const}$ (для всевозможных значений постоянной const). Или, другими словами, сечения графика функции плоскостями, параллельными плоскости $OXY$. Применим функцию **ContourPlot**.

```
In[10]:= ContourPlot[g[x, y], {x, -5, 5}, {y, -5, 5}, ContourShading → False]
         контурный график                  оцвечивание конт···  ложь
```

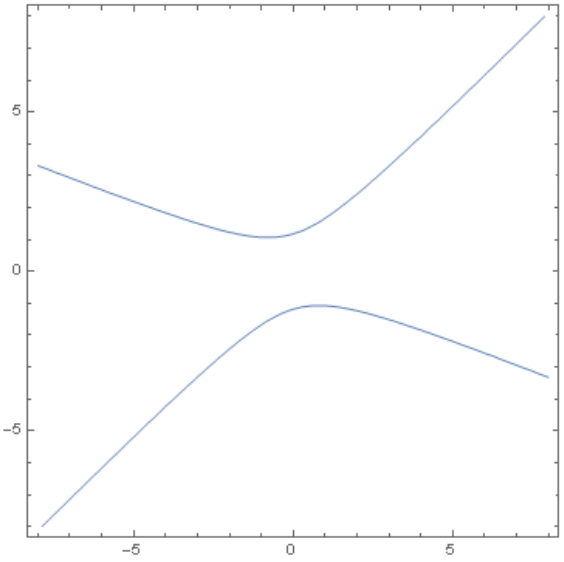
Out[10]=




Кривую $g(x, y) = 0$ будем рассматривать как линию нулевого уровня функции $g(x, y)$ и построим ее с помощью функции **ContourPlot**.

```
In[11]:= ContourPlot[g[x, y] == 0, {x, -8, 8}, {y, -8, 8}]
         контурный график
```

Out[11]=




Полученная кривая является гиперболой. ▲

# Лабораторная работа 1

## Основы работы с пакетом Wolfram Mathematica

**1.** Составьте и найдите значения арифметических выражений, содержащих операции сложения, вычитания, умножения, деления и возведения в степень с целыми, рациональными и действительными числами. Например, $\frac{35.4+48.6}{28}$, $17.2\cdot 4.35$, $4.3^3$, $\sqrt[5]{326}$ и т. д. *Для получения результата вычислений следует нажимать* **Shift+Enter**.

**2.** Покажите диапазон положительных чисел системы **Mathematica** – машинный нуль (**$MinMachineNumber**) и машинную бесконечность (**$MaxMachineNumber**).

**3.** Рассмотрите действие функций:

а) выделения целой и дробной частей числа **IntegerPart**[*x*] и **FractionalPart**[*x*]. Например, $x=123^2 \,/\, 768$ или $x=\pi\cdot e^{-2}$ (числа $\pi$ и $e$ обозначаются **Pi** и **E**);

б) представления числа в виде рациональной дроби **Rationalize**[*x*] и **Rationalize**[*x*, точность числа *x*]. Например, для числа $\pi$ с точностью $10^{-3}$, $10^{-8}$, $10^{-12}$;

в) **N**[*x*] **(**или *x*//*N*) для представления результата вычислений в виде десятичной дроби и **N**[*x*, число цифр результата] для вывода результата с заданной точностью. Например, вывести семь знаков числа $\pi$; вывести 28 знаков числа $3\pi^2$ и т. д.;

г) присваивания (например, **«*x*=5»**) и очистки переменных **Clear**[*x*] или **«x=.»**, **Clear**[*a*,*b*,*c*,*d*];

д) выполнения подстановок в выражения: ***expr*/.x→*value*** (здесь ***expr*** – выражение, зависящее от *x*, ***value*** – значение, которое подставляется вместо *x*). Например, подставить в выражение $x^5-2x+\cos x$ вместо $x$ числа 3, $\pi$, выражение $y+z$ (встроенная функция $\cos x$ имеет вид **Cos**[*x*]).

**4.** Задайте функции одной и нескольких переменных $f_1(x)=3x^4-2^x+5\cos x$, $f_2(x)=x^6-x^5-18x^4+14x^3+61x^2-93x-36$, $f_3(x)=3x^4+5x^3+7x^2+15x-6$, $g(x,y)=4xy-y^3+\ln(x+3y)+7$.

Выполните следующие действия:

а) найдите значения функций $f_1(x)$ и $f_2(x)$ в точках $-1$, 2, 5 и т. д.;

б) найдите значения функции $g(x, y)$ в точках $(1, 0)$, $(-2, 1)$, $(3, e)$ и т. д.;

в) примените к функциям $f_2(x)$ и $f_3(x)$ функцию **Factor**[*expr*] (*она пытается разложить многочлен на множители над полем целых чисел*), а затем функцию **Expand**[*expr*] к результату этой операции.

**5.** Создайте таблицы значений функции $f_1(x)$ на отрезке $[-4, 4]$ с шагом 0,5 с помощью функций **N**[Table[*f*1[*x*],{*x*,*x*min,*x*max,step}]] и **Table**[{*x*,*f1*[*x*]},{*x*, *x*min,*x*max,step}]//**N**. Выведите полученные списки в виде таблиц с помощью функции **TableForm**. Примените функцию **PaddedForm**, чтобы выводимые числа содержали не более восьми цифр, две из которых находятся в дробной части.

**6.** Создайте таблицу значений функции $g(x, y)$ в прямоугольнике $[-1, 4]\times[1, 3]$ с шагом 0,5 по $x$ и шагом 0,4 по $y$ с помощью функции **Table**[{*x*,*y*,*g*[*x*,*y*]},{*x*,*x*min,*x*max,*step*},{*y*,*y*min,*y*max,*step*}]//**N**.

**7.** Постройте:

а) графики функций одной переменной $f_1(x)$ и $f_2(x)$ с помощью функции **Plot**[*f*[*x*],{*x*,*x*min,*x*max}];

б) точки графика функции $f_1(x)$ из таблицы, полученной в п. 5;

в) графики функций $f_1(x)$ и $f_2(x)$ в одной системе координат с помощью функции **Plot**[{*f*1[*x*], *f*2[*x*]},{*x*, *x*min, *x*max}], а также функции **Show**;

г) график функции двух переменных $g(x, y)$ с помощью функции **Plot3D**[*g*[*x*,*y*],{*x*,*x*min,*x*max},{*y*,*y*min,*y*max}];

д) контурный график (линии уровня) функции двух переменных $g(x, y)$ с помощью функции **ContourPlot**[*g*[*x*,*y*]},{*x*,*x*min,*x*max},{*y*,*y*min,*y*max}, ContourShading–>False];

е) кривую $g(x, y) = 0$ с помощью функции **ContourPlot**.

# 3. Численное решение нелинейных уравнений

Приближенное или численное нахождение изолированных действительных корней нелинейного уравнения $f(x)=0$ состоит из двух этапов:

1) ***отделение корней***, т. е. установление всех возможно тесных промежутков, в каждом из которых содержится один, и только один, корень уравнения;

2) ***уточнение корней***, т. е. доведение их до заданной степени точности.

Отметим два способа отделения действительных корней уравнения – *аналитический* и *графический*.

Для ***аналитического отделения корней*** используют следующие теоремы математического анализа.

1. Теорема Больцано – Коши. Если функция $f(x)$ непрерывна на отрезке $[a,b]$ и на его концах принимает значения разных знаков, т. е.

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

то внутри отрезка $[a,b]$ существует по крайней мере один корень уравнения $f(x)=0$.

2. Если функция $f(x)$ непрерывна на отрезке $[a,b]$, $f(a)\cdot f(b)<0$, производная $f'(x)$ существует и сохраняет постоянный знак в интервале $(a,b)$, то внутри отрезка $[a,b]$ существует ровно один корень уравнения $f(x)=0$.

Таким образом, чтобы отделить корни аналитически, нужно:

– найти критические точки $x_1, x_2, ..., x_k, \ldots$ функции $f(x)$, т. е. точки из области определения функции, в которых производная $f'(x)$ равна нулю, бесконечности или в которых она не существует;

– вычислить значения функции $f(x)$ в этих точках и ее предельные значения на концах области определения;

– по знакам функции $f(x)$ и ее производной $f'(x)$ определить интервалы, в которых уравнение имеет ровно один корень;

– сузить полученные интервалы до нужной длины, так чтобы на его концах функция принимала значения разных знаков.

Критические точки также могут оказаться корнями четной кратности функции $f(x)$.

Решив задачу *отделения корней*, фактически находят приближенные значения корней с погрешностями, не превосходящими длины отрезка, содержащего тот или иной корень.

***Графическое отделение корней*** состоит в построении графика функции $y = f(x)$ и нахождении тех значений $x$, при которых график пересекает ось абсцисс. Они и являются корнями уравнения $f(x) = 0$.

Если график функции $y = f(x)$ построить трудно, то от исходного уравнения $f(x) = 0$ следует перейти к равносильному уравнению вида

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

таким образом, чтобы графики функций $y = \varphi_1(x)$ и $y = \varphi_2(x)$ были достаточно просты. Абсциссы точек пересечения этих графиков и будут корнями уравнения.

После выполнения этапа отделения корней переходят к следующему этапу приближенного решения уравнений – ***уточнению корней***.

Пусть искомый корень уравнения отделен, т. е. найден отрезок $[a, b]$, на котором имеется только один корень уравнения. Для вычисления этого корня с требуемой точностью $\varepsilon$ строят последовательность приближений $x_n$, сходящуюся к нему. Начальное приближение $x_0$ выбирают из отрезка $[a, b]$. Вычисления продолжают до тех пор, пока не будет выполнено неравенство

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

При этом считают, что $x_n$ – это и есть корень уравнения, найденный с заданной точностью $\varepsilon$.

При выборе метода построения последовательности приближений к корню большую роль играют такие свойства метода, как простота, надежность, экономичность. Одной из его важнейших характеристик является *скорость сходимости*.

Последовательность $(x_n)$, сходящаяся к числу $x^*$, имеет скорость сходимости порядка $\alpha$, если

$$|x_n - x^*| = O\left(|x_{n-1} - x^*|^{\alpha}\right) \text{ при } n \to \infty .$$

При $\alpha = 1$ сходимость называется линейной, при $1 < \alpha < 2$ – сверхлинейной, при $\alpha = 2$ – квадратичной. С ростом $\alpha$ алгоритм, как правило, усложняется и условия сходимости становятся более жесткими.

Рассмотрим наиболее распространенные из итерационных методов уточнения корней уравнения $f(x)=0$.

### Метод половинного деления (бисекции, дихотомии)

Это простейший и надежный алгоритм уточнения простого корня.

Пусть на отрезке $[a,b]$ содержится только один корень уравнения $f(x)=0$. В качестве первого приближения $x_1$ к корню берут середину отрезка $[a,b]$. Если $f(x_1)=0$, то процесс окончен. В противном случае в качестве нового отрезка $[a_1,b_1]$ выбирают ту половину $[a,x_1]$ или $[x_1,b]$ отрезка $[a,b]$, на концах которой функция принимает значения разных знаков. Очередное приближение к корню является серединой такого отрезка. И так процесс продолжают далее.

Расчетные формулы метода имеют вид:

$$x_n=\left(a_{n-1}+b_{n-1}\right)/2,$$

$$a_n=a_{n-1},\ \ b_n=x_n,\ \text{если}\ \ f(a_{n-1})\cdot f(x_n)<0,$$

$$a_n=x_n,\ \ b_n=b_{n-1},\ \text{если}\ \ f(a_{n-1})\cdot f(x_n)>0,$$

где $[a_0,b_0]=[a,b],\ n=1,2,...$

Погрешность между точным значением корня $\xi$ и приближенным $x_n$ не превосходит длины отрезка $[a_n,b_n]$:

$$\left|\xi-x_n\right|\le b_n-a_n=\frac{b-a}{2^n}.$$

Метод сходится для любых непрерывных функций, в том числе недифференцируемых, устойчив к ошибкам округления. Скорость сходимости линейная. Для достижения заданной точности $\varepsilon$ требуется $\log_2\frac{b-a}{\varepsilon}$ итераций.

Метод неприменим для нахождения корней четной кратности.

### Метод хорд (пропорциональных частей (или отрезков))

Метод хорд является более быстрым, чем метод половинного деления, способом нахождения корня уравнения $f(x)=0$.

Пусть функция $f(x)$ непрерывна на $[a,b]$ и на этом отрезке существует ровно один корень уравнения (рис. 3.1.). Метод хорд сходится, если вторая производная $f''(x)$ сохраняет знак на отрезке $[a,b]$.

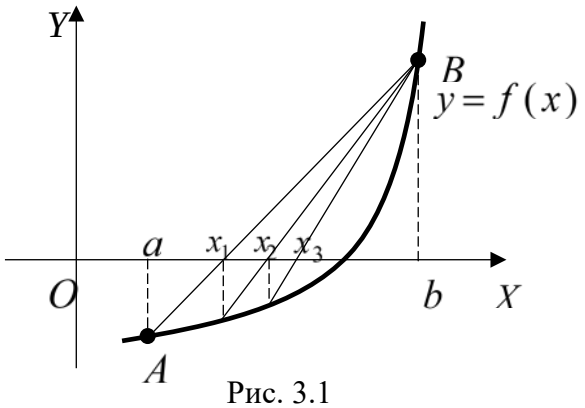


Рис. 3.1

За последовательное приближение $x_{n+1}$ к корню принимают абсциссу точки пересечения с осью $OX$ хорды, соединяющей точки $(x_n, f(x_n))$ и $(a, f(a))$ (либо $(b, f(b))$. Возможны два случая:

1) если $f(a)\cdot f''(x)>0$ на отрезке $[a,b]$, то

$$\begin{cases} x_0 = b, \\ x_{n+1} = a - \dfrac{f(a)}{f(x_n)-f(a)}(x_n - a) \end{cases}$$

или, что равносильно, $x_{n+1} = x_n - \dfrac{f(x_n)}{f(x_n)-f(a)}(x_n - a)$;

2) если $f(b)\cdot f''(x)>0$ на отрезке $[a,b]$, то

$$\begin{cases} x_0 = a, \\ x_{n+1} = x_n - \dfrac{f(x_n)}{f(b)-f(x_n)}(b - x_n). \end{cases}$$

Метод сходится линейно, но близость двух очередных приближений не всегда означает, что корень найден с требуемой точностью. Более надежным практическим критерием окончания итераций в методе хорд является выполнение неравенства

$$\frac{(x_{n+1}-x_n)^2}{|x_{n+1}+x_{n-1}-2x_n|} < \varepsilon .$$

### Метод Ньютона (касательных)

Пусть функция $f(x)$ непрерывна на отрезке $[a,b]$, ее производные $f'(x)$ и $f''(x)$ сохраняют знак на отрезке $[a,b]$ (рис. 3.2.).

Если $x_n$ – $n$-е приближение к корню уравнения $f(x)=0$, то в качестве следующего приближения $x_{n+1}$ берут абсциссу точки пересечения с осью $OX$ касательной к графику функции $f(x)$ в точке $x_n$.

Таким образом, $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$.

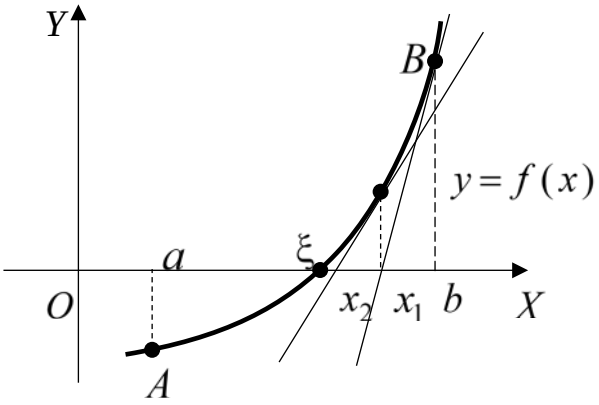


Рис. 3.2

Метод Ньютона сходится при любом начальном приближении $x_0$, но скорость сходимости существенно зависит от выбора точки $x_0$. В качестве начального приближения $x_0$ следует брать тот конец отрезка $[a, b]$, для которого выполняется условие

$$f(x_0) \cdot f''(x_0) > 0.$$

В этом случае метод имеет квадратичную скорость сходимости для простого корня.

### Модифицированный метод Ньютона

Чтобы избежать многократного вычисления производной функции $f(x)$ в методе Ньютона, используют следующую расчетную формулу:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)},$$

где $f'(x_0) \neq 0$.

Метод имеет линейную скорость сходимости.

### Метод секущих

Этот метод также является модификацией метода Ньютона, в котором производная $f'(x_n)$ заменена ее разностным приближением:

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

Таким образом, расчетная формула метода имеет вид

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \cdot f(x_n).$$

Метод является двушаговым, каждое новое приближение $x_{n+1}$ к корню строится с использованием двух предыдущих приближений $x_{n-1}$ и $x_n$, поэтому для начала итерационного процесса следует выбрать два приближения $x_0$ и $x_1$ из отрезка $[a, b]$. Метод имеет линейную скорость сходимости.

## Метод простой итерации
## (последовательных приближений, итераций)

Пусть дано уравнение $f(x)=0$, где $f(x)$ – функция, непрерывная на отрезке $[a, b]$.

В первую очередь от этого уравнения следует перейти к равносильному уравнению вида

$$x=\varphi(x),$$

что может быть выполнено бесконечным числом способов (один из них указан ниже).

Расчетная формула метода имеет вид

$$x_{n+1}=\varphi(x_n),$$

где $x_0$ (начальное приближение) – любое число из отрезка $[a, b]$.

Достаточным условием сходимости итерационного процесса (при любом начальном приближении) является выполнение неравенства

$$|\varphi'(x)|<1 \text{ для любого } x\in[a, b].$$

Например, в качестве $\varphi(x)$ можно взять функцию вида

$$\varphi(x)=x-\lambda f(x),$$

причем $\lambda$ должно удовлетворять условию

$$|\lambda|<\frac{2}{M},$$

где $M=\max\limits_{[a,b]}|f'(x)|$ и знак числа $\lambda$ совпадает со знаком производной $f'(x)$ на отрезке $[a, b]$.

Такой вариант перехода к уравнению, пригодному для итераций (т. е. $x=\varphi(x)$), также называют **методом релаксации** (ослабления).

Метод имеет линейную скорость сходимости.

Если $|\varphi'(x)|\le q<1$ для любого $x\in[a, b]$, то точность вычислений может быть оценена с помощью одного из соотношений:

$$|\xi-x_n|\le\frac{q^n}{1-q}\cdot|x_1-x_0| \qquad \text{или} \qquad |\xi-x_n|\le|x_n-x_{n-1}|,$$

где $\xi$ – точное значение корня.

## Основные операторы и функции пакета Mathematica для решения нелинейных уравнений и организации циклов

***Plot***[ $f[x],\{x,x_{\min},x_{\max}\}$ ] – строит график функции $f$ переменной $x$ на промежутке от $x_{\min}$ до $x_{\max}$.

***Plot***[$\{f_1[x], f_2[x],...\},\{x,x_{\min},x_{\max}\}$] – строит на одном рисунке графики функций $f_1$, $f_2$, … переменной *x* на промежутке от $x_{\min}$ до $x_{\max}$.

***ContourPlot***[ $g[x,y]==0,\{x,x_{\min},x_{\max}\},\{y,y_{\min},y_{\max}\}$] – строит кривую, заданную неявно уравнением $g(x,y)=0$, в прямоугольнике $x_{\min}\le x\le x_{\max}$, $y_{\min}\le y\le y_{\max}$.

***Solve***[ $lhs==rhs, x$] – находит решение уравнения $lhs=rhs$ относительно переменной *x*.

***Solve***[$\{eqn1,eqn2,...\},\{x_1,x_2,...\}$] – находит решение системы уравнений $eqn1, eqn2, ...$ относительно переменных $x_1, x_2, ...$

***NSolve***[ $eqn, x$] – находит численное решение уравнения *eqn* относительно переменной *x*.

***NSolve***[**{**$\{eqn1,eqn2,...\},\{x_1,x_2,...\}$] – находит численное решение заданной системы уравнений.

***NSolve***[$\{eqn1,eqn2,...\},\{x_1,x_2,...\},Reals$**]** – находит численное решение заданной системы уравнений на множестве действительных чисел.

***FindRoot*** [ $lhs==rhs, \{x,x_0\}$] – находит численное решение уравнения $lhs=rhs$, если для переменной *x* выбрано начальное приближение $x_0$.

***FindRoot*** [$\{eqn1,eqn2\},\{x,x_0\},\{y,y_0\}$] – находит численное решение системы уравнений $eqn1, eqn2$ с двумя переменными *x* и *y*, если выбрано начальное приближение $(x_0, y_0)$.

***Roots***[ $eqn, x$] – находит корни полиномиального уравнения *eqn* относительно переменной *x*.

***Factor***[ $P[x]$ ] – раскладывает многочлен $P(x)$ на множители с целыми коэффициентами, если это возможно.

***Expand***[*expr*] – раскрывает скобки в выражении *expr*.

***D***[*f, x*] – находит производную (также и частную) функции *f* по указанной переменной *x*.

***D***[*f,* {*x, n*}] – находит (частную) производную *n*-го порядка функции *f* по переменной *x*.

***Do***[*expr*,{*i*,$i_{\min}$,$i_{\max}$}] – вычисляет *expr* с переменной *i*, последовательно принимающей значения от $i_{\min}$ до $i_{\max}$. Выражение *expr* может состоять из нескольких команд (функций), отделенных друг от друга точкой с запятой «;».

***Do***[*expr*,{*i*,$i_{\min}$,$i_{\max}$,*h*}] – вычисляет *expr* с переменной *i*, последовательно принимающей значения от $i_{\min}$ до $i_{\max}$ с шагом *h*.

***For***[*start,test,incr,body*] – сначала вычисляет выражение *start*, а затем раз за разом вычисляет выражения *body* (основное тело цикла) и *incr* (изменяющееся значение, от которого зависит *test*) до тех пор, пока условие *test* не перестанет давать логическое значение **True**. Каждая из частей функции ***For*** (т. е. *start, test, incr, body*) может состоять из нескольких команд (функций), отделенных друг от друга точкой с запятой «;».

***While***[*test,expr*] – выполняет *expr* до тех пор, пока условие *test* не перестанет давать логическое значение True.

***If***[*condition,tr,fls*] – возвращает *tr*, если условие *condition* является верным (True), и возвращает *fls*, если *condition* ложно (False).

***If***[*condition,tr,fls,u*] – возвращает *tr*, если условие *condition* верно (True); *fls*, если *condition* ложно (False); *u*, если в результате вычисления *condition* не было получено ни True, ни False.

*Примечание.* Каждый оператор и функция пакета **Mathematica** имеет дополнительные опции, список которых можно получить, например, с помощью команды ***Options***[*Имяфункции*].

Например, ***Options***[*NSolve*] показывает, что функция ***NSolve*** имеет только одну дополнительную опцию ***WorkingPrecision***, которая задает число верных цифр результата (по умолчанию 16).

## Примеры решения нелинейных уравнений средствами пакета Mathematica

**Пример 3.1.** Отделить графически корни уравнения

$$36x^3 + 72x^2 - 241x - 182 = 0.$$

$\Delta$ Введем функцию $f(x) = 36x^3 + 72x^2 - 241x - 182$.

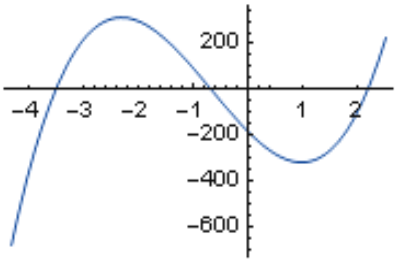
```
In[1]:= f[x_] := 36 x^3 + 72 x^2 - 241 x - 182
```

Построим ее график сначала на произвольном достаточно широком интервале, который затем уменьшим так, чтобы было понятно, между какими соседними целыми числами находятся нули функции. Возможно, потребуется построение нескольких графиков функции для отделения различных корней уравнения.

Поскольку в пакете **Mathematica** оси координат могут пересекаться не в начале координат, то чтобы не допустить ошибки при отделении корней уравнения, следует использовать опцию **AxesOrigin → {0,0}** команды **Plot**.

```
In[2]:= Plot[f[x], {x, -4.3, 2.5}, AxesOrigin → {0, 0}, ImageSize → Small]
        график функции              точка пересечения осей  размер изо…  малый
```

Out[2]=




По графику видно, что уравнение имеет три действительных корня. Они расположены в интервалах $(-4,-3)$, $(-1,0)$, $(2,3)$. ▲

**Пример 3.2.** Найти один из корней уравнения

$$36x^3 + 72x^2 - 241x - 182 = 0$$

с точностью $\varepsilon = 10^{-3}$: а) методом половинного деления; б) методом Ньютона. Указать потребовавшееся число итераций.

Δ Корни данного уравнения были отделены в примере 3.1. Доведем до заданной точности корень, расположенный на отрезке $[2,3]$.

Введем функцию, концы отрезка и точность.

```
In[1]:= f[x_] := 36 x^3 + 72 x^2 - 241 x - 182;

In[2]:= a = 2; b = 3; e = 0.001;
```

а) в соответствии с методом половинного деления организуем цикл с помощью оператора **Do**, в котором будем находить последовательные приближения к решению. Найдем середину отрезка $[a,b]$ – точку $c$. Проверим выполнение условия $f(a)\cdot f(c)<0$, чтобы выбрать ту половину отрезка $[a,b]$, внутри которой содержится корень. Эту половину снова обозначим как $[a,b]$. Оценим точность вычислений $|b-a|<\varepsilon$. Если это неравенство верно, или если $f(c)=0$, то процесс окончен.

```
In[3]:= Do[c = (a + b) / 2.;
          оператор цикла
            fc = f[c];
            If[f[a] * fc < 0, b = c, If[fc ≠ 0, a = c]];
            условный оператор          условный оператор
              If[Abs[b - a] < e || fc == 0,
              у…  абсолютное значение
                Print["Решение x=", c // N, " получено на ", n, " шаге."];
                печатать                    численное приближение
                Break[]],
                прервать цикл
          {n, 1, 100}]

Решение x=2.16699 получено на 10 шаге.
```

б) проверим выполнение условий сходимости метода Ньютона. Функция $f(x)$ принимает на концах отрезка $[2,3]$ значения разных знаков; ее первая и вторая производные ( $f'(x)=108x^2+144x-241$ и $f''(x)=216x+144$ ) положительны на этом отрезке. В качестве начального приближения возьмем правый конец отрезка $b=3$, т. к. $f(b)\cdot f''(x)>0$.

```
In[4]:= a = 2; b = 3;
```

Переменные $a$ и $b$ использовались и изменялись в цикле, реализующем метод половинного деления, поэтому снова введем концы отрезка $[a,b]=[2,3]$.

Далее организуем цикл с помощью оператора **Do**, в котором будем находить последовательные приближения к решению в соответствии с методом Ньютона.

```
In[5]:= x1 = b;
        Do[x2 = x1;
        оператор цикла
          x1 = (x1 - f[x1] / f'[x1]) // N;
                                  численное приближение
          If[Abs[x2 - x1] < e,
          у…  абсолютное значение
            Print["Решение x=", x2 // N, " получено на ", n, " шаге."];
            печатать                    численное приближение
            Break[]],
            прервать цикл
        {n, 1, 100}]

Решение x=2.16691 получено на 4 шаге.
```

Итак, чтобы получить корень уравнения с точностью $\varepsilon = 10^{-3}$, потребовалось 10 шагов для метода половинного деления и 4 шага для метода Ньютона. ▲

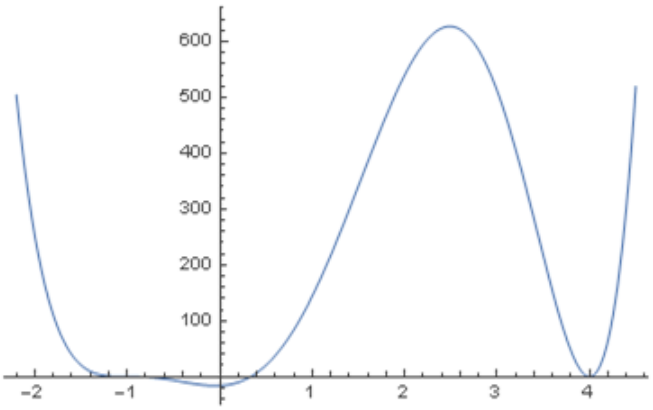
**Пример 3.3.** Отделить графически и найти с помощью встроенных функций пакета **Mathematica** корни уравнения

$$3x^6 - 16x^5 - 10x^4 + 80x^3 + 95x^2 + 8x - 16 = 0.$$

Δ Для отделения корней зададим левую часть уравнения как функцию $f(x)$ и построим ее график.

```
In[1]:= f[x_] := 3 x^6 - 16 x^5 - 10 x^4 + 80 x^3 + 95 x^2 + 8 x - 16

In[2]:= Plot[f[x], {x, -2.2, 4.5}]
        график функции
```

Out[2]=




Корни данного уравнения находятся на промежутках $(-1{,}5; -0{,}5)$, $(0, 1)$, $(3{,}5; 4{,}5)$. Рассмотрим несколько способов решения данного уравнения средствами пакета **Mathematica**: с помощью функций **Solve**, **NSolve**, **Roots**, а также посредством разложения многочлена $f(x)$ на множители функцией **Factor**.

```
In[3]:= Solve[f[x] == 0]
        решить уравнения

Out[3]= {{x → -1}, {x → -1}, {x → -1}, {x → 1/3}, {x → 4}, {x → 4}}

In[4]:= NSolve[f[x] == 0]
        численное решение уравнений

Out[4]= {{x → -1.}, {x → -1.}, {x → -1.}, {x → 0.333333}, {x → 4.}, {x → 4.}}
```

```
In[5]:= Roots[f[x] == 0, x]
        корни многочлена
Out[5]= x == 1/3 || x == -1 || x == -1 || x == -1 || x == 4 || x == 4
```

```
In[6]:= Factor[f[x]]
        факторизовать
Out[6]= (-4 + x)^2 (1 + x)^3 (-1 + 3 x)
```

Как видно, данное уравнение имеет три корня, причем корни $-1$ и 4 являются трехкратным и двукратным корнями соответственно.

Отметим еще один способ нахождения корня уравнения при заданном начальном приближении – с помощью функции **FindRoot**.

```
In[7]:= FindRoot[f[x] == 0, {x, -2}]
        найти корень
Out[7]= {x → -1.}
```

```
In[8]:= FindRoot[f[x] == 0, {x, 0}]
        найти корень
Out[8]= {x → 0.333333}
```

```
In[9]:= FindRoot[f[x] == 0, {x, 3}]
        найти корень
Out[9]= {x → 4.}
```

▲

**Пример 3.4.** Отделить графически и найти с помощью встроенных функций пакета **Mathematica** корни уравнения

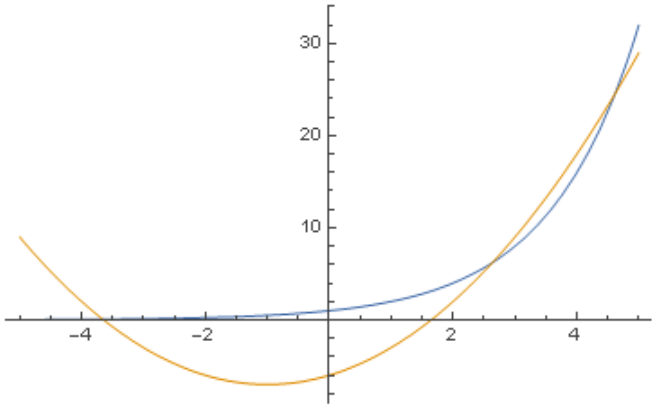
$$2^x = x^2 + 2x - 6.$$

Δ Введем две функции $f(x) = 2^x$ и $g(x) = x^2 + 2x - 6$ и построим их графики в одной системе координат. Корнями данного уравнения являются абсциссы точек пересечения этих графиков.

```
In[1]:= f[x_] := 2^x; g[x_] := x^2 + 2 x - 6;
```

```
In[2]:= Plot[{f[x], g[x]}, {x, -5, 5}]
        график функции
```

Out[2]=




Уравнение имеет три корня. Они расположены в интервалах $(-4,-3)$, $(2,3)$ и $(4,5)$.

При попытке численно решить данное уравнение с помощью встроенной функции **NSolve** программа выдает сообщение о том, что это уравнение не может быть решено методами, доступными **NSolve**.

```
In[3]:= NSolve[f[x] == g[x], x]
        численное решение уравнений

        NSolve: This system cannot be solved with the methods available to NSolve.

Out[3]= NSolve[2^x == -6 + 2 x + x^2, x]
```

Получим корни уравнения, используя функцию **FindRoot**, задавая начальное приближение к каждому из корней.

```
In[4]:= FindRoot[f[x] == g[x], {x, -4}]
        найти корень

Out[4]= {x → -3.66065}
```

```
In[5]:= FindRoot[f[x] == g[x], {x, 2}]
        найти корень

Out[5]= {x → 2.6345}
```

```
In[6]:= FindRoot[f[x] == g[x], {x, 4}]
        найти корень

Out[6]= {x → 4.61903}
```

▲

**Пример 3.5.** Решить средствами пакета **Mathematica** систему уравнений

$$\begin{cases} x^2 + y^2 - 2x = 5, \\ x^3 + \sin y = 1. \end{cases}$$

Δ Решение системы получим с помощью функции **FindRoot**, для которой нужно указать начальное приближение к решению.

Введем две функции $f(x,y) = x^2 + y^2 - 2x - 5$ и $g(x,y) = x^3 + \sin y - 1$.

```
In[1]:= f[x_, y_] = x^2 + y^2 - 2 x - 5
        g[x_, y_] = x^3 + Sin[y] - 1
                    синус
```

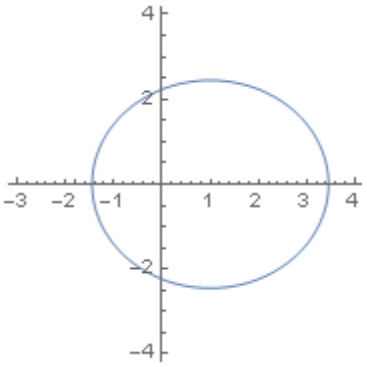
```
Out[1]= -5 - 2 x + x^2 + y^2

Out[2]= -1 + x^3 + Sin[y]
```

Решением данной системы являются точки пересечения кривых $f(x,y) = 0$ и $g(x,y) = 0$. Построим их с помощью функции **ContourPlot**, а затем объединим в одном графическом окне функцией **Show**.

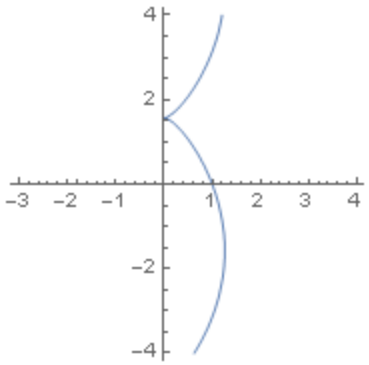
```
In[3]:= gr1 = ContourPlot[f[x, y] == 0, {x, -3, 4}, {y, -4, 4}, Axes → True,
            контурный график                                    оси    истина
        Frame → False, ImageSize → Small]
        рамка   ложь   размер изо···  малый
```

```
In[4]:= gr2 = ContourPlot[g[x, y] == 0, {x, -3, 4}, {y, -4, 4}, Axes → True,
            контурный график                                    оси    истина
        Frame → False, ImageSize → Small]
        рамка   ложь   размер изо···  малый
```

Out[3]=

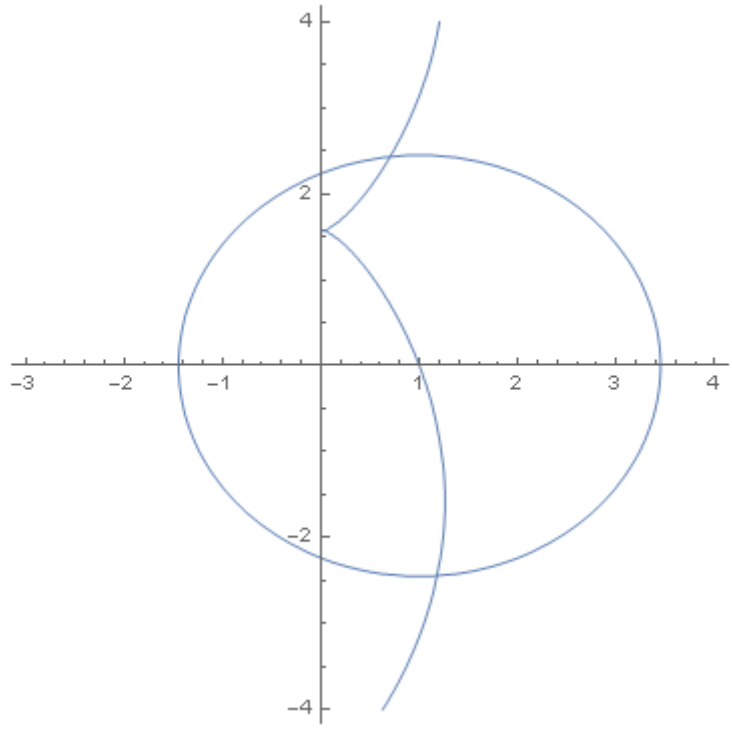



Out[4]=




```
In[5]:= Show[gr1, gr2, ImageSize → Medium]
        показать   размер изо···  средний
```

Out[5]=




По графику видно, что кривые пересекаются в двух точках. Одна из них находится вблизи точки $(1, 2)$, вторая – возле точки $(1, -2)$. Эти точки и будем использовать как начальные приближения.

```
In[6]:= FindRoot[{f[x, y] == 0, g[x, y] == 0}, {x, 1}, {y, 2}]
        найти корень
Out[6]= {x → 0.703435, y → 2.43147}

In[7]:= FindRoot[{f[x, y] == 0, g[x, y] == 0}, {x, 1}, {y, -2}]
        найти корень
Out[7]= {x → 1.18005, y → -2.44286}
```

▲

# Лабораторная работа 2

## Численное решение нелинейных уравнений

**1.** Отделите графически корни алгебраического уравнения $f(x)=0$ с помощью функции **Plot**. Найдите один из них (нецелый) с точностью $\varepsilon=10^{-3}$ методом хорд. Укажите потребовавшееся число итераций. Проиллюстрируйте графически нахождение первых двух приближений (постройте график функции и хорды).

1.1. $f(x)=15x^3-86x^2+111x-28$.

1.2. $f(x)=63x^3-248x^2+155x+66$.

1.3. $f(x)=66x^3-49x^2-209x-84$.

1.4. $f(x)=9x^3-3x^2-122x-40$.

1.5. $f(x)=26x^3-57x^2-41x+102$.

1.6. $f(x)=14x^3-179x^2+281x-66$.

1.7. $f(x)=12x^3-100x^2+237x-135$.

1.8. $f(x)=12x^3+29x^2-52x+11$.

1.9. $f(x)=6x^3-85x^2+309x-170$.

1.10. $f(x)=14x^3-151x^2+479x-396$.

1.11. $f(x)=26x^3-171x^2+336x-196$.

1.12. $f(x)=18x^3-39x^2-154x-65$.

1.13. $f(x)=36x^3-168x^2+55x+350$.

1.14. $f(x)=21x^3-61x^2+19x+21$.

1.15. $f(x)=54x^3+117x^2-276x+77$.

1.16. $f(x)=36x^3-219x^2+349x-110$.

**2.** Отделите графически и найдите с помощью функций **Solve**, **NSolve**, **Roots**, **FindRoot** корни алгебраического уравнения $f(x)=0$. Разложите многочлен $f(x)$ на множители, используя функцию **Factor**.

2.1. $f(x)=x^6+6x^5+12x^4+6x^3-9x^2-12x-4$.

2.2. $f(x)=x^6+7x^5-5x^4-95x^3-20x^2+304x-192$.

2.3. $f(x)=x^6+8x^5+17x^4-8x^3-45x^2+27$.

2.4. $f(x)=x^6+6x^5-50x^3-45x^2+108x+108$.

2.5. $f(x)=x^6-3x^5-9x^4+19x^3+12x^2-36x+16$.

2.6. $f(x)=x^6-x^5-18x^4+14x^3+61x^2-93x+36$.

2.7. $f(x)=x^6+x^5-26x^4-52x^3+152x^2+512x+384$.

2.8. $f(x) = x^6 + x^5 - 31x^4 - 13x^3 + 306x^2 - 864$.

2.9. $f(x) = x^6 + 7x^5 + 5x^4 - 55x^3 - 90x^2 + 108x + 216$.

2.10. $f(x) = x^6 - x^5 - 19x^4 - 15x^3 + 46x^2 + 28x - 40$.

2.11. $f(x) = x^6 - 19x^5 + 147x^4 - 589x^3 + 1276x^2 - 1392x + 576$.

2.12. $f(x) = x^6 - 6x^5 - 24x^4 + 82x^3 + 315x^2 + 324x + 108$.

2.13. $f(x) = x^6 + 4x^5 - 10x^4 - 24x^3 + 13x^2 + 44x + 20$.

2.14. $f(x) = x^6 - 14x^5 + 73x^4 - 188x^3 + 256x^2 - 176x + 48$.

2.15. $f(x) = x^6 + 8x^5 + 2x^4 - 36x^3 + x^2 + 52x - 28$.

2.16. $f(x) = x^6 + 12x^5 + 45x^4 + 48x^3 - 21x^2 - 60x - 25$.

**3.** Отделите графически корни трансцендентного уравнения с помощью функции **Plot**. Найдите один из них с точностью $\varepsilon = 10^{-3}$:

а) методом Ньютона;

б) методом секущих.

Укажите потребовавшееся число итераций.

3.1. $3^x = 4x + 2$.

3.2. $\log_2 x = x^2 - 3x - 1$.

3.3. $7\,\mathrm{arctg}\,4x = x^3 + 2x + 2$.

3.4. $0{,}7^x - 5 = \log_2(4 - x)$.

3.5. $x\log_3 x + x^2 - 5x + 2 = 0$.

3.6. $5\cos(2x + 1) = 3x^2 + 5x - 4$.

3.7. $7x^2 - x = 2^{3x-4} + 9$.

3.8. $\sqrt{12x^2 + x + 3} = 2^x + 2$.

3.9. $4 \cdot 3^{1-x} = 2x - 15x^2 + 37$.

3.10. $\log_2(x + 5) = 4 - \sqrt{2x^2 + 1}$.

3.11. $5x^2 - 4 + \sqrt{x + 5}x = 4\sin(3x - 1)$.

3.12. $12x - 5x^2 = 2^x - 9$.

3.13. $8\cos^2(x + 3) = 15 + 9x - 4x^2$.

3.14. $7^{x-1} + 3^x = x^4 + x - 1$.

3.15. $\log_3(x + 4) = 3 - 2x^2$.

3.16. $5^x - 4x = \ln(3x + 8)$.

**4.** Приведите уравнение (3.1–3.16) к виду, пригодному для итераций. Найдите его корни методом простых итераций с точностью $\varepsilon = 10^{-3}$. Укажите потребовавшееся число итераций.

**5.** Решите уравнение (3.1–3.16) с помощью функций **Solve**, **NSolve**, **FindRoot**.

**6.** Дана система двух нелинейных уравнений $f(x,y)=0$, $g(x,y)=0$. Используя средства пакета **Mathematica**, изобразите на одном чертеже кривые $f(x,y)=0$ и $g(x,y)=0$, и решите данную систему.

6.1. $\begin{cases} \sqrt[3]{x^2}+\sqrt[3]{y^2}=\sqrt[3]{25}, \\ x^3+y^3-9xy=0. \end{cases}$

6.2. $\begin{cases} y^2=x^3/(7-x)+1, \\ x^2-3\arccos(1-y/5)=0. \end{cases}$

6.3. $\begin{cases} \left(x^2+y^2\right)^2=21\left(x^2-y^2\right), \\ 2x^4-3y^2-4x-y^5=1. \end{cases}$

6.4. $\begin{cases} \left(x^2+y^2+2x\right)^2=4\left(x^2+y^2\right), \\ (x-y)^2=7y-5x-8. \end{cases}$

6.5. $\begin{cases} \left(x^2+y^2-2x\right)^2=24\left(x^2+y^2\right), \\ \sin(2x+y)=4y-3x+2. \end{cases}$

6.6. $\begin{cases} \sqrt[3]{(x+1)^2}+\sqrt[3]{(y-3)^2}=4, \\ 3x^2-5y^2=15. \end{cases}$

6.7. $\begin{cases} \left(x^2+y^2\right)^2=8\left(x^2-y^2\right)+65, \\ 2^{x+7y}+x-9y^2=0. \end{cases}$

6.8. $\begin{cases} y=\sinh(2y-x)-3x+1, \\ \left(x^2+y^2\right)^2=32\left(y^2-x^2\right). \end{cases}$

6.9. $\begin{cases} \left(x^2+y^2-6y\right)^2=36\left(x^2+y^2\right), \\ \sqrt[3]{(x-4)^2}+\sqrt[3]{(y-1)^2}=4. \end{cases}$

6.10. $\begin{cases} x^3+y^3=15xy, \\ y=4-5\operatorname{arctg}(x-5). \end{cases}$

6.11. $\begin{cases} \left(x^2+y^2+5y\right)^2=8\left(x^2+y^2\right), \\ 7y-3x^2+4x=\cos(y+1)-9. \end{cases}$

6.12. $\begin{cases} \left(x^2+y^2\right)^2=18\left(x^2-y^2\right)+12, \\ 2x^2-y^2-4x+6y-11=0. \end{cases}$

6.13. $\begin{cases} \sqrt[3]{(x-1)^2}+\sqrt[3]{(y-3)^2}=5, \\ \sqrt{x^2+y^2}=2\cosh(x-y-1). \end{cases}$

6.14. $\begin{cases} \left(x^2+y^2-3x\right)^2=9\left(x^2+y^2\right), \\ 2y-3x+5\ln(4x+y)=3. \end{cases}$

6.15. $\begin{cases} \left(x^2+y^2\right)^2=54xy, \\ x^2-3y^2+2x=y-2. \end{cases}$

6.16. $\begin{cases} (y+3)^2=\dfrac{x^3-4x^2+4x}{4-x}, \\ \left(x^2+y^2+6y\right)^2=30\left(x^2+y^2\right). \end{cases}$

# 4. Решение систем линейных алгебраических уравнений

Рассмотрим систему $n$ линейных алгебраических уравнений с $n$ неизвестными (СЛАУ) вида

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + .... + a_{nn}x_n = b_n \end{cases}$$

или $AX = B$, где $a_{ij}$ – коэффициенты системы, $A = \left(a_{ij}\right)$ – матрица системы, $X$ – столбец неизвестных $x_j$, $B$ – столбец свободных членов $b_i$, $i = \overline{1, n}$, $j = \overline{1, n}$.

Будем также предполагать, что определитель главной матрицы системы $A$ отличен от нуля. Случаи, когда число неизвестных не совпадает с числом уравнений или когда $\det A = 0$, можно свести к этому.

Рассмотрим два вида методов решения систем линейных уравнений:

**1. Прямые** (или **точные**) – методы, которые в предположении, что вычисления ведутся без округлений, позволяют за конечное число шагов получить точное решение системы или установить ее неразрешимость.

**2. Итерационные** – методы, позволяющие получить приближенное решение системы за конечное число шагов, количество которых зависит от требуемой точности приближения.

Эффективность методов решения СЛАУ во многом зависит от структуры и свойств матрицы системы, одним из которых является ее *обусловленность*.

## Нормы векторов и матриц. Число обусловленности матрицы. Погрешность решения СЛАУ прямыми методами

Наиболее часто используются следующие нормы векторов и согласованные с ними нормы матриц.

**1.** ***Первая норма вектора*** $\vec{x} = (x_1, x_2, ..., x_n)$ (1-норма, норма-сумма, октаэдрическая норма):

$$\left\|\vec{x}\right\|_1 = \sum_{i=1}^{n} \left|x_i\right|.$$

***Первая норма матрицы*** $A = \left(a_{ij}\right)$ (1-норма, норма-сумма, столбцовая норма):

$$\left\|A\right\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} \left|a_{ij}\right|.$$

Здесь суммируются взятые по модулю элементы каждого столбца, а затем выбирается наибольшее из полученных чисел.

**2.** ***Евклидова норма вектора*** $\vec{x} = (x_1, x_2, ..., x_n)$ (сферическая норма):

$$\left\|\vec{x}\right\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}\ .$$

***Спектральная норма матрицы*** $A$ (2-норма):

$$\left\|A\right\|_2 = \sqrt{\lambda}\ ,$$

где $\lambda$ – наибольшее собственное значение матрицы $A^T A$ (все собственные значения матриц такого вида являются неотрицательными действительными числами).

**3.** ***Норма-максимум вектора*** $\vec{x} = (x_1, x_2, ..., x_n)$ (∞-норма, кубическая норма):

$$\left\|\vec{x}\right\|_\infty = \max_{1 \le i \le n} \left|x_i\right|.$$

***Норма-максимум матрицы*** $A = \left(a_{ij}\right)$ (∞-норма, строчная норма):

$$\left\|A\right\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} \left|a_{ij}\right|.$$

Здесь суммируются взятые по модулю элементы каждой строки, а затем выбирается наибольшее из полученных чисел.

Отметим еще одну норму матрицы – ***норму Фробениуса***:

$$\left\|A\right\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2}\ .$$

***Числом обусловленности*** невырожденной матрицы $A$ называется число, которое обозначается $cond(A)$, и вычисляется по формуле

$$cond(A) = \left\|A\right\| \cdot \left\|A^{-1}\right\|.$$

Для любой матрицы $A$ в указанных нормах справедливо неравенство $cond(A) \ge 1$.

Матрица $A^{-1}$, обратная к невырожденной матрице $A$, называется ***устойчивой***, если малым изменениям элементов матрицы $A$ соответствуют малые изменения элементов матрицы $A^{-1}$. В противном случае обратная матрица называется ***неустойчивой***.

Матрица $A$ называется ***хорошо обусловленной***, если матрица $A^{-1}$ устойчива, и ***плохо обусловленной,*** если обратная матрица неустойчива.

Хорошо обусловленными являются матрицы с малым числом обусловленности, плохо обусловленными – с большим $cond(A)$. Плохо обусловленными часто являются почти вырожденные матрицы, у которых определитель близок к нулю.

Пусть в системе линейных уравнений

$$AX = B$$

правая часть оказалась заданной неточно (получила возмущение) или в процессе вычислений возникли, например, ошибки округления. Тогда фактически решается возмущенная система

$$AX = B + \Delta B \quad \text{или} \quad (A + \Delta A)X = B + \Delta B.$$

Обозначим через $X$ – решение точной системы $AX = B$, $X^*$ – решение возмущенной системы. Решение $X^*$ можно считать приближенным решением системы $AX = B$.

Оценить ***погрешность*** приближенного решения $X^*$ можно при помощи двух векторов – ***вектора ошибки***

$$\Delta X = X - X^*$$

и ***вектора невязки***

$$R = AX^* - B.$$

Вектор невязки показывает, насколько левая часть системы $AX = B$, вычисленная на приближенном решении $X^*$, отличается от правой части системы.

Если матрица $A$ линейной системы уравнений плохо обусловлена, то незначительные изменения ее коэффициентов или правых частей уравнений могут приводить к большим изменениям решения. Системы с большим числом обусловленности $cond(A)$ называются ***плохо обусловленными***.

Справедливо соотношение, связывающее предельную относительную погрешность решения $\delta_X = \|X^* - X\| / \|X^*\|$, число обусловленности матрицы системы $cond(A)$ и относительную погрешность правых частей системы $\delta_B = \|\Delta B\| / \|B + \Delta B\|$:

$$\delta_X = cond(A) \cdot \delta_B.$$

Если в системе $AX = B$ получила приращение не только правая часть $B$, но и матрица системы $A$, то при условии $\|A^{-1}\| \cdot \|\Delta A\| < 1$ верно

$$\delta_X = \frac{cond(A)}{1 - \delta_A \cdot cond(A)} \cdot (\delta_B + \delta_A),$$

где $\delta_A = \|\Delta A\| / \|A + \Delta A\|$ – относительная погрешность матрицы системы.

## Прямые методы решения систем линейных уравнений

Перечислим основные ***прямые*** (***точные***) методы решения линейных систем $AX = B$ с невырожденной матрицей $A$.

**1.** ***Матричный метод.*** Решение получают по формуле $X = A^{-1}B$.

**2.** ***Метод Крамера.*** Решение получают по формулам $x_i = \dfrac{d_i}{d}$, где $d = \det A$, $d_i$ – определитель матрицы, полученной из матрицы $A$ заменой в ней $i$-го столбца столбцом свободных членов $B$, $i = \overline{1, n}$.

**3.** ***Метод Гаусса*** **(*метод исключения неизвестных, схема единственного деления*).** Решение этим методом состоит из двух этапов.

Сначала систему $AX = B$ с помощью элементарных преобразований над строками ее расширенной матрицы $(A|B)$ приводят к равносильной системе с верхней треугольной матрицей (***прямой ход*** метода Гаусса):

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

Для этого в первую очередь получают нули в первом столбце под элементом $a_{11}$, вычитая из $i$-й строки первую строку, умноженную на $a_{i1}/a_{11}$ ($i = \overline{2, n}$). Если $a_{11} = 0$, то предварительно переставляют строки (уравнения системы) так, чтобы $a_{11} \neq 0$.

Затем получают нули во втором столбце под главной диагональю, т. е. в строках с третьей по $n$-ю, вычитая из $i$-й строки измененную вторую строку, умноженную на $a_{i2}^{(1)}/a_{22}^{(1)}$ ($i = \overline{3, n}$), и т. д. Элементы $a_{ii}^{(i-1)}$, на которые осуществляется деление, называются ***ведущими*** (или ***главными***) ***элементами*** метода Гаусса и не должны равняться нулю.

***Обратный ход*** метода Гаусса заключается в последовательном определении неизвестных, начиная с $x_n$ и заканчивая $x_1$ из системы, полученной на первом этапе:

$$x_n = b_n^{(n-1)}/a_{nn}^{(n-1)}, \quad x_i = \left(b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)}x_j\right)\Big/a_{ii}^{(i-1)}, \quad i = \overline{n-1, 1}.$$

Чтобы уменьшить влияние ошибок округлений и исключить деление на нуль, рассматривают различные модификации метода Гаусса, например, метод Гаусса *с постолбцовым выбором главного элемента*, метод *оптимального исключения* и др.

**4.** ***Метод прогонки*** решения систем с трехдиагональными матрицами. Этот метод является модификацией метода Гаусса и применяется к решению систем вида

$$\begin{cases} b_1x_1 + c_1x_2 = d_1, \\ a_2x_1 + b_2x_2 + c_2x_3 = d_2, \\ \qquad a_3x_2 + b_3x_3 + c_3x_4 = d_3, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \qquad\qquad a_{n-1}x_{n-2} + b_{n-1}x_{n-1} + c_{n-1}x_n = d_{n-1}, \\ \qquad\qquad\qquad a_nx_{n-1} + \ b_nx_n = d_n. \end{cases}$$

Матрица этой системы имеет ***трехдиагональный*** вид – ее ненулевые элементы расположены только на главной диагонали, над ней и под ней.

Система может быть записана в краткой форме:

$$\begin{cases} a_ix_{i-1} + b_ix_i + c_ix_{i+1} = d_i, \quad i = \overline{1, n}, \\ a_1 = 0, \quad c_n = 0. \end{cases}$$

Метод прогонки состоит из двух этапов: первый – ***прямая прогонка***, второй – ***обратная прогонка***.

На первом этапе систему преобразуют так, чтобы ее ненулевые коэффициенты оставались только на главной диагонали и над ней, т. е. приводят к виду

$$\begin{cases} x_i = L_ix_{i+1} + M_i, \quad i = \overline{1, n-1}, \\ x_n = M_n. \end{cases}$$

Числа $L_i$ и $M_i$ называются ***прогоночными коэффициентами*** и вычисляются по рекуррентным формулам:

$$L_1 = -\frac{c_1}{b_1}, \quad M_1 = \frac{d_1}{b_1}, \quad L_i = -\frac{c_i}{b_i + a_iL_{i-1}}, \quad M_i = \frac{d_i - a_iM_{i-1}}{b_i + a_iL_{i-1}}, \quad i = \overline{2, n}.$$

На втором этапе (*обратная прогонка*) последовательно находят неизвестные, начиная с $x_n$, по формулам

$$x_n = M_n, \quad x_i = L_ix_{i+1} + M_i, \quad i = \overline{1, n-1}.$$

Если выполняются неравенства

$$|b_i| \geq |a_i| + |c_i|, \quad i = \overline{1, n},$$

причем хотя бы для одного значения $i$ неравенство является строгим, $a_i \neq 0$, $c_i \neq 0$ (кроме $a_1 = 0$, $c_n = 0$ по условию), то знаменатели всех прогоночных коэффициентов не обращаются в нуль, и система линейных уравнений имеет единственное решение.

Количество арифметических операций, которое нужно выполнить для получения решения системы $n$ уравнений методом прогонки, пропорционально $n$.

## Итерационные методы решения систем линейных уравнений

Процесс решения системы линейных уравнений итерационными методами состоит в построении последовательности приближений к решению, сходящейся к нему. Существует большое количество итерационных методов, рассмотрим некоторые из них.

**1.** ***Метод простой итерации.*** Рассмотрим систему *n* линейных алгебраических уравнений с *n* неизвестными $AX = B$ с невырожденной матрицей *A*. Преобразуем эту систему каким-нибудь способом (их бесконечно много) к виду

$$X = \Phi X + \beta,$$

где $\Phi$ – квадратная матрица порядка *n*; $\beta$ – вектор-столбец.

Согласно методу простой итерации для начала вычислений задают начальное приближение $X^{(0)}$ (например, $X^{(0)} = \beta$), все следующие приближения к решению находят с помощью рекуррентной формулы

$$X^{(k)} = \Phi X^{(k-1)} + \beta, \quad k = 1, 2, \ldots.$$

Для того чтобы метод простых итераций сходился к решению системы $X = \Phi X + \beta$ при любом начальном приближении $X^{(0)}$, ***необходимо и достаточно***, чтобы все собственные значения матрицы $\Phi$ были по модулю меньше единицы.

На практике это бывает трудно проверить, поэтому часто пользуются ***достаточными условиями сходимости***.

Если какая-нибудь норма матрицы $\Phi$ меньше единицы ($\|\Phi\| = q < 1$), то метод простых итераций сходится к единственному решению $X^*$ системы $X = \Phi X + \beta$ при любом начальном приближении $X^{(0)}$. Причем для всех $k \in \mathbb{N}$ справедливы оценки погрешности:

1) $\left\|X^* - X^{(k)}\right\| \le \dfrac{q^k}{1-q}\left\|X^{(1)} - X^{(0)}\right\|$ (*априорная*);

2) $\left\|X^* - X^{(k)}\right\| \le \dfrac{q}{1-q}\left\|X^{(k)} - X^{(k-1)}\right\|$ (*апостериорная*).

Чем меньше норма матрицы $\Phi$, тем быстрее сходится итерационный процесс.

Чтобы найти решение системы с точностью $\varepsilon$, то, как следует из апостериорной оценки погрешности, можно использовать следующее *условие окончания итерационного процесса*:

$$\left\|X^{(k)} - X^{(k-1)}\right\| < \frac{1-q}{q} \cdot \varepsilon, \quad \text{где} \quad q = \|\Phi\|.$$

**2. *Метод Якоби.*** Этот метод является частным случаем метода простой итерации.

Пусть все диагональные элементы матрицы $A$ системы $AX = B$ отличны от нуля ($a_{ii} \neq 0$). Тогда, выражая диагональные элементы через остальные, получим систему вида $X = \Phi X + \beta$:

$$\begin{cases} x_1 = -\left(a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n - b_1\right)/a_{11}, \\ x_2 = -\left(a_{21}x_1 + a_{23}x_3 + ... + a_{2n}x_n - b_2\right)/a_{22}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = -\left(a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + .... + a_{n,n-1}x_{n-1} - b_n\right)/a_{nn} \end{cases}$$

или

$$x_i = -\frac{1}{a_{ii}}\left[\sum_{j=1,\, j\neq i}^{n} a_{ij}x_j - b_i\right], \quad i = \overline{1, n}.$$

В *методе Якоби* используется следующий алгоритм построения приближений:

$$x_i^{(k)} = -\frac{1}{a_{ii}}\left[\sum_{j=1,\, j\neq i}^{n} a_{ij}x_j^{(k-1)} - b_i\right], \quad i = \overline{1, n}.$$

Условия сходимости метода Якоби совпадают с условиями сходимости метода простой итерации. Отметим еще один вариант ***достаточного условия***: если $A$ – матрица с диагональным преобладанием, т. е. $|a_{ii}| > \sum_{j=1,\, j\neq i}^{n} |a_{ij}|$ для всех $i = \overline{1, n}$, то метод Якоби сходится при любом начальном приближении $X^{(0)}$.

**3. *Метод Зейделя.*** Это модификация метода простой итерации, в которой при $k$-й итерации для вычисления $x_i^{(k)}$ используются уже вычисленные на этом шаге новые значения $x_1^{(k)}$, $x_2^{(k)}$, …, $x_{i-1}^{(k)}$, а остальные берутся из предыдущего $(k-1)$-го шага. Это приводит, как правило, к ускорению сходимости.

Пусть система $AX = B$ приведена к виду $X = \Phi X + \beta$. Тогда расчетные формулы метода Зейделя имеют вид

$$\begin{cases} x_1^{(k)} = \varphi_{11}x_1^{(k-1)} + \varphi_{12}x_2^{(k-1)} + \cdots + \varphi_{1n}x_n^{(k-1)} + \beta_1, \\ x_2^{(k)} = \varphi_{21}x_1^{(k)} + \varphi_{22}x_2^{(k-1)} + \cdots + \varphi_{2n}x_n^{(k-1)} + \beta_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k)} = \varphi_{n1}x_1^{(k)} + \varphi_{n2}x_2^{(k)} + \cdots + \varphi_{nn}x_n^{(k-1)} + \beta_n, \end{cases}$$

или

$$x_i^{(k)} = \sum_{j=1,\ i\neq 1}^{i-1} \varphi_{ij} x_j^{(k)} + \sum_{j=i}^{n} \varphi_{ij} x_j^{(k-1)} + \beta_i, \quad i = \overline{1, n}.$$

Условия сходимости метода Зейделя совпадают с условиями сходимости метода простой итерации. Если система $AX = B$ приведена к виду $X = \Phi X + \beta$ так же, как в методе Якоби, то метод Зейделя сходится, если только матрица $A$ удовлетворяет условию усиленного доминирования главной диагонали, т. е.

$$\sum_{j=1,\ j\neq i}^{n} \left|a_{ij}\right| \le \alpha \cdot \left|a_{ii}\right|, \quad 0 < \alpha < 1 \ \text{ для всех } \ i = \overline{1, n}.$$

## Основные операции и функции пакета Mathematica для работы с векторами, матрицами и системами линейных алгебраических уравнений

В пакете **Mathematica** любой набор элементов, заключенный в фигурные скобки, является списком. Вектор – список, матрицу заменяет список списков.

Ввести матрицу в программе **Mathematica** можно несколькими способами:

1. Непосредственно ввести с клавиатуры в виде списка списков, например $\{\{a, b\}, \{c, d\}\}$.

2. На главном меню выбрать **Insert → Table/Matrix → New…** В появившемся окне необходимо выбрать **Matrix (List of lists)** и ввести количество строк, столбцов, а затем нажать **OK**. Появится матрица, которую следует заполнить числами.

3. При помощи панели инструментов выбрать **Palettes → Basic Math Assistant**. Перейти на закладку **Advanced** и щелкнуть по кнопке **Matrix** для ввода матрицы $2 \times 2$. Дополнительная строка добавляется нажатием комбинации клавиш **Ctrl+Enter**, а столбец – **Ctrl+«,»** (запятая).

***Array***[*f,n*] – создает одномерный массив (вектор) длиной *n* вида $\{f[1], f[2], ..., f[n]\}$.

***Array***[*f*, $\{n_1, n_2\}$] – создает двумерный массив (матрицу) размером $n_1 \times n_2$ с элементами $f[i, j]$, $i = \overline{1, n_1}$, $j = \overline{1, n_2}$.

***A***[[*i,j*]] – выбирает элемент $a_{ij}$ матрицы $A$.

***A***[[*i*]] – выбирает *i*-ю строку матрицы $A$.

***Append***[*lst,x*] – создает новый список, добавляя элемент *x* в конец списка *lst*.

***CharacteristicPolynomial*** [ $A, \lambda$ ] – возвращает характеристический многочлен матрицы $A$ относительно переменной $\lambda$.

***ConstantArray***$[b, \{m, n\}]$ – задает постоянную матрицу размером $m \times n$ с элементами *b*.

***Cross***$[v_1, v_2]$ или $v_1 \times v_2$ – находит векторное произведение векторов $v_1$ и $v_2$.

***Dot***$[v_1, v_2]$ или $v_1.v_2$ – вычисляет скалярное произведение векторов $v_1$ и $v_2$.

***Dot***$[A, B]$ или $A.B$ – вычисляет произведение матриц $A$ и $B$.

***Det***$[A]$ – вычисляет определитель квадратной матрицы $A$.

***DiagonalMatrix***[*list*] – создает диагональную матрицу, размещая на главной диагонали элементы списка *list*.

***Dimension***$[A]$ – определяет размерность матрицы $A$.

***Drop***$[A, i]$ – удаляет *i*-ю строку матрицы $A$.

***Drop***$[A, \{j\}, \{k\}]$ – удаляет *j*-й и *k*-й столбец матрицы $A$.

***Eigenvalues***$[A]$ – возвращает список собственных значений квадратной матрицы *A*.

***Eigenvectors***$[A]$ – возвращает список собственных векторов квадратной матрицы $A$.

***IdentityMatrix***[*n*] – задает единичную матрицу *n*-го порядка.

***Inverse***$[A]$ – находит обратную матрицу $A^{-1}$ для квадратной матрицы $A$.

**Join**$[list_1, list_2, \ldots]$ – объединяет списки $list_1$, $list_2$, … в единую цепочку.

**Join**[*A,rov*] – добавляет к матрице $A$ размером $m \times n$ строку *rov*, которая должна содержать *n* элементов.

***MatrixForm***$[A]$ или $A$//***MatrixForm*** – выводит матрицу $A$ в традиционной форме, а не в виде вложенного списка.

***MatrixPower***$[A, n]$ – возводит матрицу $A$ в степень *n*.

***MatrixRank***$[A]$ – вычисляет ранг матрицы $A$.

***Minors***$[A, k]$ – возвращает список миноров порядка *k* матрицы $A$.

***Norm***[*expr, p*] – возвращает *p*-норму вектора или матрицы *expr*.

***Prepend***[*lst, x*] – создает новый список, добавляя *x* в начало списка *lst*.

***RowReduce***$[C]$ – приводит матрицу $C$ к ступенчатому виду, выполняя элементарные преобразования по строкам.

***Tranpose***$[A]$ – транспонирует матрицу $A$.

***FindRoot*** $[\{eqn_1, eqn_2, ..., eqn_n\}, \{x_1, x_1^{(0)}\}, \{x_2, x_2^{(0)}\}, \ldots, \{x_n, x_n^{(0)}\}]$ – находит численное решение системы *n* уравнений $eqn_1, eqn_2, ..., eqn_n$ с *n* неизвестными $x_1$, $x_2$, … $x_n$, если задано начальное приближение $(x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})$.

***LinearSolve***[ $A, B$ ] – находит решение системы линейных алгебраических уравнений вида $AX = B$.

***NSolve***[{ $\{eqn_1, eqn_2, ...\}, \{x_1, x_2, ...\}$] – находит численное решение заданной системы уравнений.

***NSolve***[ $\{eqn_1, eqn_2, ...\}, \{x_1, x_2, ...\}$, *Reals*] – находит численное решение заданной системы уравнений на множестве действительных чисел.

***Solve***[ $\{eqn_1, eqn_2, ...\}, \{x_1, x_2, ...\}$ ] – находит решение системы уравнений $eqn_1, eqn_2, \ldots$ относительно переменных $x_1, x_2, \ldots$ .

## Примеры решения систем линейных уравнений и исследования погрешности их решений средствами пакета Mathematica

**Пример 4.1.** Решить две системы линейных уравнений $AX = B$ и $AX = B + \Delta B$ (точную и возмущенную), где

$$A = \begin{pmatrix} 1 & 10 \\ 100 & 1001 \end{pmatrix},\ B = \begin{pmatrix} 11 \\ 1101 \end{pmatrix},\ \Delta B = \begin{pmatrix} 0{,}01 \\ 0 \end{pmatrix},\ X = \begin{pmatrix} x \\ y \end{pmatrix}.$$

Найти число обусловленности матрицы $A$ в нормах $\|\cdot\|_1$ и $\|\cdot\|_\infty$. Найти предельную относительную погрешность возмущенной системы. Сравнить полученные решения, оценив их абсолютную и относительную погрешности.

Δ Введем данные матрицы в виде списков:

```
In[1]:= A = {{1, 10}, {100, 1001}}; B = {11, 1101}; dB = {0.01, 0}; X = {x, y};
```

Команда **MatrixForm** выдает результат в матричной форме:

```
In[2]:= A // MatrixForm
        матричная форма

Out[2]//MatrixForm=
        1    10
        100  1001
```

Запишем систему в матричной форме, операцию произведения матриц зададим с помощью точки (уравнение в системе **Mathematica** формируется двойным знаком равенства «==»):

```
In[3]:= A.X == B

Out[3]= {x + 10 y, 100 x + 1001 y} == {11, 1101}
```

Решим систему (или матричное уравнение) с помощью универсальной функции **Solve**:

```
In[4]:= sol = Solve[A.X == B, {x, y}]
        решить уравнения
Out[4]= {{x → 1, y → 1}}
```

Полученное решение является вложенным списком второго уровня. Понизим уровень, т. е. уберем внешние скобки, с помощью функции **Flatten**.

```
In[5]:= Flatten[sol]
        уплостить
Out[5]= {x → 1, y → 1}
```

Решение **X** данной системы сохраним в виде вектора **sol1** – списка с явными значениями переменных:

```
In[6]:= sol1 = X /. Flatten[sol]
            уплостить
Out[6]= {1, 1}
```

Чтобы использовать их отдельно, можно поступить следующим образом:

```
In[7]:= {x, y} = sol1
Out[7]= {1, 1}

In[8]:= x
Out[8]= 1

In[9]:= y
Out[9]= 1
```

Перейдем к решению возмущенной системы $AX = B + \Delta B$. Снова воспользуемся функцией **Solve**. Полученное решение сохраним в виде вектора **sol2**.

```
In[10]:= Clear[x, y]
         очистить
In[11]:= Solve[A.X == B + dB, {x, y}]
         решить уравнения
Out[11]= {{x → 11.01, y → 6.51911×10^-14}}
```

```
In[12]:= sol2 = Flatten[%]
          уплостить
Out[12]= {x → 11.01, y → 6.51911×10^-14}

In[13]:= sol2 = X /. sol2
Out[13]= {11.01, 6.51911×10^-14}
```

Решим возмущенную систему вторым способом, используя функцию **LinearSolve**, предназначенную для реализации методов решения именно линейных систем:

```
In[14]:= sol21 = LinearSolve[A, B + dB]
         решить линейные уравнени
Out[14]= {11.01, 0.}
```

Как видно, решение, полученное с помощью функции **Solve**, из-за погрешности ее методов оказалось неточным. В дальнейших вычислениях будем использовать решение **sol21** возмущенной системы.

Для нахождения числа обусловленности системы найдем обратную матрицу с помощью функции **Inverse**:

```
In[15]:= A1 = Inverse[A]
          обратная мат
Out[15]= {{1001, -10}, {-100, 1}}
```

Вычислим нормы матриц, не используя встроенную функцию системы **Mathematica**.

По определению норма-максимум матрицы $A$ равна $\|A\|_\infty = \max\limits_{1\le i\le n} \sum\limits_{j=1}^{n} |a_{ij}|$.

Для каждой из матриц ($A$ и $A^{-1} = A1$) сформируем вектор, координатами которого являются суммы взятых по модулю элементов каждой строки матрицы. Затем найдем наибольшую из координат, т. е. норму-максимум матриц $A$ и $A^{-1}$.

```
In[16]:= t1 = Table[Sum[Abs[A[[i, j]]], {j, 1, 2}], {i, 1, 2}]
           таблица  абсолютное значение
Out[16]= {11, 1101}

          t2 = Table[Sum[Abs[A1[[i, j]]], {j, 1, 2}], {i, 1, 2}]
           таблица  абсолютное значение
Out[17]= {1011, 101}

In[18]:= norm1 = Max[t1]
           максимум
Out[18]= 1101

          norm2 = Max[t2]
           максимум
Out[19]= 1011
```

Число обусловленности матрицы $A$ в норме-максимум равно

```
In[20]:= condA = norm1 norm2

Out[20]= 1 113 111
```

Аналогичным образом найдем число обусловленности матрицы $A$ в норме-сумме.

```
In[21]:= t11 = Table[Sum[Abs[A[[i, j]]], {j, 1, 2}]]
           таблица  i=1  абсолютное значение

Out[21]= {101, 1011}

         t21 = Table[Sum[Abs[A1[[i, j]]], {j, 1, 2}]]
               таблица  i=1  абсолютное значение

Out[22]= {1101, 11}

In[23]:= norm11 = Max[t11]
                  максимум

Out[23]= 1011

         norm21 = Max[t21]
                  максимум

Out[24]= 1101

In[25]:= condA1 = norm11 norm21

Out[25]= 1 113 111
```

Число обусловленности матрицы $A$ равно 1113111, т. е. является величиной порядка $10^6$. Матрица $A$ плохо обусловлена.

Для анализа реакции решения на возмущение правой части системы найдем вектор ошибки $\Delta X = X - X^*$:

```
In[26]:= deltaX = sol21 - sol1

Out[26]= {10.01, -1.}
```

Абсолютную погрешность решения $\|\Delta X\| = \|X^* - X\|$ вычислим в норме-сумме, используя функцию **Norm**:

```
In[27]:= nd = Norm[deltaX, 1]
           норма

Out[27]= 11.01
```

Тогда относительная погрешность решения возмущенной системы $\delta_X = \|X^* - X\| / \|X^*\|$ в этой норме равна

```
In[28]:= nrd = nd / Norm[sol21, 1]

Out[28]= 1.
```

или 100 %.

Вычислим прогнозируемую предельную относительную погрешность решения возмущенной системы $\delta_X = cond(A)\cdot\delta_B$, где $\delta_B = \|\Delta B\|/\|B+\Delta B\|$:

```
In[29]:= pr = condA1 * Norm[dB, 1] / Norm[B + dB, 1]

Out[29]= 10.0099
```

или 1001 %.

По определению относительная погрешность решения не превосходит его предельную относительную погрешность. Это условие выполнено.

**Пример 4.2.** Решить методом Гаусса систему линейных уравнений $AX = B$, где $A = (a_{ij})$, $B = (b_i)$,

$$a_{ij} = \begin{cases} 6 - i - 2j, & i + j < 5, \\ 75 - 15i, & i + j = 5, \\ i + j - 6, & i + j > 5, \end{cases} \qquad b_i = \frac{i}{2}\cdot(i - 35) + 80, \qquad i = \overline{1,4},\ j = \overline{1,4}.$$

$\Delta$ Введем матрицы $A$ и $B$ с помощью функции **Table** и условного оператора **If**.

```
In[1]:= A = Table[If[i + j > 5, i + j - 6,
                    If[i + j == 5, 75 - 15 i, 6 - i - 2 j]], {i, 4}, {j, 4}];
          B = Table[i/2 (i - 35) + 80, {i, 4}];

In[3]:= {MatrixForm[A], MatrixForm[B]}

Out[3]= {{{3, 1, -1, 60}, {2, 0, 45, 0}, {1, 30, 0, 1}, {15, 0, 1, 2}},
         {{63}, {47}, {32}, {18}}}
```

Организуем прямой ход метода Гаусса с постолбцовым выбором главного элемента. Найдем уравнение с максимальным по модулю элементом в первом столбце. Поменяем это уравнение местами с первым уравнением и исключим переменную $x_1$ из остальных уравнений. Аналогичным образом поступим со вторым столбцом, рассматривая новую систему без первого уравнения и т. д.

Введем обозначения: $n$ – количество уравнений системы и ее неизвестных, $\mathrm{Coef}$ – пока еще пустой список ведущих коэффициентов системы (именно на них производится деление на каждом шаге).

```
In[4]:= n = Length[B];
        длина
        Coef = List[]; (* Список ведущих коэффициентов*)
               список
```

Для реализации прямого хода составим несколько вложенных циклов.

```
In[6]:= For[k = 1, k ≤ n - 1, k++,
        цикл ДЛЯ
  (* Выбрать уравнение p, в котором находится ведущий коэффициент A[p,k]*)
    t = Abs[A[[k, k]]]; p = k;
        абсолютное значение
    For[m = k + 1, m ≤ n, m++, If[Abs[A[[m, k]]] > t, p = m]];
    цикл ДЛЯ                   у··· абсолютное значение
    If[Abs[A[[p, k]]] < $MachineEpsilon, Return[{"Решение не найдено", MatrixForm[A]}]];
    у··· абсолютное значе··· машинная эпсилон вернуть управление матричная форма
  (* Если ведущий коэффициент по модулю меньше константы $MachineEpsilon,
  то закончить вычисления*)
    Coef = Append[Coef, A[[p, k]]]; (* Добавить ведущий коэффициент в список*)
           добавить в конец
    If[k < p, (* Переставить уравнение p и уравнение k*)
    условный оператор
    For[m = k, m ≤ n, m++,
    цикл ДЛЯ
      t = A[[k, m]]; A[[k, m]] = A[[p, m]]; A[[p, m]] = t;
      t = B[[k]]; B[[k]] = B[[p]]; B[[p]] = t;
    For[i = k + 1, i ≤ n, i++, (* Исключить неизвестное из уравнений k+1,...n*)
    цикл ДЛЯ
      t = A[[i, k]] / A[[k, k]]; A[[i, k]] = 0;
      For[j = k + 1, j ≤ n, j++,
      цикл ДЛЯ
        A[[i, j]] = A[[i, j]] - t × A[[k, j]]];
        B[[i]] = B[[i]] - t × B[[k]]]];
```

В результате прямого хода получена система с верхней треугольной матрицей.

```
In[7]:= {MatrixForm[A], MatrixForm[B]}
        матричная форма матричная форма
```

$$\text{Out[7]=}\left\{\begin{pmatrix} 15 & 0 & 1 & 2 \\ 0 & 1 & -\frac{6}{5} & \frac{298}{5} \\ 0 & 0 & \frac{673}{15} & -\frac{4}{15} \\ 0 & 0 & 0 & -\frac{1202597}{673} \end{pmatrix}, \begin{pmatrix} 18 \\ \frac{297}{5} \\ \frac{223}{5} \\ -\frac{1202597}{673} \end{pmatrix}\right\}$$

Далее найдем решение системы, выполняя обратный ход метода Гаусса. Введем нулевой вектор $X$.

```
X = Table[0, {n}];
    таблица значений
```

Из последнего уравнения найдем $x_4$, затем из третьего уравнения – $x_3$ и т. д. Все вычисления организуем в цикле.

```
In[9]:= X[[n]] = B[[n]] / A[[n, n]];

In[10]:= For[i = n - 1, i ≥ 1, i--,
          цикл ДЛЯ
           X[[i]] = (B[[i]] - ∑_{j=i+1}^{n} A[[i, j]] × X[[j]]) / A[[i, i]]];
```

Выведем решение $X$ и вектор невязки $R = AX - B$.

```
In[11]:= {X, A.X - B}

Out[11]= {{1, 1, 1, 1}, {0, 0, 0, 0}}
```

Так как вектор невязки является нулевым, то получено точное решение (1, 1, 1, 1).

# Лабораторная работа 3

## Решение систем линейных алгебраических уравнений

**1.** Даны матрицы $A=(a_{ij})$ и $B=(b_i)$, $i=\overline{1,7}$, $j=\overline{1,7}$. Используя средства пакета **Mathematica** (функции **Norm**, **Inverse**, **LinearSolve**):

а) найти число обусловленности матрицы $A$ в норме-максимум $\|\cdot\|_\infty$;

б) решить точную систему линейных уравнений $AX=B$;

в) решить три возмущенные системы вида $AX=B+\Delta B$, увеличив значение правой части **только последнего уравнения** системы $AX=B$ последовательно на 0,01; 0,1 и на 1 %;

г) найти прогнозируемую предельную относительную погрешность решения каждой возмущенной системы;

д) найти относительную погрешность решения каждой возмущенной системы; сделать вывод о зависимости относительной погрешности от величины возмущения и числа обусловленности матрицы $A$.

Выполнить задание для двух случаев:

1.1. $a_{ij}=\begin{cases}1, & i>j,\\ i+1, & i=j,\\ 2, & i<j,\end{cases}$ $b_i=2ki-i^2$; 1.2. $a_{ij}=\dfrac{1}{i+j-1}$, $b_i=3i-2k$,

где $i=\overline{1,7}$, $j=\overline{1,7}$, $k$ – номер вашего варианта.

**2.** Решить методом прогонки трехдиагональную систему, составить таблицу прогоночных коэффициентов $L_i$, $M_i$, $i=\overline{1,5}$. Выполнить проверку полученного решения с помощью функции **LinearSolve**.

2.1. $\begin{cases}4x_1-x_2=5,\\ 2x_1+10x_2+4x_3=18,\\ 5x_2+20x_3+3x_4=-50,\\ 4x_3+10x_4-x_5=30,\\ 2x_4-3x_5=-2.\end{cases}$ 2.2. $\begin{cases}4x_1+x_2=7,\\ x_1+11x_2+2x_3=24,\\ 3x_2+14x_3-6x_4=-37,\\ 2x_3+19x_4+7x_5=-44,\\ x_4+5x_5=26.\end{cases}$

2.3. $\begin{cases}5x_1-2x_2=17,\\ x_1-12x_2+4x_3=23,\\ 2x_2+11x_3+3x_4=32,\\ -7x_3+10x_4-x_5=25,\\ x_4-11x_5=-7.\end{cases}$ 2.4. $\begin{cases}7x_1+3x_2=-11,\\ 2x_1+16x_2-4x_3=0,\\ 5x_2-12x_3+x_4=-31,\\ 3x_3+21x_4-6x_5=-21,\\ 4x_4+7x_5=35.\end{cases}$

2.5. $\begin{cases} 4x_1 + x_2 = 11, \\ -2x_1 + 14x_2 + 5x_3 = 33, \\ 3x_2 - 17x_3 + 2x_4 = 28, \\ x_3 + 8x_4 - 3x_5 = 19, \\ 3x_4 + 5x_5 = -17. \end{cases}$

2.6. $\begin{cases} 5x_1 - 3x_2 = -10, \\ 3x_1 + 19x_2 + 4x_3 = -2, \\ -x_2 + 8x_3 + 2x_4 = 14, \\ 2x_3 - 11x_4 - 5x_5 = -26, \\ 2x_4 + 9x_5 = -3. \end{cases}$

2.7. $\begin{cases} 5x_1 + 2x_2 = -5, \\ 4x_1 - 9x_2 - 2x_3 = 45, \\ x_2 + 13x_3 + 2x_4 = 27, \\ 2x_3 + 11x_4 + 5x_5 = 34, \\ -x_4 - 7x_5 = 4. \end{cases}$

2.8. $\begin{cases} 7x_1 + 2x_2 = -14, \\ x_1 + 12x_2 + 3x_3 = 7, \\ 2x_2 - 9x_3 - 4x_4 = -23, \\ x_3 + 13x_4 + 2x_5 = -8, \\ 3x_4 + 15x_5 = 12. \end{cases}$

2.9. $\begin{cases} 7x_1 + 4x_2 = 1, \\ -2x_1 + 14x_2 - 7x_3 = 23, \\ 3x_2 + 21x_3 + 5x_4 = 17, \\ x_3 - 10x_4 + 4x_5 = 21, \\ 2x_4 - 9x_5 = -4. \end{cases}$

2.10. $\begin{cases} 8x_1 + 3x_2 = 5, \\ 3x_1 - 17x_2 - 4x_3 = 11, \\ x_2 + 7x_3 + 2x_4 = 22, \\ -2x_3 + 15x_4 + 4x_5 = 13, \\ 3x_4 + 11x_5 = -19. \end{cases}$

2.11. $\begin{cases} 20x_1 + 7x_2 = 20, \\ 3x_1 + 12x_2 + 8x_3 = 11, \\ 10x_2 + 14x_3 + 3x_4 = 14, \\ 5x_3 + 11x_4 + 3x_5 = 8, \\ 10x_4 + 11x_5 = 11. \end{cases}$

2.12. $\begin{cases} 13x_1 - 12x_2 = 13, \\ 5x_1 + 7x_2 - x_3 = 4, \\ -3x_2 + 7x_3 - x_4 = 7, \\ 9x_3 + 14x_4 - x_5 = 8, \\ x_4 + 2x_5 = 2. \end{cases}$

2.13. $\begin{cases} 14x_1 - x_2 = -1, \\ x_1 + 5x_2 - 3x_3 = 5, \\ 7x_2 + 9x_3 - x_4 = 6, \\ 11x_3 + 15x_4 + 2x_5 = 15, \\ -3x_4 + 4x_5 = -3. \end{cases}$

2.14. $\begin{cases} 7x_1 - 5x_2 = 2, \\ x_1 + 10x_2 - 4x_3 = 5, \\ 8x_2 + 12x_3 + 2x_4 = 22, \\ 10x_3 + 13x_4 - x_5 = 22, \\ x_4 - 2x_5 = -1. \end{cases}$

2.15. $\begin{cases} 8x_1 + 2x_2 = 6, \\ x_1 + 11x_2 - x_3 = -9, \\ 2x_2 + 4x_3 - x_4 = 3, \\ x_3 + 5x_4 - x_5 = -5, \\ 5x_4 + 7x_5 = 2. \end{cases}$ 2.16. $\begin{cases} 2x_1 + x_2 = 4, \\ x_1 + 3x_2 + x_3 = 10, \\ x_2 + 4x_3 - x_4 = 10, \\ 2x_3 - 4x_4 + x_5 = -5, \\ x_4 - 2x_5 = -6. \end{cases}$

**3.** Решить систему $n$-го порядка $AX = B$ методом Якоби и методом Зейделя с точностью $\varepsilon = 10^{-3}$ при $n = 10$ и $n = 20$. Сравнить число итераций, необходимых для достижения точности $\varepsilon$ этими методами. Здесь $A = (a_{ij})$ – матрица с диагональным преобладанием, $B = (b_i)$ – вектор-столбец,

$$a_{ij} = \begin{cases} 1, & i \neq j, \\ 2n, & i = j, \end{cases} \qquad b_i = (2n-1)i + \frac{n(n+1)}{2} + (3n-1)(k-1),$$

где $i = \overline{1, n}$, $j = \overline{1, n}$, $k$ – номер вашего варианта.

# 5. Интерполяция и среднеквадратичное приближение функций

Задача ***приближения***, или ***аппроксимирования***, ***функции*** $f(x)$ на некотором числовом множестве $X$ состоит в ее замене функцией определенного вида $Q(x)$ (чаще всего – многочленом), так чтобы значения функций $f(x)$ и $Q(x)$ мало отличались на множестве $X$. Эта задача возникает в следующих случаях:

1) функция $f(x)$ задана таблично, т. е. известны ее значения в конечном числе точек $x_i \in X$, $i = \overline{0,n}$;

2) функция $f(x)$ задана аналитически, но имеет слишком сложный вид.

Далее будем рассматривать только задачи приближения ***многочленами***.

В зависимости от того, что считать мерой отклонения значений двух функций, выделяют различные виды аппроксимации.

## Интерполяция

Пусть на отрезке $[a, b]$ даны $n+1$ значений аргумента $x_0 = a$, $x_1$, $x_2$, …, $x_n = b$ (***узлы интерполяции***) и соответствующие значения функции $f(x)$:

$$y_0 = f(x_0),\ y_1 = f(x_1),\ \ldots,\ y_n = f(x_n).$$

Будем считать, что функция $f(x)$ и многочлен $Q(x)$ *мало отличаются* друг от друга на отрезке $[a, b]$, если их значения совпадают в узлах интерполяции $x_i$ ($i = \overline{0,n}$), т. е.

$$Q(x_0) = y_0,\ Q(x_1) = y_1,\ \ldots,\ Q(x_n) = y_n.$$

Задача построения такого многочлена $Q(x)$ называется ***задачей интерполирования***. При этом $Q(x)$ называется ***интерполяционным многочленом***.

**Теорема.** Существует один и только один многочлен $Q(x)$ степени не выше, чем $n$, который решает задачу интерполирования функции $f(x)$ по $n+1$ узлам, т. е. удовлетворяет условиям

$$Q(x_i) = f(x_i)\ \ (i = \overline{0,n}).$$

Рассмотрим различные варианты построения интерполяционного многочлена.

**1. Интерполяционная формула Лагранжа.** ***Интерполяционный многочлен Лагранжа*** имеет вид

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

Введем обозначение:

$$\Pi_{n+1}(x)=(x-x_0)(x-x_1)...(x-x_n).$$

Тогда интерполяционную формулу Лагранжа можно переписать в виде

$$L_n(x)=\sum_{i=0}^{n} y_i \frac{\Pi_{n+1}(x)}{(x-x_i)\Pi'_{n+1}(x_i)}.$$

***Абсолютная погрешность*** $|R_n(x)|=|f(x)-L_n(x)|$ интерполяционной формулы Лагранжа:

$$|R_n(x)|\le \frac{M_{n+1}}{(n+1)!}\cdot|\Pi_{n+1}(x)|,$$

где $M_{n+1}=\max\limits_{[a,b]}|f^{(n+1)}(x)|$.

Погрешность $|R_n(x)|$ зависит от двух множителей $M_{n+1}$ и $\Pi_{n+1}$. Множитель $M_{n+1}$ полностью определяется свойствами функции $f(x)$ и регулированию не поддается. Второй множитель $\Pi_{n+1}$ можно изменять, т. к. он зависит от выбора узлов интерполирования.

***Погрешность*** интерполирования многочленом степени *n* будет ***минимальной***, если в качестве узлов интерполяции брать точки

$$x_i=\frac{a+b}{2}+\frac{b-a}{2}\cdot t_i,$$

где $t_i$ – корни многочлена Чебышёва $T_{n+1}(t)=\cos\big((n+1)\arccos t\big)$, т. е.

$$t_i=\cos\frac{\pi(2i+1)}{2n+2},\ i=\overline{0,n}.$$

Эти точки называются ***чебышёвскими узлами интерполяции***.

В случае интерполяции по чебышёвским узлам ее максимальная погрешность определяется неравенством

$$\max_{[a,b]}|R_n(x)|\le \frac{M_{n+1}}{(n+1)!}\cdot\frac{(b-a)^{n+1}}{2^{2n+1}}.$$

Такая оценка называется ***наилучшей равномерной оценкой погрешности интерполяции*** (ее невозможно улучшить для заданного *n*).

Если аналитическое выражение для функции $f(x)$ неизвестно, то оценка погрешности является невозможной.

Недостатком интерполяционной формулы Лагранжа является то, что каждое из слагаемых (они представляет собой многочлены *n*-й степени) зависит от всех узлов интерполяции $x_i$. Поэтому при увеличении числа точек $x_i$ и, следовательно, степени многочлена каждое слагаемое формулы Лагранжа нужно вычислять заново.

**2. Интерполяционные формулы Ньютона для равноотстоящих узлов.** Предположим, что все узлы интерполяции на отрезке $[a, b]$ равноудалены друг от друга:

$$x_0 = a,\ x_1 = x_0 + h,\ x_2 = x_1 + h = x_0 + 2h, \ldots,\ x_n = x_0 + nh = b,$$

т. е. $x_i = x_0 + ih,\ i = \overline{0,n}$. При этом $h$ называется ***шагом*** интерполяции, а ***узлы*** $x_i$ называются ***равноотстоящими***. Пусть известны значения функции $f(x)$ в этих узлах: $f(x_i) = y_i,\ i = \overline{0,n}$.

***Конечной разностью первого порядка*** функции $f(x)$ в точке $x_\text{i}$ называется число, определяемое равенством

$$\Delta y_i = y_{i+1} - y_i,\ \ i = \overline{0, n-1}.$$

Из конечных разностей первого порядка образуют ***конечные разности второго порядка***:

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i = y_{i+2} - 2y_{i+1} + y_i,\ \ i = \overline{0, n-2}.$$

Аналогичным образом определяются ***конечные разности третьего и более высоких порядков***:

$$\Delta^3 y_i = \Delta^2 y_{i+1} - \Delta^2 y_i,\ \ i = \overline{0, n-3}, \ldots,$$

$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i,\ \ i = \overline{0, n-k},\ k = \overline{1, n}.$$

2.1. ***Первая интерполяционная формула Ньютона*** имеет вид

$$P_n(x) = y_0 + t\,\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \ldots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0,$$

где $t = \dfrac{x - x_0}{h}$, $\Delta^k y_0$ – конечная разность порядка *k* в точке $x_0$.

Эта формула также называется ***интерполяционной формулой Ньютона для интерполирования вперед***. Ей удобно пользоваться для вычисления значе-

ний функции $f(x)$ вблизи точки $x_0 = a$ на отрезке $[a, b]$, а также для ***экстраполирования назад***, т. е. в точках, не принадлежащих отрезку $[a, b]$, но достаточно близких к точке $x_0 = a$.

***Абсолютная погрешность*** $|R_n(x)| = |f(x) - P_n(x)|$ первой интерполяционной формулы Ньютона:

$$|R_n(x)| \approx \frac{|t(t-1)...(t-n+1)(t-n)|}{(n+1)!}\Delta^{n+1}y_0.$$

2.2. ***Вторая интерполяционная формула Ньютона*** имеет вид

$$P_n(x) = y_n + q\Delta y_{n-1} + \frac{q(q+1)}{2!}\Delta^2 y_{n-2} + ... + \frac{q(q+1)...(q+n-1)}{n!}\Delta^n y_0,$$

где $q = \frac{x - x_n}{h}$; $\Delta^k y_i$ – конечная разность порядка $k$ в точке $x_i$.

Эта формула также называется ***интерполяционной формулой Ньютона для интерполирования назад***. Ее используют для интерполирования назад и ***экстраполирования вперед***, т. е. для вычисления значений функции $f(x)$ вблизи точки $x_n = b$, где $|q|$ имеет малые значения.

***Абсолютная погрешность*** второй интерполяционной формулы Ньютона:

$$|R_n(x)| \approx \frac{|q(q+1)...(q+n)|}{(n+1)!}\Delta^{n+1}y_0.$$

Заметим, что для оценки погрешности при интерполяции многочленами Ньютона $n$-й степени надо взять дополнительный узел и вычислить слагаемое $(n+1)$-й степени. Его абсолютное значение приближенно равно погрешности соответствующей интерполяционной формулы.

**3. Интерполяционная формула Ньютона для неравноотстоящих узлов.** Предположим, что узлы интерполяции $x_0 = a$, $x_1$, $x_2$, …, $x_n = b$ – ***неравноотстоящие***, т. е. расстояния между ними неравные. Пусть известны значения функции $f(x)$ в этих узлах: $f(x_i) = y_i$, $i = \overline{0,n}$.

***Разделенной разностью первого порядка*** функции $f(x)$ по точкам $x_i$ и $x_{i+1}$ называется выражение

$$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{\Delta y_i}{\Delta x_i}, \quad i = \overline{0,n-1}.$$

***Разделенной разностью второго порядка*** функции $f(x)$ по точкам $x_i$, $x_{i+1}$ и $x_{i+2}$ называется выражение

$$f(x_i, x_{i+1}, x_{i+2}) = \frac{f(x_{i+1}, x_{i+2}) - f(x_i, x_{i+1})}{x_{i+2} - x_i}, \quad i = \overline{0, n-2}.$$

Разделенные разности более высоких порядков определяются аналогичным образом с помощью рекуррентных формул. ***Разделенная разность порядка k*** имеет вид

$$f(x_i, x_{i+1}, ..., x_{i+k}) = \frac{f(x_{i+1}, x_{i+2}, ..., x_{i+k}) - f(x_i, x_{i+1}, ..., x_{i+k-1})}{x_{i+k} - x_i}, \quad i = \overline{0, n-k}.$$

Вычисление разделенных разностей можно оформить в виде таблицы (табл. 5.1).

Таблица 5.1

| $x$ | $f(x)$ | Разделенные разности | | | |
|---|---|---|---|---|---|
| | | 1 | 2 | … | $n$ |
| $x_0$ | $f(x_0)$ | $f(x_0, x_1)$ | $f(x_0, x_1, x_2)$ | … | $f(x_0, x_1, ..., x_n)$ |
| $x_1$ | $f(x_1)$ | $f(x_1, x_2)$ | $f(x_1, x_2, x_3)$ | … | – |
| $x_2$ | $f(x_2)$ | $f(x_2, x_3)$ | $f(x_2, x_3, x_4)$ | … | – |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋮ |
| $x_{n-2}$ | $f(x_{n-2})$ | $f(x_{n-2}, x_{n-1})$ | $f(x_{n-2}, x_{n-1}, x_n)$ | … | – |
| $x_{n-1}$ | $f(x_{n-1})$ | $f(x_{n-1}, x_n)$ | – | … | – |
| $x_n$ | $f(x_n)$ | – | – | … | – |

Заметим, что добавление нового узла не меняет уже вычисленные разделенные разности. Таблица будет просто дополнена новым столбцом и новыми значениями разделенных разностей внизу ее старых столбцов.

***Интерполяционная формула Ньютона для неравноотстоящих узлов*** имеет вид

$$P_n(x) = f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + \ldots +$$
$$+ f(x_0, x_1, \ldots, x_n)(x - x_0)(x - x_1)(x - x_2)\ldots(x - x_{n-1}).$$

Ее коэффициентами являются разделенные разности, расположенные в первой строке таблицы.

Данная формула используется для ***интерполирования вперед*** или ***экстраполирования назад***, т. е. для вычисления значений функции $f(x)$ вблизи точки $x_0 = a$.

***Погрешность*** формулы можно оценить, добавив еще один узел $x_{n+1}$:

$$|R_n(x)| \approx |f(x_0, x_1, \ldots, x_n, x_{n+1})(x - x_0)(x - x_1)\ldots(x - x_{n-1})(x - x_n)|.$$

Равносильный вариант многочлена Ньютона для ***интерполирования назад*** или ***экстраполирования вперед*** можно записать, воспользовавшись нижней (или побочной) диагональю в таблице разделенных разностей:

$$P_n(x) = f(x_n) + f(x_{n-1}, x_n)(x - x_n) + f(x_{n-2}, x_{n-1}, x_n)(x - x_n)(x - x_{n-1}) + \ldots +$$
$$+ f(x_0, x_1, \ldots, x_n)(x - x_n)(x - x_{n-1})(x - x_{n-2})\ldots(x - x_1).$$

Указанные формулы Ньютона являются универсальными. Их можно применять при любом расположении узлов интерполирования.

Поскольку интерполяционный многочлен для функции $f(x)$ по одной и той же системе точек $x_i$ $(i = \overline{0,n})$ единственен, то многочлены, построенные по различным формулам Ньютона и Лагранжа, совпадают. Преимущество многочленов Ньютона перед многочленом Лагранжа состоит в том, что при добавлении нового узла в многочленах Ньютона появляется только дополнительное слагаемое, а все предыдущие слагаемые не изменяются.

## Интерполяция кубическими сплайнами

Если функцию $f(x)$ нужно приблизить функцией $Q(x)$ на отрезке $[a, b]$, длина которого велика, или если функция $f(x)$ не является достаточно гладкой на $[a, b]$, то в качестве $Q(x)$ не имеет смысла использовать многочлены высоких степеней, т. к. использование большого числа узлов интерполяции на всем отрезке $[a, b]$ не всегда дает удовлетворительную точность приближения и значительно увеличивает количество вычислительных действий. В этом случае используют *кусочно-полиномиальную аппроксимацию* функции $f(x)$. Она состоит в том, что отрезок $[a, b]$ разбивают на частичные отрезки, на каждом из которых функцию $f(x)$ заменяют многочленами одной и той же невысокой степени.

Пусть функция $f(x)$ определена на отрезке $[a, b]$, отрезок $[a, b]$ разбит на $n$ частичных отрезков $[x_{k-1}, x_k]$ $(k = \overline{1,n})$ точками $a = x_0 < x_1 < x_2 < \ldots < x_n = b$.

***Сплайном*** $S_m(x)$ степени $m$ называется определенная на отрезке $[a, b]$ $l$ раз непрерывно-дифференцируемая функция, которая на каждом частичном отрезке $[x_{k-1}, x_k]$ $(k = \overline{1,n})$ является многочленом степени $m$. При этом точки $x_k$ $(k = \overline{0,n})$ называются ***узлами сплайна***.

Разность $d$ между степенью сплайна $m$ и показателем ее гладкости $l$ (т. е. порядком его наивысшей непрерывной производной) называется

***дефектом сплайна***: $d = m - l$. Для сплайна степени $m$ дефекта $d$ используется обозначение $S_m^d(x)$.

Если сплайн $S_m^d(x)$ строится для функции $f(x)$ по системе точек $x_k$ $(k = \overline{0,n})$ так, чтобы выполнялись условия $S_m^d(x_k) = f(x_k)$, то он называется ***интерполяционным сплайном*** для функции $f(x)$.

Наиболее часто применяется интерполяционный кубический сплайн (т. е. степени 3) дефекта 1. Рассмотрим его подробно.

Пусть известны значения функции $f(x)$ в точках $x_k$ $(a = x_0 < x_1 < x_2 < \ldots < x_n = b)$: $f(x_k) = y_k$, $k = \overline{0,n}$. И пусть узлы сплайна совпадают с узлами интерполяции $x_k$. Обозначим расстояние между соседними узлами $h_k = x_k - x_{k-1}$, $k = \overline{1,n}$.

***Интерполяционным кубическим сплайном дефекта 1*** для функции $f(x)$ на отрезке $[a, b]$ называется функция $S_3^1(x)$, которая на каждом частичном отрезке $[x_{k-1}, x_k]$ $(k = \overline{1,n})$ является многочленом третьей степени $g_k(x)$, т. е. имеет вид

$$S_3^1(x) = \left\{ g_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3,\ x \in [x_{k-1}, x_k] \right\}_{k=1}^{n},$$

и удовлетворяет условиям:

1) $S_3^1(x)$ дважды непрерывно дифференцируема на $[a, b]$;

2) $S_3^1(x_k) = y_k$, $k = \overline{0,n}$ (условия интерполяции);

3) $S_3^{1\,\prime\prime}(a) = 0$, $S_3^{1\,\prime\prime}(b) = 0$ (краевые условия).

Здесь краевыми условиями являются условия нулевой кривизны на концах отрезка. В общем случае краевые условия могут задаваться различными способами.

Определенный таким образом интерполяционный сплайн также называют ***естественным*** или ***чертежным сплайном***. Его коэффициенты $a_k$, $b_k$, $d_k$ $(k = \overline{1,n})$ находят по формулам:

$$a_k = y_k,$$

$$b_k = \frac{y_k - y_{k-1}}{h_k} + \frac{2}{3} h_k c_k + \frac{1}{3} h_k c_{k-1},$$

$$d_k = \frac{c_k - c_{k-1}}{3h_k},$$

где неизвестные $c_k$ являются решением системы линейных уравнений

$$\begin{cases} h_{k-1}c_{k-2} + 2(h_{k-1} + h_k)c_{k-1} + h_k c_k = 3\left(\dfrac{y_k - y_{k-1}}{h_k} - \dfrac{y_{k-1} - y_{k-2}}{h_{k-1}}\right), \quad k = \overline{2,n}, \\ c_0 = 0, \quad c_n = 0. \end{cases}$$

Эта система является линейной системой с трехдиагональной матрицей. Ее решение можно найти, например, методом прогонки. Поскольку матрица системы имеет диагональное преобладание, то метод прогонки сходится, т. е. система имеет единственное решение.

Таким образом, по заданным в узлах интерполяции значениям функции $f(x)$ при заданных краевых условиях $S_3^{1\,''}(a) = 0$, $S_3^{1\,''}(b) = 0$ можно построить единственный кубический сплайн дефекта 1, интерполирующий функцию $f(x)$. Если функция $f(x)$ четырежды непрерывно дифференцируема на отрезке $[a, b]$, то для любого фиксированного $n$ существует такая постоянная $C > 0$, что справедлива ***оценка погрешности интерполяции***

$$\left| f(x) - S_3^1(x) \right| \le C \cdot \Delta^4 \quad \text{для любого } x \in [a, b],$$

где $\Delta = \max\limits_{1 \le k \le n}(x_k - x_{k-1})$ – диаметр разбиения отрезка $[a, b]$.

## Среднеквадратичное приближение функций

Пусть значения функции $f(x)$ известны в точках $x_0 = a$, $x_1$, $x_2, \ldots, x_n = b$ отрезка $[a, b]$: $f(x_i) = y_i$, $i = \overline{0,n}$. Другими словами, функция $f(x)$ задана на дискретном множестве $X = \{x_0, x_1, \ldots, x_n\}$.

***Квадратичным отклонением*** функции $Q(x)$ от функции $f(x)$ на дискретном множестве $X$ называется величина $S$, равная

$$S = \sum_{i=0}^{n} (Q(x_i) - f(x_i))^2 .$$

Построим многочлен $m$-й степени $Q_m(x) = a_0 + a_1 x + \ldots + a_m x^m$, такой чтобы его *отклонение* от аппроксимируемой функции $f(x)$ на множестве $X$ было наименьшим.

Если при аппроксимации в качестве меры близости двух функций используется *квадратичное отклонение*, то говорят, что функция $Q_m(x)$ построена с помощью ***метода наименьших квадратов*** или выполнено ***квадратичное*** (***среднеквадратичное***) ***приближение*** функции $f(x)$.

Найдем коэффициенты $a_0$, $a_1$, …, $a_m$ многочлена $Q_m(x)$ так, чтобы величина $S$ его квадратичного отклонения от значений функции $f(x)$ на множестве $X = \{x_0, x_1, ..., x_n\}$ принимала наименьшее значение:

$$S = \sum_{i=0}^{n} \left(Q_m(x_i) - y_i\right)^2 \to \min.$$

Считая $S$ функцией $m$ переменных $a_0$, $a_1$, …, $a_m$, т. е. $S = S(a_0, a_1, \ldots, a_m)$, запишем необходимые условия ее локального минимума:

$$\frac{\partial S}{\partial a_j} = 2\sum_{i=0}^{n} \left(a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i\right) \cdot x_i^j = 0, \quad j = \overline{0,m}.$$

В результате получим систему $m+1$ линейных уравнений относительно $m+1$ неизвестных $a_0$, $a_1$, …, $a_m$:

$$\begin{cases} a_0 \sum\limits_{i=0}^{n} 1 + a_1 \sum\limits_{i=0}^{n} x_i + \ldots + a_m \sum\limits_{i=0}^{n} x_i^m = \sum\limits_{i=0}^{n} y_i, \\ a_0 \sum\limits_{i=0}^{n} x_i + a_1 \sum\limits_{i=0}^{n} x_i^2 + \ldots + a_m \sum\limits_{i=0}^{n} x_i^{m+1} = \sum\limits_{i=0}^{n} y_i x_i, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_0 \sum\limits_{i=0}^{n} x_i^m + a_1 \sum\limits_{i=0}^{n} x_i^{m+1} + \ldots + a_m \sum\limits_{i=0}^{n} x_i^{2m} = \sum\limits_{i=0}^{n} y_i x_i^m. \end{cases}$$

Можно доказать, что если среди точек $x_0$, $x_1$, $x_2$, …, $x_n$ нет совпадающих и $m \le n$, то определитель этой системы отличен от нуля, и, следовательно, система имеет единственное решение. Многочлен $Q_m(x)$ с такими коэффициентами $a_0$, $a_1$, …, $a_m$ будет иметь наименьшее квадратичное отклонение $S$ от функции $f(x)$ на множестве $X$. Он также называется многочленом (полиномом) *наилучшего среднеквадратичного* (или *квадратичного*) *приближения*.

Если $m = n$, то аппроксимирующий многочлен $Q_m(x)$ совпадает с интерполяционным многочленом для узлов $x_0$, $x_1$, $x_2$,…, $x_n$, при этом квадратичное отклонение $S = 0$.

Если $m > n$, то задачу сводят к случаю $m = n$, т. к. не имеет смысла строить многочлен высокой степени $m$.

## Основные функции пакета Mathematica, используемые для приближения функций

***InterpolatingPolynomial***[*data, x*] – строит интерполяционный многочлен по формуле Ньютона от переменной *x* для функции, заданной таблицей значений *data*. Данные *data* должны иметь вид $\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\}\}$ или $\{y_1, y_2, \ldots, y_n\}$, во втором случае считается, что *x* последовательно принимает значения 1, 2, …, *n*. Для упрощения результата рекомендуется применять функцию ***Expand***.

***Interpolation***[*data*] – строит интерполирующую функцию для функции $y=f(x)$ по списку ее значений *data*. Данные *data* должны иметь вид $\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\}\}$ или $\{y_1, y_2, \ldots, y_n\}$, во втором случае считается, что *x* последовательно принимает значения 1, 2, …, *n*. Поддерживает опцию ***Method***, для которой возможны варианты ***"Spline"*** (сплайн-интерполяция) и **"Hermite"** (интерполяция по Эрмиту), например, ***Interpolation***[*data*, ***Method->"Spline"***]. Результат возвращает в виде объекта ***InterpolatingFunction***, который в системе **Mathematica** может быть использован как любая другая обычная функция.

***SplineFit***[*data, type*] – выполняет интерполяцию функции $f(x)$ сплайном вида *type* по списку ее значений *data*. Для того чтобы функция ***SplineFit*** была доступна, предварительно необходимо загрузить пакет сплайн-интерполяции с помощью команды ***Needs*****["*Splines*`"]**. Данные *data* должны иметь вид $\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\}\}$. Возможные типы сплайна – ***Cubic***, ***Bezier***, ***CompositeBezier***. Например, ***SplineFit***[*data*,***Cubic***] – интерполяционный кубический сплайн. Результат возвращает в виде объекта ***SplineFunction***[*type*,*domain*,*interval*], который дает параметрическое представление интерполяционной кривой в виде {*x*[*t*], *y* [*t*]}. Параметр принимает значения из области *domain*. Если дать определенное значение параметра в качестве аргумента этого объекта, то он возвращает координаты соответствующей точки кривой.

***Fit***[*data*,$\{f_1, \ldots, f_m\}$,*x*] – строит с помощью метода наименьших квадратов (МНК) аппроксимирующую функцию для функции $f(x)$ по списку ее значений *data* в виде линейной комбинации функций $f_1$, …, $f_m$ относительно переменной *x*. Данные *data* должны иметь вид $\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\}\}$ или $\{y_1, y_2, \ldots, y_n\}$, во втором случае считается, что *x* последовательно принимает значения 1, 2, …, *n*. Для аппроксимации многочленом вида $Q_m(x) = a_0 + a_1 x + \ldots + a_m x^m$ в качестве функций $f_1$, $f_2$, …, $f_m$ следует указать 1, $x$, …, $x^m$.

***ChebyshevT***[*n*,*t*] – многочлен Чебышёва первого рода $T_n(t)$ *n*-й степени относительно переменной *t*.

***Piecewise***[{{$f_1$, $cond_1$},{$f_2$, $cond_2$},…,{$f_n$, $cond_n$}}] – представляет

кусочно-заданную функцию, равную выражениям $f_1$, $f_2$, …, $f_n$ в областях, определяемых соответствующими условиями $cond_1$, $cond_2$,..., $cond_n$.

***FindMaximum***$[\{f[x], a \le x \le b\}, x]$ – находит локальный максимум функции $f(x)$ на отрезке $[a, b]$.

***ParametricPlot***$[\{x[t], y[t]\}, \{t, t_{\min}, t_{\max}\}]$ – строит параметрически заданную кривую $x = x(t)$, $y = y(t)$ при $t$, принимающем значения от $t_{\min}$ до $t_{\max}$.

## Примеры интерполяции и аппроксимации функций средствами пакета Mathematica

**Пример 5.1.** Используя значения функции

$$f(x) = \frac{5x^2 - 4x\,\mathrm{arctg}(3x+2) + \ln 2}{2x^2 + 7}$$

в равноотстоящих узлах $x_i$ отрезка $[-2, 6]$ при разбиении его на шесть частей, выполнить следующие действия:

а) создать таблицу конечных разностей функции $f(x)$ по точкам $(x_i, f(x_i))$;

б) построить первый интерполяционный многочлен Ньютона для $f(x)$, проиллюстрировать графически;

в) интерполировать функцию $f(x)$ с помощью встроенных функций **InterpolatingPolynomial** и **Interpolation** пакета **Mathematica**, проиллюстрировать графически;

г) найти значения функции $f(x)$ и построенных интерполяционных функций в точке $x = 2{,}28$.

Δ Введем функцию $f(x)$, границы отрезка $a$ и $b$, количество частей разбиения отрезка $n$, вычислим шаг $h$:

```
In[1]:= f[x_] = (5 x^2 - 4 x ArcTan[3 x + 2] + Log[2]) / (2 x^2 + 7);

In[2]:= a = -2; b = 6; n = 6; h = (b - a) / n;
```

Составим таблицу значений функции $f(x)$ в равноотстоящих узлах:

```
In[3]:= data = N[Table[{a + i h, f[a + i h]}, {i, 0, n}]]
        таблица значений
Out[3]= {{-2., 0.67244}, {-0.666667, 0.369554}, {0.666667, -0.0786098},
         {2., 0.608108}, {3.33333, 1.24608}, {4.66667, 1.61062}, {6., 1.82523}}
```

а) с помощью функции **Array** пакета **Mathematica** создадим массив dif размера $(n + 1) \times (n + 1)$ с начальными индексами (0, 0), в котором будем хранить конечные разности. С учетом введенных обозначений *dif* [*i*, *k*] – это конечная разность порядка *k* в точке $x_i$.

```
In[4]:= Array[dif, {n + 1, n + 1}, {0, 0}];
        массив
```

Определим элементы массива dif , которые соответствуют пустым клеткам таблицы:

```
In[5]:= For[k = 1, k ≤ n, k++,
        цикл ДЛЯ
          For[i = n, i ≥ n - k, i--, dif[i, k] = ""]];
          цикл ДЛЯ
```

Заполним элементы первого столбца массива *dif* [*i*, 0] – конечные разности нулевого порядка. Им соответствуют значения $f(x_i)$, которые хранятся во втором столбце таблицы данных data, т. е. *data* [[*i* + 1,2]]. Затем заполним элементы массива dif с конечными разностями остальных порядков:

```
In[6]:= For[i = 0, i ≤ n, i++, dif[i, 0] = data[[i + 1, 2]]];
        цикл ДЛЯ
        For[k = 1, k ≤ n, k++,
        цикл ДЛЯ
          For[i = 0, i ≤ n - k, i++,
          цикл ДЛЯ
            dif[i, k] = dif[i + 1, k - 1] - dif[i, k - 1]]];
```

Выведем построенную таблицу конечных разностей:

```
In[8]:= tab = Array[dif, {n + 1, n + 1}, {0, 0}];
            массив

In[9]:= PaddedForm[TableForm[tab], {6, 5}]
        форма числ··· табличная форма
```

Out[9]//PaddedForm=

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.67244 | -0.30289 | -0.14528 | 1.28016 | -2.46378 | 3.42271 | -4.03342 |
| 0.36955 | -0.44816 | 1.13488 | -1.18362 | 0.95893 | -0.61071 | |
| -0.07861 | 0.68672 | -0.04874 | -0.22470 | 0.34821 | | |
| 0.60811 | 0.63797 | -0.27344 | 0.12352 | | | |
| 1.24608 | 0.36453 | -0.14992 | | | | |
| 1.61062 | 0.21461 | | | | | |
| 1.82523 | | | | | | |

б) построим первый интерполяционный многочлен Ньютона *pn*1[*x*]:

```
In[10]:= t = (x - a)/h; pn1[x_] = dif[0, 0]; p[t_] = 1;

In[11]:= For[k = 1, k ≤ n, k++,
          цикл ДЛЯ
           p[t_] = p[t] * (t - k + 1);
           pn1[x_] = pn1[x] + dif[0, k]/k! * p[t]]
```

Выведем полученный многочлен:

```
In[12]:= pn1[x]

Out[12]= 0.67244 - 0.227165 (2 + x) - 0.0544789 (2 + x) (-1 + 3 (2 + x)/4) +
         0.16002 (2 + x) (-2 + 3 (2 + x)/4) (-1 + 3 (2 + x)/4) -
         0.0769932 (2 + x) (-3 + 3 (2 + x)/4) (-2 + 3 (2 + x)/4) (-1 + 3 (2 + x)/4) +
         0.0213919 (2 + x) (-4 + 3 (2 + x)/4) (-3 + 3 (2 + x)/4) (-2 + 3 (2 + x)/4) (-1 + 3 (2 + x)/4) -
         0.00420148 (2 + x) (-5 + 3 (2 + x)/4) (-4 + 3 (2 + x)/4)
          (-3 + 3 (2 + x)/4) (-2 + 3 (2 + x)/4) (-1 + 3 (2 + x)/4)
```

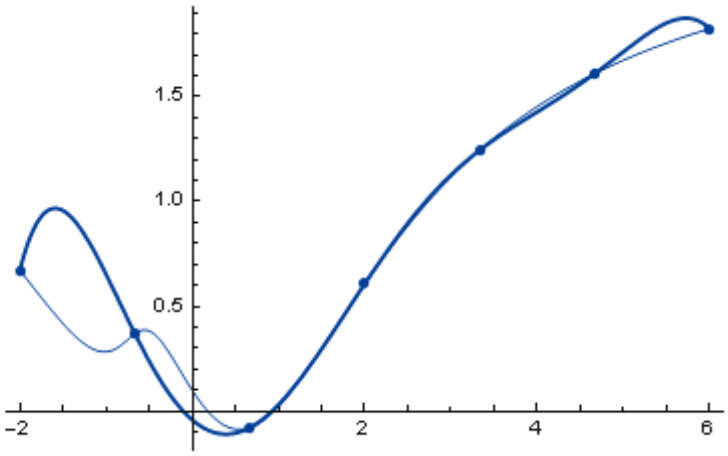
Упростим вид многочлена *pn*1[*x*]:

```
In[13]:= pn1[x_] = Simplify[pn1[x]]

Out[13]= -0.0418053 - 0.349915 x + 0.450958 x^2 +
          0.0244787 x^3 - 0.0661214 x^4 + 0.0147448 x^5 - 0.000997032 x^6
```

Построим графики (график многочлена Ньютона изображен жирной линией):

```
In[14]:= gr1 = Plot[f[x], {x, -2, 6}];
         (график функции)
         gr2 = ListPlot[data, PlotStyle → PointSize[0.015]];
         (диаграмма разб··· | стиль графика | размер точки)
         gr3 = Plot[pn1[x], {x, -2, 6}, PlotStyle → Thickness[0.0056]];
         (график функции | стиль графика | толщина)
Show[gr1, gr2, gr3]
```

Out[15]=




в) построим интерполяционный многочлен Ньютона с помощью встроенной функции **InterpolatingPolynomial** и упростим его вид:

```
In[16]:= Np[x_] = InterpolatingPolynomial[data, x]
         (интерполяционный многочлен)
In[17]:= Np[x_] = Simplify[Np[x]]
         (упростить)
```

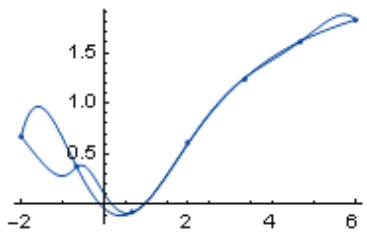
Out[17]=

$$-0.0418053 - 0.349915\,x + 0.450958\,x^2 + 0.0244787\,x^3 - 0.0661214\,x^4 + 0.0147448\,x^5 - 0.000997032\,x^6$$

Построим графики:

```
In[18]:= gr4 = Plot[Np[x], {x, -2, 6}, PlotStyle → Thickness[0.0056]];
         (график функции | стиль графика | толщина)
In[19]:= Show[gr1, gr2, gr4, ImageSize → Small]
```

Out[19]=




Найдем максимум абсолютной погрешности интерполирования многочленом $Np[x]$ на отрезке [–2, 6]:

```
In[20]:= FindMaximum[Abs[f[x] - Np[x]], {x, -2, 6}]
         найти макси···  абсолютное значение
Out[20]= {0.549889, {x → -1.40838}}
```

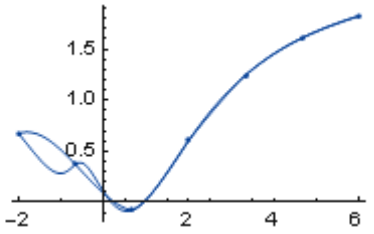
Выполним интерполяцию с помощью встроенной функции **Interpolation**:

```
In[21]:= Tfn = Interpolation[data]
Out[21]= InterpolatingFunction[ Domain: {{-2., 6.}}
                                Output: scalar ]
```

Построим графики:

```
In[22]:= gr5 = Plot[Tfn[x], {x, -2, 6}, PlotStyle → Thickness[0.0056]];
In[23]:= Show[gr1, gr2, gr5, ImageSize → Small]
```

Out[23]=




Найдем максимум абсолютной погрешности интерполирования функцией *Tfn*[*x*] на отрезке [–2, 6]:

```
In[24]:= FindMaximum[Abs[f[x] - Tfn[x]], {x, -2, 6}]
Out[24]= {0.274704, {x → -1.2546}}
```

г) вычислим значения $f(x)$ и построенных интерполяционных функций в точке $x = 2{,}28$:

```
In[25]:= {f[2.28], pn1[2.28], Np[2.28], Tfn[2.28]}
Out[25]= {0.769497, 0.77637, 0.77637, 0.753643}
```

▲

**Пример 5.2.** Построить график кусочно-заданной на отрезке [–2, 3] функции

$$f(x)=\begin{cases} a_1(x+2)^2+b_1, & -2\le x<-1,\\ a_2(x+1)^2+b_2, & -1\le x<0,\\ a_3x^2+b_3, & 0\le x<1,\\ a_4(x-1)^2+b_4, & 1\le x<2,\\ a_5(x-2)^2+b_5, & 2\le x\le 3, \end{cases}$$

где $a_k=(-1)^k(2k-1)$, при $k=\overline{1,5}$, $b_1=3{,}5$, $b_k=a_{k-1}h^2+b_{k-1}$, $k=\overline{1,4}$; $h$ – шаг разбиения отрезка $[-2,3]$ на частичные отрезки.

Вычислить *f*(2,28), найти максимальное значение функции $f(x)$ на отрезке [–2, 3].

Δ Отрезок [–2, 3] разделен точками $-1, 0, 1$ и $2$ на пять равных частичных отрезков. На каждом из промежутков $[x_{k-1}, x_k)$ $(k=\overline{1,4})$ и на отрезке $[x_4, x_5]=[2,3]$ функция $f(x)$ имеет вид $y=a_k(x-x_{k-1})^2+b_k$. Введем концы отрезка *a* и *b*, число частичных отрезков *n*, шаг *h*. Сформируем список *X* точек $x_k$ $(k=\overline{0,5})$, причем точке $x_k$ будет соответствовать элемент *X*[[*k* + 1]].

```
In[1]:= a = -2; b = 3; n = 5; h = (b - a)/n;

In[2]:= X = Table[a + h i, {i, 0, n}]
        таблица значений

Out[2]= {-2, -1, 0, 1, 2, 3}
```

Далее сформируем списки коэффициентов $a_k$ и $b_k$:

```
In[3]:= Ak = Table[(-1)^k (2 k - 1), {k, 1, n}];
        таблица значений

In[4]:= Bk = Table[0, {k, 1, n}]; Bk[[1]] = 3.5;
        таблица значений

In[5]:= For[k = 2, k ≤ n, k++, Bk[[k]] = Ak[[k - 1]] h^2 + Bk[[k - 1]]]; Bk
        цикл ДЛЯ

Out[5]= {3.5, 2.5, 5.5, 0.5, 7.5}
```

Чтобы определить кусочно-заданную функцию $f(x)$, воспользуемся встроенной функцией **Piecewise**. Аргумент этой функции должен иметь вид $\{\{f_1, cond_1\}, \{f_2, cond_2\}, ..., \{f_n, cond_n\}\}$. Создадим соответствующий список dat:

```
In[6]:= dat = Table[If[k < n, {Ak[[k]] (x - X[[k]])^2 + Bk[[k]], X[[k]] ≤ x < X[[k + 1]]},
        табл··· условный оператор
          {Ak[[k]] (x - X[[k]])^2 + Bk[[k]], X[[k]] ≤ x ≤ X[[k + 1]]}], {k, 1, n}]

Out[6]= {{3.5 - (2 + x)^2, -2 ≤ x < -1}, {2.5 + 3 (1 + x)^2, -1 ≤ x < 0},
         {5.5 - 5 x^2, 0 ≤ x < 1}, {0.5 + 7 (-1 + x)^2, 1 ≤ x < 2}, {7.5 - 9 (-2 + x)^2, 2 ≤ x ≤ 3}}
```

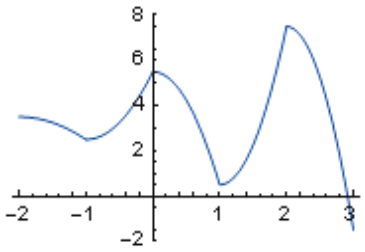
Теперь можно задать функцию $f(x)$ (обозначим ее $f1$):

```
In[7]:= f1 = Piecewise[dat]

Out[7]= { 3.5 - (2 + x)^2      -2 ≤ x < -1
        { 2.5 + 3 (1 + x)^2    -1 ≤ x < 0
        { 5.5 - 5 x^2          0 ≤ x < 1
        { 0.5 + 7 (-1 + x)^2   1 ≤ x < 2
        { 7.5 - 9 (-2 + x)^2   2 ≤ x ≤ 3
        { 0                    True
```

Построим график функции:

```
In[8]:= Plot[f1, {x, -2, 3}, ImageSize → Small]
```

Out[8]=




Вычислим значение функции в точке $x$ = 2,28 и найдем ее максимум на отрезке [–2, 3]:

```
In[9]:= f1 /. x → 2.28

Out[9]= 6.7944

In[10]:= FindMaximum[f1, {x, -2, 3}]
         найти максимум

Out[10]= {3.5, {x → -2.}}
```

▲

**Пример 5.3.** Используя значения функции

$$f(x)=\frac{5x^2-4x\,\mathrm{arctg}(3x+2)+\ln 2}{2x^2+7}$$

в равноотстоящих узлах $x_i$ отрезка [–2, 6] при разбиении его на шесть частей, выполнить следующие действия:

а) осуществить интерполяцию сплайном с помощью функции **Interpolation**[*data*, **Method->“Spline”**], проиллюстрировать графически;

б) построить интерполяционный кубический сплайн с помощью функции **SplineFit[***data*,**Cubic]**, проиллюстрировать графически;

в) построить кубический многочлен наилучшего среднеквадратичного приближения с помощью функции **Fit**, проиллюстрировать графически;

г) найти значения функции $f(x)$ и построенных функций в точке $x$ = 2,28.

Δ Введем функцию $f(x)$, границы отрезка $a$ и $b$, количество частей разбиения отрезка $n$, вычислим шаг $h$ и построим таблицу значений функции:

```
In[1]:= f[x_] = (5 x^2 - 4 x ArcTan[3 x + 2] + Log[2]) / (2 x^2 + 7);

In[2]:= a = -2; b = 6; n = 6; h = (b - a) / n;

In[3]:= data = N[Table[{a + i h, f[a + i h]}, {i, 0, n}]];
```

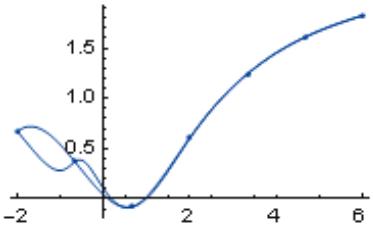
а) построим интерполяционный сплайн с помощью функции **Interpolation**:

```
In[4]:= spl1 = Interpolation[data, Method → "Spline"]

Out[4]= InterpolatingFunction[ Domain: {{-2., 6.}}
                               Output: scalar ]
```

Построим графики (график сплайна изображен жирной линией):

```
In[5]:= gr1 = Plot[f[x], {x, -2, 6}];
        gr2 = ListPlot[data, PlotStyle → PointSize[0.015]];
        gr3 = Plot[spl1[x], {x, -2, 6}, PlotStyle → Thickness[0.0056]];

In[6]:= Show[gr1, gr2, gr3, ImageSize → Small]
```

Out[6]=




б) построим интерполяционный кубический сплайн с помощью функции **SplineFit**. Предварительно загрузим пакет сплайн-интерполяции командой **Needs[“Splines`”]**, чтобы эта функция была доступна.

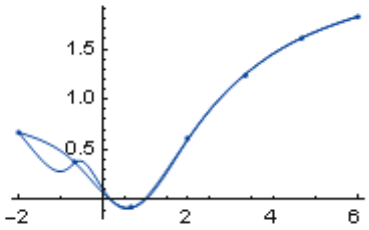
```
In[7]:= Needs["Splines`"]

In[8]:= spl2 = SplineFit[data, Cubic]

Out[8]= SplineFunction[Cubic, {0., 6.}, <>]
```

В результате получаем параметрически заданную функцию с указанной областью изменения параметра. Если интерполирование функцией **SplineFit** проводится по системе равноотстоящих узлов отрезка $[a, b]$, то между параметром $t$ сплайна и переменной $x$ существует линейная зависимость $t = \dfrac{x - a}{h}$, где $h$ – шаг интерполяции.

Построим графики:

```
In[9]:= Clear[t]; gr4 = ParametricPlot[spl2[t], {t, 0, 6}];

In[10]:= Show[gr1, gr2, gr4, ImageSize → Small]
```

Out[10]=



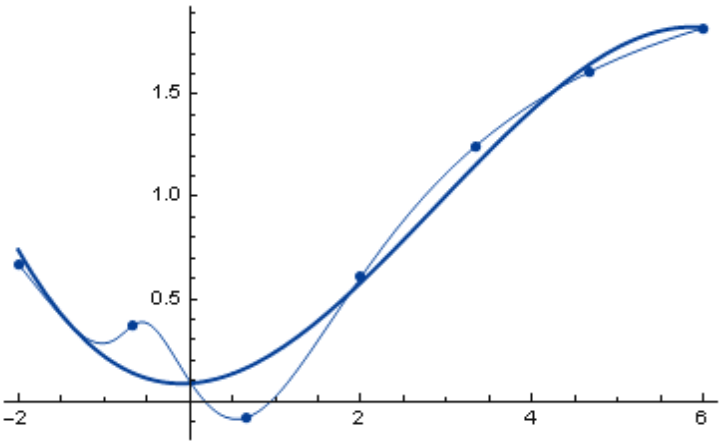

в) аппроксимируем функцию $f(x)$ многочленом третьей степени наилучшего среднеквадратичного приближения с помощью функции **Fit**.

```
In[11]:= p3 = Fit[data, {1, x, x^2, x^3}, x]

Out[11]= 0.0879585 + 0.0264223 x + 0.143234 x^2 - 0.0165582 x^3

In[12]:= p[x_] = p3

Out[12]= 0.0879585 + 0.0264223 x + 0.143234 x^2 - 0.0165582 x^3
```

Построим графики:

```
In[13]:= gr5 = Plot[p[x], {x, -2, 6}, PlotStyle → Thickness[0.0056]];

In[14]:= Show[gr1, gr2, gr5]
```

Out[14]=




г) вычислим значения $f(x)$ и построенных функций в точке $x$ = 2,28:

```
In[15]:= {f[2.28], spl1[2.28], spl2[(2.28 - a)/h], p[2.28]}

Out[15]= {0.769497, 0.77404, {2.28, 0.775649}, 0.696537}
```

Как видно, функция spl2, построенная с помощью **SplineFit**, возвращает координаты точки кривой. ▲

# Лабораторная работа 4

## Интерполяция и среднеквадратичное приближение

**1.** Создать таблицу значений функции $f(x)$, разбив отрезок [0, 6] на $n$ равных частей точками $x_i$ ($i=\overline{0,n}$). Для полученной таблично заданной в равноотстоящих узлах функции $f(x)$, выполнить следующие действия при $n=6$ и $n=10$:

а) построить интерполяционный многочлен Лагранжа $L_n(x)$, проиллюстрировать графически (изобразить точки $(x_i, f(x_i))$ и графики функций $f(x)$ и $L_n(x)$ на одном чертеже);

б) создать таблицу конечных разностей функции $f(x)$ по точкам $(x_i, f(x_i))$, $i=\overline{0,n}$;

в) построить второй интерполяционный многочлен Ньютона $P_n(x)$, проиллюстрировать графически;

г) построить интерполяционный многочлен Ньютона $Np_n(x)$ с помощью функции **InterpolatingPolynomial** пакета **Mathematica**, проиллюстрировать графически;

д) вычислить значения функции $f(x)$ и всех построенных интерполяционных многочленов $L_n(x)$, $P_n(x)$ и $Np_n(x)$ в точке $x$ = 2,4316;

е) построить график погрешности интерполирования многочленом Ньютона $R_n(x)=|f(x)-Np_n(x)|$ на отрезке [0, 6], найти максимум погрешности $P_n(x)$ на отрезке [0, 6] с помощью функции **FindMaximum** пакета **Mathematica**;

ж) исследовать зависимость погрешности интерполирования $P_n(x)$ от числа узлов интерполяции (степени многочлена $n$).

1.1. $f(x)=5\exp\left(-\frac{1}{18}x^2+\frac{1}{3}x-\frac{1}{2}\right)-2\sin\sqrt{x}$.

1.2. $f(x)=\frac{3x+\pi}{\sqrt{1+x^2+\sqrt{(1+x^2)^3}}}$.

1.3. $f(x)=\sqrt{x^3+4}\cdot\cos\left(\frac{x}{\sqrt{17}}+\frac{1}{21}\right)$.

1.4. $f(x)=\dfrac{4x^2-5x+1}{\sqrt{2+x^2+\sqrt{(2+x^2)^5}}}$.

1.5. $f(x)=\sqrt{3}\exp\left(-\dfrac{1}{22}x^3+\dfrac{1}{2}x-\dfrac{1}{4}\right)$.

1.6. $f(x)=\exp\left(x-\dfrac{x^2}{4}\right)\cdot \text{th}\left(\dfrac{x^3}{11}+\dfrac{1}{3}\right)$.

1.7. $f(x)=\dfrac{2\sqrt{21}\cdot\sin(3x^2/28)+\sqrt[3]{3}}{\sqrt{2+x^2+\sqrt{(4+x^2)^3}}}$.

1.8. $f(x)=\exp\left(2x-\dfrac{2x^2}{7}\right)\cdot \text{arctg}\left(\dfrac{3x^5}{14}+\dfrac{5}{6}\right)$.

1.9. $f(x)=3+\left(\dfrac{2}{7}x-\text{ch}\dfrac{3x}{13}\right)\cdot\ln(x^2+2x+3)$.

1.10. $f(x)=\dfrac{\text{sh}\sqrt{x^2+x+5}+\pi}{\sqrt{3x^8+11x^4+33}}$.

1.11. $f(x)=4\exp\left(-\dfrac{2}{7}x^2-\dfrac{4}{9}x+\dfrac{1}{13}\right)+7\cos\sqrt{4x+1}$.

1.12. $f(x)=\dfrac{7x+12\sin x}{\sqrt{\pi+x^2+\sqrt[3]{(1+x^2)^4}}}$.

1.13. $f(x)=\sqrt[5]{x^6+4x^2+1}\cdot\sin\left(\dfrac{2x}{\sqrt{31}}+\dfrac{1}{7}\sqrt{x+5}+\dfrac{1}{18}\right)$.

1.14. $f(x)=\left(x+\sqrt{\pi+1}\right)\cdot\exp\left(-\dfrac{4}{39}\sqrt{x^5}+\dfrac{5}{9}x+\dfrac{1}{4}\right)$.

1.15. $f(x)=\left(\dfrac{5}{11}x+\cos\dfrac{3x}{2}-\sqrt{x}\,\text{sh}\dfrac{x}{6}\right)\cdot\log_2(x^2+4x+5)$.

1.16. $f(x)=\exp\left(3x-\dfrac{x^2}{6}\right)\cdot\arccos\left(\dfrac{2x^7}{35}+1\right)$.

**2.** Создать таблицу значений функции $f(x)$ (1.1–1.16), разбив отрезок [0, 6] на $n$ частей неравноотстоящими точками $x_i$ вида

$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cdot t_i$, где $t_i$ – корни многочлена Чебышёва $T_{n+1}(t)$ ($i = \overline{0,n}$). Для полученной таблично заданной функции $f(x)$ выполнить следующие действия при $n = 6$ и $n = 10$:

а) создать таблицу разделенных разностей функции $f(x)$ по точкам $(x_i, f(x_i))$, $i = \overline{0,n}$;

б) построить интерполяционный многочлен Ньютона $Pnr_n(x)$ для неравноотстоящих узлов, проиллюстрировать графически (изобразить точки $(x_i, f(x_i))$ и графики функций $f(x)$ и $Pnr_n(x)$ на одном чертеже);

в) построить интерполирующую функцию $Intf_n(x)$ с помощью функции **Interpolation** пакета **Mathematica**, проиллюстрировать графически;

г) вычислить значения функции $f(x)$ и построенных интерполяционных многочленов $Pnr_n(x)$ и $Intf_n(x)$ в точке $x$ = 2,4316;

д) найти максимумы абсолютных погрешностей интерполирования функции $f(x)$ многочленом Ньютона $Pnr_n(x)$ и функцией $Intf_n(x)$ на отрезке [0, 6] с помощью функции **FindMaximum** пакета **Mathematica**.

**3.** Сравнить результаты заданий 1 и 2 для равноотстоящих и неравноотстоящих узлов и сделать выводы о зависимость погрешности интерполирования от числа узлов и их расположения на отрезке.

**4.** Используя таблицу значений функции $f(x)$ в равноотстоящих точках отрезка [0, 6], полученной в задании 1 при $n = 10$, выполнить следующие действия:

а) построить интерполяционный кубический сплайн дефекта 1 $S_3(x)$ для функции $f(x)$, проиллюстрировать графически (изобразить точки $(x_i, f(x_i))$ и графики функций $f(x)$ и $S_3(x)$ на одном чертеже);

б) выполнить интерполяцию сплайном $Sf(x)$ с помощью функции **Interpolation**[*data*,**Method->"Spline"**], проиллюстрировать графически;

в) построить интерполяционный кубический сплайн *Spl* с помощью функции **SplineFit**[*data*,**Cubic**] (предварительно загрузить пакет сплайн-интерполяции командой **Needs**[**"Splines`"**]), проиллюстрировать графически (для построения графика сплайна S*pl* использовать функцию **ParametricPlot**);

г) вычислить значения функции $f(x)$ и построенных интерполяционных сплайнов $S_3(x)$, $Sf(x)$ и *Spl* в точке $x$ = 2,4316.

**5.** Используя таблицу значений функции $f(x)$ в равноотстоящих точках отрезка [0, 6], полученной в задании 1 при $n=10$, выполнить следующие действия:

а) аппроксимировать функцию $f(x)$ с помощью метода наименьших квадратов многочленом первой степени $Q_1(x)$, проиллюстрировать графически (изобразить точки $(x_i, f(x_i))$ и график функции $Q_1(x)$ на одном чертеже);

б) аппроксимировать функцию $f(x)$ с помощью метода наименьших квадратов многочленом второй степени $Q_2(x)$, проиллюстрировать графически;

в) найти многочлены наилучшего среднеквадратичного приближения третьей и четвертой степеней ($Q_3(x)$ и $Q_4(x)$) с помощью функции **Fit** пакета **Mathematica**, проиллюстрировать графически;

г) вычислить значения функции $f(x)$ и построенных многочленов $Q_1(x)$, $Q_2(x)$, $Q_3(x)$ и $Q_4(x)$ в точке $x$ = 2,4316;

д) сравнить результаты, полученные в пунктах *а*, *б* и *в*, изобразив на одном чертеже точки $(x_i, f(x_i))$ и графики функций $Q_1(x)$, $Q_2(x)$, $Q_3(x)$ и $Q_4(x)$.

# 6. Численное дифференцирование и интегрирование

## Численное дифференцирование

При решении практических задач часто требуется найти производные различных порядков от функций, заданных таблично, либо от функций, заданных с помощью очень сложных аналитических выражений. Тогда прибегают к численному дифференцированию.

Чтобы получить формулы приближенного дифференцирования, данную функцию $f(x)$ заменяют на отрезке $[a, b]$ интерполирующей функцией $P(x)$ (чаще всего полиномом)

$$f(x) \approx P(x),\ x \in [a, b],$$

а затем полагают

$$f'(x) \approx P'(x),\ f''(x) \approx P''(x),\ \ldots,\ f^{(n)}(x) \approx P^{(n)}(x),\ x \in [a, b].$$

Если известна погрешность $R(x)$ интерполирующей функции $P(x)$

$$R(x) = f(x) - P(x),$$

то можно найти погрешность $r_n(x)$ численного дифференцирования:

$$r_n(x) = f^{(n)}(x) - P^{(n)}(x) = R^{(n)}(x).$$

В результате численного дифференцирования мы получаем таблицу значений производной *n*-го порядка функции $f(x)$ в узловых точках, т. е. таблично заданную функцию $f^{(n)}(x)$.

*Примечание*. Численное дифференцирование представляет собой операцию менее точную, чем интерполирование. Действительно, близость друг к другу ординат двух кривых $y = f(x)$ и $y = P(x)$ на отрезке $[a, b]$ еще не гарантирует близости на этом отрезке их производных $f'(x)$ и $P'(x)$, т. е. малого отличия угловых коэффициентов касательных к этим кривым при одинаковых значениях аргумента.

## Простейшие формулы численного дифференцирования

Для функции $f(x)$, $(n+1)$ раз непрерывно дифференцируемой в окрестности точки *c*, справедлива формула Тейлора:

$$f(x) = f(c) + \frac{f'(c)}{1!}(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \ldots + \frac{f^{(n)}(c)}{n!}(x-c)^n + R_n(x),$$

где остаточный член

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}, \quad \xi \in (c, x).$$

Пусть дана таблица значений функции $f(x)$ в равноотстоящих узлах $x_i$ отрезка $[a, b]$:

$$y_i = f(x_i), \quad x_i = x_0 + ih, \quad i = \overline{0, n}, \quad h = \frac{b-a}{n}, \quad x_0 = a, \quad x_n = b.$$

Получим некоторые простейшие формулы численного дифференцирования, используя формулу Тейлора.

**1.** Пусть функция $f(x)$ дважды непрерывно дифференцируема.

Для каждого отрезка $[x_i, x_{i+1}]$ запишем ее разложение по формуле Тейлора первого порядка в окрестности точки $x_i$:

$$f(x) = f(x_i) + \frac{f'(x_i)}{1!}(x-x_i) + \frac{f''(\xi)}{2!}(x-x_i)^2.$$

При $x = x_{i+1}$ имеем

$$y_{i+1} = y_i + y_i' h + \frac{f''(\xi)}{2!}h^2, \quad \xi \in (x_i, x_{i+1}),$$

где $y_i = f(x_i)$, $y_{i+1} = f(x_{i+1})$.

Отсюда

$$y_i' = \frac{y_{i+1} - y_i}{h} - \frac{f''(\xi)}{2}h.$$

Тогда ***приближенная формула для первой производной*** имеет вид

$$y_i' \approx \frac{y_{i+1} - y_i}{h}, \quad i = \overline{0, n-1}.$$

При этом ***погрешность*** аппроксимации производной $R \le \frac{h}{2}\max\limits_{x \in [a, b]} |f''(x)|$.

Формула имеет ***первый порядок*** точности.

Заметим, что числитель формулы представляет собой конечную разность первого порядка $\Delta y_i = y_{i+1} - y_i$, т. е.

$$y_i' \approx \frac{\Delta y_i}{h}, \quad i = \overline{0, n-1}.$$

**2.** Если $f(x)$ трижды непрерывно дифференцируема, то, выразив $y_{i+1}$ и $y_{i-1}$ через $y_i$ по формуле Тейлора второго порядка и вычтя из первого равенства второе, получим еще одну ***приближенную формулу для первой производной***:

$$y_i' \approx \frac{y_{i+1} - y_{i-1}}{2h}, \quad i = \overline{1, n-1},$$

ее ***погрешность*** $R \le \frac{h^2}{6} \max\limits_{x \in [a, b]} \left| f'''(x) \right|$. Формула имеет ***второй порядок*** точности.

**3.** Если $f(x)$ четырежды непрерывно дифференцируема, то, выразив $y_{i+1}$ и $y_{i-1}$ через $y_i$ по формуле Тейлора третьего порядка и сложив их, получим ***приближенную формулу для второй производной***:

$$y_i'' \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}, \quad i = \overline{1, n-1},$$

ее ***погрешность*** $R \le \frac{h^2}{12} \max\limits_{x \in [a, b]} \left| f^{(4)}(x) \right|$. Формула имеет ***второй порядок*** точности.

Заметим, что числитель формулы представляет собой конечную разность второго порядка $\Delta^2 y_i = y_{i+1} - 2y_i + y_{i-1}$, т. е.

$$y_i'' \approx \frac{\Delta^2 y_i}{h^2}, \quad i = \overline{1, n-1}.$$

Аналогичные ***формулы*** используют и ***для производных более высоких порядков***:

$$y_i^{(k)} \approx \frac{\Delta^k y_i}{h^k},$$

однако они не обладают хорошей точностью.

Существуют ***более точные формулы***, основанные, в частности, на первой и второй интерполяционных формулах Ньютона, формулах Гаусса и др.

Например, для первой и второй производной можно пользоваться следующими формулами:

$$y_i' \approx \frac{1}{h}\left( \Delta y_i - \frac{1}{2}\Delta^2 y_i + \frac{1}{3}\Delta^3 y_i - \frac{1}{4}\Delta^4 y_i + \ldots \right), \qquad y_i'' \approx \frac{1}{h^2}\left( \Delta^2 y_i - \Delta^3 y_i + \frac{11}{12}\Delta^4 y_i + \ldots \right),$$

где $\Delta^k y_i$ – конечная разность порядка $k$. Как правило, на практике ограничиваются первыми двумя слагаемыми в этих формулах.

## Численное интегрирование

Задача приближенного или численного интегрирования возникает в следующих случаях:

1) если первообразная интегрируемой функции $f(x)$ не может быть найдена с помощью элементарных средств или является слишком сложной;

2) если подынтегральная функция $f(x)$ задана таблично.

Задача численного интегрирования состоит в вычислении значения определенного интеграла на основании ряда значений подынтегральной функции:

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

где $x_i$ – узлы интегрирования ($x_i \in [a, b]$), $A_i$ – квадратурные коэффициенты.

Численное нахождение однократного интеграла называется ***механической квадратурой***, двойного интеграла – ***механической кубатурой***. Соответствующие формулы называются *квадратурными* и *кубатурными*.

## Квадратурные формулы прямоугольников

Как известно, определенный интеграл $\int_a^b f(x)\,dx$ от неотрицательной функции $f(x)$ равен площади криволинейной трапеции, ограниченной сверху графиком функции $y = f(x)$, снизу – осью $OX$, слева и справа – отрезками вертикальных прямых $x = a$, $x = b$.

Разобьем отрезок $[a, b]$ на $n$ равных частей точками $x_i$:

$$x_0 = a, \quad x_i = x_0 + ih, \quad i = \overline{1, n}, \quad h = \frac{b-a}{n} \quad (x_n = b).$$

На каждом из частичных отрезков $[x_{i-1}, x_i]$ $(i = \overline{1, n})$ определенный интеграл $\int_{x_{i-1}}^{x_i} f(x)\,dx$ заменим площадью прямоугольника, ширина которого равна $h$, а длина – $f(c_i)$, где $c_i$ – некоторая точка отрезка $[x_{i-1}, x_i]$:

$$\int_{x_{i-1}}^{x_i} f(x)\,dx \approx S_i = h \cdot f(c_i) = \frac{b-a}{n} \cdot f(c_i), \quad i = \overline{1, n}.$$

Тогда определенный интеграл от функции $f(x)$ по всему отрезку $[a, b]$ будет равен сумме

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx \approx \frac{b-a}{n} \cdot \sum_{i=1}^{n} f(c_i).$$

Заметим, что эта формула справедлива и для функций, принимающих не только неотрицательные значения в точках отрезка $[a, b]$.

**1.** Выберем в качестве точки $c_i$ левый конец отрезка $[x_{i-1}, x_i]$ (рис. 6.1), т. е. $c_i = x_{i-1}$, получим так называемую ***формулу левых прямоугольников*** (где $y_i = f(x_i)$):

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}\cdot\sum_{i=1}^{n} f(x_{i-1}) = \frac{b-a}{n}\cdot\left(y_0 + y_1 + \ldots + y_{n-1}\right).$$

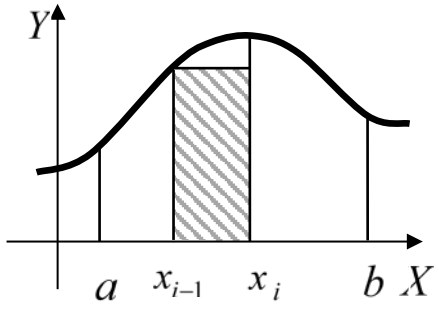


Рис. 6.1

***Погрешность*** (остаточный член) формулы левых прямоугольников: $R_n = \dfrac{(b-a)^2}{2n} f'(\xi)$, где $a \le \xi \le b$.

**2.** Возьмем в качестве точки $c_i$ правый конец отрезка $[x_{i-1}, x_i]$ (рис. 6.2), т. е. $c_i = x_i$, получим ***формулу правых прямоугольников***:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}\cdot\sum_{i=1}^{n} f(x_i) = \frac{b-a}{n}\cdot\left(y_1 + y_2 + \ldots + y_n\right).$$

***Погрешность*** формулы правых прямоугольников: $R_n = -\dfrac{(b-a)^2}{2n} f'(\xi)$, где $a \le \xi \le b$.

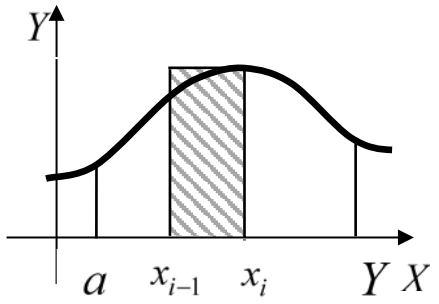


Рис. 6.2

**3.** Выберем в качестве точки $c_i$ середину отрезка $[x_{i-1}, x_i]$ (рис. 6.3), т. е.

$$c_i = \left(x_{i-1} + x_i\right)/2 = x_{i-1} + h/2 = x_{i-\frac{1}{2}}.$$

Тогда получим ***формулу средних*** (или ***центральных***) ***прямоугольников***:

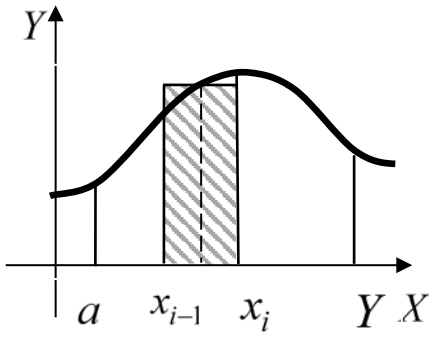


Рис. 6.3

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}\cdot\sum_{i=1}^{n} f\left(x_{i-1} + \frac{b-a}{2n}\right) = \frac{b-a}{n}\cdot\left(y_{\frac{1}{2}} + y_{\frac{3}{2}} + \ldots + y_{\frac{2n-1}{2}}\right).$$

***Погрешность*** формулы средних прямоугольников: $R_n = \dfrac{(b-a)^3}{24n^2} f''(\xi)$, где $a \le \xi \le b$.

## Квадратурная формула трапеций

Разобьем отрезок $[a, b]$ на $n$ равных частей точками $x_i$. На каждом из частичных отрезков $[x_{i-1}, x_i]$ $(i = \overline{1, n})$ определенный интеграл $\int_{x_{i-1}}^{x_i} f(x)\,dx$ заменим площадью прямоугольной трапеции с вершинами в точках $(x_{i-1}, 0)$, $(x_i, 0)$, $(x_{i-1}, y_{i-1})$, $(x_i, y_i)$, где $y_i = f(x_i)$ (рис. 6.4):

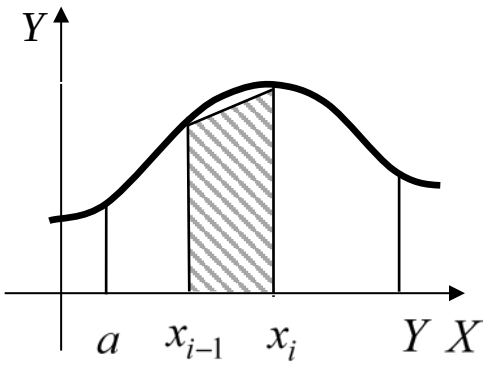



Рис. 6.4

$$\int_{x_{i-1}}^{x_i} f(x)\,dx \approx S_i = h \cdot \frac{y_{i-1} + y_i}{2}.$$

Отсюда следует ***формула трапеций*** *(общая формула трапеций)*:

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx \approx \frac{b-a}{n} \cdot \left( \frac{y_0}{2} + y_1 + y_2 + \ldots + y_{n-1} + \frac{y_n}{2} \right).$$

***Погрешность*** формулы трапеций: $R_n = -\dfrac{(b-a)^3}{12n^2} f''(\xi)$, где $a \le \xi \le b$.

Заметим, что формула средних прямоугольников и формула трапеций дают примерно одинаковую точность вычислений. Формулы правых и левых прямоугольников значительно менее точны, но более просты.

## Квадратурная формула Симпсона

Разделим отрезок $[a, b]$ на $2n$ равных частей точками $x_k$ (рис. 6.5). Пусть $h = \dfrac{b-a}{2n}$, $x_0 = a$, $x_k = x_0 + kh$, $k = \overline{1, 2n}$, $(x_{2n} = b)$.

На каждом из частичных отрезков $[x_{2i}, x_{2i+2}]$ $(i = \overline{0, n-1})$ определенный интеграл $\int_{x_{2i}}^{x_{2i+2}} f(x)\,dx$ заменим площадью криволинейной трапеции, ограниченной сверху параболой, проходящей через точки $A(x_{2i}, f(x_{2i}))$, $B(x_{2i+1}, f(x_{2i+1}))$ и $C(x_{2i+2}, f(x_{2i+2}))$.

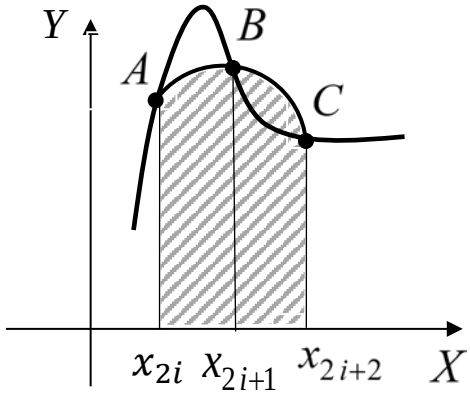



Рис. 6.5

Составив уравнение такой параболы и вычислив соответствующий интеграл, получим

$$\int_{x_{2i}}^{x_{2i+2}} f(x)\,dx \approx \frac{h}{3}\cdot\left(y_{2i} + 4y_{2i+1} + y_{2i+2}\right).$$

Отсюда следует ***общая формула Симпсона*** (или ***формула парабол***):

$$\int_a^b f(x)\,dx = \sum_{i=0}^{n-1} \int_{x_{2i}}^{x_{2i+2}} f(x)\,dx \approx$$

$$\approx \frac{h}{3}\cdot\left(y_0 + y_{2n} + 4\left(y_1 + y_3 + \ldots + y_{2n-1}\right) + 2\left(y_2 + y_4 \ldots + y_{2n-2}\right)\right),$$

где $h = \dfrac{b-a}{2n}$, $x_k = x_0 + kh$, $y_k = f(x_k)$, $k = \overline{0, 2n}$.

***Погрешность*** формулы Симпсона: $R_n = \dfrac{(b-a)^5}{180n^4} f^{(4)}(\xi)$, где $a \le \xi \le b$.

Квадратурные формулы прямоугольников, трапеций и парабол являются частными случаями ***квадратурных формул Ньютона – Котеса*** (при $n = 0$, $n = 1$ и $n = 2$), которые получаются при замене функции $f(x)$ ее интерполяционным многочленом Лагранжа степени $n$ и строятся для равноотстоящих узлов.

## Квадратурные формулы Гаусса

Квадратурная формула Гаусса для вычисления определенного интеграла от функции $f(t)$ на отрезке $[-1, 1]$ имеет вид

$$\int_{-1}^{1} f(t)\,dt = \sum_{i=1}^{n} A_i f(t_i),$$

где $A_i$ – квадратурные коэффициенты; $t_i$ – корни полинома Лежандра $n$-й степени $\chi_n(t) = \dfrac{1}{2^n n!}\cdot\dfrac{d^n}{dx^n}\left((x^2-1)^n\right)$.

По свойствам полином Лежандра $\chi_n(t)$ имеет ровно $n$ различных действительных корней, все они расположены в интервале $(-1, 1)$. Их значения можно взять, например, из таблиц. Квадратурные коэффициенты $A_i$, для которых также составлены таблицы значений, являются решением системы линейных уравнений

$$\begin{cases} \sum_{i=1}^{n} A_i = 2, \\ \sum_{i=1}^{n} A_i\, t_i = 0, \\ \ldots\ldots\ldots\ldots \\ \sum_{i=1}^{n} A_i\, t_i^{n-1} = \dfrac{1-(-1)^n}{n}. \end{cases}$$

Для вычисления интеграла $\int_a^b f(x)\,dx$ с помощью квадратурной формулы Гаусса нужно произвести замену переменной $x = \dfrac{b+a}{2} + \dfrac{b-a}{2}\cdot t$. Тогда

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\sum_{i=1}^{n} A_i\, f\left(\frac{b+a}{2} + \frac{b-a}{2}\cdot t_i\right).$$

***Погрешность*** (остаточный член) формулы Гаусса с $n$ узлами:

$R_n = \dfrac{(b-a)^{2n+1}\cdot(n!)^4}{((2n)!)^3\cdot(2n+1)}\cdot f^{(2n)}(\xi)$, где $a \le \xi \le b$.

### Уточнение значения интеграла по Ричардсону

Пусть с помощью одной и той же квадратурной формулы найдены два приближенных значения $I_{n_1}$ и $I_{n_2}$ интеграла $\int_a^b f(x)\,dx$ при $n_2 > n_1$. Более точное его значение, согласно Ричардсону, можно получить по формуле

$$I_{n_1,n_2} = I_{n_2} + \frac{n_1^k}{n_2^k - n_1^k}\left(I_{n_2} - I_{n_1}\right),$$

где $k$ – порядок остаточного члена квадратурной формулы. Для формул правых и левых прямоугольников $k = 1$, для средних прямоугольников и трапеций $k = 2$, для формулы Симпсона (парабол) $k = 4$, для формулы Гаусса с $n$ узлами $k = 2n$.

## Основные операторы и функции пакета Mathematica для дифференцирования и интегрирования

***D***[*f*,*x*] – находит производную (также и частную) функции *f* по указанной переменной *x*.

***D***[*f*,{*x*,*n*}] – находит (частную) производную *n*-го порядка функции *f* по переменной *x*.

***D***[*f*,*x*,*y*}] – находит смешанную производную $\frac{\partial^2 f}{\partial x \partial y}$. Может применяться для большего числа переменных.

***D***[*f*,{*x*, $n_1$},{*y*, $n_2$}] – находит частную производную $\frac{\partial^n f}{\partial x^{n_1} \partial y^{n_2}}$ порядка $n = n_1 + n_2$. Может применяться для большего числа переменных.

***Dt***[*f*,*x*] – возвращает полную производную функции *f* по переменной *x*.

***Dt***[*f*,{*x*,*n*}] – находит полную производную $n$-го порядка функции *f* по переменной *x*.

***Dt***[*f*] – находит полный дифференциал функции *f*.

***Integrate***[*f*,*x*] – находит первообразную (неопределенный интеграл) от функции *f* по переменной *x*. Константа при этом не добавляется.

***Integrate***[*f*, {*x*, *a*, *b*}] – вычисляет определенный интеграл от функции *f* по переменной *x* на отрезке $[a, b]$.

***NIntegrate***[*f*, {*x*, *a*, *b*}] – находит численное приближение к определенному интегралу от функции *f* на отрезке $[a, b]$.

Производные и интегралы в системе **Mathematica** можно записывать в привычной форме, используя палитру математических символов ***Basic Math Assistant***.

Отметим некоторые специальные математические функции из справочной базы данных **Mathematica** (они не могут быть выражены через элементарные функции):

– **CosIntegral**[*x*] – интегральный косинус $\mathrm{Ci}(x)$;
– **SinIntegral**[*x*] – интегральный синус $\mathrm{Si}(x)$;
– **CoshIntegral**[*x*] – интегральный гиперболический косинус;
– **SinhIntegral**[*x*] – интегральный гиперболический синус;
– **ExpIntegral**[*x*] – интегральная показательная функция $\mathrm{Ei}(x)$;
– **Erf**[*x*] **–** функция ошибок;
– **LogIntegral**[*x*] – интегральный логарифм $\mathrm{Li}(x)$;
– **LegendreP**[*n*, *x*] – полином Лежандра степени *n* относительно переменной *x*.

Для таких функций в системе **Mathematica** можно получить численные значения, построить график и др.

## Примеры численного дифференцирования и интегрирования средствами пакета Mathematica

**Пример 6.1.** Найти приближенные значения производных первого и второго порядков функции $f(x) = \ln\cos x$ в точке $x_1 = 0{,}83$, используя:

а) функцию ***D*** системы **Mathematica**;

б) формулы $y_i' \approx \dfrac{y_{i+1} - y_{i-1}}{2h}$ и $y_i'' \approx \dfrac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$ для шага $h = 0{,}01$.

$\Delta$ а) введем функцию $f(x)$, точку $x_1$ и вычислим производные с помощью функции ***D***.

```
In[1]:= f[x_] = Log[Cos[x]]; x1 = 0.83;
        на··· косинус

In[2]:= D[f[x], x]
        дифференциироват
        D[f[x], {x, 2}]
        дифференциироват

Out[2]= -Tan[x]

Out[3]= -Sec[x]^2

In[4]:= d11 = D[f[x], x] /. x → x1
        дифференциировать

Out[4]= -1.09343

In[5]:= d21 = D[f[x], {x, 2}] /. x → x1
        дифференциировать

Out[5]= -2.1956
```

б) для вычисления производных в точке $x_1$ с помощью заданных формул потребуются значения функции в двух соседних точках $x_0 = x_1 - h$ и $x_2 = x_1 + h$. Введем функции для приближенного вычисления производных:

```
In[6]:= pr1[x_, h_] = (f[x + h] - f[x - h])/(2 h); pr2[x_, h_] = (f[x + h] - 2 f[x] + f[x - h])/h^2;
```

Вычислим производные:

```
In[7]:= pr11 = pr1[x1, 0.01]

Out[7]= -1.09351

In[8]:= pr21 = pr2[x1, 0.01]

Out[8]= -2.19576
```

Определим абсолютную погрешность результата:

```
In[9]:= {Abs[d11 - pr11], Abs[d21 - pr21]}
        абсолютное значе··· абсолютное значени

Out[9]= {0.0000800335, 0.000167866}
```

▲

**Пример 6.2.** Вычислить определенный интеграл $\int_0^1 e^{-x^2}\,dx$:

а) методом левых прямоугольников, разбив отрезок интегрирования на 200 частей;

б) с помощью функции **NIntegrate**.

Δ Введем функцию, концы отрезка интегрирования, число разбиений и шаг:

```
In[1]:= f[x_] = Exp[-x^2]; a = 0; b = 1; n = 200; h = (b - a)/n;
        показательная функция
```

а) вычислим интеграл по формуле левых прямоугольников:

```
In[2]:= int1 = h * Sum[f[a + i*h], {i, 0, n-1}] // N
        чи                                              Out[2]= 0.748403
```

По умолчанию система **Mathematica** выдает на экран результат с шестью значащими цифрами. Выведем полученное значение в виде числа, содержащего 9 цифр, 8 из которых находятся в дробной части.

```
In[3]:= PaddedForm[int1, {9, 8}]
        форма числа с заполнением нулями
                                        Out[3]//PaddedForm=
                                                0.74840290
```

б) вычислим интеграл с помощью встроенной функции численного интегрирования **NIntegrate**:

```
In[4]:= int2 = NIntegrate[f[x], {x, a, b}]
        квадратурное интегрирование     Out[4]= 0.746824

In[5]:= PaddedForm[int2, {9, 8}]
        форма числа с заполнением нулями
                                        Out[5]//PaddedForm=
                                                0.74682413
```

Сравним полученные значения:

```
In[6]:= Abs[int1 - int2]
        абсолютное значение             Out[6]= 0.00157877
```

▲

**Пример 6.3.** Вычислить определенный интеграл $\int_1^2 \frac{\sin x}{x}\,dx$ с помощью квадратурной формулы Гаусса с пятью узлами.

Δ Квадратурная формула Гаусса с пятью узлами для функции $f(x)$ на отрезке $[a, b]$ имеет вид

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\sum_{i=1}^{5} A_i f\left(\frac{b+a}{2}+\frac{b-a}{2}\cdot t_i\right),$$

где $t_i$ – корни полинома Лежандра пятой степени; $A_i$ – квадратурные коэффициенты Гаусса, определяемые из системы линейных уравнений

$$\begin{cases} \sum_{i=1}^{5} A_i = 2, \\ \sum_{i=1}^{5} A_i\, t_i^{k-1} = \dfrac{1-(-1)^k}{k}, \quad k = \overline{2,5}. \end{cases}$$

Найдем $t_i$ и $A_i$, используя средства пакета **Mathematica**.

Полином Лежандра пятой степени имеет вид

```
In[1]:= LegendreP[5, t]
        P-функция Лежандра первого рода
Out[1]= 1/8 (15 t - 70 t^3 + 63 t^5)
```

Список его корней сохраним под именем **tt**:

```
In[2]:= sl = NSolve[LegendreP[5, t] == 0, t]
        числе··· P-функция Лежандра первого
Out[2]= {{t → -0.90618}, {t → -0.538469}, {t → 0.}, {t → 0.538469}, {t → 0.90618}}

In[3]:= tt = t /. sl
Out[3]= {-0.90618, -0.538469, 0., 0.538469, 0.90618}
```

Для определения квадратурных коэффициентов Гаусса $A_i$ решим линейную систему. Введем матрицу ее коэффициентов **T** и столбец свободных членов **B**:

```
In[4]:= T = Table[If[i == 1, 1, (tt[[j]])^(i-1)], {i, 5}, {j, 5}]; MatrixForm[T]
        табл··· условный оператор                                  матричная форма
Out[4]//MatrixForm=
        1          1          1          1          1
-0.90618   -0.538469  0.     0.538469   0.90618
0.821162   0.289949   0.     0.289949   0.821162
-0.74412   -0.156129  0.     0.156129   0.74412
0.674307   0.0840705  0.     0.0840705  0.674307

In[5]:= B = Table[If[EvenQ[i] == True, 0, 2/i], {i, 5}] // N
        табл··· у··· чётное чис··· истина                  чи
Out[5]= {2., 0., 0.666667, 0., 0.4}
```

Решение сохраним под именем **A**:

```
In[6]:= A = LinearSolve[T, B]
        решить линейные урави

Out[6]= {0.236927, 0.478629, 0.568889, 0.478629, 0.236927}
```

Далее введем подынтегральную функцию и отрезок интегрирования, вычислим определенный интеграл по формуле Гаусса:

$$\text{In[7]:=}\quad f[x\_] = \frac{\text{Sin}[x]}{x};\ a = 1;\ b = 2;$$

$$\text{In[8]:=}\quad \text{int} = \frac{b-a}{2} * \sum_{i=1}^{5} A[[i]] * f\left[\frac{b+a}{2} + \frac{b-a}{2} * tt[[i]]\right]$$

```
Out[8]= 0.65933
```

Выведем полученное значение в виде числа, содержащего 19 цифр, 18 из которых находятся в дробной части:

```
In[9]:= PaddedForm[int, {19, 18}]
        форма числа с заполнением нулями

Out[9]//PaddedForm=
        0.659329906435518700
```

Найдем значение этого интеграла, записав его в привычном виде с помощью палитры инструментов:

$$\text{In[10]:=}\quad \int_1^2 \frac{\text{Sin}[x]}{x}\, dx$$

```
Out[10]= -SinIntegral[1] + SinIntegral[2]
```

Система **Mathematica** выдает результат через встроенную специальную функцию **SinIntegral** (интегральный синус). Получим его численное значение:

```
In[11]:= N[-SinIntegral[1] + SinIntegral[2]]
         ч··· интегральный синус  интегральный синус

Out[11]= 0.65933
```

Рассмотрим еще один вариант вычисления этого интеграла – с помощью функции **NIntegrate**:

$$\text{In[12]:=}\quad \text{int1} = \text{NIntegrate}\left[\frac{\text{Sin}[x]}{x}, \{x, 1, 2\}\right]$$

квадратурное интегрирование

```
Out[12]= 0.65933
```

Выведем это значение в виде числа, содержащего 19 цифр, 18 из которых находятся в дробной части:

```
In[13]:= PaddedForm[int1, {19, 18}]
         форма числа с заполнением нулями
Out[13]//PaddedForm=
         0.659329906435512600
```

Сравним полученные результаты:

```
In[14]:= Abs[int - int1]
         абсолютное значение
Out[14]= 6.10623 x 10^-15
```

Столь малая погрешность объясняется тем, что система **Mathematica** по умолчанию выбирает наиболее подходящий (точный) метод численного нахождения интеграла, как правило, – квадратурный метод Гаусса. ▲

# Лабораторная работа 5

## Численное дифференцирование и интегрирование

**1.** Найти приближенные значения производных первого и второго порядков функции $f(x)$ в точке $x_0$, используя:

а) функцию **D** системы **Mathematica**;

б) формулы численного дифференцирования $y_i' \approx \frac{1}{h}\left(\Delta y_i - \frac{1}{2}\Delta^2 y_i + \frac{1}{3}\Delta^3 y_i\right)$ и $y_i'' \approx \frac{1}{h^2}\left(\Delta^2 y_i - \Delta^3 y_i\right)$ для шага $h_1 = 0{,}1$ и шага $h_2 = 0{,}01$.

Сравнить полученные значения.

1.1. $f(x) = \operatorname{tg}\ln 3x$, $x_0 = 2{,}13$.

1.2. $f(x) = 2\sin\sqrt[3]{x+1}$, $x_0 = 4{,}71$.

1.3. $f(x) = \sqrt[3]{\ln^2(\cos 5x + 4)}$, $x_0 = 1{,}82$.

1.4. $f(x) = \cos^2(\sin(2x-1))$, $x_0 = 9{,}04$.

1.5. $f(x) = (x-3)e^{\sin(x^2+2)}$, $x_0 = 5{,}12$.

1.6. $f(x) = \frac{\log_2^2(3x+2)}{x+4}$, $x_0 = 1{,}13$.

1.7. $f(x) = \operatorname{ch}^2\sqrt{x+7}$, $x_0 = 2{,}21$.

1.8. $f(x) = \sqrt{x}\cos\sqrt{x}$, $x_0 = 5{,}37$.

1.9. $f(x) = x^3 \sin\ln 2x$, $x_0 = 0{,}42$.

1.10. $f(x) = \operatorname{ctg}\sqrt{x+2}$, $x_0 = 4{,}31$.

1.11. $f(x) = 2^{\sin\left(\ln^2(x+1)\right)}$, $x_0 = 3{,}07$.

1.12. $f(x) = \operatorname{sh}(2\sqrt{x}-1)$, $x_0 = 2{,}24$.

1.13. $f(x) = \sqrt{5x + \ln(x^3+1)}$, $x_0 = 2{,}47$.

1.14. $f(x) = \cos\sqrt{x^2+1}$, $x_0 = 0{,}75$.

1.15. $f(x) = \sqrt{x^2 + 3^x}$, $x_0 = 1{,}17$.

1.16. $f(x) = \sin^2\ln(x+2)$, $x_0 = 4{,}23$.

**2.** Выполнить следующие задания:

а) вычислить с помощью формулы *второго порядка точности* и составить таблицу приближенных значений $y_i'$ производной функции $f(x)$ на отрезке $[-1, 3]$ с шагом $h = 0{,}2$;

б) изобразить на одном чертеже (на отрезке $[-1, 3]$) график функции $f'(x)$, полученной с помощью функции **D** пакета **Mathematica**, и точки $(x_i, y_i')$, соответствующие приближенным значениям производной, найденные в п. «а»).

2.1. $f(x) = \operatorname{sh}(\ln \operatorname{ch} x)$.

2.2. $f(x) = \sqrt[3]{2x+5}\cdot\log_2(x+4)$.

2.3. $f(x) = x\operatorname{ch}\frac{1}{x-5}$.

2.4. $f(x) = \sqrt[5]{\operatorname{sh} 2\sqrt[3]{x+3}}$.

2.5. $f(x)=\cos(2x+3)\cdot\ln(3x+7)$.

2.6. $f(x)=x\sin(2x+\cos x)$.

2.7. $f(x)=\ln\dfrac{3\operatorname{ch}x+4}{x^2+1}$.

2.8. $f(x)=\dfrac{\operatorname{sh}(0{,}2x+1)}{\sqrt[5]{\sin^2 x+1}}$.

2.9. $f(x)=(2x-3)\log_3(5x^2+1)$.

2.10. $f(x)=x\sin\big(x\ln(0{,}1x+7)\big)$.

2.11. $f(x)=3^{\sin^2(0{,}6x-1)}$.

2.12. $f(x)=\ln(x^3+5)\cdot\operatorname{ch}\sqrt{x+2}$.

2.13. $f(x)=\dfrac{\sqrt[3]{2x^3+5}}{\sin x+3}$.

2.14. $f(x)=\operatorname{ctg}^2\sqrt{2x^2+15}$.

2.15. $f(x)=\sqrt{x^3+2}\cdot\sin(7x+1)$.

2.16. $f(x)=\ln(4x^2+6)\cdot\cos 2^{x-4}$.

**3.** Вычислить определенный интеграл:

а) по формуле средних прямоугольников;

б) по формуле трапеций.

В обоих случаях использовать двойной просчет при $n_1=8$ и $n_2=10$ для уточнения значения интеграла по Ричардсону.

3.1. $\int\limits_{0,7}^{1,5}\dfrac{\sqrt[3]{x^2+6}}{3x+\sqrt{x^2+0{,}6}}dx$.

3.2. $\int\limits_{1,4}^{2,2}\dfrac{\sqrt{0{,}8x+1}}{0{,}4+\sqrt{2x^2+1{,}3}}dx$.

3.3. $\int\limits_{0,3}^{0,9}\dfrac{2+\sqrt[4]{3x+1{,}7}}{x+\sqrt{1{,}5x^2+1}}dx$.

3.4. $\int\limits_{3}^{4,2}\dfrac{x+2\sqrt[5]{x^2+0{,}8}}{0{,}2x^2+\sqrt{3x+1}}dx$.

3.5. $\int\limits_{1,2}^{2}\dfrac{\sqrt{4x^2+1{,}5}}{4x+\sqrt{0{,}6x^2+3}}dx$.

3.6. $\int\limits_{0,7}^{2,1}\dfrac{1+\sqrt{2{,}3x^2+0{,}2}}{2{,}4+\sqrt{1{,}6x+1{,}7}}dx$.

3.7. $\int\limits_{0,8}^{2,4}\dfrac{0{,}1x+\sqrt[4]{x+2{,}7}}{5+\sqrt{3x+0{,}1}}dx$.

3.8. $\int\limits_{0,8}^{1,6}\dfrac{\sqrt{2x+2{,}8}}{0{,}7x^2+\sqrt{x^2+1{,}3}}dx$.

3.9. $\int\limits_{0,4}^{1,2}\dfrac{\sqrt{3x^2+4{,}1}}{0{,}2x+\sqrt{1{,}5x^2+6}}dx$.

3.10. $\int\limits_{0,3}^{1,1}\dfrac{x+\sqrt[3]{5x^2+1{,}6}}{0{,}4x^2+\sqrt{1{,}9x+2}}dx$.

3.11. $\int\limits_{1,3}^{2,7}\dfrac{\sqrt{1{,}2x^2+0{,}7}}{1{,}5x+\sqrt{2x^2+0{,}9}}dx$.

3.12. $\int\limits_{1,1}^{2,5}\dfrac{2{,}4+\sqrt[4]{x^2+1{,}5}}{0{,}8x+\sqrt{5x+0{,}4}}dx$.

3.13. $\int\limits_{0,8}^{1,6}\dfrac{x+\sqrt{1{,}4x+3{,}1}}{3+\sqrt{x^2+2{,}7}}dx$.

3.14. $\int\limits_{0,9}^{2,1}\dfrac{\sqrt{4x^2+0{,}6}}{2x+\sqrt{0{,}3x^2+1{,}3}}dx$.

3.15. $\int\limits_{1,3}^{1,9}\dfrac{1{,}1+\sqrt[3]{x^2+4{,}8}}{2x+\sqrt{x+3{,}1}}dx$.

3.16. $\int\limits_{0,4}^{1,2}\dfrac{x+\sqrt{5x+2{,}4}}{2{,}7+\sqrt{4x^2+0{,}9}}dx$.

**4.** Вычислить определенный интеграл от таблично заданной функции по формуле Симпсона (парабол) для разбиений отрезка интегрирования на 8 и на 16 частей.

| № | **4.1** | | **4.2** | | **4.3** | | **4.4** | |
|---|---|---|---|---|---|---|---|---|
| | ***x*** | ***y*** | ***x*** | ***y*** | ***x*** | ***y*** | ***x*** | ***y*** |
| 1 | 1.2 | 0.1823 | 1.1 | 2.1591 | -1 | 0.3499 | 0.3 | -1.1052 |
| 2 | 1.3 | 0.2624 | 1.175 | 1.7421 | -0.95 | 0.4055 | 0.348 | -1.0799 |
| 3 | 1.4 | 0.3365 | 1.25 | 1.2749 | -0.9 | 0.3886 | 0.396 | -1.1225 |
| 4 | 1.5 | 0.4055 | 1.325 | 0.7737 | -0.85 | 0.4459 | 0.444 | -1.0598 |
| 5 | 1.6 | 0.4700 | 1.4 | 0.2616 | -0.8 | 0.4317 | 0.492 | -1.0678 |
| 6 | 1.7 | 0.5306 | 1.475 | -0.2309 | -0.75 | 0.4904 | 0.54 | -0.9782 |
| 7 | 1.8 | 0.5878 | 1.55 | -0.6672 | -0.7 | 0.4795 | 0.588 | -0.9554 |
| 8 | 1.9 | 0.6419 | 1.625 | -1.0064 | -0.65 | 0.5392 | 0.636 | -0.8462 |
| 9 | 2 | 0.6931 | 1.7 | -1.2056 | -0.6 | 0.5325 | 0.684 | -0.7949 |
| 10 | 2.1 | 0.7419 | 1.775 | -1.2249 | -0.55 | 0.5930 | 0.732 | -0.6713 |
| 11 | 2.2 | 0.7885 | 1.85 | -1.0324 | -0.5 | 0.5915 | 0.78 | -0. 5930 |
| 12 | 2.3 | 0.8329 | 1.925 | -0.6103 | -0.45 | 0.6521 | 0.828 | -0.4594 |
| 13 | 2.4 | 0.8755 | 2 | 0.0391 | -0.4 | 0.6570 | 0.876 | -0.3548 |
| 14 | 2.5 | 0.9163 | 2.075 | 0.8881 | -0.35 | 0.7171 | 0.924 | -0.2147 |
| 15 | 2.6 | 0.9555 | 2.15 | 1.8807 | -0.3 | 0.7297 | 0.972 | -0.0844 |
| 16 | 2.7 | 0.9933 | 2.225 | 2.9330 | -0.25 | 0.7885 | 1.02 | 0.0593 |
| 17 | 2.8 | 1.0296 | 2.3 | 3.9381 | -0.2 | 0.8105 | 1.068 | 0.2150 |

| № | **4.5** | | **4.6** | | **4.7** | | **4.8** | |
|---|---|---|---|---|---|---|---|---|
| | ***x*** | ***y*** | ***x*** | ***y*** | ***x*** | ***y*** | ***x*** | ***y*** |
| 1 | 0.2 | 1.2336 | 0.5 | 1.2506 | 0.5 | 0.4431 | 1.04 | 0.9519 |
| 2 | 0.25 | 1.2680 | 0.552 | 1.2938 | 0.56 | 0.4697 | 1.1 | 0.9181 |
| 3 | 0.3 | 1.3702 | 0.604 | 1.3828 | 0.62 | 0.5096 | 1.16 | 0.8534 |
| 4 | 0.35 | 1.3944 | 0.656 | 1.4115 | 0.68 | 0.5234 | 1.22 | 0.8278 |
| 5 | 0.4 | 1.5219 | 0.708 | 1.4927 | 0.74 | 0.5519 | 1.28 | 0.7734 |
| 6 | 0.45 | 1.5334 | 0.76 | 1.5111 | 0.8 | 0.5517 | 1.34 | 0.7537 |
| 7 | 0.5 | 1.6904 | 0.812 | 1.5877 | 0.86 | 0.5667 | 1.4 | 0.7071 |
| 8 | 0.55 | 1.6862 | 0.864 | 1.5993 | 0.92 | 0.5517 | 1.46 | 0.6917 |
| 9 | 0.6 | 1.8776 | 0.916 | 1.6742 | 0.98 | 0.5513 | 1.52 | 0.6513 |
| 10 | 0.65 | 1.8542 | 0.968 | 1.6819 | 1.04 | 0.5212 | 1.58 | 0.6392 |
| 11 | 0.7 | 2.0854 | 1.02 | 1.7575 | 1.1 | 0.5039 | 1.64 | 0.6036 |
| 12 | 0.75 | 2.0390 | 1.072 | 1.7640 | 1.16 | 0.4586 | 1.7 | 0.5941 |
| 13 | 0.8 | 2.3163 | 1.124 | 1.8428 | 1.22 | 0.4234 | 1.76 | 0.5625 |
| 14 | 0.85 | 2.2423 | 1.176 | 1.8501 | 1.28 | 0.3633 | 1.82 | 0.5549 |
| 15 | 0.9 | 2.5728 | 1.228 | 1.9344 | 1.34 | 0.3095 | 1.88 | 0.5265 |
| 16 | 0.95 | 2.4657 | 1.28 | 1.9446 | 1.4 | 0.2355 | 1.94 | 0.5206 |
| 17 | 1 | 2.8576 | 1.332 | 2.0366 | 1.46 | 0.1630 | 2 | 0.4950 |

| № | 4.9 | | 4.10 | | 4.11 | | 4.12 | |
|---|---|---|---|---|---|---|---|---|
| | ***x*** | ***y*** | ***x*** | ***y*** | ***x*** | ***y*** | ***x*** | ***y*** |
| 1 | 0.1 | 0.4664 | -0.5 | -0.6966 | -0.2 | 0.3437 | 0.484 | 0.2159 |
| 2 | 0.156 | 0.5356 | -0.42 | -0.5420 | -0.124 | 0.2474 | 0.56 | 0.2887 |
| 3 | 0.212 | 0.5992 | -0.34 | -0.4175 | -0.048 | 0.1327 | 0.636 | 0.3566 |
| 4 | 0.268 | 0.6415 | -0.26 | -0.2996 | 0.028 | 0.0917 | 0.712 | 0.4449 |
| 5 | 0.324 | 0.6902 | -0.18 | -0.1994 | 0.104 | 0.2222 | 0.788 | 0.5206 |
| 6 | 0.38 | 0.7206 | -0.1 | -0.1048 | 0.18 | 0.3172 | 0.864 | 0.6218 |
| 7 | 0.436 | 0.7620 | -0.02 | -0.0203 | 0.256 | 0.4051 | 0.94 | 0.7021 |
| 8 | 0.492 | 0.7854 | 0.06 | 0.05797 | 0.332 | 0.4770 | 1.016 | 0.8140 |
| 9 | 0.548 | 0.8224 | 0.14 | 0.1316 | 0.408 | 0.5528 | 1.092 | 0.8965 |
| 10 | 0.604 | 0.8410 | 0.22 | 0.1978 | 0.484 | 0.6133 | 1.168 | 1.0177 |
| 11 | 0.66 | 0.8750 | 0.3 | 0.2636 | 0.56 | 0.6827 | 1.244 | 1.1006 |
| 12 | 0.716 | 0.8901 | 0.38 | 0.3204 | 0.636 | 0.7358 | 1.32 | 1.2298 |
| 13 | 0.772 | 0.9219 | 0.46 | 0.3803 | 0.712 | 0.8013 | 1.396 | 1.3118 |
| 14 | 0.828 | 0.9343 | 0.54 | 0.4296 | 0.788 | 0.8488 | 1.472 | 1.4481 |
| 15 | 0.884 | 0.9645 | 0.62 | 0.4848 | 0.864 | 0.9116 | 1.548 | 1.5282 |
| 16 | 0.94 | 0.9746 | 0.7 | 0.5279 | 0.94 | 0.9547 | 1.624 | 1.6711 |
| 17 | 0.996 | 1.0036 | 0.78 | 0.5794 | 1.016 | 1.0156 | 1.7 | 1.7487 |

| № | 4.13 | | 4.14 | | 4.15 | | 4.16 | |
|---|---|---|---|---|---|---|---|---|
| | ***x*** | ***y*** | ***x*** | ***y*** | ***x*** | ***y*** | ***x*** | ***y*** |
| 1 | -0.5 | 0.5359 | -1.3 | 4.8232 | 0.292 | 0.3799 | 1 | 1.0100 |
| 2 | -0.42 | 0.4851 | -1.228 | 3.3376 | 0.38 | 0.3990 | 1.04 | 0.9153 |
| 3 | -0.34 | 0.4371 | -1.156 | 2.7039 | 0.468 | 0.4055 | 1.08 | 0.8659 |
| 4 | -0.26 | 0.3716 | -1.084 | 1.9866 | 0.556 | 0.4319 | 1.12 | 0.7892 |
| 5 | -0.18 | 0.3017 | -1.012 | 1.6670 | 0.644 | 0.4449 | 1.16 | 0.7505 |
| 6 | -0.1 | 0.2073 | -0.94 | 1.2484 | 0.732 | 0.4800 | 1.2 | 0.6875 |
| 7 | -0.02 | 0.0735 | -0.868 | 1.0554 | 0.82 | 0.5004 | 1.24 | 0.6568 |
| 8 | 0.06 | 0.1555 | -0.796 | 0.7886 | 0.908 | 0.5458 | 1.28 | 0.6042 |
| 9 | 0.14 | 0.2839 | -0.724 | 0.6593 | 0.996 | 0.5748 | 1.32 | 0.5796 |
| 10 | 0.22 | 0.3901 | -0.652 | 0.4827 | 1.084 | 0.6326 | 1.36 | 0.5352 |
| 11 | 0.3 | 0.4977 | -0.58 | 0.3913 | 1.172 | 0.6714 | 1.4 | 0.5153 |
| 12 | 0.38 | 0.5925 | -0.508 | 0.2743 | 1.26 | 0.7438 | 1.44 | 0.4774 |
| 13 | 0.46 | 0.6981 | -0.436 | 0.2092 | 1.348 | 0.7937 | 1.48 | 0.4611 |
| 14 | 0.54 | 0.7899 | -0.364 | 0.1345 | 1.436 | 0.8829 | 1.52 | 0.4284 |
| 15 | 0.62 | 0.8984 | -0.292 | 0.0904 | 1.524 | 0.9447 | 1.56 | 0.4150 |
| 16 | 0.7 | 0.9905 | -0.22 | 0.0477 | 1.612 | 1.0524 | 1.6 | 0.3867 |
| 17 | 0.78 | 1.1044 | -0.148 | 0.0227 | 1.7 | 1.1261 | 1.64 | 0.3755 |

**5.** Вычислить определенный интеграл с помощью квадратурной формулы Гаусса с *n* узлами. Выполнить проверку, применив функцию **NIntegrate**.

5.1. $\int_{1,1}^{2} \frac{\ln 2x}{4x+1}\,dx,\ \ n=4.$

5.2. $\int_{1,3}^{2,7} \sqrt{x+1}\sin^2(2x-1)\,dx,\ \ n=6.$

5.3. $\int_{0,8}^{2,1} \frac{\cos(5x+2)}{x}\,dx,\ \ n=7.$

5.4. $\int_{1,4}^{2,9} \frac{e^{-x^2+5x}}{3x+0{,}8}\,dx,\ \ n=4.$

5.5. $\int_{1}^{2,1} \sqrt{2x+1}\ln(x+3)\,dx,\ \ n=6.$

5.6. $\int_{1,3}^{2,6} \frac{\mathrm{ch}(x+3)}{5x+2}\,dx,\ \ n=7.$

5.7. $\int_{0,4}^{3} \frac{\mathrm{sh}(2x-3)}{x+1}\,dx,\ \ n=4.$

5.8. $\int_{1,6}^{3,2} \frac{3+\cos(x+2)}{x+2{,}4}\,dx,\ \ n=6.$

5.9. $\int_{0,2}^{1,1} \frac{e^{2x-1}}{4x+2{,}8}\,dx,\ \ n=7.$

5.10. $\int_{1,5}^{2,9} \frac{x-2\sin(4x+3)}{1{,}6x+0{,}7}\,dx,\ \ n=4.$

5.11. $\int_{1,7}^{3,1} \frac{x+\ln(x+1{,}2)}{1{,}5x^2+1}\,dx,\ \ n=6.$

5.12. $\int_{0,8}^{2,4} \frac{\mathrm{sh}(3x+0{,}4)}{5x+4{,}1}\,dx,\ \ n=7.$

5.13. $\int_{0,6}^{2,5} (x+1)\cos(x^2+2)\,dx,\ \ n=4.$

5.14. $\int_{2}^{3,2} \frac{\sin(3x-1)}{2x+3}\,dx,\ \ n=6.$

5.15. $\int_{0,2}^{1} \frac{\ln 3x^2}{x^2+1}\,dx,\ \ n=7.$

5.16. $\int_{1,5}^{2,8} \sqrt{x+2{,}5}\,\mathrm{ch}(x-0{,}8)\,dx,\ \ n=4.$

# 7. Численные методы решения обыкновенных дифференциальных уравнений и их систем

## Численное решение задачи Коши для обыкновенных дифференциальных уравнений

Обыкновенным дифференциальным уравнением (ОДУ) $n$-го порядка называется уравнение вида

$$F(x, y, y', ..., y^{(n)}) = 0,$$

где $x$ – независимая переменная; $y(x)$ – искомая функция; $y', ..., y^{(n)}$ – ее производные; $F$ – заданная функция $(n+2)$-х переменных.

Если дифференциальное уравнение $n$-го порядка можно записать в виде

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

то такое уравнение называется уравнением *в нормальной форме* или уравнением, *разрешенным относительно старшей производной*.

*Решением* дифференциального уравнения $n$-го порядка на интервале $(a, b)$ называется $n$ раз непрерывно дифференцируемая функция $y = \varphi(x)$, которая обращает данное уравнение в тождество на этом интервале.

**Задача Коши** (или *начальная задача*) для обыкновенного дифференциального уравнения $n$-го порядка состоит в следующем: найти решение уравнения

$$F(x, y, y', ..., y^{(n)}) = 0,$$

удовлетворяющее *начальным условиям*

$$y(x_0) = y_0, \quad y'(x_0) = y_1, \ \ldots, \quad y^{(n-1)}(x_0) = y_{n-1},$$

где $x_0$, $y_0$, $y_1$, $y_{n-1}$ – заданные числа.

Рассмотрим задачу Коши для ОДУ первого порядка:

$$\begin{cases} y' = f(x, y), \\ y(x_0) = y_0. \end{cases}$$

Будем считать, что функция $f(x, y)$ в некоторой области удовлетворяет всем необходимым требованиям и задача поставлена корректно, т. е. решение задачи Коши существует и единственно. В большинстве случаев интегрирование таких уравнений невозможно не только в элементарных, но и в специальных функциях. Поэтому были созданы различные методы приближенного решения дифференциальных уравнений – аналитические, графические, численные. В результате применения численных методов искомую функцию получают ***в табличном виде***, т. е. в виде таблиц значений функции $y(x)$ в узловых точках $x_i$.

Выберем достаточно малый шаг $h$ и построим систему равноотстоящих точек (сетку):

$$x_i = x_0 + ih, \quad i = 0, 1, 2, ...,$$

которые называются узлами сетки.

Выполним дискретизацию задачи Коши, т. е. заменим дифференциальное уравнение разностным уравнением

$$y_{i+1} = \Phi(x_i, y_{i-p+1}, y_{i-p+2}, ..., y_i, y_{i+1}) \qquad (i \geq p-1),$$

которое необходимо решить на каждом шаге для нахождения $y_{i+1}$.

Выбор функции $\Phi$ определяет метод численного решения дифференциального уравнения. Если она не зависит от $y_{i+1}$, то получают ***явный*** метод (явную формулу для вычисления $y_{i+1}$), в противном случае – ***неявный*** метод. Метод, дающий формулу для вычисления $y_{i+1}$ по $m$ предыдущим значениям $y_{i-m+1}$, $y_{i-m+2}$, …, $y_i$, называется $m$-шаговым. Существуют две группы численных методов решения задачи Коши: одношаговые (или методы Рунге – Кутта) и многошаговые разностные методы.

Говорят, что ***метод сходится*** *в точке* $x^*$, если построена последовательность сеток, таких что

$$x^* = x_0 + nh \quad (h \to 0, n \to \infty) \quad \text{и} \quad y(x^*) - y_n \to 0 \quad \text{при} \quad h \to 0.$$

Если существует такое $p > 0$, что $y(x^*) - y_n = O(h^p)$ при $h \to 0$, то говорят, что метод имеет ***p-й порядок точности***.

Погрешность метода численного решения в точке $x_i$ определяется нормой разности $y(x_i) - y_i$, где $y(x_i)$ – точное, а $y_i$ – приближенное решение. ***Локальной погрешностью*** называют ошибку на данном шаге при условии, что предыдущие значения верны. ***Глобальная погрешность*** – это разность между вычисленным и точным решением, определяемым начальным условием.

## Метод Эйлера (метод ломаных)

Этот метод является простейшим явным одношаговым методом.

Найдем решение уравнения $y' = f(x, y)$, удовлетворяющее начальному условию $y(x_0) = y_0$.

Согласно методу Эйлера уравнение $y' = f(x, y)$ заменяется разностным уравнением

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i),$$

где $x_i = x_0 + ih$, $i = 0, 1, 2, ....$ Решение этого уравнения находится явным образом по рекуррентной формуле

$$y_{i+1} = y_i + hf(x_i, y_i), \quad i = 0, 1, 2, ...$$

Геометрически метод Эйлера состоит в том, что интегральную кривую $y = y(x)$, проходящую через точку $(x_0, y_0)$ (т. е. график решения задачи Коши), заменяют ломаной с вершинами в точках $M_i(x_i, y_i)$. Эта ломаная называется ***ломаной Эйлера***. Ее звенья $M_iM_{i+1}$ – отрезки прямых с угловыми коэффициентами $f(x_i, y_i)$.

На каждом шаге метод Эйлера дает ***погрешность*** порядка $h^2$ (***локальная*** погрешность).

***Глобальная погрешность*** метода имеет первый порядок, т. е. равна $O(h)$.

## Метод Эйлера – Коши

Этот метод является модификацией метода Эйлера.

Сначала в точке $x_{i+1}$ находят «грубое» приближение (или *предиктор*) к значению $y(x_{i+1})$ по формуле метода Эйлера:

$$y_{i+1}^{(0)} = y_i + hf(x_i, y_i),$$

которое затем уточняют по формуле

$$y_{i+1} = y_i + \frac{h}{2}\left(f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(0)})\right),$$

получая так называемый *корректор*.

***Локальная*** погрешность метода – $O(h^3)$, ***глобальная*** погрешность имеет второй порядок, т. е. равна $O(h^2)$.

## Метод Рунге – Кутта

Идея построения методов Рунге – Кутта порядка $p$ состоит в следующем. Приближения к решениям $y(x_{i+1})$ ищут по формуле вида

$$y_{i+1} = y_i + h\varphi(x_i, y_i, h),$$

где $\varphi(x, y, h)$ – некоторая функция, приближающая отрезок ряда Тейлора $p$-го порядка и не содержащая частных производных функции $f(x, y)$:

$$y(x_{i+1}) = y(x_i) + hy'(x_i) + \frac{1}{2}h^2y''(x_i) + \ldots + \frac{1}{p!}h^p y^{(p)}(x_i) + o(h^{p+1}).$$

Метод Эйлера является методом Рунге – Кутта первого порядка точности. Для построения методов Рунге – Кутта порядка $p > 1$ функцию $\varphi(x, y, h)$ полагают зависящей от нескольких параметров. Их значения подбирают, сравнивая выражение $y_{i+1} = y_i + h\varphi(x_i, y_i, h)$ с многочленом Тейлора степени $p$.

Расчетные формулы ***метода Рунге – Кутта четвертого порядка*** имеют вид

$$y_{i+1} = y_i + \frac{1}{6}\left(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}\right),$$

где

$$k_1^{(i)} = hf(x_i, y_i), \quad k_2^{(i)} = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1^{(i)}}{2}\right),$$

$$k_3^{(i)} = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2^{(i)}}{2}\right), \quad k_4^{(i)} = hf\left(x_i + h, y_i + k_3^{(i)}\right),$$

причем $x_i = x_0 + ih$, $i = 0, 1, 2, ...$

***Локальная*** погрешность метода – $O(h^5)$, ***глобальная*** погрешность метода равна $O(h^4)$.

Формулы более высокого порядка точности, как правило, не используются в связи с их громоздкостью, возрастающей значительно быстрее, чем точность формулы.

## Численное решение задачи Коши для системы обыкновенных дифференциальных уравнений

### Метод Эйлера

Формула Эйлера легко обобщается на случай систем обыкновенных дифференциальных уравнений в нормальной форме:

$$\begin{cases} y_1' = f_1(x, y_1, ..., y_n), \\ y_2' = f_2(x, y_1, ..., y_n), \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ y_n' = f_n(x, y_1, ..., y_n) \end{cases}$$

с начальными условиями

$$y_1(x_0) = y_1^{(0)}, \ \ y_2(x_0) = y_2^{(0)}, \ \ldots, \ y_n(x_0) = y_n^{(0)}.$$

Приближенные значения $y_k^{(i+1)}$ точного решения $y_k(x_{i+1})$ вычисляются по формулам:

$$y_k^{(i+1)} = y_k^{(i)} + h\, f_k(x_i, y_1^{(i)}, y_2^{(i)}, ..., y_n^{(i)}), \quad k = \overline{1, n}, \quad i = 0, 1, 2, ...$$

В частности, для системы двух уравнений вида

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

с начальными условиями

$$y(x_0) = y_0, \quad z(x_0) = z_0,$$

расчетные формулы метода Эйлера имеют вид

$$y_{i+1} = y_i + h\, f(x_i, y_i, z_i), \quad z_{i+1} = z_i + h\, g(x_i, y_i, z_i),$$

где $x_i = x_0 + ih;\ i = 0, 1, 2, \ldots$ .

## Метод Рунге – Кутта

Расчетные формулы метода Рунге – Кутта четвертого порядка для системы двух уравнений

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

с начальными условиями

$$y(x_0) = y_0, \quad z(x_0) = z_0$$

имеют вид

$$y_{i+1} = y_i + \frac{1}{6}\left(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}\right),$$

$$z_{i+1} = z_i + \frac{1}{6}\left(r_1^{(i)} + 2r_2^{(i)} + 2r_3^{(i)} + r_4^{(i)}\right),$$

где

$$k_1^{(i)} = hf(x_i, y_i, z_i), \quad r_1^{(i)} = hg(x_i, y_i, z_i),$$

$$k_2^{(i)} = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1^{(i)}}{2}, z_i + \frac{r_1^{(i)}}{2}\right), \quad r_2^{(i)} = hg\left(x_i + \frac{h}{2}, y_i + \frac{k_1^{(i)}}{2}, z_i + \frac{r_1^{(i)}}{2}\right),$$

$$k_3^{(i)} = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2^{(i)}}{2}, z_i + \frac{r_2^{(i)}}{2}\right), \quad r_3^{(i)} = hg\left(x_i + \frac{h}{2}, y_i + \frac{k_2^{(i)}}{2}, z_i + \frac{r_2^{(i)}}{2}\right),$$

$$k_4^{(i)} = hf\left(x_i + h, y_i + k_3^{(i)}, z_i + r_3^{(i)}\right), \quad r_4^{(i)} = hg\left(x_i + h, y_i + k_3^{(i)}, z_i + r_3^{(i)}\right),$$

причем $x_i = x_0 + ih,\ i = 0, 1, 2, \ldots$ .

## Основные функции пакета Mathematica, используемые при решении дифференциальных уравнений и их систем

***DSolve***[*eqn*, *y*[*x*], *x*] – решает дифференциальное уравнение *eqn* с функцией *y* и независимой переменной *x*.

***DSolve***[*eqn*, *y*[*x*], {*x*, *a*, *b*}] – находит решение дифференциального уравнения *eqn* относительно функции $y(x)$ на отрезке $[a, b]$.

***DSolve***[{*eqn*1,*eqn*2, …}, {*y*1[*x*], *y*2[*x*], …}, *x*] – находит решение системы дифференциальных уравнений *eqn*1, *eqn*2, … относительно функций *y*1, *y*2, … с независимой переменной *x*.

***NDSolve***[*eqn*, *y*[*x*],{*x*, *a*, *b*}] – находит численное решение дифференциального уравнения *eqn* относительно функции $y(x)$ на отрезке $[a, b]$.

***NDSolve***[{*eqn*1,*eqn*2, …}, {*y*1[*x*], *y*2[*x*], …}, {*x*, *a*, *b*}] – находит численное решение системы дифференциальных уравнений *eqn*1, *eqn*2, … относительно функций *y*1, *y*2, … на отрезке $[a, b]$.

***Floor***[*x*] – выделяет целую часть выражения *x*;

***Floor***[*x*, *a*] – определяет наибольшее число, кратное *a*, которое меньше или равно *x*.

***Prepend***[*expr*,*elem*] **–** добавляет в начало списка *expr* список значений *elem*.

***ListPlot***[*list*, ***PlotJoined→True***] – строит точки из списка *list* и соединяет их отрезками прямых.

## Примеры численного решения дифференциальных уравнений и их систем средствами пакета Mathematica

**Пример 7.1.** Решить задачу Коши $y' = -xy$, $y(0) = 1$:

а) методом Эйлера на отрезке $[0, 1]$ с шагом $h = 0{,}1$;

б) с помощью функции **DSolve**.

Сравнить полученные решения в узлах таблицы. Проиллюстрировать графиками.

Δ а) введем функцию $f(x, y)$ – правую часть уравнения, границы отрезка, начальные значения $x_0$, $y_0$, шаг $h$, найдем число точек разбиения отрезка $n$ (не считая $x_0$):

```
In[1]:= f[x_, y_] = -x y;

In[2]:= a = 0; b = 1; x0 = 0; y0 = 1; h = 0.1; n = Floor[(b - a)/h];
                                                  округление вниз
```

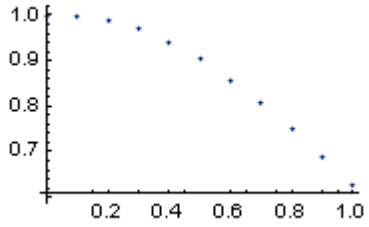
Составим таблицу приближенных значений функции $y(x)$, вычисленных с помощью метода Эйлера:

```
In[3]:= x = x0; y = y0; eul1 = Table[{x, y} = {x + h, y + h f[x, y]}, {i, n}];
                                  таблица значений

In[4]:= eul1 = Prepend[eul1, {x0, y0}]
                      добавить в начало

Out[4]= {{0, 1}, {0.1, 1.}, {0.2, 0.99}, {0.3, 0.9702},
         {0.4, 0.941094}, {0.5, 0.90345}, {0.6, 0.858278},
         {0.7, 0.806781}, {0.8, 0.750306}, {0.9, 0.690282}, {1., 0.628157}}
```

Изобразим полученные точки на графике:

```
In[5]:= gr1 = ListPlot[eul1, ImageSize → Small]
                  диаграмма разб··· размер изо··· малый
```

Out[5]=




б) решим задачу Коши с помощью функции **DSolve**.

```
In[6]:= Clear[x, y]
        очистить

In[7]:= sol = DSolve[{y'[x] == f[x, y[x]], y[x0] == y0}, y[x], x]
                  решить дифференциальные уравнения

Out[7]= {{y[x] → e^(-x^2/2)}}
```

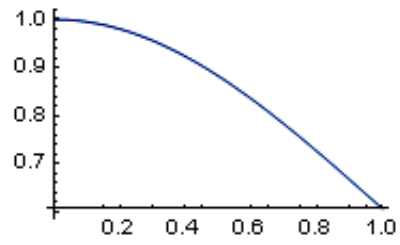
Решение сохраним в виде функции $y1(x)$:

```
In[8]:= y1[x_] = y[x] /. Flatten[sol]
                         уплостить

Out[8]= e^(-x^2/2)
```

Построим ее график на отрезке $[0,1]$:

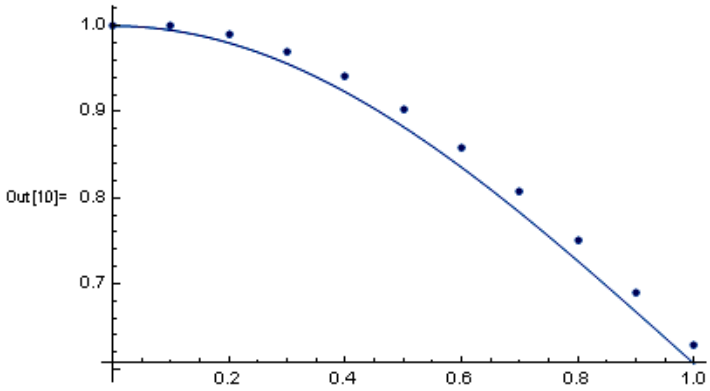
```
In[9]:= gr2 = Plot[y1[x], {x, 0, 1}, ImageSize → Small]
                  график функции  размер изо··· малый
```

Out[9]=



Сравним полученные решения графически:

```
In[10]:= Show[gr1, gr2, ImageSize → Medium]
         показать  размер изо···  средний
```




Оценим погрешность приближенного решения, найдя вектор ошибки и его норму:

```
In[11]:= delta = Table[y1[x0 + (i - 1) h] - eul1[[i, 2]], {i, n + 1}]
          таблица значений

Out[11]= {0., -0.00498752, -0.00980133, -0.0142025, -0.0179777, -0.0209533,
          -0.0230075, -0.0240765, -0.0241574, -0.0233051, -0.0216258}

In[12]:= Norm[delta]
         норма

Out[12]= 0.0615491
```

▲

**Пример 7.2.** Решить задачу Коши $y' = 0{,}3x + y^2$, $y(0) = 0{,}6$:

а) методом Рунге – Кутта четвертого порядка на отрезке $[0, 1]$ с шагом $h = 0{,}1$;

б) с помощью функций **DSolve** и **NDSolve**.

Сравнить полученные решения.

Δ а) введем функцию $f(x, y)$ – правую часть уравнения, границы отрезка, начальные значения $x_0$, $y_0$, шаг $h$, найдем число точек разбиения отрезка $n$ (не считая $x_0$):

```
In[1]:= f[x_, y_] = 0.3 x + y^2;

In[2]:= a = 0; b = 1; x0 = 0; y0 = 0.6; h = 0.1; n = Floor[(b - a)/h];
                                                    округление вниз
```

Организуем цикл и создадим таблицу **sol1** приближенных значений решения дифференциального уравнения, полученных с помощью метода Рунге – Кутта:

```
In[3]:= sol1 = List[{x0, y0}];
        список
In[4]:= x = x0; y = y0;
        For[k = 1, k < n + 1, k++,
        цикл ДЛЯ
          k1[x_, y_] := h * f[x, y];
          k2[x_, y_] := h * f[x + h / 2, y + k1[x, y] / 2];
          k3[x_, y_] := h * f[x + h / 2, y + k2[x, y] / 2];
          k4[x_, y_] := h * f[x + h, y + k3[x, y]];
          x = x + h; y = y + (k1[x, y] + 2 * k2[x, y] + 2 * k3[x, y] + k4[x, y]) / 6;
          sol1 = Append[sol1, {x, y}]]
                 добавить в конец
```

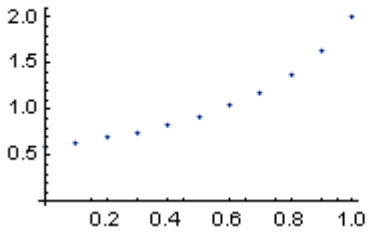
Выведем таблицу:

```
In[6]:= sol1
Out[6]= {{0, 0.6}, {0.1, 0.643059}, {0.2, 0.695247},
         {0.3, 0.758465}, {0.4, 0.835315}, {0.5, 0.929462}, {0.6, 1.04623},
         {0.7, 1.19367}, {0.8, 1.38454}, {0.9, 1.6403}, {1., 2.00017}}
```

Изобразим полученные точки на графике:

```
In[7]:= gr1 = ListPlot[sol1, ImageSize → Small]
          диаграмма разб···  размер изо···  малый
```

Out[7]=




б) решим данную задачу Коши с помощью функций **DSolve** и **NDSolve**. Полученные аналитическое и численное решения обозначим **sol2** и **sol3** соответственно.

```
In[8]:= Clear[x, y]
        очистить
In[9]:= sol2 = DSolve[{y'[x] == f[x, y[x]], y[x0] == y0}, y[x], x];
        решить дифференциальные уравнения
        y1[x_] = y[x] /. Flatten[sol2]
                         уплостить
```

Out[10]=
$$-\left\{\left\{0.273861\left(1.\,x^{3/2}\,\text{BesselJ}\left[-\frac{4}{3},\,0.365148\,x^{3/2}\right]-2.4589\,x^{3/2}\,\text{BesselJ}\left[-\frac{2}{3},\,0.365148\,x^{3/2}\right]+1.82574\,\text{BesselJ}\left[-\frac{1}{3},\,0.365148\,x^{3/2}\right]-1.\,x^{3/2}\,\text{BesselJ}\left[\frac{2}{3},\,0.365148\,x^{3/2}\right]\right)\Big/\left(x\left(1.\,\text{BesselJ}\left[-\frac{1}{3},\,0.365148\,x^{3/2}\right]-1.22945\,\text{BesselJ}\left[\frac{1}{3},\,0.365148\,x^{3/2}\right]\right)\right)\right\}\right\}$$

Как видно, аналитическое решение выражается через функции Бесселя.

Численное решение система **Mathematica** выдает в виде некоторой интерполирующей функции:

```
In[11]:= sol3 = NDSolve[{y'[x] == f[x, y[x]], y[x0] == y0}, y[x], {x, 0, 1}]
          численно решить ДУ
```

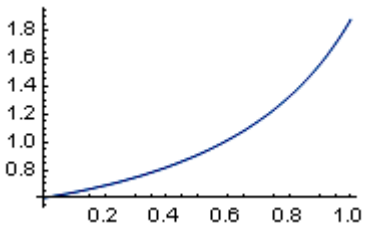
```
Out[11]= {{y[x] → InterpolatingFunction[ Domain: {{0., 1.}}
                                         Output: scalar ][x]}}
```

Заметим, что по умолчанию функция **NDSolve** использует метод Рунге – Кутта.

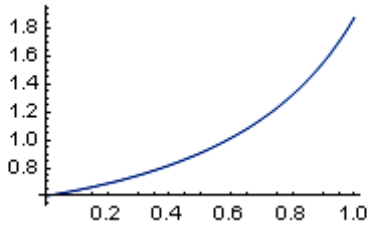
Построим графики этих решений.

```
In[12]:= gr2 = Plot[y1[x], {x, 0, 1}, ImageSize → Small]
          график функции          размер изо···  малый
In[13]:= gr3 = Plot[Evaluate[y[x] /. sol3], {x, 0, 1}, ImageSize → Small]
          гра···  вычислить                      размер изо···  малый
```

Out[12]=

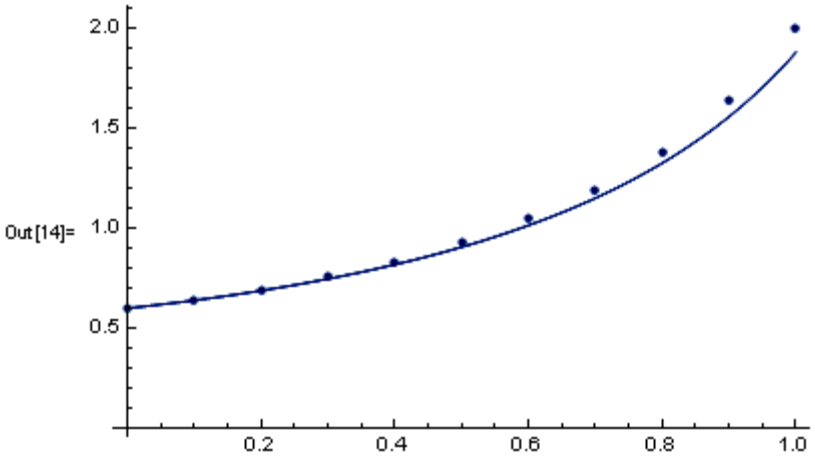



Out[13]=




Сравним решения, полученные тремя способами (методом Рунге – Кутта, с помощью функций **DSolve** и **NDSolve**), графически:

```
In[14]:= Show[gr1, gr2, gr3, ImageSize → Medium]
          показать          размер изо···  средний
```




Графики решений **sol2** и **sol3** совпали.

Очевидно, метод Рунге – Кутта дает более точное приближенное решение, чем метод Эйлера. ▲

# Лабораторная работа 6

## Численное решение задачи Коши для ОДУ первого порядка и их систем

**1.** Решить задачу Коши для дифференциального уравнения первого порядка на отрезке $[0,1]$:

а) методом Эйлера – Коши с шагом $h_1 = 0{,}1$ и $h_2 = 0{,}05$, построить графики полученных решений;

б) методом Рунге – Кутта четвертого порядка с шагом $h_1 = 0{,}1$ и $h_2 = 0{,}05$, построить графики полученных решений;

в) с помощью функций **DSolve** и **NDSolve**, построить графики.

Сравнить все полученные решения. Сделать выводы о точности методов в зависимости от шага сетки.

1.1. $y' = \cos(2x + y) + x - y,\ y(0) = 0.$ 1.2. $y' - 0{,}1x^2 = 2xy,\ y(0) = 0{,}2.$

1.3. $y' + 1{,}25y^2 = 0{,}6\sin x + 1,\ y(0) = 0.$ 1.4. $y' = 0{,}3xy + y^2,\ y(0) = 0{,}4.$

1.5. $y' = \dfrac{\cos y}{x+2} - 0{,}3y^2,\ y(0) = 0.$ 1.6. $y' = 1 + 4y\sin x - 1{,}5y^2,\ y(0) = 0.$

1.7. $y' = \cos(x + y) - x - y,\ y(0) = 0.$ 1.8. $y' - 2x = y^2,\ y(0) = 0{,}3.$

1.9. $y' = 5\cos y - \sin x + x^2,\ y(0) = 0{,}5.$ 1.10. $y' = y - \dfrac{3x}{y},\ y(0) = 1.$

1.11. $y' = x^3 + y^2,\ y(0) = 1.$ 1.12. $y' = 0{,}2x^2 + 5y^2,\ y(0) = 0{,}1.$

1.13. $y' = y^2 + 0{,}7\sin 2x,\ y(0) = 0.$ 1.14. $y' = 4x + 1{,}3y^2,\ y(0) = 0{,}12.$

1.15. $y' = 2{,}5x^2 - 0{,}9y^2,\ y(0) = 0{,}2.$ 1.16. $y' = y\sin 3x + 0{,}2y^2,\ y(0) = 0.$

**2.** Решить задачу Коши для системы двух дифференциальных уравнений на отрезке $[0,1]$:

а) методом Эйлера с шагом $h_1 = 0{,}1$ и $h_2 = 0{,}05$, построить графики полученных решений;

б) методом Рунге – Кутта четвертого порядка с шагом $h_1 = 0{,}1$ и $h_2 = 0{,}05$, построить графики полученных решений;

в) с помощью функций **DSolve** и **NDSolve**, построить графики.

Сравнить все полученные решения.

1.1. $\begin{cases} y' + 2z = 0, & y(0) = 0, \\ z' - 3y' - 1 = 0, & z(0) = 0. \end{cases}$

1.2. $\begin{cases} y' - y - 3z = 0, & y(0) = 0{,}3, \\ z' = -y + 5z, & z(0) = 0{,}1. \end{cases}$

1.3. $\begin{cases} y' = 2y + z + 3e^{-x}, & y(0) = 0{,}1, \\ z' = y - 2z + 4e^{x}, & z(0) = 0{,}2. \end{cases}$

1.4. $\begin{cases} y' = -y + 2z, & y(0) = 0, \\ z' = -2y' - 5z, & z(0) = 0{,}1. \end{cases}$

1.5. $\begin{cases} y' - y - 3z = 2x, & y(0) = 0{,}4, \\ z' - 4y' - 7z = 0, & z(0) = 0. \end{cases}$

1.6. $\begin{cases} y' - y^2 - 3z = 0, & y(0) = 0{,}2, \\ z' - 4y' - 7z = 0, & z(0) = 0. \end{cases}$

1.7. $\begin{cases} y' + z = 1, & y(0) = -1, \\ z' + \dfrac{2y}{(x+1)^2} = \ln(x+1), & z(0) = 1. \end{cases}$

1.8. $\begin{cases} y' + 5z - 4y = 0, & y(0) = 0{,}4, \\ z' - 3y + 4z = 0, & z(0) = 1. \end{cases}$

1.9. $\begin{cases} y' + 5z = 2, & y(0) = 0{,}5, \\ z' + y + 3z = 0, & z(0) = 2. \end{cases}$

1.10. $\begin{cases} y' = y^2 / z, & y(0) = 1, \\ z' - 0{,}5y = 0, & z(0) = 0{,}6. \end{cases}$

1.11. $\begin{cases} y' = 2y - 3z + 0{,}8x, & y(0) = 0{,}4, \\ z' = 4y - 7z - 2e^{-3x}, & z(0) = 1. \end{cases}$

1.12. $\begin{cases} y' - 0{,}7y = z + \sin 2x, & y(0) = 0, \\ z' = y + 0{,}4z, & z(0) = 1. \end{cases}$

1.13. $\begin{cases} y' + 5y + z = 2, & y(0) = 1, \\ z' + 2y - z = \cos x, & z(0) = 1. \end{cases}$

1.14. $\begin{cases} y' = 4z + e^{-x}, & y(0) = 0, \\ z' = 3y - z + 7, & z(0) = 1. \end{cases}$

1.15. $\begin{cases} y' = 0{,}3y + 4z, & y(0) = 1{,}5, \\ z' = 1{,}6y' - z - 8, & z(0) = 0{,}1. \end{cases}$

1.16. $\begin{cases} y' - 4y + z = x^2, & y(0) = 1, \\ z' = 3y + 5z, & z(0) = 0{,}6. \end{cases}$

## Список использованных источников


1. Калиткин, Н. П. Численные методы / Н. П. Калиткин. – СПб. : БХВ-Петербург, 2011. – 592 с.

2. Самарский, А. А. Численные методы : учеб. пособие для вузов / А. А. Самарский, А. В. Гулин. – М. : Наука, 1989. – 432 с.

3. Численные методы решения задач на ПЭВМ : пособие для вузов. В 2 ч. Ч. 1 / Р. М. Жевняк [и др.]. – Минск : Технопринт, 2004. – 150 с.

4. Численные методы решения задач на ПЭВМ : пособие для вузов. В 2 ч. Ч. 2 / Р. М. Жевняк [и др.]. – Минск : Технопринт, 2005. – 236 с.

5. Бахвалов, Н. С. Численные методы / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. – 9-е изд. – М. : Лаборатория знаний, 2020. – 636 с.

6. Вержбицкий, В. М. Численные методы: линейная алгебра и нелинейные уравнения : учеб. пособие / В. М. Вержбицкий. – 2-е изд., испр. – М. : Оникс 21 век, 2005. – 432 с.

7. Бахвалов, Н. С. Численные методы. Решения задач и упражнения : учеб. пособие для вузов / Н. С. Бахвалов, А. А. Корнев, Е. В. Чижонков. – 2-е изд., испр. и доп. – М. : Лаборатория знаний, 2016. – 352 с.

8. Демидович, Б. П. Основы вычислительной математики / Б. П. Демидович, И. А. Марон. – СПб. : Лань, 2009. – 672 с.

9. Морозов, А. А. Программирование задач численного анализа в системе Mathematica : учеб. пособие / А. А. Морозов, В. Б. Таранчук. – Минск : БГПУ, 2005. – 145 с.

10. Дьяконов, В. П. Mathematica 5.1/5.2/6. Программирование и математические вычисления / В. П. Дьяконов. – М. : ДМК-Пресс, 2008. – 576 с.

11. Тарадаев, Е. П. Использование среды Wolfram Mathematica для решения численных задач : учеб. пособие / Е. П. Тарадаев. – СПб. : Лань, 2020. – 57 с.

12. Математика. Применение пакета Mathematica. В 2 ч. Ч. 1 : Линейная алгебра. Аналитическая геометрия. Введение в математический анализ : пособие / О. А. Вагнер, Л. А. Фомичёва. – Минск : БГУИР, 2019. – 180 с.

13. Математика. Применение пакета Mathematica. В 2 ч. Ч. 2 : Дифференцирование функций нескольких переменных. Интегральное исчисление функций одной и нескольких переменных. Дифференциальные уравнения. Ряды. Операционное исчисление : пособие / Л. А. Фомичёва [и др.]. – Минск : БГУИР, 2021. – 147 с.

14. Высшая математика (Раздел «Численные методы») [Электронный ресурс] : электрон. учеб.-метод. комплекс. – Минск : БГУИР, 2007.

# Содержание

Св. план 2023, поз. 36

*Учебное издание*

**Степанова** Татьяна Сергеевна
**Баркова** Елена Александровна
**Князева** Людмила Павловна
**Самсонов** Павел Анатольевич

# ЧИСЛЕННЫЕ МЕТОДЫ В КОМПЬЮТЕРНОЙ МАТЕМАТИКЕ

ПОСОБИЕ

Редактор *А. Ю. Шурко*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. А. Долгая*

Подписано в печать 24.02.2025. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 7,67. Уч.-изд. л. 8,0. Тираж 100 экз. Заказ 147.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск