Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный
университет имени П.М. Машерова»

Т.Г. Алейникова, А.И. Шербаф

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

Практикум

*Рекомендовано учебно-методическим объединением
по педагогическому образованию в качестве практикума
для студентов учреждений высшего образования,
обучающихся по специальностям:
1-02 05 01 Математика и информатика,
1-02 05 02 Физика и информатика*

*Витебск
ВГУ имени П.М. Машерова
2020*

УДК 519.6(075.8)
ББК 22.19я73
А45

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 5 от 21.05.2020.

Авторы: доцент кафедры информатики и информационных технологий ВГУ имени П.М. Машерова, кандидат физико-математических наук **Т.Г. Алейникова;** доцент кафедры информатики и методики преподавания информатики УО «БГПУ имени Максима Танка», кандидат физико-математических наук **А.И. Шербаф**

Р е ц е н з е н т ы :
кафедра прикладной математики и информатики
УО «БрГУ имени А.С. Пушкина»;
заведующий отделом вычислительной математики Института математики
НАН Беларуси, кандидат физико-математических наук *Г.Ф. Громыко*

Алейникова, Т.Г.

А45 Вычислительные методы : практикум / Т.Г. Алейникова, А.И. Шербаф. – Витебск : ВГУ имени П.М. Машерова, 2020. – 98 с.
ISBN 978-985-517-746-4.

В учебном издании излагаются краткие теоретические сведения о методах решения базовых задач вычислительной математики, на доступном уровне описываются алгоритмы реализации методов в электронных таблицах MS Excel, математическом пакете Maple и на языке программирования Python.

УДК 519.6(075.8)
ББК 22.19я73

ISBN 978-985-517-746-4

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Изучение основ вычислительной математики будущими учителями математики, физики и информатики является важным элементом в их профессиональной подготовке и способствует развитию прикладного характера обучения. Студенты знакомятся с различными алгоритмами решения широкого круга задач, получают навыки практической реализации этих алгоритмов с помощью современных компьютерных и информационных технологий, а также овладевают умениями грамотно интерпретировать полученные результаты и применять их в практической деятельности.

Настоящий практикум представляет собой сборник лабораторных работ по вычислительным методам. Он составлен для студентов педагогического вуза в соответствии с учебными программами учреждения высшего образования по учебным дисциплинам «Вычислительные методы» и «Вычислительные методы и компьютерное моделирование» для специальностей: 1-02 05 01 Математика и информатика, 1-02 05 02 Физика и информатика. Данное издание будет также полезно студентам других специальностей физико-математического профиля, имеющих в своих учебных планах вычислительные методы.

Материал практикума охватывает ряд базовых разделов вычислительной математики: элементы теории погрешностей, решение алгебраических и трансцендентных уравнений с одной переменной, системы линейных алгебраических уравнений, аппроксимация функций, численное дифференцирование и численное интегрирование, обыкновенные дифференциальные уравнения.

Каждая тема содержит методические указания, краткую теоретическую справку и подробные решения типовых задач несколькими способами. Приводятся примеры, как используемые вычислительные методы реализуются с помощью табличного процессора MS Excel, математического пакета Maple и языка программирования Phyton. Применение электронных таблиц упрощает работу студента с данными и позволяет получать результаты без проведения расчетов вручную или специального программирования. Использование пакета Maple дает возможность наглядного представления результатов вычислений, встроенные функции пакета, содержащие реализацию того или иного численного метода, позволяют студенту освободить время для обдумывания алгоритма. Реализация вычислительных алгоритмов на одном из самых популярных языков программирования Phyton способствует развитию системного мышления студентов и закреплению навыков программирования. В рассмотренных примерах демонстрируются разнообразные специализированные инструменты реализации вычислительных методов, которыми располагают современные программные средства, в виде встроенных функций, процедур, модулей и библиотек. Это дает студентам возможность приобрести практический опыт их использования применительно к конкретным задачам и методам. Иллюстрация различных методов и алгоритмов на одних и тех же примерах позволяет сопоставлять и интерпретировать полученные результаты, оценивать точность решения.

В практикуме каждая изучаемая тема предполагает выполнение лабораторной работы. Все работы включают общее задание, одинаковое для всех 20-ти вариантов задач, что обеспечивает индивидуальную работу студента. Выполнение заданий в различных средах предназначено для закрепления теоретического материала дисциплины и способствует формированию у студентов навыков практической реализации изучаемых вычислительных алгоритмов. Вариативность образовательной деятельности студентов обеспечивается не только различными методами и средствами реализации их алгоритмов, но и методическими указаниями по совершенствованию предлагаемых решений и применению современных способов повышения их эффективности.

В заключительном разделе практикума приводится тест для самоконтроля знаний по изучаемой дисциплине. Он содержит 30 вопросов, которые охватывают все рассмотренные темы. Наличие ответов позволяет студенту оценить уровень своей подготовленности, выявить пробелы в знаниях для дальнейшего самообразования.

Авторы выражают благодарность рецензентам – коллективу кафедры и заведующему кафедрой прикладной математики и информатики физико-математического факультета Брестского государственного университета имени А.С. Пушкина, кандидату физико-математических наук Д.В. Грицуку, заведующему отделом вычислительной математики Института математики НАН Беларуси, кандидату физико-математических наук Г.Ф. Громыко за ценные замечания. Авторы признательны проректору БГПУ имени Максима Танка, кандидату физико-математических наук С.И. Васильцу и заведующему кафедрой информатики и методики преподавания информатики БГПУ имени Максима Танка, кандидату педагогических наук С.В. Вабищевич за помощь и поддержку.

Тема 1. Методы оценки погрешности вычислений

Методические указания

Задачу нахождения погрешности функции по заданным погрешностям приближенных аргументов называют ***основной задачей*** *теории погрешностей.*

Определим, как вычислить погрешность функции, аргументы которой заданы приближенно.

Пусть $y = f(x)$ – функция, для которой необходимо найти погрешность, a – приближенное исходное данное, и известны НГa и ВГa. Если исходное данное задано верными цифрами, то его границы можно установить исходя из определения верной цифры.

Цифра в записи приближенного числа называется ***верной*** (в широком смысле), если абсолютная погрешность числа не превосходит единицы ее разряда (в строгом смысле – половины разряда).

Необходимо определить $y = f(x)$, НГy и ВГy, а также Δy.

1.1. Метод границ

Для нахождения границ результата вычисляют $y1 = f($НГ$y)$ и $y2 = f($ВГ$y)$, а затем меньшее из этих значений принимают за НГy, а большее – за ВГy и округляют: нижнее с недостатком, а верхнее с избытком, сохраняя все совпадающие знаки и еще один (различный).

Рассмотрим случай двух переменных: $z = f(a, b)$.

$$\left.\begin{array}{l} z = f(\text{НГ}_a, \text{НГ}_b) \\ z = f(\text{НГ}_a, \text{ВГ}_b) \\ z = f(\text{ВГ}_a, \text{ВГ}_b) \\ z = f(\text{ВГ}_a, \text{НГ}_b) \end{array}\right\} min \to \text{НГ}_z, max \to \text{ВГ}_z$$

Замечание. Прежде чем производить расчет для всех возможных вариантов (для n переменных – 2^n), необходимо попытаться оценить характер зависимости функции от некоторых переменных и подобрать комбинацию для максимального и минимального значений.

1.2. Дифференциальная оценка погрешности

Пусть $y = f(x_1, x_2, \ldots, x_n)$ – дифференцируемая функция n переменных. Предельная абсолютная погрешность функции вычисляется по формуле:

$$\Delta f = \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}\right| \Delta x_i$$

Здесь Δx_i – предельные абсолютные погрешности аргументов функции.

Обратная задача теории погрешностей *состоит в том, что по заданной абсолютной погрешности функции необходимо определить, каковы должны быть предельные абсолютные погрешности ее аргументов.*

Для решения обратной задачи обычно пользуются принципом «равных влияний», т.е. одинакового вклада каждого аргумента в погрешность функции. Тогда погрешность аргумента x_i можно рассчитать по формуле:

$$\Delta x_i = \frac{\Delta f}{n \cdot \left|\frac{\partial f}{\partial x_i}\right|}$$

Пример 1.1. Оценить погрешность вычисления по формуле методом границ в MS EXCEL. Аргументы заданы приближенно, все цифры в записи – верные в широком смысле.

$$L = \frac{v_0^2 \cdot \sin 2\alpha}{g} \text{ при } v_0 = 3.51,\ \alpha = 0.68,\ g = 9.8.$$

Решение.

	A	B	C	D	E	F
1						
2						
3						
4	Метод границ оценки погрешностей вычислений					
5						
6						
7						
8	3,510	0,680	9,800			
9	0,010	0,010	0,100			
10	3,520	0,690	9,700			
11	3,500	0,670	9,900			
12						
13	Lmax	Lmin				
14	1,254181	1,204564				
15	L1	L2				
16	1,260	1,200	1,230	1,230	0,030	0,030
17						
18	L=1,23±0,03					

1. Вводим в ячейки **A8, C8** и **D8** входные данные.
2. Вводим в ячейки **A9, C9** и **D9** соответствующие значения погрешностей входных данных. Эти погрешности определяем исходя из определения верной цифры (в широком смысле). Их предельные значения: $\Delta v_0 = 0.01, \Delta\alpha = 0.01, \Delta g = 0.1.$
3. Вводим формулы расчета верхней границы данных в ячейки **A10 [=A8+A9]; B10 [=B8+B9]**.
4. Вводим формулы расчета нижней границы данного в ячейку **C10 [=C8-C9]**.
5. Вводим формулы расчета нижней границы данных в ячейки **A11 [=A8-A9]; B11 [=B8-B9]**.
6. Вводим формулы расчета верхней границы данного в ячейку **C11 [=C8+C9]**.

7. Вводим формулу расчета максимального значения заданной функции в ячейку **A14 [=A10^2*SIN(2*B10)/C10]**.
8. Вводим формулу расчета минимального значения заданной функции в ячейку **B14 [=A11^2*SIN(2*B11)/C112]**.
9. Вводим результат округления функции сверху в ячейку **A16 [=ОКРВВЕРХ(A14;0,01)];** результат округления функции снизу – в ячейку **B16 [=ОКРВНИЗ(B14;0,01)];** среднее арифметическое полученных результатов – в ячейку **C16[=(A16+B16)/2];** результат округления до трех цифр – в ячейку **D16[=ОКРУГЛ(C16;3)]; E16 [=(A16-B16)/2];** результат округления до трех цифр – в ячейку **F16 [=ОКРУГЛ(E16;3)]**.

Ответ: L=1,23±0,03.

Пример 1.2. Оценить погрешность вычисления по формуле методом границ. Аргументы заданы приближенно, все цифры в записи – верные в широком смысле. Реализовать решение на языке программирования Python.

$$L = \frac{v_0^2 \cdot \sin 2\alpha}{g} \text{ при } v_0 = 3.51,\ \alpha = 0.68,\ g = 9.8.$$

Решение.

Исходные данные (переменные `v0, alfa, g`) заданы верными цифрами (в широком смысле), что дает возможность определить их предельные абсолютные погрешности (`dv, dalfa, dg`). Функция `fun(a, b, c)` описывает заданную формулу, она вызывается с аргументами – наборами границ исходных данных, обеспечивающими минимальное и максимальное значения результата. Функция `poz(a, b)` возвращает количество знаков дробной части, которые необходимо сохранить при округлении нижней и верхней границ. Функции `round_up(a, k)` и `round_floor(a, k)` необходимы для округления с избытком и недостатком с точностью до `k`-го знака в дробной части. В них используются встроенные функции `ceil(a)` и `floor(a)` из модуля `math`.

```
#метод границ
import math
def fun(a,b,c):
    return a**2*math.sin(2*b)/c
def poz(a, b):
    r = math.fabs(a-b)
    k = 0
    while r < 1:
        k+=1
        r*=10
    return k
def round_up(a, k):
    a = a*10**k
    a = math.ceil(a)
    a = a/10**k
    return a
def round_floor(a, k):
    a = a*10**k
    a = math.floor(a)
    a = a/10**k
    return a
```

```
v0, alfa, g = 3.51, 0.68, 9.8
dv, dalfa, dg = 0.01, 0.01, 0.1
Lmax = fun(v0 + dv, alfa + dalfa, g - dg)
Lmin = fun(v0 - dv, alfa - dalfa, g + dg)
n = poz(Lmax, Lmin)
L1 = round_up(Lmax, n)
L2 = round_floor(Lmin, n)
L = (L1 + L2)/2
dL = round((L1 - L2)/2, n)
print('L =', L, 'абс.погр.=', dL)
```

Результат выполнения программы:

```
L = 1.23 абс.погр.= 0.03
```

Пример 1.3. Оценить погрешность вычисления по формуле методом границ и дифференциальным способом в математическом пакете Maple. Аргументы заданы приближенно, все цифры в записи – верные в широком смысле.

$$L = \frac{v_0^2 \cdot sin\,2\alpha}{g} \text{ при } v_0 = 3.51,\ \alpha = 0.68,\ g = 9.8.$$

Решение.

$f := (a, b, c) \rightarrow \dfrac{a^2 \cdot \sin(2 \cdot b)}{c} :$

$v_0 := 3.51 : \alpha := 0.68 : g := 9.8 :$

$\Delta v := 0.01 : \Delta\alpha := 0.01 : \Delta g := 0.1 :$

$Lmax := f\left(v_0 + \Delta v, \alpha + \Delta\alpha, g - \Delta g\right)$

$$1.254181235$$

$Lmin := f\left(v_0 - \Delta v, \alpha - \Delta\alpha, g + \Delta g\right)$

$$1.204564206$$

$L1 := 1.26 : L2 := 1.20 :$

$L := evalf\left(\dfrac{L1 + L2}{2}, 3\right)$

$$1.23$$

$\Delta L := evalf\left(\dfrac{L1 - L2}{2}, 3\right)$

$$0.030$$

$L = 1.23 \pm 0.03 :$

Реализуем оба способа вычисления погрешности и сравним полученные результаты. Исходные данные (v_0, α, g) заданы верными цифрами, что дает возможность определить их предельные абсолютные погрешности ($\Delta v, \Delta\alpha, \Delta g$). Определяем функцию $f(a, b, c)$, соответствующую формуле для вычисления L. Затем вычисляем минимальное и максимальное значения L, округляем их и находим среднее арифметическое и предельную погрешность результата.

В дифференциальном способе вычисляются частные производные функции, значение которых в точке (v_0, α, g) подставляется в формулу оценки. Полученная погрешность и результат округляются.

$$f := (a, b, c) \rightarrow \frac{a^2 \cdot \sin(2 \cdot b)}{c} :$$

$$v_0 := 3.51 : \alpha := 0.68 : g := 9.8 :$$

$$\Delta v := 0.01 : \Delta\alpha := 0.01 : \Delta g := 0.1 :$$

$$Dfa := \frac{d}{d\,a} f(a, b, c) :$$

$$Dfb := \frac{d}{d\,b} f(a, b, c) :$$

$$Dfc := \frac{d}{d\,c} f(a, b, c) :$$

$$D1 := subs\left(a = v_0, b = \alpha, c = g, Dfa\right) :$$

$$D2 := subs\left(a = v_0, b = \alpha, c = g, Dfb\right) :$$

$$D3 := subs\left(a = v_0, b = \alpha, c = g, Dfc\right) :$$

$$\Delta L := |D1| \cdot \Delta v + |D2| \cdot \Delta\alpha + |D3| \cdot \Delta g$$

$$0.02480974170$$

$$L := f\left(v_0, \alpha, g\right)$$

$$1.229325479$$

$$L = 1.229 \pm 0.025 :$$

Сравнение полученных результатов свидетельствует о большей точности дифференциальной оценки по сравнению со способом границ.

Пример 1.4. В математическом пакете Maple вычислить предельные погрешности аргументов функции L, которые обеспечивают абсолютную погрешность $\Delta L = 0.025$.

$$L = \frac{v_0^2 \cdot \sin 2\alpha}{g} \text{ при } v_0 = 3.51,\ \alpha = 0.68,\ g = 9.8.$$

Решение.

Это пример решения обратной задачи теории погрешности. Реализация представлена как продолжение решения примера 1.3. Отметим, что результат отличается от исходных данных в прямой задаче, поскольку рассчитан на равновесный вклад по каждой переменной.

$$\Delta L := 0.025 :$$

$$\Delta v := \frac{\Delta L}{3 \cdot |D1|}; \ \Delta\alpha := \frac{\Delta L}{3 \cdot |D2|}; \ \Delta g := \frac{\Delta L}{3 \cdot |D3|};$$

0.01189676799

0.01584012698

0.06643209473

Ответ: $\Delta v = 0.012$, $\Delta\alpha = 0.016$, $\Delta g = 0.67$

Лабораторная работа № 1

Задание:

1. Вычислить погрешность функции, аргументы которой заданы с точностью до верных цифр:
   a) методом границ;
   b) дифференциальным способом.
2. Решить обратную задачу теории погрешностей, используя значение погрешности функции, полученное в задании 1.

Варианты заданий:

1. $Z = \frac{e^t}{r^2 - \cos s}$, $t = 1.38, r = -3.4, s = 0.0742$	11. $T = f \cdot \frac{\sqrt{c+d}}{3c-d}$, $f = 45.92, c = 1.64, d = 0.7591$
2. $Y = \cos 2x \cdot \sqrt{\frac{m^2+1}{k}}$, $x = 0.654, m = -13.81, k = 127.3$	12. $V = \frac{-b+\sqrt{b^2-a\cdot c}}{16(a+b)}$, $a = 1.547, b = 13.5, c = 5.24$
3. $S = \sqrt{p(p-a)(p-b)(p-c)}$, $p = \frac{a+b+c}{2}$, $a = 9.41, b = 10.5, c = 6.182$	13. $P = \frac{\sin s - e^3}{6 \cdot s - t^e}$, $s = 4.247, e = 2.7, t = 3.17$
4. $X = \frac{u\sqrt{d-t}}{u+d}$, $u = 1.57, d = 13.65, t = 2.3024$	14. $H = (x-1)^2 + \frac{4z(y-3)}{\ln(x+y)}$, $x = 7.1053, y = 10.247, z = 0.21$
5. $N = \frac{h^2}{e^t} - r$, $h = 12.1, t = 2.0401, r = 7.254$	15. $B = h^3 \cdot \frac{q^2+t}{12\sqrt{qh}}$, $h = 5.2, q = 13.214, t = -2.39$

6. $Q=\frac{(3n-1)^2\cdot(k+d)}{k-d}$, $n=1.53, k=28.647, d=3.004$	16. $W=(1+\ln(y-3))\cdot\frac{x^2}{\cos z}$, $x=2.9, y=5.63, z=13.655$
7. $F=\alpha^2\left(1+\frac{2\beta}{\alpha}+\frac{\gamma^2}{\alpha^2}\right)$, $\alpha=2.347, \beta=0.32, \gamma=5.8047$	17. $M=\frac{\cos\left(q+\frac{\pi}{6}\right)(v-1)^2}{\sin mq+0.2v}$, $q=49.14, v=9.2485, m=15.6$
8. $L=\frac{a^3b^2}{\sqrt{k(7.6+a)}}$, $a=2.83, b=7.452, k=3.1$	18. $H=\frac{l^2}{k\left(r+\sqrt{r^2-\frac{l^2}{k}}\right)}$, $l=1.52, r=6320, k=4.0$
9. $S=v_0t+\frac{gt^2}{2}$, $v_0=3.51, t=17, g=9.8$	19. $U=e^{k\omega t}\left(\cos\frac{\omega}{2}-2\cos\frac{3\omega}{2}\right)$, $k=-0.98, \omega=0.786, t=15.6$
10. $A=\pi r^2\left(cos\,2t-\frac{sin\,r}{r-1}\right)$, $\pi=3.14, r=6.107, t=0.7284$	20. $B=\frac{(2k-1)^2}{a+c}$, $k=1.01, a=1.47, c=3.303$

Тема 2. Решение алгебраических и трансцендентных уравнений с одной переменной

Методические указания

Приближенное нахождение вещественных корней уравнения $f(x)=0$ обычно складывается из двух этапов:

1) отделение корней, т.е. установление наиболее коротких промежутков [a; b], в которых находится один, и только один, корень уравнения;
2) уточнение приближенных корней, иначе говоря, вычисление их до заданной степени точности.

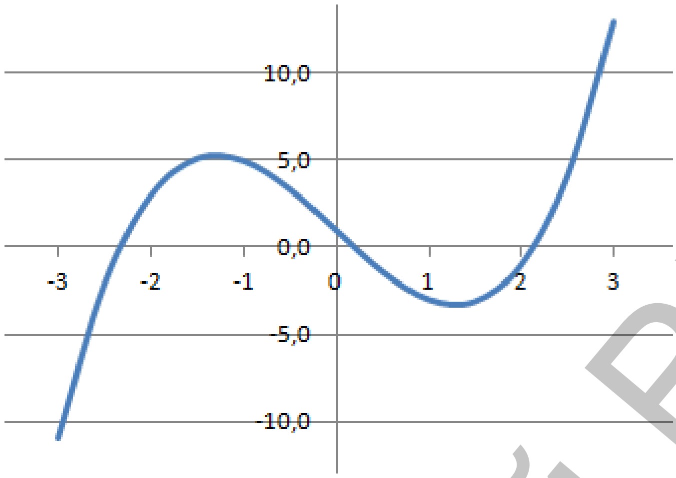
2.1. Отделение корней

Прежде чем приступить к решению уравнения с помощью вычислительного метода, необходимо отделить корень. **Отделение корней** можно выполнить графически, таблично или аналитически. В графическом способе координаты точек пересечения графика функции $y=f(x)$ с осью абсцисс являются корнями уравнения $f(x)=0$.

Пример 2.1. Отделить корень уравнения $x^3-5x+1=0$ графическим способом в MS EXCEL.

Решение.

Сначала построим график функции $y=x^3-5x+1$ на отрезке [–3; 3]. Для этого заполним столбец значениями x из указанного отрезка с небольшим шагом, например, равным 0.1. Еще один столбец заполним соответствующими значениями функции. По данным таблицы построим диаграмму – график.



Из графика функции видно, что корни уравнений находятся на следующих отрезках [–3; –2], [0; 1] и [2; 3].

Пример 2.2. Отделить корень уравнения $x^3 - 5x + 1 = 0$ табличным способом. Реализовать решение на языке программирования Python.

Решение.

Чтобы отделить корни уравнения табличным способом, достаточно табулировать функцию с небольшим шагом, определяя отрезки изменения знака функции. Приведем реализацию алгоритма на Python. Задаем минимальное и максимальное значения `x`, шаг табуляции `h`. Результат сохраняем в списке `l`.

```
def fun(x):
    return x**3-5*x+1
x_min = -3
x_max = 3
h = 1
l = []
for x in range(x_min, x_max, h):
    if fun(x)*fun(x+h) < 0:
        l.append([x, x+h])
if len(l)>0:
    print('Отрезки изоляции корней',l)
else:
    print('В диапазоне',[x_min,x_max],'корней нет')
```

Результат выполнения программы:

```
Отрезки изоляции корней [[-3, -2], [0, 1], [2, 3]]
```

В примере шаг табуляции `h = 1`, в данном случае это дает удовлетворительный результат. Однако выбор шага может быть неудачным, если корень попадает на границу деления или внутри отрезка оказывается более одного корня уравнения.

2.2. Метод половинного деления

Для нахождения приближенного значения методом половинного деления корня уравнения $f(x) = 0$, отделенного на отрезке [a; b], делим отрезок пополам. Из двух половин отрезка выбираем ту, на концах которой функция *f(x)* имеет разные знаки. Выбранный отрезок снова делим пополам и повторяем те же действия. Процесс деления отрезков пополам продолжаем до тех пор, пока длина отрезка, в котором содержится корень, не станет меньше заданной погрешности *e*. Любая точка из этого отрезка (например, середина) дает значение корня с требуемой точностью.

Пример 2.3. Найти корень уравнения $x^3 - 5x + 1 = 0$ на отрезке [0; 1] с точностью *e* = 0,001 в MS EXCEL.

Решение.

Найдите корни заданного уравнения методом половинного деления отрезка с заданной точностью

	A	B	C	D	E	F	G
8					точность	0,001	
9	a	b	c	f(a)	f(c)	d	условие
10	0,000000	1,000000	0,500000	1,000000	-1,375000		
11	0,000000	0,500000	0,250000	1,000000	-0,234375	0,500000	продолжать
12	0,000000	0,250000	0,125000	1,000000	0,376953	0,250000	продолжать
13	0,125000	0,250000	0,187500	0,376953	0,069092	0,125000	продолжать
14	0,187500	0,250000	0,218750	0,069092	-0,083282	0,062500	продолжать
15	0,187500	0,218750	0,203125	0,069092	-0,007244	0,031250	продолжать
16	0,187500	0,203125	0,195313	0,069092	0,030888	0,015625	продолжать
17	0,195313	0,203125	0,199219	0,030888	0,011813	0,007813	продолжать
18	0,199219	0,203125	0,201172	0,011813	0,002282	0,003906	продолжать
19	0,201172	0,203125	0,202148	0,002282	-0,002482	0,001953	продолжать
20	0,201172	0,202148	0,201660	0,002282	-0,000100	0,000977	стоп

1. Вводим в ячейки **A10** и **B10** границы отрезка существования корня, в нашем случае 0 и 1.
2. Вводим формулу расчета середины отрезка ***c=(a+b)/2*** в ячейку **C10** **[=(A10+B10)/2]**.
3. Формулы расчета **f(a)** и **f(c)** вводим в ячейки **D10 [=A10^3-5*A10+1]** и **E10 [=B10^3-5*B10+1]**.
4. Формулы расчета границ нового отрезка вводим в ячейки соответственно **A11 [= ЕСЛИ(D10*E10<0; A10;C10)]** и **B11 [= ЕСЛИ(D10*E10<0; C10;B10)**.

5. Распространяем формулы расчета ***c***, ***f(a)*** и ***f(c)*** на ячейки **C11:E11**.
6. Вводим формулу расчета модуля длины полученного отрезка ***d*** **=** |***b-a***| в ячейку **F11** [=**ABS(B11-A11**)].
7. Условие достижения заданной точности вводим в ячейку **G11 [=ЕСЛИ(F11<F$8; "стоп"; "продолжать")]**.
8. Выделяем ячейки **A11:G11** и распространяем их содержимое до **A20:G20** (до появления команды «стоп»).

В качестве ответа выбираем содержимое любой из ячеек **A20, B20 или C20.**

Ответ: корень с точностью 0,001 равен 0,202.

Пример 2.3. Найти корень уравнения $x^3 - 5x + 1 = 0$ на отрезке [0; 1] с точностью $e = 0{,}001$ методом половинного деления. Реализовать решение на языке программирования Python.

Решение.

Опишем две функции: `fun(x)`, соответствующую левой части заданного уравнения, и `round_eps(x, e)`, с помощью которой округлим результат с заданной точностью `eps`. Предлагаемый алгоритм рассчитан на то, что вводимые пользователем значения `[a, b]` являются результатом отделения корней и не требуют дополнительной проверки на единственность корня. Функция округления рассчитана только на положительные значения `eps`, меньшие 1. Основная часть программы помимо ввода и вывода данных содержит цикл, в котором реализуется алгоритм метода половинного деления. Для проверки условия достижения заданной точности вычисляется длина отрезка `[a, b]`, которая на каждой итерации сравнивается с `eps`.

```
#метод половинного деления
def fun(x):
    return x**3-5*x+1
def round_eps(x, e):
    k = 0
    while e < 1:
        e *=10
        k +=1
    return round(x,k)
print('Введите a, b, eps(0<eps<1): ')
a,b,eps = map(float, input().split())
while b-a > eps:
    c = (a+b)/2
    if fun(a)*fun(c)<0:
        b = c
    else:
        a = c
```

```
root = round_eps(c, eps)
print('Корень с точностью',eps, 'равен',root)
```

Результат работы программы для отрезка [0; 1].

```
Корень с точностью 0.001 равен 0.202
```

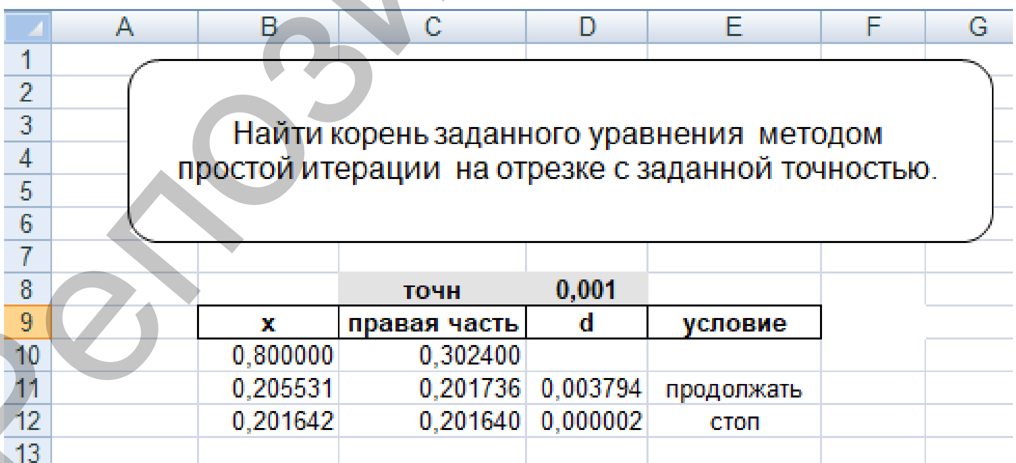
Возможна модификация предложенной программы с целью повышения ее универсальности. Для этого необходимо реализовать проверку существования на отрезке единственного корня, а функцию округления расширить на случай, когда точность больше 1.

2.3. Метод простой итерации

Пример 2.4. Найти корень уравнения $x^3 - 5x + 1 = 0$ на отрезке [0; 1] с точностью *e* = 0.001 методом простой итерации в MS EXCEL.

Решение.

Для нахождения корня заданного уравнения на отрезке [0; 1] методом простой итерации преобразуем уравнение $x^3 - 5x + 1 = 0$ к итерационному виду $x = \frac{x^3+1}{5}$. Так как модуль производной правой части $\left|\frac{3x^2}{5}\right|$<1 на отрезке [0; 1], то достаточное условие сходимости метода выполняется. В качестве нулевого приближения можно выбрать любую точку из отрезка [0; 1]. Подставив выбранное нулевое приближение в правую часть преобразованного уравнения, получим первое приближение к корню, которое подставим в правую часть преобразованного уравнения; повторяем те же действия до тех пор, пока модуль разности двух соседних приближений будет меньше заданной точности ε.



	A	B	C	D	E	F	G
8			точн	0,001			
9		x	правая часть	d	условие		
10		0,800000	0,302400				
11		0,205531	0,201736	0,003794	продолжать		
12		0,201642	0,201640	0,000002	стоп		
13							

1. Вводим в ячейку **B10** нулевое приближение корня, например, 0.1.
2. Вводим формулу правой части преобразованного уравнения в ячейку **C10 [=(B10^3+1)/5]**.
3. Распространяем эту формулу расчета на ячейку **C11 [=(B11^3+1)/5]**.
4. Вводим формулу расчета нового приближения в ячейку **B11 [=(C10:3+1)/5]**.

5. Вводим модуль разности двух приближений к корню **d** = **|x₀-x₁|** в ячейку **D11 [=ABS(C11-B11)]**.
6. Вводим условие достижения заданной точности в ячейку **E11 [=ЕСЛИ(D11<D$8; "стоп"; "продолжать"]**.
7. Выделяем ячейки **B11:E11** и распространяем их содержимое до **B12:E12** (до появления команды «стоп»).

В качестве ответа выбираем содержимое любой из ячеек **B12 или C12.**

Ответ: корень с точностью 0.001 равен 0.202.

Пример 2.5. Найти корень уравнения $x^3 - 5x + 1 = 0$ на отрезке [2; 3] с точностью $e = 0.001$ методом простой итерации. Реализовать решение в математическом пакете Maple.

Решение.

Используем универсальный способ приведения уравнения к итерационному виду $x = x - \alpha \cdot f(x)$, параметр α выбирается таким образом, чтобы выполнялось условие сходимости.

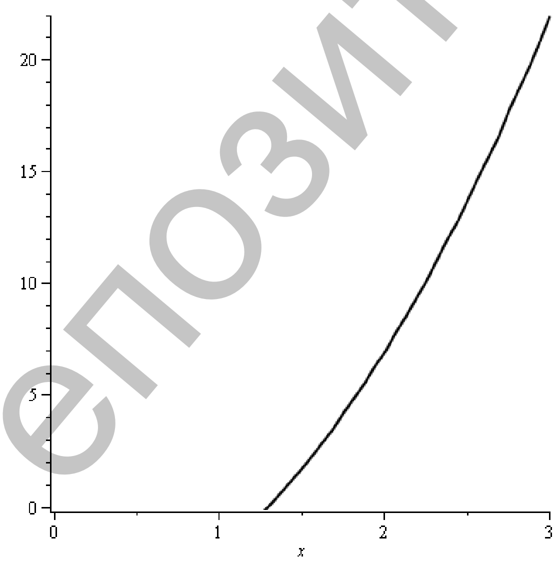
Найдем корень уравнения $x^3 - 5x + 1 = 0$ на отрезке [2; 3], применив оценку $M = max|f'(x)|$. Реализуем решение в Maple.

$f := x \rightarrow x^3 - 5 \cdot x + 1 :$

$df := x \rightarrow 3 \cdot x^2 - 5 :$

$a := 2 : b := 3 : eps := 0.001 :$

$plot(df(x), x = 0..3)$



$M := |df(b)|$

22

Как видно из графика производной $f'(x)$ на отрезке [2; 3], максимальное значение достигается в точке 3, и оно равно 22, тогда параметр $\alpha = \frac{1}{22}$.

В случае, когда производная на отрезке отрицательна, параметр вычисляется как $\alpha = -\frac{1}{max|f'(x)|}$.

Уравнение в итерационном виде, удовлетворяющем условию сходимости метода простой итерации:

$$x = x - \frac{1}{22} \cdot (x^3 - 5x + 1)$$

Вычислительный итерационный процесс можно строить по формуле:

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

Поскольку не надо хранить значения решения для всех итераций, достаточно использовать две переменные: xp, xs, которые соответствуют двум соседним приближениям x_{i+1}, x_i. Критерий выхода из итерационного процесса используем следующий:

$$\frac{q}{1-q}\cdot|x_{i+1}-x_i|<\varepsilon, \text{ где } q=1-\frac{m}{M}, m=min|f'(x)|.$$

Продолжение решения в Maple:

$$\alpha := \frac{1}{M}$$

$$\frac{1}{22}$$

$$m := |df(a)|$$

$$7$$

$$q := evalf\left(1-\frac{m}{M}\right)$$

$$0.6818181818$$

$$xp := a : xs := evalf(xp - \alpha \cdot f(xp))$$

$$2.045454545$$

$$\textbf{while } \frac{q}{1-q}\cdot|xs-xp| > eps \textbf{ do}$$
$$xp := xs :$$
$$xs := evalf(xp - \alpha \cdot f(xp)) :$$
$$\textbf{end do:}$$
$$print(xs)$$

$$2.127768619$$

$$fsolve(f(x)=0, x=2..3)$$

$$2.128419064$$

В результате получаем корень 2.127768619 ≈ 2.128 с точностью 0.001. Сравнение корня со значением, полученным встроенными средствами с помощью функции *fsolve()*, свидетельствует об успешном достижении заданной точности.

2.4. Метод Ньютона (касательных)

Приближение к корню в методе Ньютона вычисляется по рекуррентной формуле:

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

В качестве начального приближения x_0 выбирается любая точка отрезка [a; b], в которой функция f и ее вторая производная имеют одинаковые знаки, т.е. выполняется условие Фурье $f(x_0)f''(x_0) > 0$.

Пример 2.6. Найти корень уравнения $x^3 - 5x + 1 = 0$ на отрезке [0; 1] с точностью *e* = 0.001 методом Ньютона в MS EXCEL.

Решение.

Найдем производную левой части заданного уравнения. В качестве нулевого приближения выбираем левый конец отрезка [0; 1].

Найти корень уравнения методом Ньютона на отрезке с заданной точностью

	A	B	C	D	E
8			точность	0,001	
9	x	f(x)	производная	d	условие
10	0,000000	1,000000	-5,000000		
11	0,200000	0,008000	-4,880000	0,200000	продолжать
12	0,201639	0,000002	-4,878025	0,001639	продолжать
13	0,201640	0,000000	-4,878024	0,000000	стоп

Первое приближение к корню получаем по формуле $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$, которое подставляем в правую часть выражения выше, получаем второе приближение к корню; повторяем те же действия до тех пор, пока модуль разности двух соседних приближений не будет меньше заданной точности ε.

1. Вводим в ячейку **A10** нулевое приближение корня, например, 0.
2. Вводим формулу функции в ячейку **B10 [=A10^3-5*A10+1]**.
3. Вводим формулу производной в ячейку **C10 [=3*A10^2-5]**.
4. Вводим формулу расчета нового приближения в ячейку **A11 [=A10-B10/C10]**.
5. Распространяем формулы расчета функции и производной на ячейки **B11:C11**.
6. Вводим модуль разности двух приближений к корню **d = |x₀-x₁|** в ячейку **D11 [=ABS(A10-A11)]**.
7. Вводим условие достижения заданной точности в ячейку **E11 [=ЕСЛИ(D11<D$8; "стоп"; "продолжать"]**.
8. Выделяем ячейки **A11:F11** и распространяем их содержимое до **A13:E13** (до появления команды «стоп»).

В качестве ответа выбираем содержимое ячейки **A13.**

Ответ: корень с точностью 0.001 равен 0.202.

Пример 2.7. Найти корень уравнения $x^3 - 5x + 1 = 0$ на отрезке [0; 1] с точностью *е* = 0.001 методом Ньютона в математическом пакете Maple.

Решение.

Для реализации метода Ньютона понадобится задать две функции: левую часть уравнения и ее производную по *x*. В качестве начального приближения выберем левый конец отрезка [a; b]. Очередное приближение к корню сохраним в переменной r, а достижение заданной точности eps будем оценивать по значению функции в корне $|f(r)|$.

$$
\begin{aligned}
&restart\\
&f := x \rightarrow x^3 - 5\cdot x + 1:\\
&df := x \rightarrow 3\cdot x^2 - 5:\\
&a := 0 : b := 1 : eps := 0.001:\\
&r := a:\\
&\textbf{while } |f(r)| > eps \textbf{ do}\\
&\quad r := evalf\left(r - \frac{f(r)}{df(r)}\right)\\
&\quad \textbf{end do}:\\
&print(r)
\end{aligned}
$$

Результат: 0.2016393443.

Можно сравнить этот результат с тем, который выдает встроенная функция Maple для численного решения уравнений:

$$fsolve(f(x) = 0, x = 0..1)$$

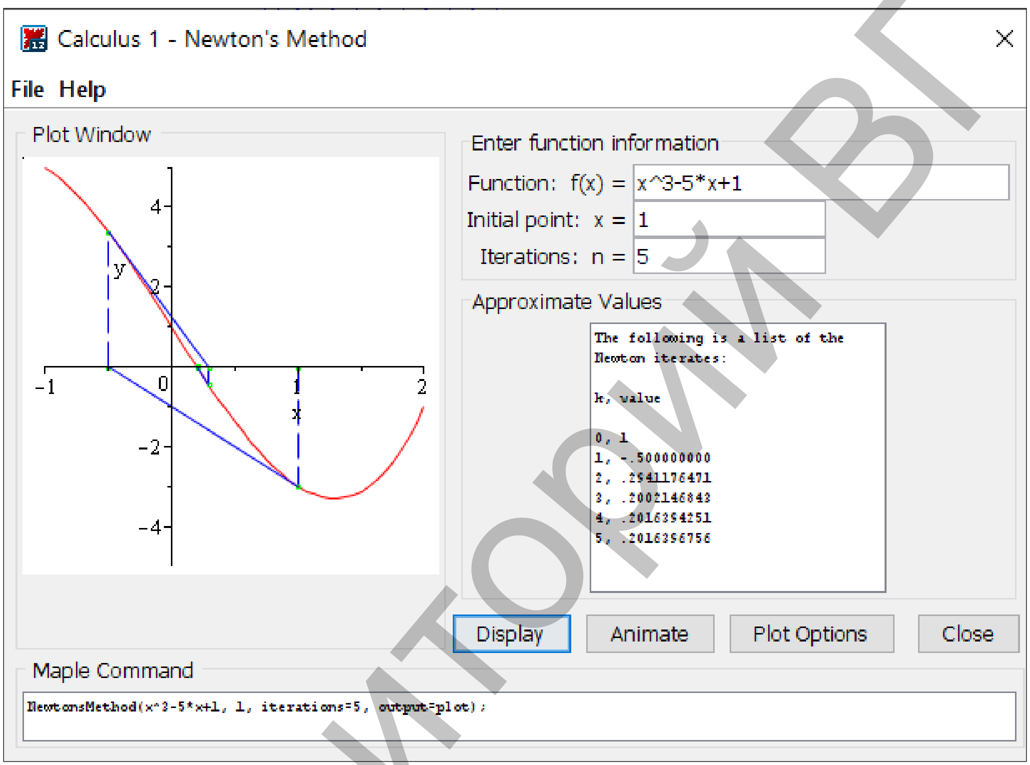
Получим значение: 0.2016396757.

Сравнение результатов показывает, что заданная точность решения $eps = 0.001$ обеспечивается в приведенной реализации метода Ньютона. Однако следует отметить, что этот алгоритм не всегда приводит к удовлетворительному результату. Для более универсального решения задачи можно добавить в программу реализацию выбора начального приближения в соответствии с условием Фурье и проверку значения производной на каждой итерации (она не должна равняться нулю).

Следует добавить, что математический пакет Maple располагает расширенными средствами, позволяющими визуализировать некоторые численные методы, в частности метод Ньютона. Чтобы получить графическую иллюстрацию решения уравнения $x^3 - 5x + 1 = 0$ на отрезке [0; 1], воспользуемся командой *NewtonsMethodTutor*() из пакета расширений *Student[Calculus1]*.

$$with(Student[Calculus1]):$$
$$NewtonsMethodTutor\left(x^3 - 5\cdot x + 1, 0..1\right)$$

Обязательным параметром этой команды является функция – левая часть уравнения, в качестве второго параметра можно указать значение начального приближения или диапазон – отрезок, на котором осуществляется поиск решения. В открывшемся интерактивном окне есть возможность уточнить количество итераций, вид и формат вывода результатов.



На рисунке построены касательные к графику функции в соответствии с методом Ньютона, в отдельной части окна выводятся числовые значения приближений. Можно получить динамическую иллюстрацию решения, если нажать кнопку *Animate*. Приведенный рисунок иллюстрирует, насколько важен выбор начального приближения. Если взять $x_0 = 1$ (не выполняется условие Фурье), итерационные приближения сначала выходят за пределы заданного отрезка. Но потом за 5 итераций решение сходится к корню довольно быстро. Если же взять $x_0 = 0$, то такую же точность получим уже на 3-й итерации. Этот эксперимент подтверждает квадратичную скорость сходимости метода Ньютона.

2.5. Метод секущих

Метод секущих позволяет избежать вычисления производной функции $f(x)$. Сначала зададим два приближения из отрезка существования корня, x_0 и x_1, так как метод является двухшаговым. Второе приближение к корню находим по формуле:

$x_2 = x_1 - (x_1 - x_0)\frac{f(x_1)}{f(x_1) - f(x_0)}$. Подставив полученное приближение x_2 и приближение x_1 в правую часть этого выражения, получим очередное приближение к корню x_3. Повторяем вычисления по рекуррентной формуле:

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

до тех пор, пока модуль разности двух соседних приближений не будет меньше заданной точности ε.

Пример 2.8. Найти корень уравнения $x^3 - 5x + 1 = 0$ на отрезке [0; 1] с точностью $e = 0.001$ методом секущих. Реализовать решение на языке программирования Python.

Решение.

```
#метод секущих
import math
def fun(x):
    return x**3-5*x+1
def round_eps(x, e):
    k = 0
    while e < 1:
        e *=10
        k +=1
    return round(x,k)

print('Введите x0, x1, eps(0<eps<1): ')
x0,x1,eps = map(float, input().split())
x2 = x1-(x1-x0)*fun(x0)/(fun(x1)-fun(x0))

while math.fabs(x0-x1) > eps:
    x2 = x1-(x1-x0)*fun(x1)/(fun(x1)-fun(x0))
    x0 = x1
    x1 = x2

root = round_eps(x2, eps)
print('Корень с точностью',eps, 'равен',root)
```

В программе используются два начальных приближения, это левый конец отрезка (`x0 = 0`) и близкое к нему значение (`x1 = 0.1`).

Входные данные:

```
Введите x0, x1, eps(0<eps<1):
0 0.1 0.0001
```

Результат:

```
Корень с точностью 0.0001 равен 0.2016
```

Лабораторная работа № 2

Задание:

1. Отделить корни уравнения. Выбрать один из отрезков, на котором существует единственный корень, для дальнейшего уточнения.
2. Корень уравнения, выбранный в задании 1, вычислить с точностью ε (задать самостоятельно):
   a) методом половинного деления;
   b) методом Ньютона;
   c) методом секущих;
   d) методом простой итерации.
3. Проверить полученное решение, подставив его в исходное уравнение.

Варианты заданий:

1. $sinx + 0{,}5x^2 - 2x - 1 = 0$	2. $e^x - 1 + (x + 0{,}2)^3 = 0$
3. $5x^3 + \cos(x + 3) - 2x = 0$	4. $0{,}5e^x + (x - 1)^3 + 0{,}3 = 0$
5. $x^3 + tgx - 1{,}2x - 0{,}5 = 0$	6. $sinx + 0{,}4e^{0{,}2x} - x = 0$
7. $0{,}1x^4 - \cos(0{,}6x) - 0{,}3 = 0$	8. $\cos(3x) - 0{,}1x^2 + x + 1 = 0$
9. $4\cos(2x) + lnx - 1{,}2 = 0$	10. $lnx - x^2 + 2 = 0$
11. $-4\sin(2x) - lnx + 1{,}3 = 0$	12. $x^2 - 4\cos(2x) - x = 0$
13. $5\sin(2x) - \sqrt{1 + x} - 1 = 0$	14. $tg(2x) + x^3 + x = 0$
15. $x^2 + \cos(x + 2) - 1 = 0$	16. $0{,}3x^2 - 2x + 1 = 0$
17. $x^2 + sinx - 2{,}3 = 0$	18. $e^{0{,}5x} - x^3 + 1 = 0$
19. $x^4 - 3x^2 + 5x - 4 = 0$	20. $\sin(x + 2) - x^2 + x = 0$

Тема 3. Решение систем линейных алгебраических уравнений (СЛАУ)

Методические указания

3.1. Методы Гаусса и Жордана–Гаусса

Одним из наиболее распространенных точных методов решения СЛАУ выступает метод Гаусса, основной идеей которого является последовательное исключение переменных. Если в системе линейных уравнений много неизвестных, схема метода Гаусса, позволяющая получить точное решение, становится громоздкой и сложной. При таких условиях для нахождения корней системы иногда более удобно пользоваться приближенными итерационными методами.

Пример 3.1. Реализовать решение системы линейных алгебраических уравнений методом Гаусса в MS EXCEL.

$$\begin{cases} 7.2x_1 + 0.45x_2 - 3.6x_3 = -3 \\ 1.5x_1 + 5.0x_2 - 2.1x_3 = 1 \\ -1.7x_1 + 2.8x_2 + 8.4x_3 = 4 \end{cases}$$

Решение.

Найдем неизвестные заданной системы уравнений последовательным исключением неизвестных. Прямой ход **метода Гаусса** состоит в сведении исходной матрицы к треугольному виду с единичными коэффициентами по главной диагонали. Первое уравнение системы делим на 7.2. Чтобы исключить переменную x_1 из второго и третьего уравнений, полученное первое уравнение умножаем на – 1.5 и вычитаем из второго уравнения, затем его умножаем на –1.7 и вычитаем из третьего уравнения. Измененное второе уравнение системы делим на коэффициент при x_2, затем его умножаем на новый коэффициент при x_2 в третьем уравнении и вычитаем из третьего уравнения. Обратный ход метода Гаусса состоит в последовательном нахождении переменных x_3, x_2 и x_1. Для этого измененное третье уравнение делим на коэффициент при x_3 и получаем значение переменной x_3. Переменная x_2 находится из второго уравнения преобразованной системы, переменная x_3 – первого уравнения преобразованной системы.

Реализуем описанный алгоритм:

1. Введем коэффициенты системы уравнений в ячейки **A6:D8**.
2. Вводим в ячейку **A10 [=A6/$A$6]** и распространяем эту формулу расчета на ячейки **B10:D10**.
3. Исключаем переменную x_1 из второго уравнения, для этого вводим следующую формулу расчета в ячейку **A11 [=A7-A10*$A$7]** и распространяем ее на ячейки **B11:D11**.

4. Исключаем переменную x_1 из третьего уравнения, для этого вводим следующую формулу расчета в ячейку **A12 [=A8-A10*$A$8]** и распространяем ее на ячейки **B12:D12**.
5. Вводим в ячейку **B14 [=B11/$B$11]** и распространяем эту формулу расчета на ячейки **C14:D14.**
6. Исключаем переменную x_2 из последующего уравнения, для этого вводим следующую формулу расчета в ячейку **B15 [=B12-B14*$B$12]** и распространяем ее на ячейки **C15:D15**.
7. Вводим в ячейку **C17 [=C15/$C$15]** и распространяем эту формулу расчета на ячейку **D17,** значение этой ячейки есть искомое значение переменной x_3.
8. Для нахождения значения переменной x_2 вводим следующую формулу расчета в ячейку **D20 [=D14-C14*D19]**.
9. Для нахождения значения переменной x_1 вводим следующую формулу расчета в ячейку **D21 [=D10-C10*D19-B10*D20]**.
10. Для проверки найденного решения подставим его в исходную систему **G6 [=A6*D21+B6*D20+C6*D19], G7 [=A7*D21+B7*D20+C7*D19]** и **G8 [=A8*D21+B8*D20+C8*D19]**.

	A	B	C	D	E	F	G
1							
2	**Нахождение корней системы линейных**						
3	**алгебраических уравнений**						
4							
5							
6	7,200	0,450	-3,600	-3,000		ПРОВЕРКА	-3,000
7	1,500	5,000	-2,100	1,000			1,000
8	-1,700	2,800	8,400	4,000			4,000
9							
10	1,000	0,063	-0,500	-0,417			
11	0,000	4,906	-1,350	1,625			
12	0,000	2,906	7,550	3,292			
13							
14		1,000	-0,275	0,331			
15		0,000	8,350	2,329			
16							
17			1,000	0,279			
18							
19			X3=	0,279			
20			X2=	0,408			
21			X1=	-0,303			

Ответ: в ячейках **D19:D21** находятся значения искомых переменных, $x_1 = –0,303$; $x_2 = 0,408$ и $x_3 = 0,279$.

Пример 3.2. Реализовать решение системы линейных алгебраических уравнений методами Гаусса и Жордана–Гаусса в Maple.

$$\begin{cases} 7.2x_1 + 0.45x_2 - 3.6x_3 = -3 \\ 1.5x_1 + 5.0x_2 - 2.1x_3 = 1 \\ -1.7x_1 + 2.8x_2 + 8.4x_3 = 4 \end{cases}$$

Решение.

С команды *restart* рекомендуется начинать решение, поскольку это обеспечивает очистку памяти от результата предыдущих вычислений при каждом новом выполнении алгоритма.

Воспользуемся пакетом *LinearAlgebra* и подключим его с помощью команды *with*. Матрица в этом пакете может быть задана с помощью скобок **< >** и разделителей **| ,** (элементов в строке и столбце соответственно). Зададим матрицу *A*, столбец свободных членов – вектор *b* и расширенную матрицу системы *Ab*:

$restart$

$with(LinearAlgebra):$

$A1 := GaussianElimination(Ab)$

$$\begin{bmatrix} 7.20000000000000016 & 0.450000000000000010 & -3.60000000000000008 & -3. \\ 0. & 4.90625000000000000 & -1.35000000000000008 & 1.62500000000000000 \\ 0. & 0. & 8.34968152866242086 & 2.32908704883227191 \end{bmatrix}$$

$x := Vector(3):$

$x[3] := \dfrac{A1(3,4)}{A1(3,3)}$

$$0.2789432197$$

$x[2] := \dfrac{A1(2,4) - x[3]\cdot A1(2,3)}{A1(2,2)}$

$$0.4079639943$$

$x[1] := \dfrac{A1(1,4) - x[3]\cdot A1(1,3) - x[2]\cdot A1(1,2)}{A1(1,1)}$

$$-0.3026928064$$

$$\begin{bmatrix} 7.2 & 0.45 & -3.6 & -3 \\ 1.5 & 5.0 & -2.1 & 1 \\ -1.7 & 2.8 & 8.4 & 4 \end{bmatrix}$$

Для решения системы используем команду *GaussianElimination* пакетом *LinearAlgebra*, которая соответствует прямому ходу метода Гаусса и приводит исходную матрицу к верхнему треугольному виду с помощью алгоритма исключения неизвестных. Обратный ход метода Гаусса реализуется в обратном порядке, снизу вверх по уравнениям приведенной к треугольному виду матрицы. Решение сохраняется в векторе *x*.

Результат получен с точностью 10 значащих цифр (по умолчанию в Maple), о чем свидетельствует проверка найденного решения с помощью его подстановки в исходную систему. Операция умножения матрицы на вектор в пакете *LinearAlgebra* задается точкой. Точность вычислений можно регулировать с помощью системной переменной *Digits*, присвоив ей значение, отличное от 10.

#проверка подстановкой

$A.x$

$$\begin{bmatrix} -2.99999999956500040 \\ 1.00000000053000004 \\ 4.00000000040000004 \end{bmatrix}$$

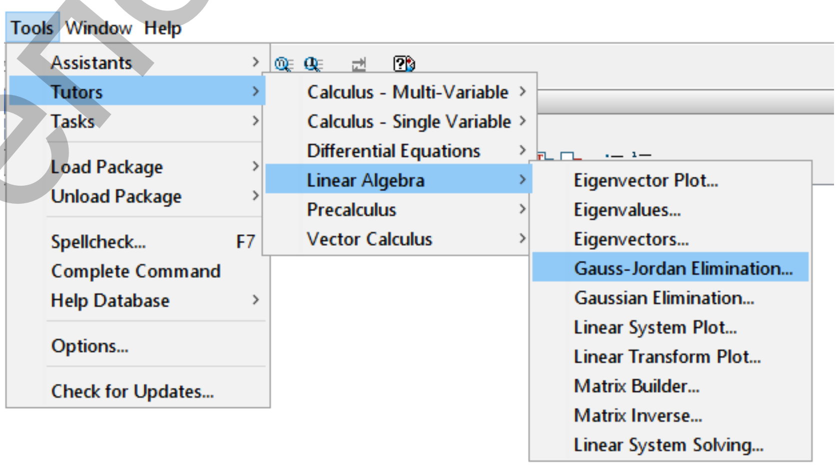
Рассмотрим модификацию метода исключения неизвестных – метод **Жордана–Гаусса**. В этом способе объединяются прямой и обратный ход, матрица при этом приводится к единичному виду, а корни системы оказываются на месте свободных членов расширенной матрицы. В пакете *LinearAlgebra* этот алгоритм реализует команда *ReducedRowEchelonForm*.

$A2 := ReducedRowEchelonForm(Ab)$

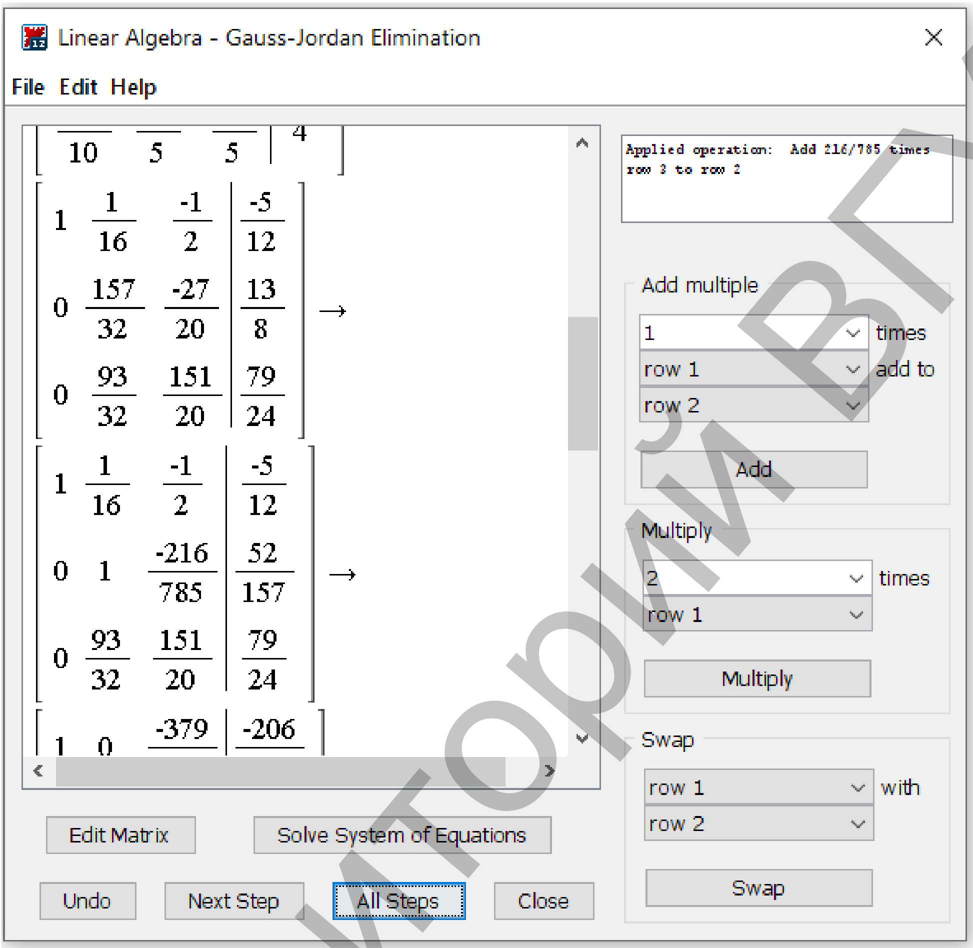
$$\begin{bmatrix} 1. & 0. & 0. & -0.302692806502797285 \\ 0. & 1. & 0. & 0.407963994091279314 \\ 0. & 0. & 1. & 0.278943219637579620 \end{bmatrix}$$

Четвертый столбец полученной матрицы содержит искомые корни системы.

Математический пакет Maple содержит надстройку ***Tutors*** (доступна в меню *Tools*), позволяющую в интерактивном режиме проиллюстрировать пошаговые преобразования в вычислительных методах. Используем это средство для реализации метода Жордана–Гаусса (*Gauss–Jordan Elimination*).



Внесем исходные данные расширенной матрицы из примера (*Edit Matrix*). При этом заметим, что Maple представит коэффициенты в виде обыкновенных дробей независимо от способа их ввода. Выполним алгоритм метода по шагам в открывшемся интерактивном окне.



Отметим, что имеющиеся в правой части окна возможности позволяют вмешиваться в процесс решения, например, менять местами строки или умножать на обратный разрешающему элемент. Это необходимо в случае, если разрешающий элемент близок к 0, и надо делать выбор главного элемента или осуществляется схема единственного деления.

$$\left[\begin{array}{ccc|c} \frac{36}{5} & \frac{9}{20} & \frac{-18}{5} & -3 \\ \frac{3}{2} & 5 & \frac{-21}{10} & 1 \\ \frac{-17}{10} & \frac{14}{5} & \frac{42}{5} & 4 \end{array}\right] \rightarrow \ldots \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & \frac{-3968}{13109} \\ 0 & 1 & 0 & \frac{5348}{13109} \\ 0 & 0 & 1 & \frac{10970}{39327} \end{array}\right]$$

Ответ: корни системы уравнений получим также, как и все решение, в форме обыкновенных дробей в последнем столбце конечной матрицы.

3.2. Метод итерации

Пример 3.3. Реализовать решение системы линейных алгебраических уравнений методом итерации с точностью 0.001 в MS EXCEL.

$$\begin{cases} 7.2x_1 + 0.45x_2 - 3.6x_3 = -3 \\ 1.5x_1 + 5.0x_2 - 2.1x_3 = 1 \\ -1.7x_1 + 2.8x_2 + 8.4x_3 = 4 \end{cases}$$

Решение.

Преобразуем исходную систему к итерационному виду. Для этого выразим искомые переменные x_1, x_2 и x_3, соответственно, из первого, второго и третьего уравнений. Получим систему:

$$\begin{cases} x_1 = -\dfrac{0.45}{7.2} \cdot x_2 + \dfrac{3.6}{7.2} \cdot x_3 - \dfrac{3}{7.2} \\ x_2 = -\dfrac{1.5}{5} \cdot x_1 + \dfrac{2.1}{5} \cdot x_3 + \dfrac{1}{5} \\ x_3 = \dfrac{1.7}{8.4} \cdot x_1 - \dfrac{2.8}{8.4} \cdot x_2 + \dfrac{4}{8.4} \end{cases}$$

Проверим условие сходимости метода простой итерации, для чего вычислим норму матрицы приведенной системы. Используем кубическую норму матрицы:

$$\|\alpha\| = \max\left(\left|-\frac{0.45}{7.2}\right| + \left|\frac{3.6}{7.2}\right|, \left|-\frac{1.5}{5}\right| + \left|\frac{2.1}{5}\right|, \left|\frac{1.7}{8.4}\right| + \left|-\frac{2.8}{8.4}\right|\right) = \frac{4.05}{7.2}$$

Поскольку это значение меньше единицы, то достаточное условие сходимости метода выполняется.

Выбранные нулевые приближения подставляем в правую часть всех уравнений системы. Полученные первые приближения сравниваем с начальными приближениями по модулю. Повторяем эти действия до тех пор, пока модули разности двух соседних приближений каждой переменной будут меньше заданной точности ε.

Выполнение алгоритма в EXCEL:

1. Вводим нулевые приближения к каждой переменной в ячейки **B10:D10,** например, значения столбца свободных членов.
2. Вводим формулы расчета первых приближений:
   переменной x_1 в ячейку **B11 [=-0,45/7,2*C10+3,6/7,2*D10-3/7,2];**
   переменной x_2 в ячейку **C11 [=-1,5/5*B10+2,1/5*D10+1/5];**
   переменной x_3 в ячейку **D11 [=1,7/8,4*B10-2/8,4*C10+4/8,4]**.
3. Вводим модули разности двух приближений по каждой переменной в ячейки **E11:G11**
   E11 [=ABS(B10-B11)]; F11 [=ABS(C10-C11)]; **G11 [=ABS(D10-D11)]**.

4. Вводим значение максимум модуля разностей приближений в ячейку **H11 [=МАКС(E11:G11)]**.
5. Вводим условие достижения заданной точности в ячейку **I11 [=ЕСЛИ(H11<$H$7;"стоп";"продолжить")]**.
6. Выделяем ячейки **B11:I11** и распространяем их содержимое до **B19:I19** (до появления команды «стоп»).

	A	B	C	D	E	F	G	H	I
1									
2		Решить систему уравнений							
3		методом итераций с заданной точностью							
4									
5									
6								**точность**	
7								**0,001**	
8									
9		**x1**	**x2**	**x3**	**d1**	**d2**	**d3**		**условие**
10		-3,000	1,000	4,000					
11		1,521	2,780	-0,464	4,521	1,780	4,464	4,521	продолжить
12		-0,823	-0,451	-0,143	2,343	3,231	0,322	3,231	продолжить
13		-0,460	0,387	0,460	0,363	0,838	0,603	0,838	продолжить
14		-0,211	0,531	0,254	0,249	0,144	0,206	0,249	продолжить
15		-0,323	0,370	0,256	0,112	0,161	0,002	0,161	продолжить
16		-0,312	0,405	0,288	0,011	0,035	0,031	0,035	продолжить
17		-0,298	0,414	0,278	0,013	0,010	0,009	0,013	продолжить
18		-0,303	0,406	0,278	0,005	0,008	0,001	0,008	продолжить
19		-0,303	0,408	0,279	0,000	0,001	0,002	0,002	продолжить
20		-0,302	0,408	0,279	0,001	0,001	0,000	0,001	стоп

Ответ: в ячейках **B19:D19** находятся значения искомых переменных:

$$x_1 = -0.303;\quad x_2 = 0.408 \text{ и } \quad x_3 = 0.279.$$

В этом алгоритме достижение заданной точности контролируется по величине кубической нормы вектора разности соседних решений.

Пример 3.4. Реализовать решение системы линейных алгебраических уравнений методом итерации с точностью 0.001 на языке программирования Python.

$$\begin{cases} 7.2x_1 + 0.45x_2 - 3.6x_3 = -3 \\ 1.5x_1 + 5.0x_2 - 2.1x_3 = 1 \\ -1.7x_1 + 2.8x_2 + 8.4x_3 = 4 \end{cases}$$

Решение.

```
#СЛАУ метод итерации
import math
def transform(matrix):
    m = matrix
```

```
    for i in range(n):
        r = matrix[i][i]
        for j in range(n+1):
            m[i][j] = matrix[i][j]/r
    for i in range(n):
        m[i][i] = 0
        for j in range(n):
            m[i][j] = - m[i][j]
    return(m)
def iteration(matrix, vector):
    s = [0 for i in range(n)]
    for i in range(n):
        for j in range(n):
            s[i] = s[i] + matrix[i][j]*vector[j]
        s[i] +=matrix[i][n]
    return(s)
def norm(vector1, vector2):
    d = [0 for i in range(n)]
    for i in range(n):
        d[i] = math.fabs(vector1[i] - vector2[i])
    return max(d)
def round_eps(vector, e):
    k = 0
    while e < 1:
        e *=10
        k +=1
    d = [0 for i in range(n)]
    for i in range(n):
        d[i] = round(vector[i],k)
    return d
n =3
a = [[7.2,0.45,-3.6,-3],
     [1.5,5.0,-2.1,1],
     [-1.7,2.8,8.4,4]]
eps = 0.001
alfa = transform(a)
x1 = [0 for i in range(n)]
x2 = iteration(alfa,x1)
k = 0
while norm(x1,x2) > eps:
    x1 = x2
    x2 = iteration(alfa,x1)
    k+=1
```

```
print('Решение системы с точностью', eps)
print(round_eps(x2, eps))
print('Количество итераций',k)
```

Результат выполнения программы:

```
Решение системы с точностью 0.001
[-0.303, 0.408, 0.279]
Количество итераций 7
```

Функция `transform` приводит систему к итерационному виду. В качестве параметра она принимает расширенную матрицу системы и возвращает приведенную матрицу с нулевыми диагональными коэффициентами. Учитывая диагональное преобладание, свойственное исходной матрице, приведенная система соответствует условию сходимости метода простой итерации (норма матрицы меньше 1). Реализация метода осуществляется с помощью функции `iteration`. Начальное приближение – нулевой вектор сгенерирован как список. Условие достижения заданной точности контролируется с помощью кубической нормы вектора разности соседних итераций (функция `norm`) и является критерием завершения цикла `while`. В цикле наращивается переменная `k` – счетчик итераций. Результат выводится с точностью `eps`, что обеспечивает функция округления `round_eps`.

3.3. Метод Зейделя

Пример 3.5. Реализовать решение системы линейных алгебраических уравнений методом Зейделя с точностью 0.001 на языке Python.

$$\begin{cases} 7.2x_1 + 0.45x_2 - 3.6x_3 = -3 \\ 1.5x_1 + 5.0x_2 - 2.1x_3 = 1 \\ -1.7x_1 + 2.8x_2 + 8.4x_3 = 4 \end{cases}$$

Решение.

Алгоритмически метод Зейделя подобен методу простой итерации. Отличие состоит в том, что при вычислении новых компонент вектора решений используются уже найденные на текущей итерации, а не на предыдущей. В программе из примера 3.4 вместо функции `iteration` используем функцию `zeidel`.

```
def zeidel(matrix, vector):
    z = [vector[i] for i in range(n)]
    for i in range(n):
        s = 0
        for j in range(n):
            s += matrix[i][j]*z[j]
        z[i] = s + matrix[i][n]
    return(z)
```

В этой функции создается список `z` – копия списка `vector`. Уравнивать их по ссылке на один объект нельзя, поскольку их компоненты совпадают только вначале, по завершению функции эти списки содержат решения на соседних итерациях. Решение достигается за 3 итерации, что быстрее, чем в методе итерации, где потребовалось 7 итераций.

Результат выполнения программы:

```
Решение системы с точностью 0.001
[-0.303, 0.408, 0.279]
Количество итераций 3
```

Лабораторная работа № 3

Задание:

1. Решить систему линейных алгебраических уравнений:
   a) методом Гаусса;
   b) методом Жордана–Гаусса;
   c) методом итерации;
   d) методом Зейделя.
2. Проверить полученное решение, подставив его в исходное уравнение.

Варианты заданий:

1. $\begin{cases} 7{,}8x_1 + 1{,}3x_2 - 3{,}2x_3 = -7{,}0 \\ 2{,}4x_1 + 6{,}1x_2 - 2{,}7x_3 = 1{,}2 \\ -0{,}7x_1 + 1{,}8x_2 + 6{,}9x_3 = -2{,}3 \end{cases}$	2. $\begin{cases} 3{,}8x_1 + 0{,}7x_2 - 1{,}2x_3 = 2{,}0 \\ 1{,}3x_1 - 6{,}4x_2 - 2{,}1x_3 = -1{,}8 \\ 0{,}5x_1 + 1{,}8x_2 + 4{,}8x_3 = 3{,}0 \end{cases}$
3. $\begin{cases} -5{,}5x_1 + 0{,}1x_2 - 1{,}7x_3 = 4{,}2 \\ 1{,}4x_1 + 6{,}1x_2 - 0{,}7x_3 = -0{,}2 \\ 0{,}4x_1 + 1{,}1x_2 - 3{,}7x_3 = -1{,}2 \end{cases}$	4. $\begin{cases} 6{,}4x_1 + 1{,}1x_2 - 1{,}1x_3 = 5{,}7 \\ 1{,}7x_1 + 5{,}3x_2 + 0{,}7x_3 = 3{,}2 \\ 1{,}4x_1 + 0{,}1x_2 - 3{,}7x_3 = 1{,}8 \end{cases}$
5. $\begin{cases} 7{,}2x_1 + 2{,}1x_2 - 1{,}7x_3 = -1{,}2 \\ 2{,}7x_1 - 7{,}1x_2 + 2{,}7x_3 = 4{,}0 \\ 1{,}4x_1 + 2{,}1x_2 - 5{,}7x_3 = 9{,}2 \end{cases}$	6. $\begin{cases} 8{,}4x_1 - 0{,}1x_2 + 1{,}6x_3 = 4{,}2 \\ 1{,}4x_1 + 5{,}1x_2 - 0{,}8x_3 = 8{,}1 \\ 2{,}4x_1 + 1{,}1x_2 - 6{,}7x_3 = 1{,}3 \end{cases}$
7. $\begin{cases} -2{,}4x_1 + 0{,}1x_2 - 0{,}7x_3 = 7{,}4 \\ 0{,}4x_1 + 3{,}9x_2 + 1{,}2x_3 = 1{,}9 \\ 0{,}4x_1 + 1{,}1x_2 - 5{,}7x_3 = 1{,}0 \end{cases}$	8. $\begin{cases} 1{,}4x_1 + 0{,}1x_2 - 0{,}1x_3 = 4{,}3 \\ 0{,}5x_1 + 2{,}0x_2 - 0{,}7x_3 = -1{,}0 \\ 0{,}4x_1 + 0{,}1x_2 + 2{,}7x_3 = -2{,}2 \end{cases}$
9. $\begin{cases} 2{,}3x_1 + 0{,}2x_2 - 0{,}7x_3 = 1{,}8 \\ 0{,}2x_1 - 4{,}1x_2 - 2{,}2x_3 = 1{,}2 \\ 0{,}4x_1 + 1{,}1x_2 + 4{,}7x_3 = 1{,}9 \end{cases}$	10. $\begin{cases} 2{,}4x_1 + 6{,}1x_2 - 2{,}7x_3 = 1{,}2 \\ 2{,}4x_1 + 6{,}1x_2 - 2{,}7x_3 = 1{,}2 \\ 2{,}4x_1 + 6{,}1x_2 - 2{,}7x_3 = 1{,}2 \end{cases}$

11. $\begin{cases} 8{,}0x_1 + 2{,}1x_2 - 1{,}5x_3 = 4{,}2 \\ 1{,}3x_1 + 6{,}1x_2 - 2{,}0x_3 = 0{,}2 \\ 1{,}4x_1 + 1{,}1x_2 - 9{,}7x_3 = 3{,}8 \end{cases}$	12. $\begin{cases} -9{,}0x_1 + 1{,}1x_2 - 0{,}7x_3 = 1{,}2 \\ 2{,}4x_1 + 5{,}1x_2 - 0{,}1x_3 = 1{,}2 \\ -2{,}4x_1 + 1{,}5x_2 - 2{,}7x_3 = 1{,}2 \end{cases}$
13. $\begin{cases} -7{,}8x_1 + 2{,}1x_2 - 1{,}3x_3 = 3{,}2 \\ 1{,}4x_1 - 6{,}6x_2 - 2{,}7x_3 = 1{,}6 \\ -0{,}4x_1 + 1{,}0x_2 - 7{,}7x_3 = 0{,}2 \end{cases}$	14. $\begin{cases} 4{,}4x_1 + 0{,}1x_2 - 1{,}4x_3 = -4{,}2 \\ 0{,}9x_1 + 6{,}8x_2 - 1{,}2x_3 = -2{,}0 \\ 0{,}7x_1 + 2{,}1x_2 - 5{,}7x_3 = 3{,}2 \end{cases}$
15. $\begin{cases} 4{,}7x_1 + 0{,}3x_2 - 2{,}4x_3 = -7{,}2 \\ 0{,}8x_1 + 6{,}9x_2 - 1{,}2x_3 = -1{,}0 \\ 1{,}7x_1 + 1{,}1x_2 - 5{,}4x_3 = 5{,}4 \end{cases}$	16. $\begin{cases} 11{,}4x_1 + 6{,}1x_2 - 1{,}7x_3 = -7{,}2 \\ 3{,}9x_1 + 12{,}8x_2 - 1{,}2x_3 = -8{,}0 \\ 3{,}7x_1 + 2{,}1x_2 - 10{,}7x_3 = 13{,}2 \end{cases}$
17. $\begin{cases} 2{,}9x_1 - 0{,}1x_2 - 0{,}7x_3 = 1{,}9 \\ 0{,}3x_1 + 3{,}1x_2 - 0{,}2x_3 = 3{,}2 \\ 0{,}8x_1 + 0{,}1x_2 + 2{,}7x_3 = 1{,}8 \end{cases}$	18. $\begin{cases} -2{,}4x_1 + 0{,}1x_2 - 2{,}1x_3 = 1{,}5 \\ 0{,}3x_1 + 6{,}1x_2 - 0{,}7x_3 = 2{,}2 \\ 1{,}0x_1 + 0{,}1x_2 - 5{,}7x_3 = -3{,}8 \end{cases}$
19. $\begin{cases} 14{,}4x_1 + 5{,}1x_2 - 3{,}4x_3 = -4{,}2 \\ 4{,}9x_1 + 16{,}8x_2 - 6{,}2x_3 = -2{,}0 \\ 5{,}7x_1 + 2{,}9x_2 - 15{,}7x_3 = 3{,}2 \end{cases}$	20. $\begin{cases} 14{,}4x_1 + 5{,}1x_2 - 3{,}4x_3 = -4{,}2 \\ 4{,}9x_1 + 16{,}8x_2 - 6{,}2x_3 = -2{,}0 \\ 5{,}7x_1 + 2{,}9x_2 - 15{,}7x_3 = 3{,}2 \end{cases}$

Тема 4. Метод наименьших квадратов для обработки экспериментальных данных

Методические указания

Пусть в процессе эксперимента путем измерений получена таблица некоторой функциональной зависимости $y = f(x)$:

x_i	x_1	x_2	…	x_n
$f(x_i)$	$f(x_1)$	$f(x_2)$	…	$f(x_n)$

Требуется найти формулу, выражающую эту зависимость аналитически. Сформулируем задачу таким образом, чтобы учитывался характер исходной функции: найти функцию F заданного вида $y = F(x)$, принимающую в точках $x_1, x_2, \ldots, x_n$ значения, близкие к табличным $y_1, y_2, \ldots, y_n$.

На практике вид приближающей функции F определяется следующим образом. По заданной таблице значений $f(x)$ строится ее точечный график, а потом проводится гладкая кривая, приближенно отображающая характер расположения точек. По полученной кривой определяется вид приближающей функции. Выберем некоторую зависимость $y = F(x, a, b, \ldots, c)$; здесь $a, b, \ldots, c$ являются неизвестными параметрами. **Метод наименьших квадратов** основан на минимизации суммы квадратов отклонений выбранной функции от исследуемых данных. Он позволяет найти значения неизвестных параметров, при которых функция невязки

$$\Phi = \sum_{i=1}^{n} [y_i - F(x_i, a, b, \dots, c)]^2$$

принимает наименьшее значение. Для этого приравняем нулю частные производные по всем неизвестным параметрам $a, b, \dots, c$.

$$\begin{cases} \frac{\partial \Phi}{\partial a} = 0 \\ \frac{\partial \Phi}{\partial b} = 0 \\ \dots \\ \frac{\partial \Phi}{\partial c} = 0 \end{cases} \text{ или } \begin{cases} \sum_{i=1}^{n} [y_i - F(x_i, a, b, \dots, c) F_a(x_i, a, b, \dots, c)] = 0 \\ \sum_{i=1}^{n} [y_i - F(x_i, a, b, \dots, c) F_b(x_i, a, b, \dots, c)] = 0 \\ \dots \\ \sum_{i=1}^{n} [y_i - F(x_i, a, b, \dots, c) F_c(x_i, a, b, \dots, c)] = 0 \end{cases}$$

Из полученной системы уравнений можно найти неизвестные параметры $a, b, \dots, c$, например, методом Гаусса. Мы получим приближающую функцию в аналитическом виде $y = F(x, a, b, \dots, c)$.

Рассмотрим, как построить **линейную** функциональную зависимость между исследуемыми данными. Эта зависимость описывается формулой $y = ax + b$. Задача заключается в нахождении значений коэффициентов зависимости a и b, при которых выражение $\Phi = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$ принимает наименьшее значение. Другими словами, при найденных коэффициентах a и b сумма квадратов отклонений экспериментальных данных от построенной прямой будет наименьшей.

Из равенства нулю частных производных этой функции по переменным a и b следует:

$$\begin{cases} \frac{\partial \Phi}{\partial a} = 0 \\ \frac{\partial \Phi}{\partial b} = 0 \end{cases} \text{ или } \begin{cases} \sum_{i=1}^{n} \big(y_i - (ax_i + b)\big) x_i = 0 \\ \sum_{i=1}^{n} \big(y_i - (ax_i + b)\big) = 0 \end{cases}$$

Откуда получим:

$$\begin{cases} a \sum_{i=1}^{n} x_i^2 + b \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \\ a \sum_{i=1}^{n} x_i + nb = \sum_{i=1}^{n} y_i \end{cases}$$

Решаем полученную систему уравнений любым методом (например, методом исключения неизвестных или методом Крамера) и получаем значения искомых коэффициентов.

Рассмотрим, как построить **квадратичную** зависимость между исследуемыми данными в виде $y = ax^2 + bx + c$. Функция невязки $\Phi = \sum_{i=1}^{n}[y_i - (ax_i^2 + bx_i + c)]^2$ принимает минимальное значение, если ее частные производные по *a* и *b* равны нулю. Неизвестные коэффициенты *a, b* и *c* будем находить из полученной системы линейных уравнений:

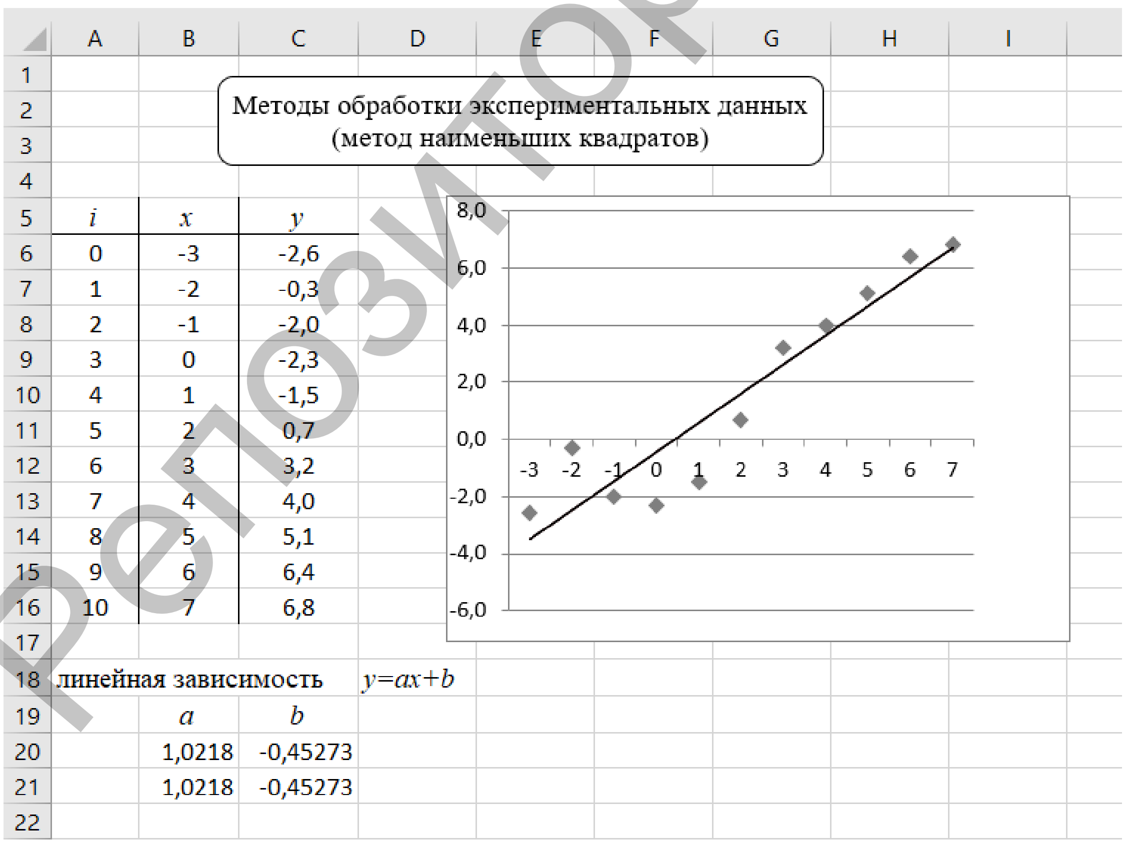
$$\begin{cases} \dfrac{\partial\Phi}{\partial a} = 0 \\ \dfrac{\partial\Phi}{\partial b} = 0 \\ \dots \\ \dfrac{\partial\Phi}{\partial c} = 0 \end{cases} \rightarrow \begin{cases} a\sum_{i=1}^{n} x_i^4 + b\sum_{i=1}^{n} x_i^3 + c\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i^2 y_i \\ a\sum_{i=1}^{n} x_i^3 + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \\ a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i + cn = \sum_{i=1}^{n} y_i \end{cases}$$

Здесь *n* равно количеству элементов в выборке. Систему относительно неизвестных *a*, *b*, и *c* решаем, например, методом Гаусса.

Пример 4.1. По данной таблице значений x и y построить линейную функцию методом наименьших квадратов в MS EXCEL.

x_i	–3	–2	–1	0	1	2	3	4	5	6	7
$f(x_i)$	–2,6	–0,3	–2	–2,3	–1,5	0,7	3,2	4,0	5,1	6,4	6,8

Решение.



Методы обработки экспериментальных данных
(метод наименьших квадратов)

i	*x*	*y*
0	-3	-2,6
1	-2	-0,3
2	-1	-2,0
3	0	-2,3
4	1	-1,5
5	2	0,7
6	3	3,2
7	4	4,0
8	5	5,1
9	6	6,4
10	7	6,8

линейная зависимость *y=ax+b*

	a	*b*
	1,0218	-0,45273
	1,0218	-0,45273

1. Вводим значения $x_i, i = \overline{0,10}$; в ячейки **B6:B16**, значения $y_i, i = \overline{0,10}$; – в ячейки **C6:C16**.
2. Строим диаграмму рассеивания по заданным экспериментальным данным, выбираем тип *Точечная*. Выделив точки на диаграмме, нажимаем правую кнопку мыши и выбираем «*Добавить линию тренда*», выбираем «*Линейная*».
3. Для вычисления параметра *a* вводим следующую формулу в ячейку **B20 [=НАКЛОН(C6:C16;B6:B16)]**.
4. Для вычисления параметра *b* вводим следующую формулу в ячейку **C20 [=ОТРЕЗОК(C6:C16;B6:B16)]**.

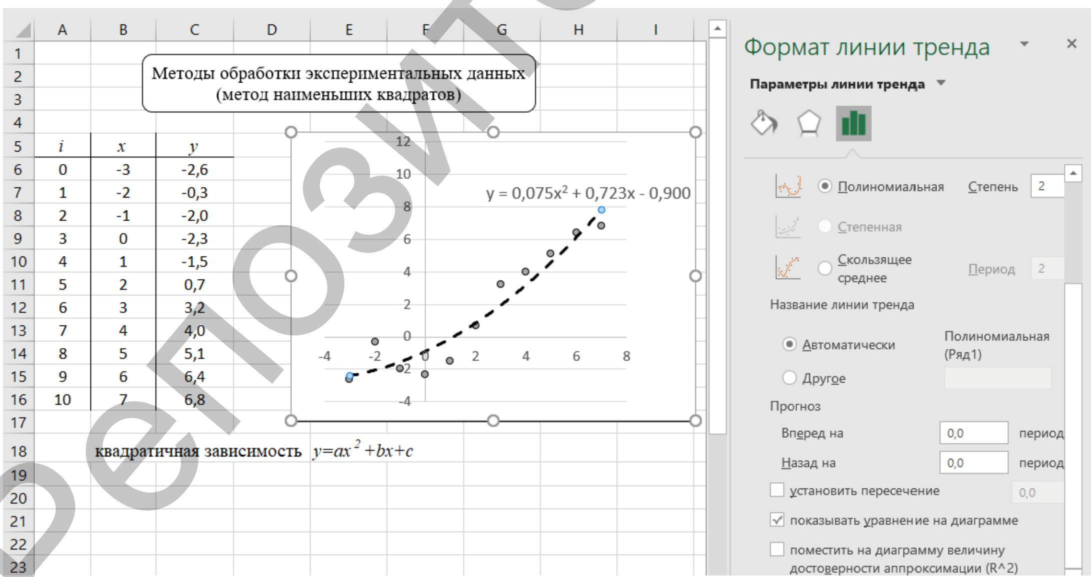
Замечание. Можно вычислить одновременно два параметра a и b. Для этого выделяем в строке две ячейки **B21:C21**, *вводим* **=ЛИНЕЙН(C6:C16;B6:B16)** *и одновременно нажимаем* ***CTRL+SHIFT+ENTER.***

Ответ: *y = 1,0218x – 0,45273*.

Пример 4.2. По данной таблице значений x и y построить квадратичную функцию методом наименьших квадратов.

x_i	–3	–2	–1	0	1	2	3	4	5	6	7
$f(x_i)$	–2,6	–0,3	–2	–2,3	–1,5	0,7	3,2	4,0	5,1	6,4	6,8

Решение.



1. Вводим значения $x_i, i = \overline{0,10}$; в ячейки **B6:B16**, значения $y_i, i = \overline{0,10}$; – в ячейки **C6:C16**.
2. Строим диаграмму рассеивания по заданным экспериментальным данным, выбираем тип *График*. Выделив точки на диаграмме, нажимаем правую кнопку мыши и выбираем «*Добавить линию тренда*», отмечаем «*Полиномиальная*», «Степень 2», а также опцию

«показывать уравнение на диаграмме». На диаграмму можно также вывести сумму квадратов отклонений (опция с R^2).

Замечание. Если необходимо изменить формат вывода уравнения, достаточно в контекстном меню вызвать формат подписи линии тренда.

Ответ: $y = 0{,}075x^2 + 0{,}723x - 0{,}900$.

Пример 4.3. По данной таблице значений x и y построить линейную и квадратичную функции методом наименьших квадратов в математическом пакете Maple.

x_i	–3	–2	–1	0	1	2	3	4	5	6	7
$f(x_i)$	–2,6	–0,3	–2	–2,3	–1,5	0,7	3,2	4,0	5,1	6,4	6,8

Решение.

Зададим исходные данные в виде списков. Точность вычислений регулируем с помощью системной константы *Digits*.

```
restart
x := [-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7] :
y := [-2.6, -0.3, -2, -2.3, -1.5, 0.7, 3.2, 4.0, 5.1, 6.4, 6.8] :
n := 11 :
Digits := 5 :
```

Составим систему линейных алгебраических уравнений для вычисления параметров *a* и *b* линейной функции в соответствии с методом наименьших квадратов. Обозначения для коэффициентов системы отражают алгоритм их подсчета (*Mx* – среднее заданных значений *x*, *My* – *y*, *Mxy* – произведений *xy*, *Mx2* – квадратов *x*).

$$Mx := \frac{1}{n}\cdot\sum_{i=1}^{n} x[i] : My := \frac{1}{n}\cdot\sum_{i=1}^{n} y[i] :$$

$$Mx2 := \frac{1}{n}\cdot\sum_{i=1}^{n} x[i]^2 : Mxy := \frac{1}{n}\cdot\sum_{i=1}^{n} x[i]\cdot y[i] :$$

$$slau := \{Mx2\cdot a + Mx\cdot b = Mxy, Mx\cdot a + b = My\};$$

$$\{2\,a + b = My, 14\,a + 2\,b = Mxy\}$$

Полученную систему уравнений *slau* решаем встроенной функцией *solve*, а команда *assign* позволяет присвоить значения параметрам *a* и *b*. Линейную функцию описываем с помощью оператора-стрелки.

```
r := solve(slau, {a, b}) :
assign(r) : a, b;
1.0218, -0.45274
f := (u) → a·u + b
u → a u + b
```

Таким образом, уравнение приближающей линейной функции имеет вид:

$$y = 1.022x - 0.452$$

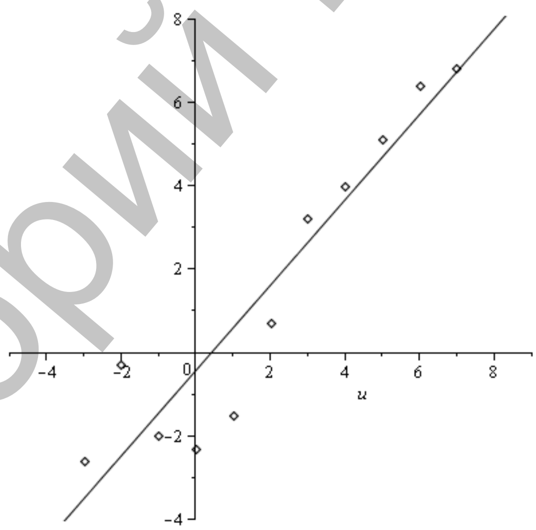
Оценим полученное решение, вычислив невязку (сумму квадратов отклонений).

$$err := \sqrt{\sum_{i=1}^{n} |y[i] - f(x[i])|^2}$$

3.9544

Проиллюстрируем полученное решение графически. Для получения точечного графика используем команду *pointplot* пакета *plots*. В качестве параметров функции укажем координаты *x*, *y*, а также цвет меток (*color*). Можно также добавить в качестве параметров форму (*symbol*) и размер меток (*symbolsize*). Поскольку второй график линейной функции строим с помощью другой команды *plot*, то сначала точки обоих графиков сохраняем в переменных, а затем объединяем для вывода командой *display* пакета *plots*.

$with(plots):$
$p := pointplot(x, y, color = black):$
$g := plot(f(u), color = black):$
$display(p, g, view = [-5..9, -4..8])$



Сравним полученное решение с тем, которое выдает встроенная функция *leastsquare()* из пакета *stat*[*fit*].

$Digits := 3:$
$with(stats):$
$l := fit[leastsquare[[u, v], v = a1 \cdot u + b1]]([x, y])$
$v = 1.02\, u - 0.455$
$r := fit\left[leastsquare\left[[u, v], v = a1 \cdot u^2 + b1 \cdot u + c1\right]\right]([x, y])$
$v = 0.0758\, u^2 + 0.720\, u - 0.909$

Параметрами функции являются координаты исходных точек, вид приближающей функции *v(u)* задается в квадратных скобках (или в форме индекса). С помощью системной переменной *Digits* ограничиваем вывод тремя значащими цифрами. Полученный ответ совпадает с тем, что был ранее рассчитан по методу наименьших квадратов. Приведем также решение для приближающей функции в виде полинома второй степени.

Пример 4.4. По данной таблице значений x и y построить линейную функцию методом наименьших квадратов. Реализовать решение на языке Python.

x_i	–3	–2	–1	0	1	2	3	4	5	6	7
$f(x_i)$	–2,6	–0,3	–2	–2,3	–1,5	0,7	3,2	4,0	5,1	6,4	6,8

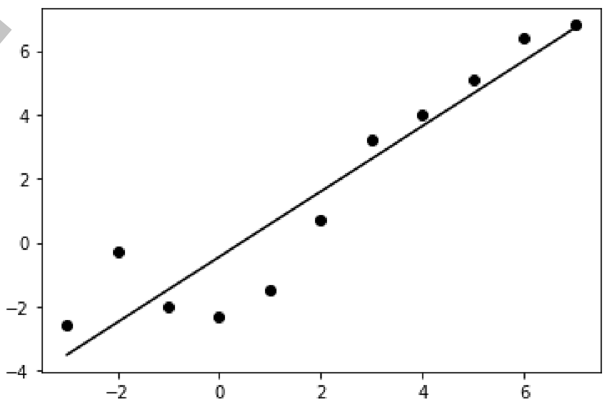
Решение.

На языке программирования Python имеется большое количество готовых средств для реализации вычислительных методов. Наиболее известные из них сосредоточены в библиотеках для научных и математических расчетов NumPy и SciPy. В решении данной задачи используем пакет `numpy` для построения приближающей функции и пакет `matplotlib.pyplot` для вывода графиков. Найдем решение с помощью функции `lstsq()` модуля `linalg`. Построим график исходной функции и приближающей.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7])
y = np.array([-2.6, -0.3, -2, -2.3, -1.5, 0.7, 3.2, 4.0,
5.1, 6.4, 6.8])
A = np.vstack([x, np.ones(len(x))]).T
a, b = np.linalg.lstsq(A, y, rcond=None)[0]
print('y =',round(a,3),'*x +', round(b,3))
plt.plot(x, y, 'o', color='black')
plt.plot(x, a*x + b, color='black')
plt.show()
```

Результат:

```
y = 0.075 x**2 + 0.723 *x + -0.9
```



Пример 4.5. По данной таблице значений x и y построить квадратичную функцию методом наименьших квадратов. Реализовать решение на языке Python.

x_i	–3	–2	–1	0	1	2	3	4	5	6	7
$f(x_i)$	–2,6	–0,3	–2	–2,3	–1,5	0,7	3,2	4,0	5,1	6,4	6,8

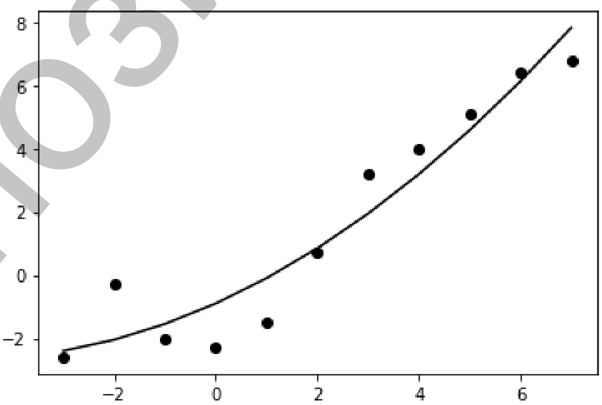
Решение.

Для полиномиального приближения воспользуемся функцией `polyfit` библиотеки `numpy`, которая в качестве параметров принимает массивы `x`, `y` и степень полинома, а возвращает вектор коэффициентов полинома.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7])
y = np.array([-2.6, -0.3, -2, -2.3, -1.5, 0.7, 3.2, 4.0,
5.1, 6.4, 6.8])
k = np.polyfit(x, y, 2)
print('y =',round(k[0],3),'x**2 +', round(k[1],3),'*x
+',round(k[2],3))
plt.plot(x, y, 'o')
plt.plot(x, k[0]*x**2 + k[1]*x+k[2])
plt.show()
```

Результат:

```
y = 0.075 x**2 + 0.723 *x + -0.9
```



Пример 4.6. По данной таблице значений x и y построить степенную функции методом наименьших квадратов. Реализовать решение на языке Python.

x_i	–3	–2	–1	0	1	2	3	4	5	6	7
$f(x_i)$	–2,6	–0,3	–2	–2,3	–1,5	0,7	3,2	4,0	5,1	6,4	6,8

Решение.

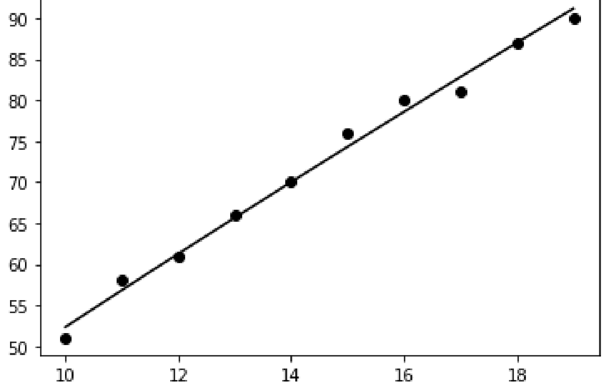
Задача построения приближения степенной функции в виде $y = mx^n$ сводится к нахождению коэффициентов линейной функции. Прологарифмируем выражение для функции, получим $\ln y = \ln m + n \ln x$. Если заменить переменные $u = \ln y, v = \ln x$, то функция приобретает вид линейной $u = ax + b$, где $a = n, b = \ln m$. Таким образом, исходные данные необходимо прологарифмировать (они должны быть положительными), затем для новых данных найти коэффициенты линейной функции, а в заключение выполнить обратную замену для параметров степенной функции $m = e^b, n = a$.

Реализуем описанный алгоритм:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
y = np.array([51, 58, 61, 66, 70, 76, 80, 81, 87, 90])
v = np.array([np.log(x[i]) for i in range(len(x))])
u = np.array([np.log(y[i]) for i in range(len(y))])
A = np.vstack([v, np.ones(len(v))]).T
a, b = np.linalg.lstsq(A, u, rcond=None)[0]
m = np.exp(b)
n = a
print('y =',round(m,3),'*x**', round(n,3))
err = np.sqrt(sum((y[i]-m*x[i]**n)**2 for i in
range(len(x))))
print('Оценка отклонения:', round(err,3))
plt.plot(x, y, 'o', color='black')
plt.plot(x, m*x**n, color='black')
```

Результат:

```
y = 7.155 *x** 0.864
Оценка отклонения: 3.579
```



Лабораторная работа № 4

Задание:

По данной таблице значений x и y найти **методом наименьших квадратов** различные эмпирические формулы и сравнить качество полученных приближений.

Варианты заданий:

1.

x_i	–4	–3	–2	–1	0	1	2	3	4	5	6
$f(x_i)$	1,6	–1,3	–2,1	–2,3	–1,5	0,2	2,3	3,7	5,1	6,9	7,0

2.

x_i	0	1	2	3	4	5	6	7	8	9
$f(x_i)$	3,6	–1,3	–3,0	–3,3	–2,5	1,7	4,2	5,0	6,1	7,4

3.

x_i	–1	0	1	2	3	4	5	6	7	8	9
$f(x_i)$	0,6	–0,3	–0,2	–1,3	–1,7	1,7	3,2	4,5	–2,6	5,4	3,8

4.

x_i	–5	–4	–3	–2	–1	0	1	2	3	4	5
$f(x_i)$	0,6	–0,3	1,2	–0,7	–1,0	0,7	1,2	4,0	3,1	5,4	4,8

5.

x_i	–3	–2	–1	0	1	2	3	4	5
$f(x_i)$	2,6	1,3	3,0	3,7	4,5	2,9	3,2	4,0	6,8

6.

x_i	1	2	3	4	5	6	7	8	9	10
$f(x_i)$	0,8	–0,4	–0,1	1,3	2,5	1,7	3,2	4,6	3,9	5,4

7.

x_i	2	3	4	5	6	7	8	9	10
$f(x_i)$	4,5	3,5	7,5	1,5	6,0	9,0	13,0	5,0	7,2

8.

x_i	1	2	3	4	5	6	7	8	9	10	11
$f(x_i)$	2,6	1,3	3,2	4,0	2,5	4,9	5,6	3,5	6,1	5,9	7,8

9.

x_i	–1	0	2	3	4	5	6	7	8	9	10
$f(x_i)$	5,6	6,3	4,5	5,0	7,0	8,6	4,2	5,7	9,1	6,4	9,9

10.

x_i	–4	–3	–2	–1	0	1	2	3	4	5	6
$f(x_i)$	–1,6	1,3	3,6	4,0	2,7	3,9	1,6	3,5	5,2	6,9	5,8

11.

x_i	–2	1	0	1	2	3	4	5	6	7
$f(x_i)$	4,6	1,5	3,2	5,0	2,5	6,7	4,6	5,5	8,1	6,9

12.

x_i	–3	–2	–1	0	1	2	3	4	5
$f(x_i)$	7,6	6,3	5,2	8,0	7,0	6,4	7,8	9,5	6,1

13.

x_i	1	2	3	4	5	6	7	8	9	10
$f(x_i)$	2,8	1,5	3,4	4,2	2,7	5,1	5,8	3,7	6,3	6,0

14.

x_i	0	1	2	3	4	5	6	7	8	9	10
$f(x_i)$	7,6	6,3	8,2	6,0	4,5	6,9	7,6	5,5	8,1	7,9	10,8

15.

x_i	1	2	3	4	5	6	7	8	9	10	11
$f(x_i)$	12,6	11,0	8,2	13,0	12,3	14,9	15,6	13,5	16,1	15,9	17,0

16.

x_i	1	2	3	4	5	6	7	8	9
$f(x_i)$	2,6	1,3	3,2	4,0	2,5	4,9	5,6	3,5	6,1

17.

x_i	–1	0	2	3	4	5	6	7	8	9	10
$f(x_i)$	–7,6	–5,3	–3,4	–4,2	–2,0	0,1	2,6	1,1	3,1	4,9	2,8

18.

x_i	–3	–2	–1	0	1	2	3	4	5	6
$f(x_i)$	–2,6	–1,3	–3,2	1,0	2,5	3,8	1,7	3,5	4,1	3,0

19.

x_i	1	2	3	4	5	6	7	8
$f(x_i)$	2,6	1,3	3,2	4,0	2,5	4,9	5,6	3,5

20.

x_i	1	2	3	4	5	6	7	8	9	10	11
$f(x_i)$	0,6	1,3	2,2	1,5	2,5	3,9	4,6	4,0	6,1	4,8	7,8

Тема 5. Интерполирование

Методические указания

5.1. Интерполяционный многочлен Лагранжа

Простейшая задача интерполирования заключается в следующем. На отрезке [a; b] заданы $n+1$ точки, $x_0, x_1, \dots, x_n$, которые называются узлами интерполирования, а также значения некоторой функции $f(x)$ в этих точках $f(x_0) = y_0, f(x_1) = y_1, \dots, f(x_n) = y_n$. Требуется построить функцию $F(x)$ (интерполирующая функция), принимающую в узлах интерполирования те же значения, что и $f(x)$, т.е. такую, что $F(x_0) = y_0, F(x_1) = y_1, \dots, F(x_n) = y_n$.

В такой общей постановке задача может иметь бесчисленное множество решений или совсем не иметь решений. Однако эта задача становится однозначно разрешимой, если вместо произвольной функции $F(x)$ искать полином $P_n(x)$ степени не выше *n*, удовлетворяющий условиям выше.

Интерполяционные формулы обычно используют для приближенного вычисления значений данной функции $f(x)$ для значений аргумента *x*, отличных от узлов интерполирования. Такая операция называется **интерполированием.**

Рассмотрим интерполирование с помощью интерполяционного многочлена Лагранжа (чаще всего используемого на практике). Многочлен **Лагранжа** имеет вид:

$$L_n(x) = \sum_{k=0}^{n} \frac{(x-x_0)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)} y_k$$

Пример 5.1. Функция $f(x)$ задана таблицей

x	0	2	5	7	12	14
f(x)	5,00	–20,94	–96,88	–155,02	124,02	747,64

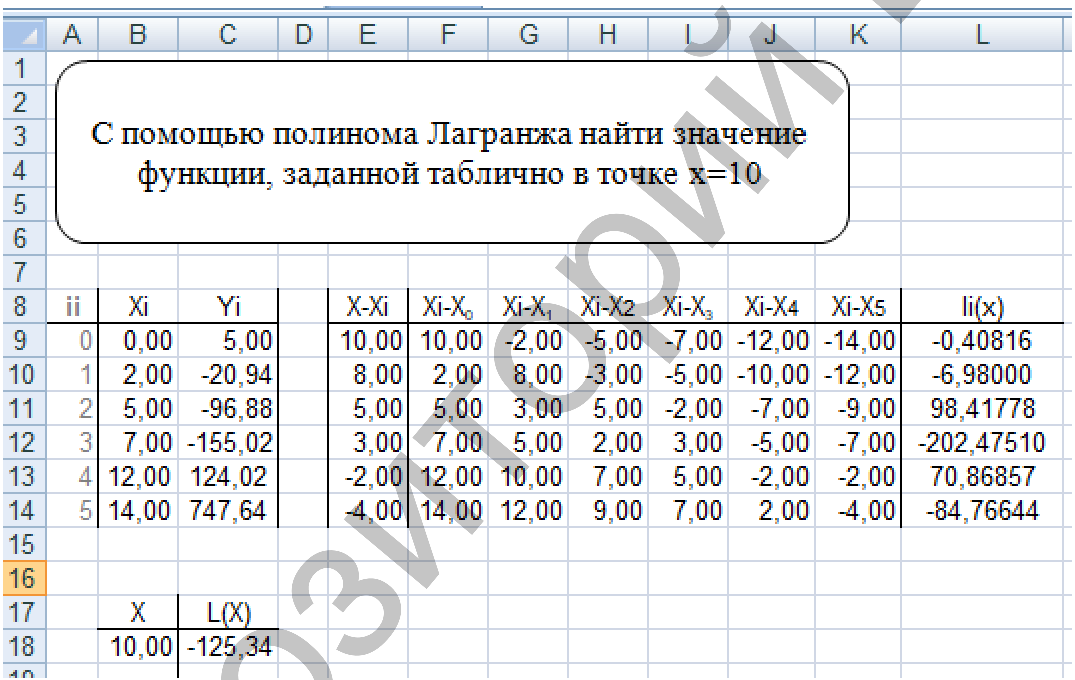
Найти значение этой функции в точке *x* = 10 с помощью многочлена Лагранжа в MS EXCEL.

Решение.

1. Вводим значения x_i , $i = \overline{0;5}$ в ячейки **[B9]:[B14]**, значения $f(x_i)$, $i = \overline{0;5}$ – в ячейки **[C9]:[C14];** заданное значение *x* – в ячейку **[B18]**.
2. В столбец **E9:E14** вводим значения разностей $(x - x_i), i = \overline{0;5}$. Для этого формулу расчета разности $(\boldsymbol{x_1} - \boldsymbol{x_0})$ вводим в ячейку **E9** **[=$B$18-B9]** и распространяем содержимое этой ячейки на **E10:E14**.
3. В ячейках **F9:K14** создаем матрицу разностей $(\boldsymbol{x_i} - \boldsymbol{x_k}); i, k = \overline{0;5}$. Для этого проводим следующие действия: для упрощения алгоритма

введем значение ячейки **E9** в ячейку **F9;** значение **E10 –** в ячейку **G10**; **E11** – в **H11**; **E12** – в **I12**; **E13** – в **J13** и **E14** – в **K14**; введем формулу расчета разности $(\boldsymbol{x_1} - \boldsymbol{x_0})$ в ячейку **F10 [=B10-$B$9]** и распространим содержимое на ячейки **F11:F14;** аналогичным образом в остальные ячейки вводим разности $(\boldsymbol{x_i} - \boldsymbol{x_k}), i = \overline{1{,}5}, k = \overline{0;5}$.

4. В столбце **L9:L14** вычисляем коэффициенты $l_i(x) = \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} y_i, \ = \overline{0;5}$. Для этого введем формулу расчета в ячейку **L9 [=C9*ПРОИЗВЕД($E$9:$E$14)/ПРОИЗВЕД(F9:K9)]** и распространим содержимое этой ячейки на ячейки **L10:L14**.
5. Вводим сумму коэффициентов $l_i(x)$, $i = \overline{0;5}$ в ячейку **C18 [=СУММ(L9:L14)]**.



	A	B	C	D	E	F	G	H	I	J	K	L
1–6	С помощью полинома Лагранжа найти значение функции, заданной таблично в точке x=10											
7												
8	ii	Xi	Yi		X-Xi	Xi-X0	Xi-X1	Xi-X2	Xi-X3	Xi-X4	Xi-X5	li(x)
9	0	0,00	5,00		10,00	10,00	-2,00	-5,00	-7,00	-12,00	-14,00	-0,40816
10	1	2,00	-20,94		8,00	2,00	8,00	-3,00	-5,00	-10,00	-12,00	-6,98000
11	2	5,00	-96,88		5,00	5,00	3,00	5,00	-2,00	-7,00	-9,00	98,41778
12	3	7,00	-155,02		3,00	7,00	5,00	2,00	3,00	-5,00	-7,00	-202,47510
13	4	12,00	124,02		-2,00	12,00	10,00	7,00	5,00	-2,00	-2,00	70,86857
14	5	14,00	747,64		-4,00	14,00	12,00	9,00	7,00	2,00	-4,00	-84,76644
15												
16												
17		X	L(X)									
18		10,00	-125,34									

В качестве ответа выбираем содержимое ячейки **C18,** *f(10)*= –125,34.

Пример 5.2. Функция $f(x)$ задана таблицей

x	0	2	5	7	12	14
$f(x)$	5,00	–20,94	–96,88	–155,02	124,02	747,64

Найти значение этой функции в точке x = 10 с помощью многочлена Лагранжа. Реализовать решение на языке программирования Python.

Решение.

Для реализации решения задачи на языке программирования Python используем научные библиотеки: `numpy` и `matplotlib`. Для сокращения записи в программе используем их синонимы (`np, plt`).

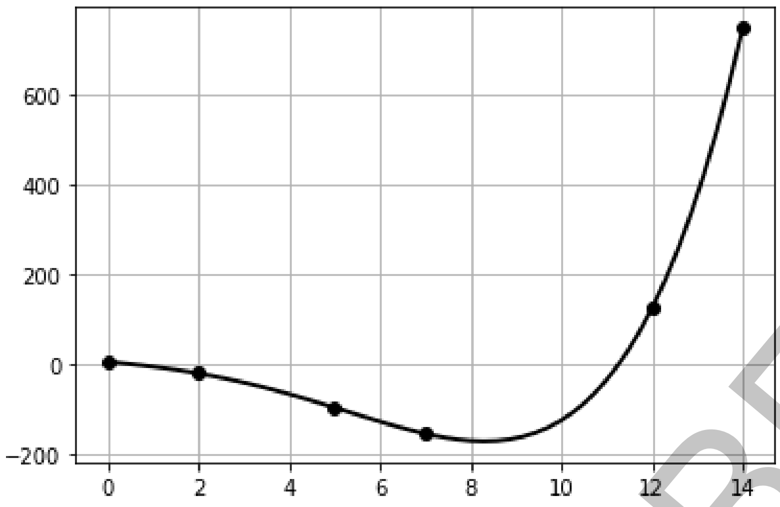
```
#интерполяционный многочлен Лагранжа
import numpy as np
```

```
import matplotlib.pyplot as plt
def lagrange(x,y,t):
    l = 0
    for j in range(len(y)):
        p1=1; p2=1
        for i in range(len(x)):
            if i!=j:
                p1 = p1*(t-x[i])
                p2 = p2*(x[j]-x[i])
        l = l + y[j]*p1/p2
    return l

x = np.array([0.00,2.00,5.00,7.00,12.00,14.00],
dtype=float)
y = np.array([5.00,-20.94,-96.88,-155.02,124.02,747.64],
dtype=float)
xnew = np.linspace(np.min(x),np.max(x),50)
ynew = [lagrange(x,y,i) for i in xnew]
print(lagrange(x,y,10))
plt.plot(x, y,'o',xnew, ynew, lw = 2, color="black")
plt.grid(True)
plt.show()
```

Сохраним исходные данные в массивах, а результат проиллюстрируем графиками построенного многочлена и таблично заданной функции. Реализация вычислений по формуле Лагранжа реализована в функции `lagrange(x,y,t)`. Параметрами функции являются массивы табличных значений исходных данных `x` и `y`, а `t` – абсцисса искомой точки (в примере это 10). С помощью функции `linspace()` модуля `numpy` формируем массив (из 50 элементов) равномерно расположенных на заданном отрезке абсцисс `xnew`, значения соответствующих ординат генерируем в списке `ynew` подстановкой в функцию вычисления многочлена Лагранжа координат `xnew`. По этим точкам строим график функции, в параметрах метода `matplotlib.pyplot.plot` указываем два набора координат (исходная таблица и полученная по формуле многочлена Лагранжа), тип маркера для точек, а также толщину и цвет линии. Методы `grid()` и `show()` необходимы для отображения координатной сетки и окна с графиком.

Результат выполнения программы:

```
-125.34335600907032
```



Как видно из рисунка, график интерполяционного многочлена проходит через все заданные табличные точки функции. Этот факт может быть критерием для тестирования программы. В качестве тестовых значений аргумента `t` функции `lagrange(x, y, t)` необходимо подставить поочередно все значения массива `x`, полученный результат должен совпадать с `y`.

Лабораторная работа № 5

Задание:

Используя многочлен Лагранжа, найти приближенное значение функции, заданной таблицей, в указанных точках.

Варианты заданий:

1.

x	10	13	17	21	25	28
$f(x)$	–514	–460	–382	156	1766	3956

x = 11; 15; 27

2.

x	11	13	15	17	20	22
$f(x)$	42,26	35,39	29,88	16,97	6,05	4,21

x = 12; 14; 19

3.

x	1	2	3	7	9	12
$f(x)$	–2,20	–6,78	–17,01	–34,77	–45,16	–56,01

x = 1,5; 3; 5

4.

x	2	3	6	7	9	11
$f(x)$	16,00	21,9	77,08	95,32	114,02	137,64

$x = 5; 8; 10$

5.

x	0,2	0,7	1,0	3,0	3,7	5,00
$f(x)$	0,60	0,72	0,77	0,95	1,14	1,38

$x = 1,5; 2,7; 4,0$

6.

x	1	3,1	4	5,7	7	9,1
$f(x)$	38	29	25	21	19	17

$x = 1,5; 2,7; 4,0$

7.

x	0,1	0,31	0,7	0,9	1,2	1,4
$f(x)$	1,38	0,9	0,35	0,23	0,15	0,10

$x = 0,2; 1,0; 1,3$

8.

x	1	3	7	9	12	14
$f(x)$	–12	–10	–6	–1	5	8

$x = 5; 11; 13$

9.

x	10	12	16	21	30	44
$f(x)$	112	132	155	176	210	325

$x = 11; 18; 35$

10.

x	1	3	7	11	12	16
$f(x)$	–83	–77	–69,8	–55	–37,7	5

$x = 4; 6; 10$

11.

x	3,5	5,0	7,2	9,3	11	13,8
$f(x)$	112	132	155	176	210	325

$x = 4; 8; 12$

12.

x	0,10	0,23	0,43	0,58	0,63	0,80
$f(x)$	–8,3	–7,7	–6,9	–5,5	–4,0	–1,5

$x = 0,2; 0,5; 0,65$

13.

x	8	11	13	16	20	24
$f(x)$	111	132	154	178	198	221

$x = 10;\ 14;\ 22$

14.

x	23	25	28	32	37	44
$f(x)$	110	136	158	171	188	211

$x = 24;\ 29;\ 41$

15.

x	35	50	72	93	110	138
$f(x)$	112	132	155	176	210	325

$x = 44;\ 60;\ 120$

16.

x	17	23	25	28	31	36
$f(x)$	–8,3	–7,7	–6,9	–5,5	–4,0	–1,5

$x = 22;\ 26;\ 33$

17.

x	13	15	18	21	24	28
$f(x)$	11,50	14,61	20,8	23,21	26,18	28,32

$x = 14;\ 19;\ 25$

18.

x	2,3	2,5	2,8	3,2	3,7	4,4
$f(x)$	11,0	13,6	15,8	17,1	18,8	21,1

$x = 2{,}4;\ 2{,}9;\ 4{,}1$

19.

x	1,0	3,7	4,8	6,2	7,7	9,0
$f(x)$	–10	–8	–5	–1	0	2

$x = 2{,}4;\ 4{,}4;\ 8{,}5$

20.

x	0,23	0,25	0,28	0,32	0,37	0,44
$f(x)$	110	136	158	171	188	211

$x = 0{,}24;\ 0{,}29;\ 0{,}41$

5.2. Интерполяционный многочлен Ньютона

Часто интерполирование проводится для функций, заданных таблицей с равноудаленными значениями аргумента. Для таких таблиц построение интерполяционных формул (и вычисления по таким формулам) значительно упрощается. Узлы интерполирования называются равноудаленными, если

$x_{i+1} - x_i = h - const;\ \ i = \overline{0, n-1}.$

Конечные разности для функции $y = f(x)$ имеют вид:

$\Delta y_i = y_{i+1} - y_i$ – конечные разности первого порядка;

$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$ – конечные разности второго порядка;

....

$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$ – конечные разности k-ого порядка.

Приведем горизонтальную таблицу конечных разностей для $n = 5$:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0	Δy_0	$\Delta^2 y_0$	$\Delta^3 y_0$	$\Delta^4 y_0$	$\Delta^5 y_0$
x_1	y_1	Δy_1	$\Delta^2 y_1$	$\Delta^3 y_1$	$\Delta^4 y_1$	
x_2	y_2	Δy_2	$\Delta^2 y_2$	$\Delta^3 y_2$		
x_3	y_3	Δy_3	$\Delta^2 y_3$			
x_4	y_4	Δy_4				
x_5	y_5					

Рассмотрим две формы записи многочлена Ньютона.

Первая интерполяционная формула Ньютона

$P_n(x) = y_0 + q\Delta y_0 + \frac{q(q-1)\Delta^2 y_0}{2!} + \dots + \frac{q(q-1)\dots(q-n+1)}{n!}\Delta^n y_0$, где $q = \frac{x - x_0}{h}$.

Заметим, что в этой формуле используется верхняя горизонтальная строка таблицы конечных разностей. Данная формула применяется для интерполирования в точках x, близких к началу таблицы.

Вторая интерполяционная формула Ньютона

$$P_n(x) = y_n + q\Delta y_{n-1} + \frac{q(q+1)\Delta^2 y_{n-2}}{2!} + \cdots + \frac{q(q+1)\ldots(q+n-1)}{n!}\Delta^n y_0,$$

где $q = \frac{x - x_n}{h}$.

В этой формуле используется побочная диагональная строка таблицы конечных разностей. Она применяется для интерполирования в точках x, близких к концу таблицы.

Пример 5.3. Функция $f(x)$ задана таблицей

x	0,05	0,10	0,15	0,20	0,25	0,30
$f(x)$	0,1202	0,2317	0,3410	0,4795	0,5841	0,6438

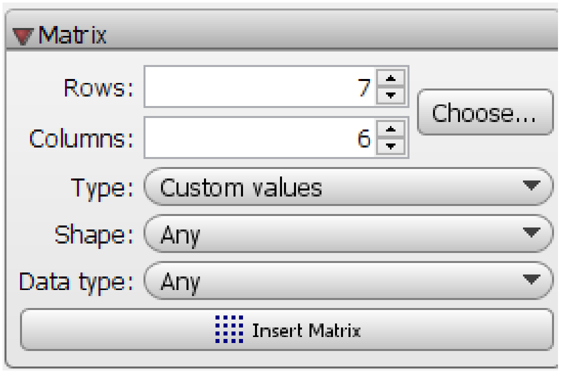
Найти значение этой функции в точке x = 0,13 с помощью многочлена Ньютона в Maple.

Решение.

Сохраним исходные данные в массивах x и y. Учитывая равномерность шага, заполнение массива x производим в цикле.

```
# интерполяционный многочлен Ньютона
restart
n := 6 :
h := 0.05 :
x := array(1 ..n) :
x[1] := 0.05 :
for i from 2 to n do
  x[i] := x[i - 1] + h
end do:
y := array(1 ..n, [0.12020, 0.23170, 0.34100, 0.47950, 0.58410, 0.6438])
```

Воспользуемся средством «Matrix» панели инструментов, чтобы заполнить таблицу конечных разностей. Устанавливаем размерность матрицы $(n+1) \times n$ (учитываем строку заголовков) и нажимаем кнопку «Insert Matrix». В появившемся шаблоне вносим заголовки в первую строку матрицы. Далее нужно внести в первый столбец значения из массива y. В данном случае первый столбец заполнялся вручную, но можно было сделать это программой с помощью циклического алгоритма.



$$R := \begin{bmatrix} y & \Delta y & \Delta^2 y & \Delta^3 y & \Delta^4 y & \Delta^5 y \\ y_1 & m_{2,2} & m_{2,3} & m_{2,4} & m_{2,5} & m_{2,6} \\ y_2 & m_{3,2} & m_{3,3} & m_{3,4} & m_{3,5} & m_{3,6} \\ y_3 & m_{4,2} & m_{4,3} & m_{4,4} & m_{4,5} & m_{4,6} \\ y_4 & m_{5,2} & m_{5,3} & m_{5,4} & m_{5,5} & m_{5,6} \\ y_5 & m_{6,2} & m_{6,3} & m_{6,4} & m_{6,5} & m_{6,6} \\ y_6 & m_{7,2} & m_{7,3} & m_{7,4} & m_{7,5} & m_{7,6} \end{bmatrix}$$

Чтобы вычислить конечные разности для дальнейшего использования в формуле многочлена Ньютона, применим вложенные циклы:

for j **from** 2 **to** n **do**
  for i **from** 2 **to** $n - j + 2$ **do**
    $R[i, j] := R[i + 1, j - 1] - R[i, j - 1]$
  end do
end do

Для отображения числовых значений матрицы R с точностью до 4-х значащих цифр используем функцию *evalf()* с двумя параметрами.

$evalf(R, 4)$

$$\begin{bmatrix} y & \Delta y & \Delta^2 y & \Delta^3 y & \Delta^4 y & \Delta^5 y \\ 0.12020 & 0.1115 & -0.00220 & 0.03140 & -0.09450 & 0.1466 \\ 0.23170 & 0.1093 & 0.02920 & -0.06310 & 0.05210 & m_{3,6} \\ 0.34100 & 0.1385 & -0.03390 & -0.01100 & m_{4,5} & m_{4,6} \\ 0.47950 & 0.1046 & -0.04490 & m_{5,4} & m_{5,5} & m_{5,6} \\ 0.58410 & 0.05970 & m_{6,3} & m_{6,4} & m_{6,5} & m_{6,6} \\ 0.6438 & m_{7,2} & m_{7,3} & m_{7,4} & m_{7,5} & m_{7,6} \end{bmatrix}$$

Как видим, часть матрицы осталась не инициализированной, поскольку эти элементы не нужны для последующих вычислений.

Реализуем алгоритм построения интерполяционного многочлена Ньютона по первой формуле.

#первая интерполяционная формула Ньютона

$q := \frac{t-x_1}{h}$:

$p := 1$:

$s := y_1$:

for i **from** 1 **to** $n-1$ **do**

$p := p\cdot(q-i+1)$:

$s := s + p\cdot\frac{R[2, i+1]}{i!}$:

end do:

Переменная q аналогична обозначению в формуле, t – независимая переменная для многочлена (вместо x), p накапливает произведение $q(q-1)\dots(q-n+1)$, а s – итоговую сумму. Элементы матрицы *R[2, i+1]* – конечные разности $\Delta^i y_0$.

Функция *normal()* упрощает полученное выражение, а *unapply()* превращает его в функцию *nuton1(t)*.

Вычислим значение полученного многочлена в точке $x = 0{,}13$ и сравним его с результатом, полученным встроенной функцией интерполирования *interp()*.

$xnew := 0.13 : ynew := nuton1(xnew)$

$$0.2918413572$$

$interp(x, y, 0.13)$

$$0.2918413568$$

Значения совпадают с точностью до 10^{-9} (вычислительная погрешность).

#вторая интерполяционная формула Ньютона

$q := \frac{t-x_n}{h}$:

$p := 1$:

$s := y_n$:

for i **from** 1 **to** n − 1 **do**

$p := p\cdot(q+i-1)$:

$s := s + p\cdot\frac{R[n-i+1, i+1]}{i!}$:

end do:

Вторая интерполяционная формула Ньютона использует конечные разности, записанные на побочной диагонали матрицы *R*.

Подстановка координаты неузловой точки:

$$xnew := 0.13 : ynew := nuton2(xnew)$$
$$0.2918413582$$

Результат вычислений по второй интерполяционной формуле Ньютона оказывается на один порядок менее точным, чем по первой. Это связано с расположением точки $x = 0{,}13$ (она ближе к началу отрезка).

Если взять точку, расположенную ближе к правому концу отрезка, то

$$xnew := 0.26 : ynew := nuton2(xnew)$$
$$0.597415745$$
$$interp(x, y, 0.26)$$
$$0.5974157375$$

результат оказывается более точным.

Лабораторная работа № 6

Задание:

Заданный отрезок монотонного возрастания (убывания) заданной функции разбить на 10 равных частей, найти шаг *h*. С помощью первой или второй формулы Ньютона найти значение заданной функции в заданных промежуточных точках. Найденные значения сравнить с точными значениями функции в этих точках.

Варианты заданий:

1. $y = sinx$ $a = 0{,}0;\ b = 2{,}0;\ x^* = 0{,}10;\ 1{,}95$	2. $y = cosx$ $a = 0{,}0;\ b = 1{,}5;\ x^* = 0{,}10;\ 1{,}45$
3. $y = tgx$ $a = 0{,}0;\ b = 1{,}0;\ x^* = 0{,}07;\ 0{,}95$	4. $y = ctgx$ $a = 2{,}0;\ b = 2{,}6;\ x^* = 2{,}02;\ 2{,}57$
5. $y = e^x$ $a = 0{,}0;\ b = 2{,}0;\ x^* = 0{,}10;\ 0{,}9$	6. $y = lnx$ $a = 1{,}0;\ b = 2{,}0;\ x^* = 1{,}06;\ 1{,}95$

7. $y = 2^x$ $a = 0{,}0;\ b = 3{,}0;\ x^* = 0{,}25;\ 2{,}8$	8. $y = \frac{1}{x}$ $a = 1{,}0;\ b = 3{,}0;\ x^* = 1{,}15;\ 2{,}9$
9. $y = \sqrt[3]{x}$ $a = 1{,}0;\ b = 3{,}0;\ x^* = 1{,}15;\ 2{,}9$	10. $y = e^{3x}$ $a = 0{,}0;\ b = 1{,}4;\ x^* = 0{,}15;\ 1{,}37$
11. $y = tg3x+1$ $a = 1{,}9;\ b = 2{,}5;\ x^* = 1{,}93;\ 2{,}77$	12. $y = cos2x+2$ $a = 1{,}0;\ b = 2{,}5;\ x^* = 1{,}10;\ 2{,}43$
13. $y = \sqrt[5]{x}$ $a = 1{,}0;\ b = 2{,}5;\ x^* = 1{,}10;\ 2{,}30$	14. $y = ln5x$ $a = 3{,}0;\ b = 4{,}5;\ x^* = 3{,}10;\ 4{,}40$
15. $y = \frac{1}{3x} + 2$ $a = 2{,}0;\ b = 4{,}5;\ x^* = 2{,}20;\ 4{,}40$	16. $y = sin\pi x+3$ $a = 0{,}0;\ b = 2{,}0;\ x^* = 0{,}03;\ 1{,}90$
17. $y = \sqrt[4]{x}$ $a = 1{,}5;\ b = 3{,}0;\ x^* = 1{,}55;\ 2{,}83$	18. $y = e^{2x+1}$ $a = 0{,}1;\ b = 1{,}6;\ x^* = 0{,}15;\ 1{,}57$
19. $y = \frac{1}{2x} + 1{,}5$ $a = 2{,}7;\ b = 4{,}7;\ x^* = 2{,}80;\ 4{,}60$	20. $y = cos2x+1$ $a = 1{,}1;\ b = 2{,}6;\ x^* = 1{,}15;\ 2{,}53$

Тема 6. Численное дифференцирование

Методические указания

К численному (приближенному) дифференцированию чаще всего прибегают, когда приходится вычислять производные от функций, заданных таблично, или когда непосредственное дифференцирование затруднительно. Для вывода формул численного дифференцирования заменяют функцию *f(x)*, заданную на отрезке [a; b] интерполирующей функцией *P(x)* (чаще всего полиномом), а затем полагают $f'(x) = P'(x)$ при *x* Y [a; b]. Аналогично поступают при нахождении производных высших порядков функции *f(x)*.

Рассмотрим формулы численного дифференцирования, основанные на первой интерполяционной формуле Ньютона. Функцию *y = f(x)* приближенно заменим первой интерполяционной формулой Ньютона:

$$y(x) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_0 + \frac{q(q-1)(q-2)}{3!}\Delta^3 y_0 + \ldots + \frac{q(q-1)\ldots(q-n+1)}{n!}\Delta^n y_0,$$

где $q = \frac{x-x_0}{h}$, $h = x_{i+1} - x_i, i = 0,1,2 \ldots$

Преобразуя данный полином, получим:

$$y(x) = y_0 + q\Delta y_0 + \frac{(q^2 - q)\Delta^2 y_0}{2} + \frac{(q^3 - 3q^2 + 2q)\Delta^3 y_0}{6} + \frac{(q^4 - 6q^3 + 11q^2 - 6q)\Delta^4 y_0}{24} + \frac{(q^5 - 10q^4 + 35q^3 - 50q^2 + 24q)\Delta^5 y_0}{120} + \cdots$$

Так как $\frac{dy}{dx} = \frac{dy}{dq} \cdot \frac{dq}{dx} = \frac{1}{h} \cdot \frac{dy}{dq}$, то

$$y'(x) = \frac{1}{h}\left(\Delta y_0 + \frac{(2q-1)\Delta^2 y_0}{2} + \frac{(3q^2 - 6q + 2)\Delta^3 y_0}{6} + \frac{(4q^3 - 18q^2 + 22q - 6)\Delta^4 y_0}{24} + \frac{(5q^4 - 40q^3 + 105q^2 - 100q + 24)\Delta^5 y_0}{120} + \cdots\right)$$

Аналогично, так как $y''(x) = \frac{d(y')}{dx} = \frac{d(y')}{dq} \cdot \frac{dq}{dx} = \frac{1}{h} \cdot \frac{d(y')}{dq}$,

то $y''(x) = \frac{1}{h^2}(\Delta^2 y_0 + (q-1)\Delta^3 y_0 + \frac{6q^2 - 18q + 11}{12}\Delta^4 y_0 \ldots)$.

Таким же способом можно вычислить производные функции *y(x)* любого порядка.

Заметим, что при нахождении производных $y'(x)$, $y''(x)$ в фиксированной точке x в качестве x_0 нужно выбирать ближайшее табличное значение аргумента.

Формулы для нахождения производных значительно упрощаются, если исходным значением x оказывается один из узлов таблицы. Поскольку в этом случае каждый узел можно считать исходным, то, положив $x = x_0$, т.е. $t = 0$, получим следующие формулы:

$$y'(x) = \frac{1}{h}\left(\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} + \cdots + (-1)^{n-1}\frac{\Delta^n y_0}{n}\right),$$

$$y''(x) = \frac{1}{h^2}\left(\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0 - \frac{5}{6}\Delta^5 y_0 + \cdots\right).$$

Пример 6.1. Используя значения функции $y(x) = \sqrt{x}$ в заданных точках, вычислить приближенные значения первой и второй производных этой функции в точках $x_0 = 30$ и $x^* = 31{,}5$ с помощью MS EXCEL. Найти точные значения производных, полученные результаты сравнить.

Решение.

1. Вводим в ячейки **C5, E5** и **G5** значение шага h и точки, в которых надо найти значения производных.
2. Вводим значение x_0 в ячейку **B7**.
3. Вводим формулу расчета следующего узла в ячейку **B8 [=B7+$C$5]** и распространяем ее до ячейки **B17**.
4. Вводим значение $y(x_0)$ в ячейку **C7 [=КОРЕНЬ(B7)]** и распространяем до ячейки **C17**.

5. Вводим формулу расчета первой конечной разности в ячейку **D7 [=C8-C7]** и распространяем ее до ячейки **D16**.
6. Вводим формулу расчета второй конечной разности в ячейку **E7 [=D8-D7]** и распространяем ее до ячейки **E15**.
7. Вводим формулу расчета третьей конечной разности в ячейку **F7 [=E8-E7]** и распространяем ее до ячейки **F14**.
8. Вводим формулу расчета четвертой конечной разности в ячейку **G7 [=F8-F7]** и распространяем ее до ячейки **G13**.

Замечание. Вычислять конечные разности более высоких порядков нет смысла ввиду их малости.

	A	B	C	D	E	F	G
1							
2		Вычислить первую и вторую производные					
3		функции в заданных точках					
4							
5		*h*=	1	*x*=	30	*x*=	31,5
6		*x*	*yi*	*d1y*	*d2y*	*d3y*	*d4y*
7		30	5,4772	0,09054	-0,00145	0,00007	-0,00001
8		31	5,5678	0,08909	-0,00138	0,00006	0,00000
9		32	5,6569	0,08771	-0,00132	0,00006	0,00000
10		33	5,7446	0,08639	-0,00126	0,00005	0,00000
11		34	5,8310	0,08513	-0,00121	0,00005	0,00000
12		35	5,9161	0,08392	-0,00116	0,00005	0,00000
13		36	6,0000	0,08276	-0,00111	0,00004	0,00000
14		37	6,0828	0,08165	-0,00107	0,00004	
15		38	6,1644	0,08058	-0,00103		
16		39	6,2450	0,07956			
17		40	6,3246				
18							
19		*q*=	1,5				
20		*y′(31,5)*=	0,089087078		*y′′(31,5)*=	-0,00141414	
21		точн	0,089087081		точн	-0,00141408	
22							
23		y′(30)=	0,091286976		y′′(30)=	-0,00152096	

9. Вводим формулу расчета величины q в ячейку **C19[=(G5-B7)/C5]**.
10. Вводим формулу расчета первой производной в точке x = 31,5 в ячейку **C20 [=(D7+(2*C19-1)*E7/2+(3*C19^2-6*C19+2)*F7/6 +(4*C19^3-18*C19^2+22*C19-6)*G7/24)/C5]**.
11. Вводим формулу расчета второй производной в точке x = 31,5 в ячейку **F20 [=(E7+(C19-1)*F7+(6*C19^2-18*C19+11)*G7/12)/(C5^2)]**.
12. Вводим точное значение первой производной в точке x = 31,5 в ячейку **C21[=1/(2*КОРЕНЬ(G5))]**.
13. Вводим точное значение второй производной в точке x = 31,5 в ячейку **F21[==-1/(4*G5^(3/2))]**.
14. Вводим формулу расчета первой производной в точке x = 30 в ячейку **C23 [=(D7-E7/2+F7/3-G7/4)/C5]**.

15. Вводим формулу расчета второй производной в точке $x = 30$ в ячейку **F23 [=(E7-F7+11/12*G7)/(C5^2)]**.
16. Вводим точное значение первой производной в точке $x = 30$ в ячейку **C24[=1/(2*КОРЕНЬ(E5))]**.
17. Вводим точное значение второй производной в точке $x = 30$ в ячейку **F24[=-1/(4*КОРЕНЬ(E5^3))]**.

Ответ: $y'(31{,}5) = 0{,}0891$; $y''(31{,}5) = -0{,}0014$;

$y'(30) = 0{,}0913$; $y''(30) = -0{,}0015$.

Пример 6.2. Используя значения функции $y(x) = \sqrt{x}$ в заданных точках, вычислить приближенные значения первой и второй производных этой функции в точках $x_0 = 30$ и $x^* = 31{,}5$ с помощью MS EXCEL. Найти точные значения производных, полученные результаты сравнить.

Решение.

Координаты точек вычислим с шагом $h = 1$, начиная с $x_0 = 30$ до $x_n = 40$, сохраним их в массивах x и y. Для конечных разностей определим двумерный массив R, первый столбец заполняем значениями массива y, столбцы с 1-го по 4-й заполняем не полностью по формулам конечных разностей.

```
f := u → √u :
n := 10 :
h := 1 :
x := array(0..n) :
y := array(0..n) :
x[0] := 30 : y[0] := evalf(f(x[0])) :
for i from 1 to n do
x[i] := x[i-1] + h;
y[i] := evalf(f(x[i]));
end do:

R := array(0..n, 0..n) :
k := iquo(n, 2)   #целочисленное деление
for i from 0 to n do
R[i, 0] := y[i]
end do:
for j from 1 to k -1 do
for i from 0 to n - j do
R[i, j] := evalf(R[i + 1, j - 1] - R[i, j - 1]) :
end do:
end do:
```

Задаем выражения для вычисления первой (*df1*) и второй (*df2*) производных.

$$df1 := \left(R[0,1] + \frac{(2\cdot q-1)\cdot R[0,2]}{2} + \frac{\left(3\cdot q^2-6\cdot q+2\right)\cdot R[0,3]}{6} + \frac{\left(4\cdot q^3-18\cdot q^2+22\cdot q-6\right)\cdot R[0,4]}{24}\right)\cdot\frac{1}{h}:$$

$$df2 := \left(R[0,2] + (\mathrm{q}-1)\cdot R[0,3] + \frac{\left(6\cdot q^2-18\cdot q+11\right)\cdot R[0,4]}{12}\right)\cdot\frac{1}{h^2}:$$

Вычисляем значение производных в точках 31.5 и 30 по формулам численного дифференцирования. Определяем переменную q, подставляем t с помощью команды *subs*.

$$p1 := subs\left(q=\frac{t-x_0}{h}, df1\right); p2 := subs\left(q=\frac{t-x_0}{h}, df2\right);$$

$$subs(t=31.5, p1); subs(t=31.5, p2);$$

$$0.08908707592$$

$$-0.001414135750$$

$$subs(t=30, p1); subs(t=30, p2);$$

$$0.09128698230$$

$$-0.001520978500$$

$$p := \frac{\mathrm{d}}{\mathrm{d}u}f(u) : \mathrm{pp} := \frac{\mathrm{d}}{\mathrm{d}u}p :$$

$$p1 := unapply(p, u); p2 := unapply(pp, u);$$

$$u \rightarrow \frac{1}{2\sqrt{u}}$$

$$u \rightarrow -\frac{1}{4u^{3/2}}$$

$$evalf(p1(31.5)); evalf(p2(31.5));$$

$$0.08908708065$$

$$-0.001414080645$$

$$evalf(p1(30)); evalf(p2(30));$$

$$0.09128709293$$

$$-0.001521451549$$

Вычисляем производные с помощью встроенных средств дифференцирования, преобразуем полученные выражения в функции командой *unapply* и находим значения в заданных точках.

Сравнивая результаты, делаем вывод о том, что ответы совпадают с точностью до 10^{-8}.

Лабораторная работа № 7

Задание:

1. Вычислить значения первой и второй производных заданных функций в заданных точках.
2. Сравнить полученные результаты с точным решением.

Варианты заданий:

1. $y(x) = \sqrt[3]{x}$; $x_0 = 40$; $x^* = 41{,}7$; $h = 1$; $n = 10$	2. $y(x) = \frac{1}{x}$; $x_0 = 10$; $x^* = 11{,}5$; $h = 1$; $n = 10$
3. $y(x) = e^x$; $x_0 = 0$; $x^* = 0{,}15$; $h = 0{,}1$; $n = 10$	4. $y(x) = cosx$; $x_0 = 0$; $x^* = 0{,}15$; $h = 0{,}1$; $n = 10$
5. $y(x) = x^4$; $x_0 = 2$; $x^* = 2{,}3$; $h = 0{,}5$; $n = 10$	6. $y(x) = lnx$; $x_0 = 0{,}9$; $x^* = 0{,}96$; $h = 0{,}1$; $n = 10$
7. $y(x) = x^3 + 2x$; $x_0 = 1$; $x^* = 1{,}7$; $h = 2$; $n = 10$	8. $y(x) = \sqrt{x+2}$; $x_0 = 2$; $x^* = 2{,}4$; $h = 0{,}6$; $n = 10$
9. $y(x) = sinx$; $x_0 = 0$; $x^* = 0{,}15$; $h = 0{,}1$; $n = 10$	10. $y(x) = \sqrt{x-3}$; $x_0 = 5$; $x^* = 5{,}5$; $h = 0{,}1$; $n = 10$
11. $y(x) = \frac{1}{x^2}$; $x_0 = 1$; $x^* = 1{,}15$; $h = 0{,}2$; $n = 10$	12. $y(x) = \frac{1}{x+1}$; $x_0 = 0$; $x^* = 0{,}14$; $h = 0{,}1$; $n = 10$
13. $y(x) = \sqrt{x+2}$; $x_0 = 2$; $x^* = 2{,}4$; $h = 0{,}6$; $n = 10$	14. $y(x) = x^3$; $x_0 = 0{,}1$; $x^* = 0{,}15$; $h = 0{,}1$; $n = 10$
15. $y(x) = 2\ln(x+1)$; $x_0 = 0$; $x^* = 0{,}15$; $h = 0{,}1$; $n = 10$	16. $y(x) = sin2x$; $x_0 = 0$; $x^* = 0{,}07$; $h = 0{,}1$; $n = 10$
17. $y(x) = e^{x+2}$; $x_0 = 0$; $x^* = 0{,}7$; $h = 0{,}3$; $n = 10$	18. $y(x) = x^4 - x$; $x_0 = 0{,}1$; $x^* = 0{,}2$; $h = 0{,}2$; $n = 10$

19. $y(x) = x^3 + 2x$; $x_0 = 1$; $x^* = 1{,}7$; $h = 2$; $n = 10$	20. $y(x) = \ln(x+1)$; $x_0 = 0{,}9$; $x^* = 0{,}96$; $h = 0{,}1$; $n = 10$

Тема 7. Численное интегрирование функций

Методические указания

На практике большое число задач сводится к вычислению значения определенного интеграла некоторой функции. Если функция *f(x)* непрерывна на интервале [a, b] и известна ее первообразная *F(x),* то определенный интеграл от этой функции в пределах от a до b можно вычислить по формуле Ньютона–Лейбница $\int_a^b f(x)\,dx = F(b) - F(a)$, где $F'(x)$ = *f(x).* Часто первообразную *F(x)* невозможно выразить через элементарные функции. Кроме того, подынтегральная функция *f(x)* может задаваться в виде таблицы ее значений на фиксированном конечном множестве точек. В этом случае понятие первообразной теряет смысл, поэтому для вычисления интеграла применяют численные (приближенные) методы. Задача численного интегрирования состоит в нахождении приближенного значения определенного интеграла с помощью некоторой приближенной формулы через известные значения подынтегральной функции *f(x)* в заданных точках.

7.1. Методы прямоугольников и трапеций

Метод прямоугольников – метод численного интегрирования функции одной переменной, заключающийся в замене подынтегральной функции на многочлен нулевой степени, то есть константу, на каждом элементарном отрезке. Если рассмотреть график подынтегральной функции, то метод будет заключаться в приближенном вычислении площади под графиком суммированием площадей конечного числа прямоугольников, ширина которых будет определяться расстоянием между соответствующими соседними узлами интегрирования, а высота – значением подынтегральной функции в этих узлах.

Формула левых прямоугольников имеет вид $\int_a^b y\,dx = h\sum_0^{n-1} y(x_i)$;

правых прямоугольников – $\int_a^b y\,dx = h\sum_1^n y(x_i)$;

средних прямоугольников – $\int_a^b y\,dx = h\sum_1^n y(x_i - \frac{h}{2})$.

Метод трапеций – метод численного интегрирования функции одной переменной, заключающийся в замене на каждом элементарном отрезке подынтегральной функции на многочлен первой степени, то есть линейную функцию. Площадь под графиком функции аппроксимируется прямоугольными трапециями.

Формула трапеций – $\int_a^b y\,dx = h\sum_{i=0}^{n-1}(\frac{y(x_{i+1}+x_i}{2})$.

Пример 7.1. Найти значение определенного интеграла $\int_1^2 (e^x + 2x)\,dx$ с помощью методов левых, правых, средних прямоугольников и метода трапеций в MS EXCEL. Результаты сравнить с точным значением интеграла.

Решение.

Системой узлов отрезок интегрирования [1; 2] разделим на n равных частей $x_i = x_0 + ih, i = \overline{0,n}$; $x_0 = 1, x_n = 2, h = \frac{b-a}{n}$; и вычислим значения подынтегральной функции в полученных узлах $y_i = e^{x_i} + 2x_i$, $i = \overline{0,n}$. Положим n = 20, $h = \frac{2-1}{20} = 0{,}05$.

Выполнение алгоритма в EXCEL:

Вычислить заданный определенный интеграл с помощью формул левых, правых, средних прямоугольников и формулы трапеций

i	x	f(x)	лев	прав	сред	трап
0	1	4,7183	0,0000	0,0000	0,0000	0,0000
1	1,05	4,9577	0,2359	0,2479	0,2419	0,2419
2	1,1	5,2042	0,4838	0,5081	0,4959	0,4959
3	1,15	5,4582	0,7440	0,7810	0,7624	0,7625
4	1,2	5,7201	1,0169	1,0670	1,0418	1,0420
5	1,25	5,9903	1,3029	1,3665	1,3345	1,3347
6	1,3	6,2693	1,6024	1,6800	1,6409	1,6412
7	1,35	6,5574	1,9159	2,0079	1,9615	1,9619
8	1,4	6,8552	2,2438	2,3506	2,2968	2,2972
9	1,45	7,1631	2,5865	2,7088	2,6472	2,6477
10	1,5	7,4817	2,9447	3,0829	3,0132	3,0138
11	1,55	7,8115	3,3188	3,4734	3,3955	3,3961
12	1,6	8,1530	3,7093	3,8811	3,7945	3,7952
13	1,65	8,5070	4,1170	4,3064	4,2109	4,2117
14	1,7	8,8739	4,5423	4,7501	4,6454	4,6462
15	1,75	9,2546	4,9860	5,2129	5,0985	5,0995
16	1,8	9,6496	5,4488	5,6953	5,5710	5,5721
17	1,85	10,0598	5,9313	6,1983	6,0637	6,0648
18	1,9	10,4859	6,4342	6,7226	6,5772	6,5784
19	1,95	10,9287	6,9585	7,2691	7,1125	7,1138
20	2	11,3891	7,5050	7,8385	7,6703	7,6717

a	b	N	h
1	2	20	0,05

точное значение интеграла

I=	7,6708
Iлев =	7,5050
Iправ=	7,8385
Iсред=	7,6703
Iтрап=	7,6717

1. Вводим пределы интегрирования 1 и 2 в ячейки **H6:I6**, число интервалов 20 – в ячейку **J6**.
2. Вводим формулу вычисления шага h разбиения отрезка интегрирования в ячейку **K6 [=(I6-H6)/J6]**.
3. Вводим нули в ячейки **D6, E6, F6** и **G7**.
4. Вводим ссылку на левый предел интегрирования в ячейку **B6 [=H6]**.
5. Вводим формулу расчета следующего узла интегрирования в ячейку **B7 [=B6+$K$6]** и распространяем ее до ячейки **B26**.

6. Вводим формулу расчета подынтегральной функции в ячейку **C6 [=EXP(B6)+2*B6]** и распространяем ее до ячейки **C20**.
7. Вводим значение площади левого прямоугольника в ячейку **D7 [=D6+$K$6*C6]** и распространяем до ячейки **D20**.
8. Вводим значение площади правого прямоугольника в ячейку **E7 [=E6+$K$6*C7]** и распространяем до ячейки **E20**.
9. Вводим значение площади среднего прямоугольника в ячейку
10. **F7 [=F6+$L$6*(EXP(B7-$L$6/2)+2*(B7-$L$6/2))]** и распространяем до ячейки **F20**.
11. Вводим значение площади трапеции в ячейку **G7 [=F6+$K$6*(C6+C7)/2]** и распространяем до ячейки **G20**.

Значение заданного интеграла, вычисленное по методу левых прямоугольников, содержится в ячейке **D20,** по методу правых прямоугольников – в ячейке **E20,** по методу средних прямоугольников – в ячейке **F20,** по методу трапеций **–** в ячейке **G20.** Сравниваем эти значения с точным решением, вычисленным в ячейке **J9**.

Ответ: $I_{лев}$=7,5050; $I_{прав}$=7,8385; $I_{сред}$=7,6703; $I_{трап}$=7,6717.

Пример 7.2. Найти значение определенного интеграла $\int_1^2 (e^x + 2x)\,dx$ с помощью методов левых, правых, средних прямоугольников и метода трапеций в Maple. Результаты сравнить с точным значением интеграла.

Решение.

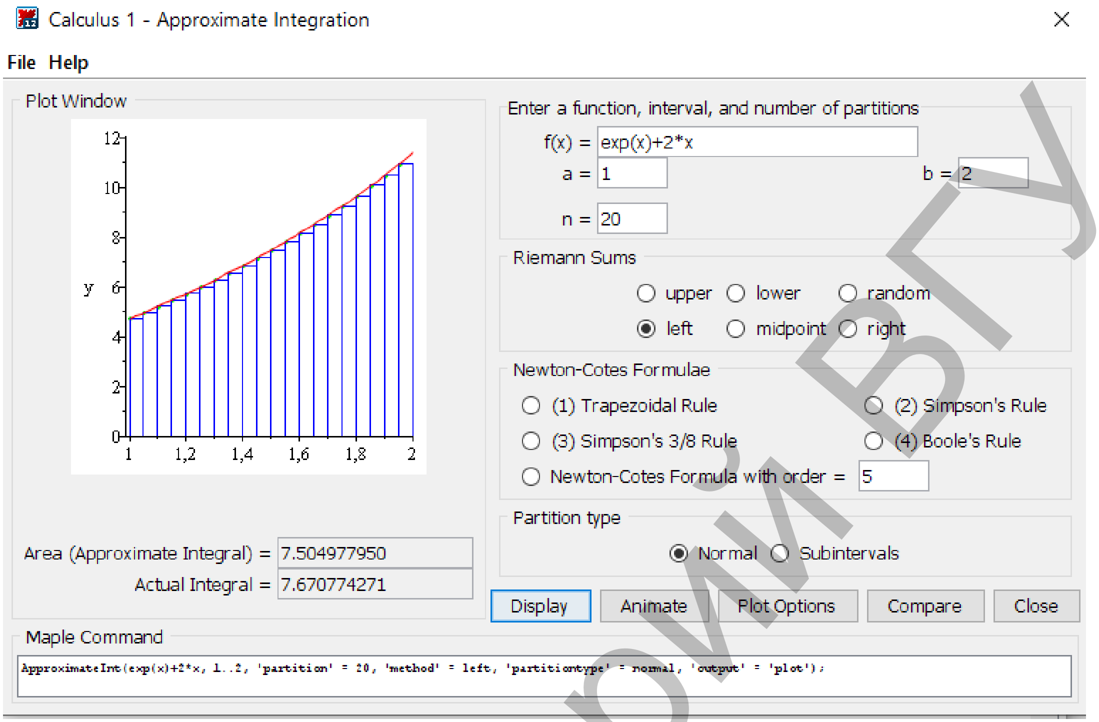
Стандартная функция в Maple, предназначенная для вычисления интеграла – *int(f, x = x1..x2),* дает следующий результат:

$$int(\exp(x) + 2*x, x = 1..2)$$

$$-e + 3 + e^2$$

Чтобы представить его не в символьной, а в числовой форме, можно использовать функцию *evalf(%)*, % означает последнее полученное выражение. Получим значение: 7.670774271.

В Maple можно наглядно продемонстрировать различные методы численного интегрирования, если использовать специальный пакет расширения для изучения математики **Student**, а точнее – его подпакет Calculus1 (вычисления с функциями одной переменной). Он содержит интерактивные инструменты визуализации решения. Доступ к нему можно получить из меню (Tools → Tutors → Calculus-Single Variable → **Aproximate Integration**). В появившемся интерактивном окне вводим функцию, интервал и число разбиений, выбираем метод (на рисунке – левых прямоугольников). В левой части окна отображаются графическая иллюстрация метода и числовые значения: приближенной площади криволинейной трапеции и стандартной функции интегрирования. В нижней части окна формируется команда Maple, которую можно скопировать для дальнейшего использования. Далее выбираем способ представления результата, например Display или Animate.



Такой же результат можно получить с помощью команды Maple:

$$with(Student[Calculus1]):$$
$$ApproximateInt(\exp(x)+2*x, 1..2, 'partition' = 20,$$
$$'method' = left, 'output' = 'plot');$$

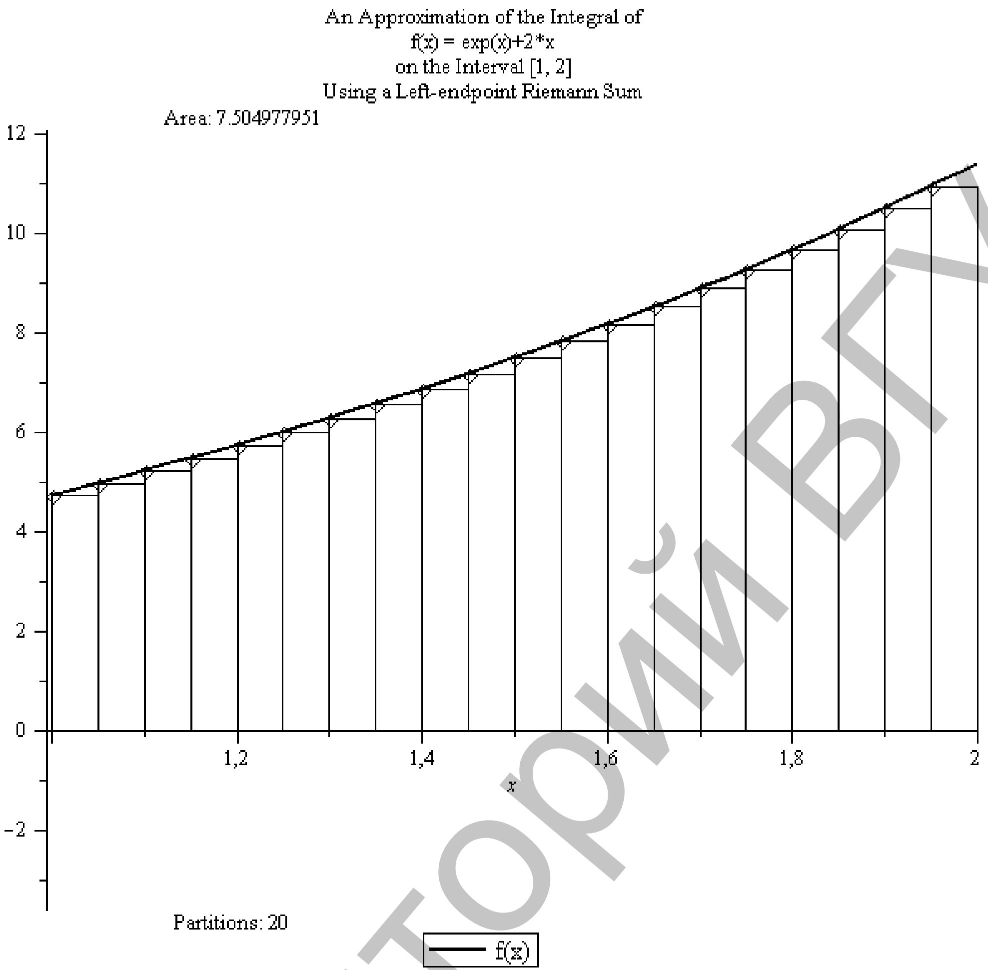
В этой команде в качестве параметра *'output'* можно указать значение *'sum'*, которое можно затем вычислить и получить числовое приближенное значение интеграла выбранным методом.

$$ApproximateInt(\exp(x)+2*x, 1..2, \mathrm{'partition'} = 20, \mathrm{'method'}$$
$$= \mathrm{left}, \mathrm{'output'} = \mathrm{'sum'});$$

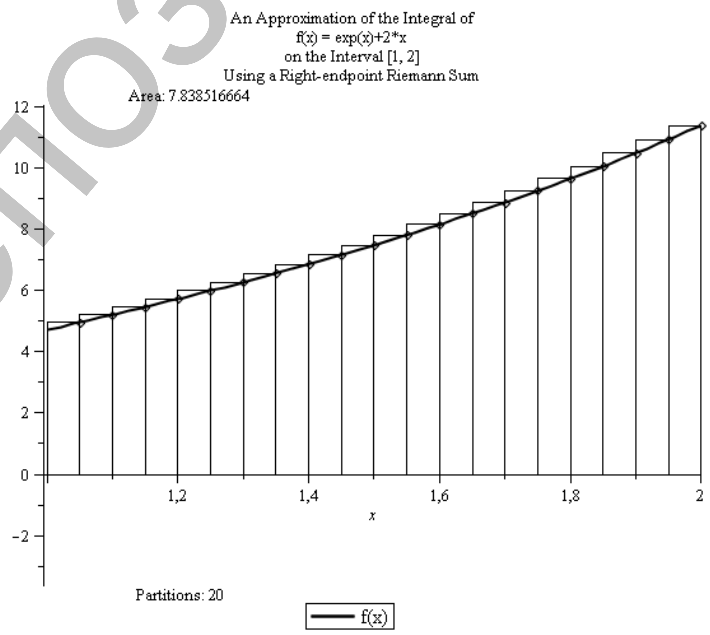
$$\frac{1}{20}\sum_{i=0}^{19}\left(e^{1+\frac{1}{20}i}+2+\frac{1}{10}i\right)$$

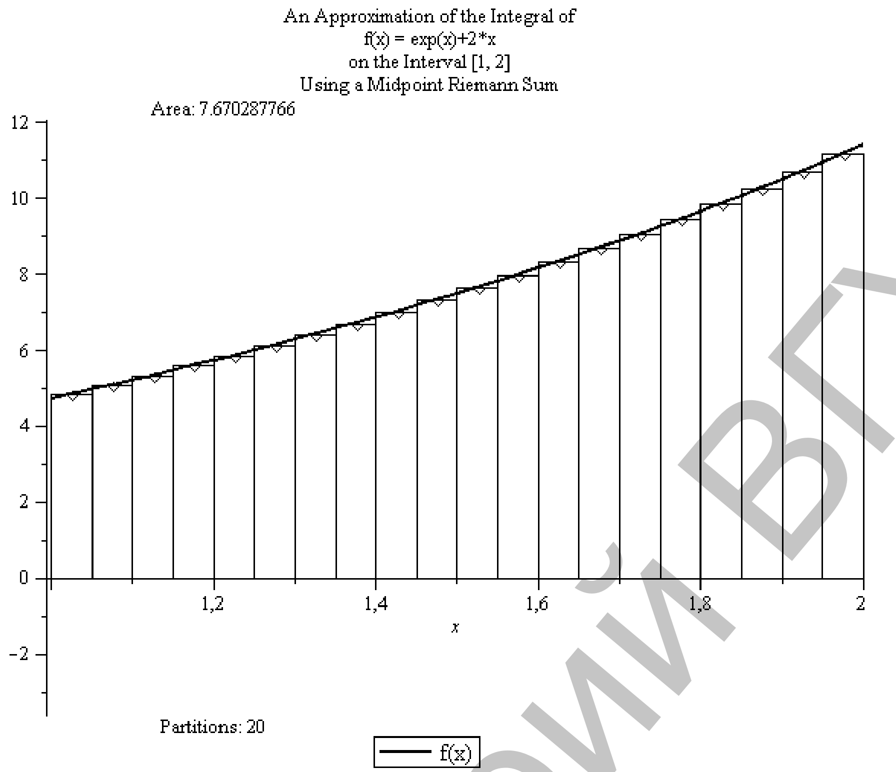
$$evalf(\%)$$
$$7.504977950$$

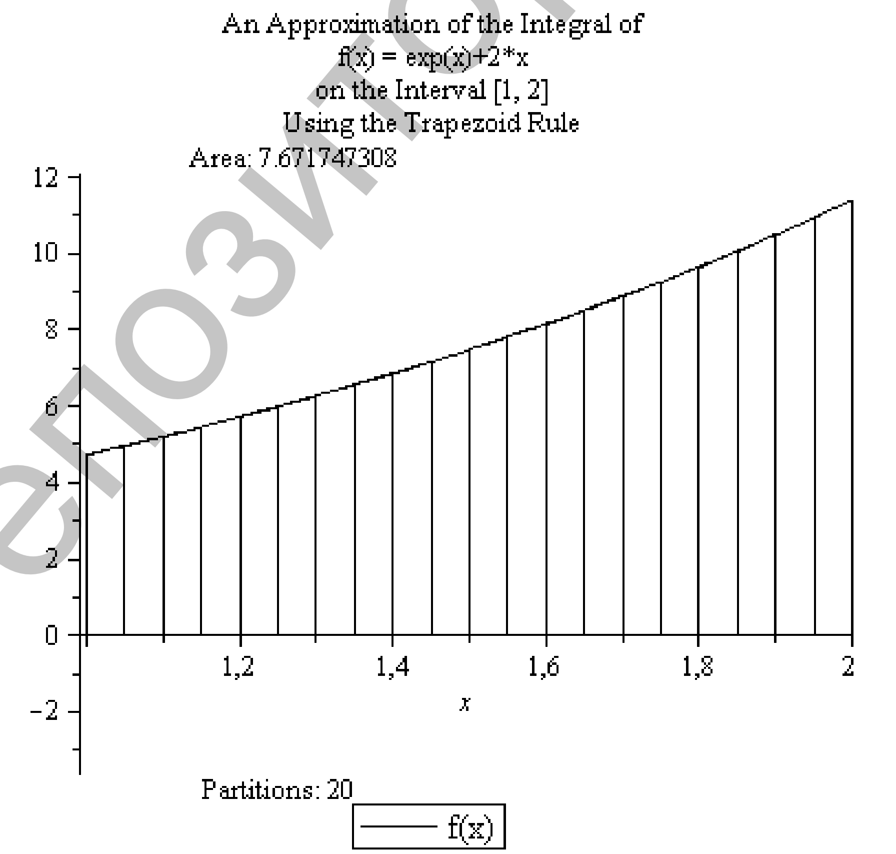
Получим графическую иллюстрацию метода левых прямоугольников.



Изменяя значение параметра *'metod'* на *'right'*, *'midpoint'*, *'trapezoid'*, получим интеграл методом правых прямоугольников, средних или трапеций.




An Approximation of the Integral of
f(x) = exp(x)+2*x
on the Interval [1, 2]
Using a Midpoint Riemann Sum
Area: 7.670287766
12
10
8
6
4
2
0
-2
1,2
1,4
1,6
1,8
2
x
Partitions: 20
f(x)


An Approximation of the Integral of
f(x) = exp(x)+2*x
on the Interval [1, 2]
Using the Trapezoid Rule
Area: 7.671747308
12
10
8
6
4
2
0
-2
1,2
1,4
1,6
1,8
2
x
Partitions: 20
f(x)

Сравнение приближенных значений интеграла, полученных разными методами, с точным (7.670774271…) свидетельствует о том, что у методов средних прямоугольников и трапеций точность на порядок выше относительно шага h, чем у левых и правых прямоугольников.

Пример 7.3. Найти значение определенного интеграла $\int_1^2 (e^x + 2x)\,dx$ с помощью методов левых, правых, средних прямоугольников и метода трапеций. Результаты сравнить с точным значением интеграла. Реализовать решение на языке программирования Python.

Решение.

Опишем подынтегральную функцию `fun(x)`, а также функции для каждого из методов.

```
import math

def fun(x):
    return math.exp(x)+2*x
.
def middle_rectangle(f,a,b,n):
'''Интеграл f(x) от a до b
методом сред. прямоугольн.
при n разбиениях'''

    h = (b - a)/n
    s = 0
    x = a + h/2
    for i in range(n):
        s = s + f(x)
        x = x + h
    s = s*h
    return s

def left_rectangle(f,a,b,n):
'''Интеграл f(x) от a до b
методом левых прямоугольн.
при n разбиениях'''
    h = (b - a)/n
    s = 0
    x = a
    for i in range(n):
        s = s + f(x)
        x = x + h
    s = s*h
    return s

def right_rectangle(f,a,b,n):
'''Интеграл f(x) от a до b
методом правых прямоугольн.
при n разбиениях'''

    h = (b - a)/n
    s = 0
    x = a + h
    for i in range(n):
        s = s + f(x)
        x = x + h
    s = s*h
    return s

def trapeze(f,a,b,n):
'''Интеграл f(x) от a до b
    методом трапеций
    при n разбиениях'''
    h = (b - a)/n
    s = (f(a) + f(b))/2
    x = a + h
    for i in range(n - 1):
        s = s + f(x)
        x = x + h
    s = s*h
    return s
```

В функции методов добавлены документ-строки (текст в тройных кавычках), которые не только удобны для комментирования кода, но и доступны через встроенную атрибут-переменную, например, `left_rectangle.__doc__`.

Вызовем поочередно функции методов с аргументами из примера 7.3 и выведем на печать полученные значения.

```
print('метод сред. прям.:', middle_rectangle(fun,1,2,20))
print('метод лев. прям.:',left_rectangle(fun,1,2,20))
print('метод прав. прям.:',right_rectangle(fun,1,2,20))
print('метод трапеций:',trapeze(fun,1,2,20))
```

Результат:

```
метод сред. прям.: 7.670287766959564
метод лев. прям.: 7.50497795114033
метод прав. прям.: 7.838516664663911
метод трапеций: 7.671747307902121
```

7.2. Метод Симпсона (парабол)

Рассмотрим более точную формулу численного интегрирования – **Симпсона** (формулу парабол):

$$\int_a^b y\,dx = \frac{h}{3}\cdot[(y_0 + y_{2m}) + 4(y_1 + y_3 + \cdots + y_{2m-1}) + 2(y_2 + y_4 + \cdots y_{2m-2})]$$

Заметим, что число разбиений отрезка интегрирования должно быть четным, $n = 2m$, $m = 1, 2, \ldots$. Суть метода заключается в приближении подынтегральной функции на отрезке [a; b] интерполяционным многочленом второй степени, то есть приближение графика функции на отрезке интегрирования параболой.

Пример 7.4. Найти значение определенного интеграла $\int_1^2 (e^x + 2x)\,dx$ с помощью метода Симпсона в MS EXCEL. Результат сравнить с точным значением интеграла.

Решение.

1. Вводим пределы интегрирования 1 и 2 в ячейки **A6:B6**, число интервалов 20 – в ячейку **C6**.
2. Вводим формулу вычисления шага *h* разбиения отрезка интегрирования в ячейку **D6 [=(B6-A6)/C6]**.
3. Вводим ссылку на левый предел интегрирования в ячейку **B8 [=A6]**.
4. Вводим формулу расчета следующего узла интегрирования в ячейку **B9 [=B9+$D$6]** и распространяем ее до ячейки **B28**.
5. Вводим формулу расчета подынтегральной функции в ячейку **C8 [=EXP(B8)+2*B8]** и распространяем ее до ячейки **C28**.

6. Вводим формулу расчета суммы значений подынтегральной функции в четных узлах в ячейку
D8 [=СУММ(ЕЧЁТН(СТРОКА(C10:C26))*(C10:C26))] и нажимаем одновременно **SHIFT+ CTRL+ ENTER.**
7. Вводим формулу расчета суммы значений подынтегральной функции в нечетных узлах в ячейку
E8 [=СУММ(ЕНЕЧЁТ(СТРОКА(C9:C27))*(C9:C27))] и нажимаем одновременно **SHIFT+ CTRL+ ENTER.**

Замечание. Значения четных и нечетных узлов должны располагаться в ячейках с четными и нечетными номерами соответственно.

8. Вводим формулу расчета значения интеграла в ячейку
F8 [=D6/3*(C8+C28+2*D8+4*E8)].

	A	B	C	D	E	F	G	H	I
1									
2	Вычислить заданный определенный интеграл								
3	с помощью формулы Симпсона								
4									
5	a	b	N	h					
6	1	2	20	0,05					
7	i	x	f(x)						
8	0	1	4,718	68,693	76,688	7,67077			
9	1	1,05	4,958						
10	2	1,1	5,204		точное значение интеграла			I=	7,6708
11	3	1,15	5,458						
12	4	1,2	5,72						
13	5	1,25	5,99						
14	6	1,3	6,269						
15	7	1,35	6,557						
16	8	1,4	6,855						
17	9	1,45	7,163						
18	10	1,5	7,482						
19	11	1,55	7,811						
20	12	1,6	8,153						
21	13	1,65	8,507						
22	14	1,7	8,874						
23	15	1,75	9,255						
24	16	1,8	9,65						
25	17	1,85	10,06						
26	18	1,9	10,49						
27	19	1,95	10,93						
28	20	2	11,39						

Ответ: $I_{Симп} = 7{,}67077$; $I_{точн} = 7{,}6708$.

Пример 7.5. Найти значение определенного интеграла $\int_1^2 (e^x + 2x)\,dx$ методом Симпсона. Реализовать решение на языке программирования Python.

Решение.

Отметим, что в алгоритме метода Симпсона в цикле используется двойной шаг, чтобы за одну итерацию суммировать y_i для нечетных и четных i с различными множителями 4 и 2. Поскольку нечетных точек разбиения больше, чем четных, для последнего нечетного i значение $f(b–h)$ добавляется после цикла. Следует отметить, что в этой функции необходима проверка четности параметра n. Чтобы избежать получения ошибочного результата при вызове функции с нечетным значением числа разбиений, в программе предусмотрена обработка исключений. С помощью ключевого слова `raise` внутри функции описываем пользовательское исключение типа `ValueError` с соответствующим сообщением. Затем в блоке `try/except` перехватывается исключение и выводится на печать сообщение об ошибке.

```
import math

def fun(x):
    return math.exp(x)+2*x

def simpson(f,a,b,n):
    '''Интеграл f(x) от a до b
    методом Симпсона (парабол)
    при n разбиениях
    '''
    if n%2 != 0:
        raise ValueError('число разбиений нечетное')
    h = (b - a)/n
    m = n // 2
    s = f(a) + f(b)
    x = a + h
    for i in range(m-1):
        s = s + 4*f(x) + 2*f(x+h)
        x = x + 2*h
    s = s + 4*f(b-h)
    s = s*h/3
    return s
try:
    print('метод Симпсона:',simpson(fun,1,2,11))
    print('метод Симпсона:',simpson(fun,1,2,20))

except ValueError as msg:
    print('ошибка:',msg)
```

Приведем результат работы программы для n = 20 и n = 51.

```
введите количество разбиений:20
метод Симпсона: 7.670774432603015

введите количество разбиений:51
ошибка: число разбиений нечетное
```

Сравнив результаты с известным решением, можем отметить, что метод Симпсона – наиболее точный из рассмотренных, поэтому он чаще остальных используется в библиотечных функциях численного интегрирования.

7.3. Метод Монте-Карло

При вычислении интеграла по формулам прямоугольников отрезок интегрирования [a; b] разбивается на n одинаковых интервалов, на концах которых вычисляются значения подынтегральной функции. Вычисляя значения функции в случайных точках, можно получить статистический вариант метода прямоугольников, который эффективен при вычислении кратных интегралов. Вычислим заданный интеграл методом **Монте-Карло**, который похож на описанный выше метод прямоугольников. Вместо равномерного разделения области интегрирования на интервалы и суммирования площадей получившихся «столбиков» в область интегрирования вбрасываются случайные точки, строятся такие же «столбики», ширина которых определяется как $\frac{b-a}{n}$, затем суммируются их площади. Получать случайные точки будем с помощью преобразования равномерно распределенных на отрезке [0; 1] случайных величин. Приведем моделирующую формулу для равномерного распределения на отрезке [a; b] $x_i = a + \gamma_i(b - a)$, где γ_i – случайные числа, равномерно распределенные на отрезке [0; 1]. Таким образом, значение интеграла вычисляется по формуле $\int_a^b y(x)\,dx \approx \frac{b-a}{n}\sum_{i=1}^{n} y(x_i)$.

Пример 7.5. Найти значение определенного интеграла $\int_1^2 (e^x + 2x)\,dx$ методом Монте-Карло в MS EXCEL. Результат сравнить с точным значением интеграла.

Решение.

1. Вводим пределы интегрирования 1 и 2 в ячейки **A6:B6**, число интервалов 20 – в ячейку **C6**.
2. Вводим функцию расчета равномерно распределенных на отрезке [a; b] случайных чисел в ячейку **B8** **[=СЛЧИС()*($B$6-$A$6)+$A$6]** (функция **СЛЧИС()** возвращает равномерно распределенное случайное число, большее или равное 0 и меньшее 1, которое изменяется при пересчете) и распространяем ее до ячейки **B27**.

3. Вводим формулу расчета подынтегральной функции в ячейку **C8 [=EXP(B8)+2*B8]** и распространяем ее до ячейки **C27**.
4. Вводим формулу расчета значения данного интеграла в ячейку **D8 [=(B6-A6)/C6*СУММ(C8:C27)]**.

	A	B	C	D	E	F	G	H	I
1									
2	Вычислить заданный определенный интеграл								
3	с помощью метода Монте-Карло								
4									
5	*a*	*b*	*N*						
6	1	2	20						
7	i	*x*	*f(x)*						
8	1	2	11,34	7,62882					
9	2	1,75	9,272						
10	3	1,91	10,61		точное значение интеграла			I=	7,6708
11	4	1,22	5,848						
12	5	1,85	10,07						
13	6	1,51	7,573						
14	7	1,11	5,24						
15	8	1,33	6,436						
16	9	1,54	7,725						
17	10	1,18	5,64						
18	11	1,59	8,067						
19	12	1,64	8,437						
20	13	1,11	5,273						
21	14	1,11	5,26						
22	15	1,93	10,77						
23	16	1,29	6,227						
24	17	1,19	5,673						
25	18	1,8	9,64						
26	19	1,29	6,203						
27	20	1,47	7,266						

Ответ: $I_{М\text{-}К} = 7{,}63$, $I_{точн} = 7{,}6708$.

Пример 7.6. Найти значение определенного интеграла $\int_1^2 (e^x + 2x)\,dx$ методом Монте-Карло. Реализовать решение на языке программирования Python с помощью собственной функции.

Решение.

```
import random, math

def fun(x):
    return math.exp(x)+2*x
def monte_karlo(f, a, b, n):
    s = 0
    for i in range(n):
        x = random.uniform(a, b)
```

```
        s += f(x)
    s = (b-a)/n*s
    return s
print(monte_karlo(fun,1,2,1000))
```

В этой программе для получения случайного числа используется функция `uniform` модуля `random`, которая генерирует равномерно распределенные значения на отрезке [a; b].
При многократном выполнении программы получаем разные результаты с учетом случайного выбора узлов. Приведем для примера два результата:

```
7.609246330487758
7.621917207608362
```

В данной теме приведены программы, реализующие алгоритмы рассмотренных методов с помощью простых средств (циклов, функций). Они хорошо работают для гладких монотонных функций. Для вычисления интегралов от функций, имеющих особенности (сингулярности, разрывы) и т.п., необходимо использовать инструменты библиотек для научных и математических расчетов на языке Python. Наиболее известные из них `NumPy` и `SciPy`.

Пример 7.7. Найти значение определенного интеграла $\int_1^2 (e^x + 2x)\,dx$ методом Монте-Карло. Реализовать решение на языке программирования Python с помощью библиотечной функции.

Решение.

Приведем результат вычисления интеграла из примера с помощью функции `quad()` пакета `integrate` библиотеки `SciPy`. Возвращаемым значением этой функции является кортеж, в котором первый элемент содержит оценочное значение интеграла, а второй элемент – верхнюю границу ошибки.

```
import scipy.integrate as integrate
import math

def fun(x):
    return math.exp(x)+2*x

result = integrate.quad(fun,1,2)
print(result)
```

Результат: `(7.670774270471606, 8.516270211779814e-14)`

Лабораторная работа № 8

Задание:

1. Вычислить значение заданного определенного интеграла с помощью приближенных методов:
   a) прямоугольников;
   b) трапеций;
   c) Симпсона;
   d) Монте-Карло.
2. Найти точное значение. Результаты сравнить.

Варианты заданий:

1. $\int_{1}^{2,4}(2x^3+sinx)\,dx$	2. $\int_{2}^{4}(3x^2-4x)\,dx$
3. $\int_{2}^{3,2}(\pi lnx+2)\,dx$	4. $\int_{1}^{2}(e^x+sinx)\,dx$
5. $\int_{-2}^{1}(9+2x-x^2)\,dx$	6. $\int_{0}^{1,6}(sinx+cosx)\,dx$
7. $\int_{1}^{2,2}(x^4+sin2x+x)\,dx$	8. $\int_{0,7}^{1,9}(lnx+3x^2)\,dx$
9. $\int_{3}^{4,4}(tgx+x)\,dx$	10. $\int_{2,1}^{3,3}(cos3x+\sqrt{x})\,dx$
11. $\int_{0,01}^{1,2}(x^3+4x^2+e^{3x})\,dx$	12. $\int_{1,8}^{3}(ctgx+cosx)\,dx$
13. $\int_{-3}^{-1,8}(x^4+5x-sinx)\,dx$	14. $\int_{0}^{1}(e^{3x}-x^4+1)\,dx$
15. $\int_{2}^{3}(x^3+5x+cosx)\,dx$	16. $\int_{2,8}^{4}(sin5x+x)\,dx$
17. $\int_{5,1}^{6,1}(cos3x+3x)\,dx$	18. $\int_{2,2}^{3,4}(sin(x+3)+0,5x^2)dx$
19. $\int_{3}^{4}(x^3-5x^2+sinx)\,dx$	20. $\int_{0}^{1,5}(e^{2x+1}-x^3+1,7)\,dx$

Тема 8. Приближенное решение обыкновенных дифференциальных уравнений. Задача Коши

Методические указания

Пусть требуется решить задачу Коши

$$\begin{cases} \acute{y} = f(x, y) \\ y(x_0) = y_0 \end{cases}, \text{ где } x, x_0 \in [a, b]$$

Часто при решении практических задач точные методы решения задачи Коши для обыкновенного дифференциального уравнения первого порядка оказываются либо совсем беспомощными, либо их решение связывается с недопустимыми затратами усилий и времени. В этом случае используются методы численного (приближенного) решения обыкновенных дифференциальных уравнений.

Решить поставленную задачу Коши численно – для заданной последовательности чисел $x_0, x_1, \dots, x_n$ из отрезка [a; b] и числа y_0, не определяя самого решения $y = y(x)$, вычислить (приближенно) значения $y_1, y_2, \dots, y_n$ этого решения в заданных точках. Другими словами, приближенное решение поставленной задачи позволяет вместо отыскания ее решения $y = y(x)$ получить таблицу значений этой функции для заданной последовательности аргументов:

x_i	x_0	x_1	…	x_n
$y(x_i)$	y_0	y_1	…	y_n

8.1. Методы Эйлера и Эйлера–Коши

С помощью **метода Эйлера** можно вычислить значения искомой функции $y(x_i)$ по формуле $y_{i+1} = y_i + hf(x_i, y_i)$, где $h = x_{i+1} - x_i, i = \overline{0, n-1};$ т.е. построить ломаную, аппроксимирующую искомую интегральную кривую. Метод Эйлера имеет невысокую точность. Более точным является усовершенствованный метод Эйлера, когда сначала вычисляют промежуточные значения:

$$x_{i+\frac{1}{2}} = x_i + \frac{h}{2}, \quad y_{i+\frac{1}{2}} = y_i + \frac{h}{2} f(x_i, y_i),$$

а потом полагают $y_{i+1} = y_i + hf\left(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}}\right)$.

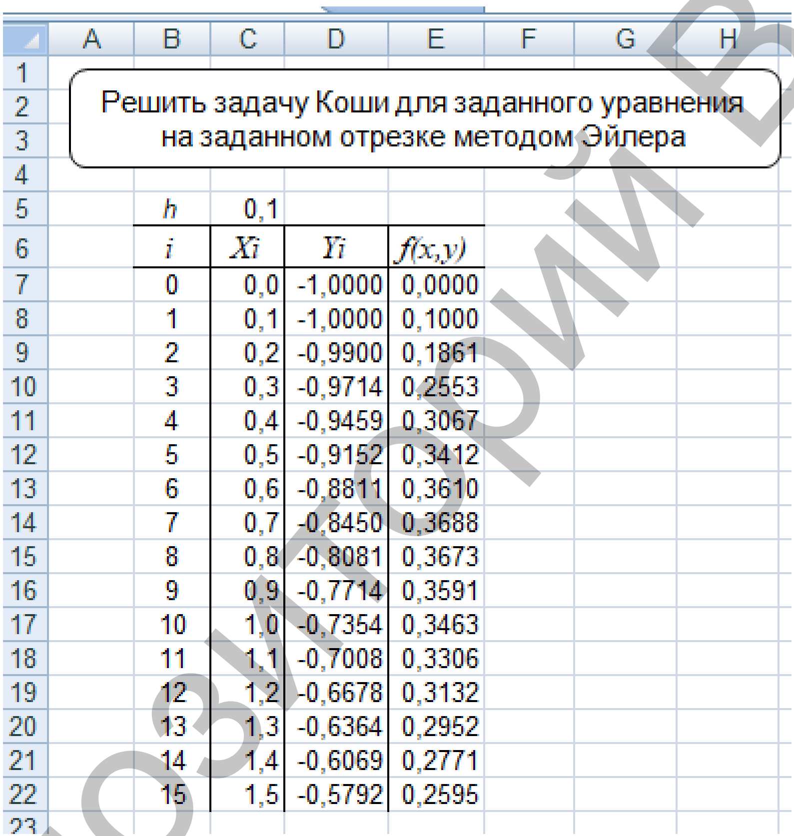
Другой модификацией метода Эйлера является усовершенствованный метод Эйлера–Коши. Его сущность заключается в следующем: сначала выбирают «грубое» приближение к решению – $\tilde{y}_{i+1} = y_i + hf(x_i, y_i)$, вычисляют $\tilde{f}_{i+1} = f(x_{i+1}, \tilde{y}_{i+1})$, а затем приближенно полагают

$$y_{i+1} = y_i + h\frac{f(x_i,y_i)+f(x_{i+1},\tilde{y}_{i+1})}{2}.$$

Пример 8.1. Решить задачу Коши на отрезке [0; 1,5] для заданного уравнения методом Эйлера в MS EXCEL.

$$\begin{cases} \acute{y} = y + (1+x)y^2 \\ y(0) = -1 \end{cases}, h = 0{,}1$$

Решение.



Решить задачу Коши для заданного уравнения на заданном отрезке методом Эйлера

h	0,1		
i	*Xi*	*Yi*	*f(x,y)*
0	0,0	-1,0000	0,0000
1	0,1	-1,0000	0,1000
2	0,2	-0,9900	0,1861
3	0,3	-0,9714	0,2553
4	0,4	-0,9459	0,3067
5	0,5	-0,9152	0,3412
6	0,6	-0,8811	0,3610
7	0,7	-0,8450	0,3688
8	0,8	-0,8081	0,3673
9	0,9	-0,7714	0,3591
10	1,0	-0,7354	0,3463
11	1,1	-0,7008	0,3306
12	1,2	-0,6678	0,3132
13	1,3	-0,6364	0,2952
14	1,4	-0,6069	0,2771
15	1,5	-0,5792	0,2595

1. Вводим значение шага *h* в ячейку **C5**.
2. Вводим значение x_0 в ячейку **C7**.
3. Вводим формулу расчета следующего узла в ячейку **C8 [=C7+$C$5]** и распространяем ее до ячейки **C22**.
4. Вводим значение $y(x_0)$ в ячейку **D7**.
5. Вводим формулу расчета значений $f(x_i, y_i)$ в ячейку **E7** **[=D7+(1+C7)*D7^2]** и распространяем ее до ячейки **E22**.
6. Вводим формулу расчета значений искомой функции в ячейку **D8 [=D7+$C$5*E7]** и распространяем ее до ячейки **D22.**

Ответ: решением задачи Коши для заданного уравнения является функция, заданная таблично. Для значений x_i в ячейках **C6:C22** значения искомой функции $y(x_i)$ находятся в ячейках **D6:D22**.

Пример 8.2. Решить задачу Коши на отрезке [0; 1,5] для заданного уравнения методом Эйлера в Maple. Найти точное аналитическое решение поставленной задачи и сравнить с приближенным.

$$\begin{cases} y' = y + (1+x)y^2 \\ y(0) = -1 \end{cases}, h = 0{,}1$$

Решение.

Воспользуемся командой *dsolve* математического пакета Maple, чтобы найти точное решение задачи Коши из примера.

$$odu := \frac{d}{dx}\, y(x) = y(x) + (1+x)\cdot y(x)^2$$

$$\frac{d}{dx}\, y(x) = y(x) + (1+x)\, y(x)^2$$

$$dsolve(\{odu, y(0) = -1\}, y(x))$$

$$y(x) = -\frac{1}{x + e^{-x}}$$

Таким образом, решение обыкновенного дифференциального уравнения с заданным начальным условием в аналитической форме имеет вид:

$$y = -\frac{1}{x + e^{-x}}.$$

Для реализации метода Эйлера в Maple создадим процедуру.

$eiler :=$ **proc**(f :: procedure, x0, y0, xend, h)
local x, y, n, i :
$x_0 :=$ x0 : $y_0 :=$ y0 :
$n := \text{round}\left(\frac{xend - x0}{h}\right)$:
for i **from** 1 **to** n **do**
$x_i := x_{i-1} + \text{h}$:
$y_i := y_{i-1} + h\cdot f(x_{i-1}, y_{i-1})$
end do:
return y
end proc:

Явная типизация первого параметра как *procedure*, а не *function* объясняется способом задания функции правой части дифференциального уравнения $fun := (x, y) \rightarrow y + (1 + x) \cdot y^2$.

Вызовем процедуру и выведем на печать полученные табличные значения

$r := eiler(fun, 0., -1., 1.5, 0.1)$:
$i := 0$:
for x **from** 0 **by** 0.1 **to** 1.5 **do**
printf("%4.1f,%7.3f\n", x, r_i) :
$i := i + 1$
end do:

Учитывая форматный вывод, получим следующий результат

```
 0.0, -1.000
 0.1, -1.000
 0.2, -0.990
 0.3, -0.971
 0.4, -0.946
 0.5, -0.915
 0.6, -0.881
 0.7, -0.845
 0.8, -0.808
 0.9, -0.771
 1.0, -0.735
 1.1, -0.701
 1.2, -0.668
 1.3, -0.636
 1.4, -0.607
 1.5, -0.579
```

8.2. Метод Рунге–Кутта

Наибольшее распространение из приближенных методов решения обыкновенных дифференциальных уравнений получил метод Рунге–Кутта. Этот метод является одним из методов повышенной точности и принадлежит к многошаговым методам численного интегрирования задачи Коши.

Вычисление приближенного значения искомой функции $y(x_{i+1})$, $i = \overline{0, n-1}$ или y_{i+1} при решении задачи Коши в точке x_{i+1} методом Рунге–Кутта заключается в выполнении следующих операций:

1) на каждом *i*-ом шаге вычисляются коэффициенты:

$$K_1 = f(x_i, y_i),$$

$$K_2 = f(x_i + \frac{h}{2}, y_i + h\frac{K_1}{2}),$$

$$K_3 = f(x_i + \frac{h}{2}, y_i + h\frac{K_2}{2}),$$

$$K_{24} = f(x_i + h, y_i + hK_3).$$

2) значения искомой функции вычисляются по формуле:

$$y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)\ .$$

Пример 8.3. Решить задачу Коши на отрезке [0; 1,5] для заданного уравнения методом Рунге–Кутта в MS EXCEL.

$$\begin{cases} \acute{y} = y + (1 + x)y^2 \\ y(0) = -1 \end{cases},\ h = 0{,}1$$

Решение.

Решить задачу Коши для заданного уравнения
на заданном отрезке методом Рунге-Кутта

h= 0,1

i	Xi	Yi	k1		k2		k3		k4
0	0,0	-1,0000	0,0000	-1,0000	0,0500	-0,9975	0,0473	-0,9953	0,0944
1	0,1	-0,9952	0,0942	-0,9905	0,1377	-0,9883	0,1349	-0,9817	0,1748
2	0,2	-0,9816	0,1747	-0,9729	0,2102	-0,9711	0,2077	-0,9608	0,2393
3	0,3	-0,9608	0,2393	-0,9488	0,2665	-0,9475	0,2644	-0,9343	0,2879
4	0,4	-0,9343	0,2878	-0,9199	0,3071	-0,9189	0,3055	-0,9037	0,3214
5	0,5	-0,9037	0,3214	-0,8877	0,3336	-0,8870	0,3326	-0,8705	0,3419
6	0,6	-0,8705	0,3419	-0,8534	0,3482	-0,8531	0,3477	-0,8357	0,3516
7	0,7	-0,8357	0,3516	-0,8181	0,3532	-0,8181	0,3531	-0,8004	0,3528
8	0,8	-0,8004	0,3528	-0,7828	0,3508	-0,7829	0,3510	-0,7653	0,3476
9	0,9	-0,7654	0,3476	-0,7480	0,3430	-0,7482	0,3434	-0,7310	0,3378
10	1,0	-0,7311	0,3378	-0,7142	0,3314	-0,7145	0,3320	-0,6979	0,3249
11	1,1	-0,6979	0,3249	-0,6817	0,3173	-0,6820	0,3181	-0,6661	0,3100
12	1,2	-0,6661	0,3101	-0,6506	0,3018	-0,6510	0,3026	-0,6359	0,2941
13	1,3	-0,6359	0,2942	-0,6212	0,2857	-0,6216	0,2865	-0,6073	0,2778
14	1,4	-0,6073	0,2779	-0,5934	0,2693	-0,5938	0,2702	-0,5803	0,2616
15	1,5	-0,5803	0,2616	-0,5673	0,2533	-0,5677	0,2541	-0,5549	0,2457

1. Вводим значение шага h в ячейку **C6**.
2. Вводим значение x_0 в ячейку **C9**.
3. Вводим формулу расчета следующего узла в ячейку **C10 [=C9+$C$6]** и распространяем ее до ячейки **C24**.
4. Вводим значение $y(x_0)$ в ячейку **D9**.
5. Вводим формулу расчета коэффициента K_1 в ячейку **E9** **[=D9+(1+C9)*D9^2)]**.
6. Вводим формулу $y(x_0) + h\frac{K_1}{2}$ в ячейку **F9 [=D9+$C$6*E9/2]**.

7. Вводим формулу расчета коэффициента K_2 в ячейку **G9 [=F9+(1+C9+$C$6/2)*F9^2].**
8. Вводим формулу $y(x_0) + h\frac{K_3}{2}$ в ячейку **H9 [=D9+$C$6*(G9/2].**
9. Вводим формулу расчета коэффициента K_3 в ячейку **I9 [=H9+(1+C9+$C$6/2)*H9^2)].**
10. Вводим формулу $y(x_0) + hK_3$ в ячейку **J9 [=D9+$C$6*I9].**
11. Вводим формулу расчета коэффициента K_4 в ячейку **K9 [=J9+(1+C9+$C$6)*J9^2)].**
12. Вводим формулу расчета значений искомой функции в ячейку **D10 [=D9+$C$6*((E9+G9+I9+K9)/6].**
13. Выделяем ячейки **E9:K9** и распространяем их содержимое до ячеек **E10:K10.**
14. Выделяем ячейки **D10:K10** и распространяем их содержимое до ячеек **D24:K24.**

Ответ: решением задачи Коши для заданного уравнения является функция, заданная таблично. Значения x_i – в ячейках **C9:C24,** значения искомой функции $y(x_i)$ – в ячейках **D9:D24.**

Пример 8.4. Решить задачу Коши на отрезке [0; 1,5] для заданного уравнения методом Рунге–Кутта в Maple.

$$\begin{cases} \acute{y} = y + (1+x)y^2 \\ y(0) = -1 \end{cases}, h = 0,1$$

Решение.

Приведем решение задачи Коши в Maple с использованием встроенной функции *dsolve()* с опцией *numeric*. По умолчанию в этом случае выбирается метод Рунге–Кутта.

$$odu := \frac{d}{dx} y(x) = y(x) + (1+x)\cdot y(x)^2 :$$
$$n := 15 :$$
$$h := 0.1 :$$
$$r := array\left(\left[seq(i\cdot 0.1, i = 0..15)\right]\right) :$$
$$dsoli := dsolve(\{odu, y(0) = -1\}, numeric, output = r)$$

Решение получаем в табличной форме.

x	$y(x)$
,00000	–1,00000
,10000	–,99519
,20000	–,98161
,30000	–,96078
,40000	–,93430
,50000	–,90373
,60000	–,87046
,70000	–,83571
,80000	–,80043
,90000	–,76536
1,00000	–,73106
1,10000	–,69790
1,20000	–,66614
1,30000	–,63592
1,40000	–,60731
1,50000	–,58034

Пример 8.5. Решить задачу Коши на отрезке [0; 1,5] для заданного уравнения приближенными методами.

$$\begin{cases} \acute{y} = y + (1 + x)y^2 \\ y(0) = -1 \end{cases}, h = 0{,}1$$

Реализовать решения на языке программирования Python с помощью самостоятельно разработанных функций. Построить график и оценить погрешность решения.

Решение.

Реализуем различные методы численного решения на языке Python и сохраним в отдельном файле (odulib.py), чтобы использовать в дальнейшем как модуль.

Параметры функций описаны в документ-строках. Отметим, что в качестве параметра используется шаг аргумента для приближенного решения (`h`). Чтобы не пропустить решение в конечной точке (`xend`) из-за вычислительной погрешности, в цикл добавлен уточняющий выбор для шага: `min(h, xend - x0)`. Для работы с массивами используется библиотека `numpy`.

```
import numpy as np

def eiler(f, x0, y0, xend,
h):
'''Решение задачи Коши
{y'=f(x,y),y(x0)=y0}
методом Эйлера с шагом h,
возвращает массивы значений
x, y'''
    x = []
    y = []
    x.append(x0)
    y.append(y0)
    while x0 < xend:
        h = min(h, xend -
x0)
        y0 = y0 + h *
f(x0,y0)
        x0 = x0 + h
        x.append(x0)
        y.append(y0)
    return np.array(x),
np.array(y)
```

```
def eiler_koshi(f, x0, y0,
        xend, h):
'''Решение задачи Коши
{y'=f(x,y),y(x0)=y0}
методом Эйлера-Коши с шагом
h, возвращает массивы значе-
ний x и y'''
    x = []
    y = []
    x.append(x0)
    y.append(y0)
    while x0 < xend:
        h = min(h, xend -
x0)
        x1 = x0 + h
        y1 = y0 + h *
f(x0,y0)
        y0 = y0 +
h*(f(x0,y0) + f(x1,y1))/2.
        x0 = x0 + h
        x.append(x0)
        y.append(y0)
```

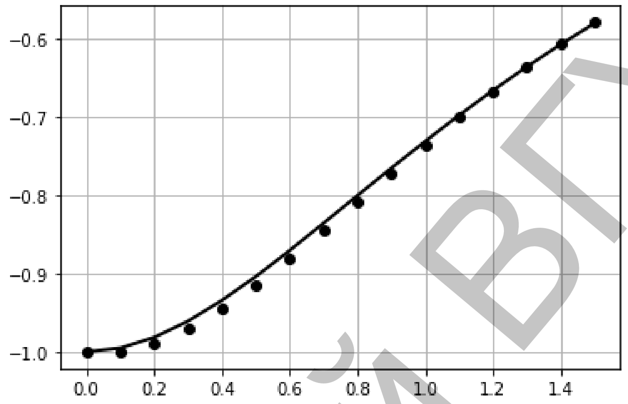
```
    return np.array(x),
np.array(y)
def increment(f, x, y, h):
    k0 = h * f(x,y)
    k1 = h * f(x + h/2., y +
k0/2.)
    k2 = h * f(x + h/2., y +
k1/2.)
    k3 = h * f(x + h, y +
k2)
    return (k0 + 2.*k1 +
2.*k2 + k3)/6.

def runge_kutta(f, x0, y0,
xend, h):
'''Решение задачи Коши
{y'=f(x,y),y(x0)=y0}
методом Рунге-Кутта с шагом
h, возвращает массивы значе-
ний x и y'''
    x = []
    y = []
    x.append(x0)
    y.append(y0)
    while x0 < xend:
        h = min(h, xend -
x0)
        y0 = y0 + incre-
ment(f, x0, y0, h)
        x0 = x0 + h
        x.append(x0)
        y.append(y0)
    return np.array(x),
np.array(y)
```

Обратимся к функциям модуля `odulib` из другой программы.

```
import matplotlib.pyplot as plt
import math
import odulib as odu

def fun(x, y):
    return y + (1 + x)*y**2

def fun_res(x):
    return -1./(x+math.exp(-x))

xnew, ynew = odu.eiler(fun, 0, -1, 1.5, 0.1)
yres = [fun_res(a) for a in xnew]
err = [math.fabs(fun_res(xnew[i])-ynew[i])\
      for i in range(len(xnew))]

for i in range(len(xnew)):
    print('%.1f %.3f %.4f' % (xnew[i],ynew[i],err[i]))

plt.plot(xnew, ynew, 'o',xnew,yres, lw=2, color="black")
plt.grid(True)
plt.show()
```

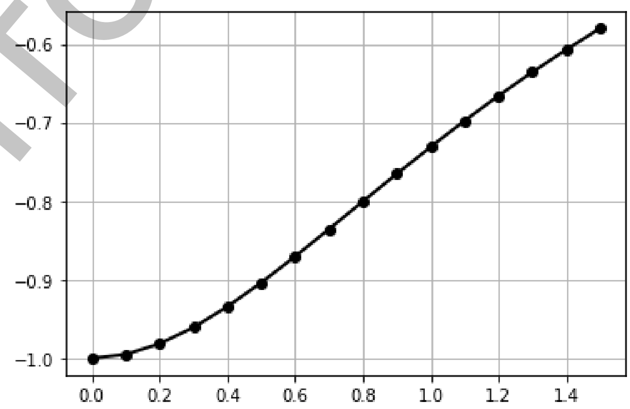
Функция `fun(x, y)` описывает правую часть ОДУ, `fun_res(x)` – точное решение задачи Коши, в списке `err` вычислена погрешность. В результате выполнения программы получим табличную функцию решения методом Эйлера, погрешность и график точной и табличной функций.

```
0.0 -1.000 0.0000
0.1 -1.000 0.0048
0.2 -0.990 0.0084
0.3 -0.971 0.0106
0.4 -0.946 0.0116
0.5 -0.915 0.0115
0.6 -0.881 0.0106
0.7 -0.845 0.0093
0.8 -0.808 0.0077
0.9 -0.771 0.0060
1.0 -0.735 0.0044
1.1 -0.701 0.0029
1.2 -0.668 0.0016
1.3 -0.636 0.0005
1.4 -0.607 0.0004
1.5 -0.579 0.0011
```



Для сравнения приведем результаты решения методом Рунге–Кутта.

```
0.0 -1.0000 0.00000000
0.1 -0.9952 0.00000032
0.2 -0.9816 0.00000059
0.3 -0.9608 0.00000079
0.4 -0.9343 0.00000090
0.5 -0.9037 0.00000091
0.6 -0.8705 0.00000086
0.7 -0.8357 0.00000075
0.8 -0.8004 0.00000060
0.9 -0.7654 0.00000044
1.0 -0.7311 0.00000029
1.1 -0.6979 0.00000014
1.2 -0.6661 0.00000001
1.3 -0.6359 0.00000009
1.4 -0.6073 0.00000018
1.5 -0.5803 0.00000025
```



Аналогичным образом можно получить решение методом Эйлера–Коши и убедиться, что по точности он занимает промежуточное значение относительно рассмотренных выше.

Лабораторная работа № 9

Задание:

1. Решить задачу Коши для следующих обыкновенных дифференциальных уравнений первого порядка методом:
   a) Эйлера;
   b) Эйлера–Коши;
   c) модифицированным Эйлера;
   d) Рунге–Кутта.
2. Оценить погрешность, сравнив с заданным точным решением.

Варианты заданий:

Задача Коши	Точное решение
1. $\begin{cases} \acute{y} = y + (1+x)y^2 \\ y(1) = 0 \end{cases}$	$y = x^2 \ln x$
2. $\begin{cases} \acute{y} = -\frac{xy}{1+x^2} \\ y(0) = 2 \end{cases}$	$y = \dfrac{2}{\sqrt{1+x^2}}$
3. $\begin{cases} \acute{y} = \frac{y+\sqrt{x^2+y^2}}{x} \\ y(1) = 0 \end{cases}$	$y = -\dfrac{1}{2}(1-x^2)$
4. $\begin{cases} \acute{y} = \frac{y^2x^2-(2x+1)y+1}{x} \\ y(1) = 0 \end{cases}$	$y = \dfrac{x-1}{x^2}$
5. $\begin{cases} \acute{y} = -y\cos x + \cos x \sin x \\ y(0) = -1 \end{cases}$	$y = \sin x - 1$
6. $\begin{cases} \acute{y} = \frac{1}{x^2} - \frac{y}{x} - \frac{y^2}{4} \\ y(1) = 2 \end{cases}$	$y = \dfrac{2}{x}$
7. $\begin{cases} \acute{y} = -xy^2 + \frac{xy}{1-x^2} \\ y(0) = -1 \end{cases}$	$y = \dfrac{1}{x^2-1}$
8. $\begin{cases} \acute{y} = \frac{xy}{y+1} - x \\ y(1) = 2 \end{cases}$	$y = \sqrt{10-x^2} - 1$

9. $\begin{cases} \acute{y} = \frac{y}{y+1} - x \\ y(1) = 2 \end{cases}$	$y = (x+1)\left(\frac{1}{3}(x - \ln x + 1) - 1\right)$
10. $\begin{cases} \acute{y} = 3x^2 - \frac{y}{x} + 2 \\ y(0) = 3.4 \end{cases}$	$y = \frac{3}{4}x^3 + x$
11. $\begin{cases} \acute{y} = x^2\sqrt{y} \\ y(0) = 1 \end{cases}$	$y = \frac{1}{36}x^6 + \frac{1}{3}x^3 + 1$
12. $\begin{cases} \acute{y} = -\frac{y}{x} + \ln x \\ y(1) = -1.5 \end{cases}$	$y = \frac{1}{2}x\ln x - \frac{1}{4}x - \frac{5}{4x}$
13. $\begin{cases} \acute{y} = -\frac{2x}{1+x^2} \cdot y - \frac{1}{1+x^2} \\ y(1) = 3 \end{cases}$	$y = \frac{7-x}{1+x^2}$
14. $\begin{cases} \acute{y} = \frac{y+\sqrt{y^2-x^2}}{x} \\ y(1) = 1 \end{cases}$	$y = \frac{1+x^2}{2}$
15. $\begin{cases} \acute{y} = \frac{2y}{e^{2x}+1} \\ y(1) = -1 \end{cases}$	$y = -\frac{(e^2+1)e^{2x-2}}{e^{2x}+1}$
16. $\begin{cases} \acute{y} = \sin x\,(1-y) \\ y(0) = 0 \end{cases}$	$y = 1 - \frac{e^{\cos x}}{e}$
17. $\begin{cases} \acute{y} = \frac{xe^{-x}}{1-y} \\ y(0) = 0 \end{cases}$	$y = 1 - \sqrt{2e^{-x}(x+1) - 1}$
18. $\begin{cases} \acute{y} = \frac{xy}{x^2+2} \\ y(1) = -4 \end{cases}$	$y = -\frac{4\sqrt{3}}{3}\sqrt{x^2+2}$
19. $\begin{cases} \acute{y} = 2\sin x - y + 1 \\ y(0) = 0 \end{cases}$	$y = 1 - \cos x + \sin x$
20. $\begin{cases} \acute{y} = \frac{2x^2+\sin 3x}{y} \\ y(0) = 1 \end{cases}$	$y = \frac{1}{3}\sqrt{15 + 12x^3 - 6\cos 3x}$

Тест

Вопрос 1

Какая из формул соответствует методу половинного деления решения нелинейных уравнений?

- $x_{i+1} = \varphi(x_i)$
- $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$
- $x_i = \dfrac{a_i + b_i}{2}$
- $x_{i+1} = x_i - f(x_i)\dfrac{b - x_i}{f(b) - f(x_i)}$
- $x_{i+1} = x_i - \dfrac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i)$

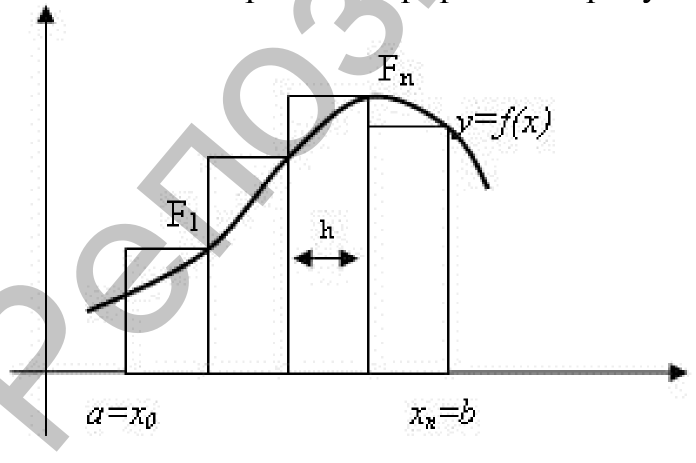
Вопрос 2

Задача численного дифференцирования состоит в...

- вычислении функций по значениям производных
- вычислении производной функции в точках, заданных значениями этой функции
- вычислении производных функций по заданным значениям этой функции
- вычислении дифференциала функции по некоторым точкам
- приближенном вычислении производных функций в некоторой точке по заданным в конечном числе точек значениям этой функции

Вопрос 3

Какой из методов проиллюстрирован на рисунке?



- левых прямоугольников
- правых прямоугольников
- Гаусса
- средних прямоугольников
- Симпсона

Вопрос 4

У приближенного числа –5,18 все значащие цифры верные в широком смысле. Чему равна абсолютная погрешность этого числа?

- 0,01
- 0,05
- 0,005
- 0,5
- 0,02

Вопрос 5

Какая из составляющих общей погрешности решения задачи численными методами связана с округлением вычислений из-за конечности разрядной сетки ЭВМ?

- неустранимая погрешность
- погрешность модели
- погрешность метода
- вычислительная погрешность
- случайная погрешность

Вопрос 6

Укажите последовательность методов решения ОДУ, составленную в порядке возрастания их точности:

- Эйлера–Коши, Эйлера, Рунге–Кутта
- Рунге–Кутта, Эйлера, Эйлера–Коши
- Эйлера, Эйлера-Коши, Рунге–Кутта
- Эйлера–Коши, Рунге–Кутта, Эйлера
- Эйлера, Рунге–Кутта, Эйлера–Коши

Вопрос 7

Какому условию должен удовлетворять параметр q в соотношении $|x^* - x_{i+1}| \le q|x^* - x_i|$, где x^* – точное решение, а x_{i+1} – приближенное, чтобы итерационный процесс сходился?

- $q = 0$
- $q > 0$
- $q < 1$
- $q > 1$
- $0 < q \le 1$

Вопрос 8

Прямой ход метода Гаусса приводит матрицу СЛАУ к...

- верхнему треугольному виду
- нижнему треугольному виду
- диагональному виду
- ленточному виду
- трехдиагональному виду

Вопрос 9
Приведенные формулы –

$$x_1^{(k+1)} = \alpha_{11}x_1^{(k)} + \alpha_{12}x_2^{(k)} + \ldots + \alpha_{1n}x_n^{(k)} + \beta_1$$
$$x_2^{(k+1)} = \alpha_{21}x_1^{(k+1)} + \alpha_{22}x_2^{(k)} + \ldots + \alpha_{2n}x_n^{(k)} + \beta_2$$
$$x_3^{(k+1)} = \alpha_{31}x_1^{(k+1)} + \alpha_{32}x_2^{(k+1)} + \alpha_{32}x_3^{(k)} + \ldots + \alpha_{2n}x_n^{(k)} + \beta_2$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$x_n^{(k+1)} = \alpha_{n1}x_1^{(k+1)} + \alpha_{n2}x_2^{(k+1)} + \ldots + \alpha_{nn-1}x_{n-1}^{(k+1)} + \alpha_{nn}x_n^{(k)} + \beta_n$$

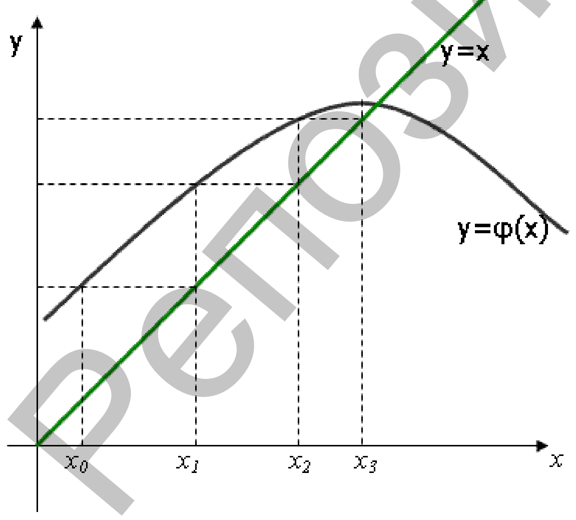
это формулы…

- ○ метода простой итерации
- ○ метода Зейделя
- ○ метода релаксации
- ○ сходимости итерационного метода
- ○ метода Гаусса

Вопрос 10
У приближенного числа –0,00111 все значащие цифры верные в строгом смысле. Чему равна абсолютная погрешность этого числа?

- ○ 5.0E-6
- ○ 1.0E-6
- ○ 5.0E-7
- ○ 1.0E-5
- ○ 5.0E-5

Вопрос 11
Какому методу решения нелинейных уравнений соответствует рисунок?



- ○ хорд
- ○ секущих
- ○ касательных
- ○ половинного деления
- ○ простой итерации

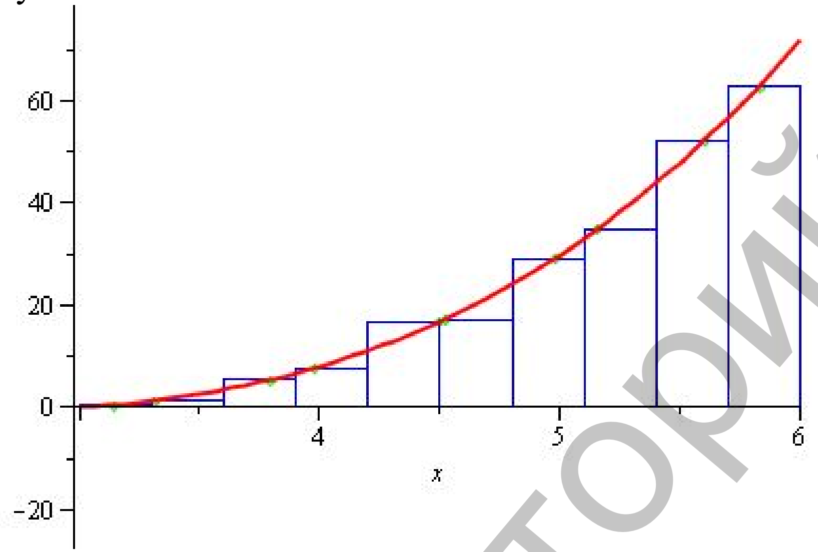
Вопрос 12
Укажите итерационный метод решения СЛАУ:
- метод Крамера
- метод Гаусса
- метод Зейделя
- метод Жордана–Гаусса
- метод прогонки

Вопрос 13
Выберите систему, для которой выполняется условие сходимости итерационного метода:
- $\begin{cases} x_1 = 5x_1 - 0.3x_2 - 1.7x_3 + 4 \\ x_2 = 2x_1 + 6x_2 + 3.7x_3 + 0.7 \\ x_3 = -4x_1 + 0.2x_2 - 11x_3 - 1.8 \end{cases}$
- $\begin{cases} x_1 = x_1 - 3x_2 - 2x_3 + 5 \\ x_2 = 2x_1 + x_2 + 3x_3 + 4 \\ x_3 = -x_1 + 2x_2 - 1.4x_3 - 2 \end{cases}$
- $\begin{cases} x_1 = 3x_1 - 14x_2 - 2x_3 + 1 \\ x_2 = 2.16x_1 + 8x_2 + 14x_3 + 3 \\ x_3 = 4x_1 + 12x_2 - 1.1x_3 - 8 \end{cases}$
- $\begin{cases} x_1 = 15x_1 - 3x_2 - 7x_3 - 4 \\ x_2 = 0.36x_2 + 5.7x_3 + 11 \\ x_3 = 3.9x_1 + 4.7x_3 \end{cases}$
- $\begin{cases} x_1 = -1.5x_1 + 6x_2 - 7.3x_3 - 2 \\ x_2 = 2.3x_1 - 0.36x_2 + 1.96x_3 - 27 \\ x_3 = -4.1x_1 + 2.6x_3 \end{cases}$

Вопрос 14
Приведенная формула:
$$\Delta_z = \left|f'_x(x,y)\right| \cdot \Delta_x + \left|f'_y(x,y)\right| \cdot \Delta_y$$
позволяет вычислить:
- производную функции
- дифференциал функции
- дифференциальную оценку погрешности функции
- погрешность функции по методу границ
- предельные погрешности аргументов

Вопрос 15
Какой метод решения нелинейных уравнений имеет квадратичную скорость сходимости?
- половинного деления
- касательных
- хорд
- секущих
- простой итерации

Вопрос 16
Какой из методов численного интегрирования проиллюстрирован на рисунке?



- метод левых прямоугольников
- метод правых прямоугольников
- метод средних прямоугольников
- статистический вариант метода прямоугольников
- метод трапеций

Вопрос 17
Интерполяция – построение приближающей функции в виде...
- многочлена
- экспоненты
- тригонометрической функции
- показательной функции
- функции, вид которой определяется набором исходных данных

Вопрос 18
Как иначе называют метод Монте-Карло для численного интегрирования?
- метод трапеций
- метод парабол
- статистический вариант метода прямоугольников
- метод средних прямоугольников
- метод правых прямоугольников

Вопрос 19

Приведенные формулы:

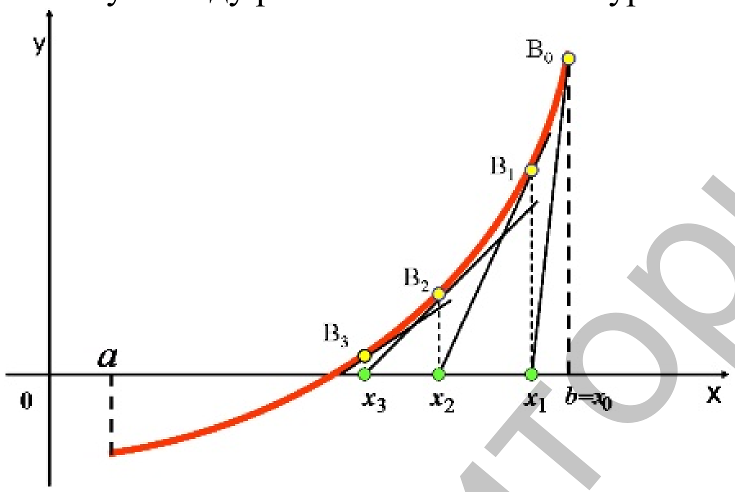
$$x_n = \frac{f_n^{(n-1)}}{a_{nn}^{(n-1)}}, x_k = \frac{f_k^{(k-1)} - \sum_{j=k+1}^{n} a_{kj}^{(k-1)} \cdot x_j}{a_{kk}^{(k-1)}}; k = n-1, n-2, ..., 1.$$

– это формулы...

- метода Гаусса
- прямого хода метода Гаусса
- обратного хода метода Гаусса
- метода Жордана–Гаусса
- метода Зейделя

Вопрос 20

Какому методу решения нелинейных уравнений соответствует рисунок?



- половинного деления
- хорд
- секущих
- касательных
- простой итерации

Вопрос 21

Пусть *f(x)* – исходная функция, *F(x)* – приближающая функция. Выберите выражение, которым определяется погрешность в методе наименьших квадратов?

- $R = \sum_{i=1}^{n} |f(x_i) - F(x_i)|^2$
- $R = |f(x) - F(x)|$
- $R = \sum_{i=1}^{n} \left(f(x_i^2) - F(x_i^2) \right)$
- $R = \sum_{i=1}^{n} \left(f(x_i)^2 - F(x_i)^2 \right)$
- $R = |f(x) - F(x)|^2$

Вопрос 22

Какой форме итерационного многочлена соответствует формула?

$$M_n(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2!h^2}(x - x_0)(x - x_1) +$$

$$\ldots + \frac{\Delta^n y_0}{n!h^n}(x - x_0)\ldots(x - x_{n-1})$$

- Лагранжа
- Ньютона
- Эйткена
- Стирлинга
- сплайнам

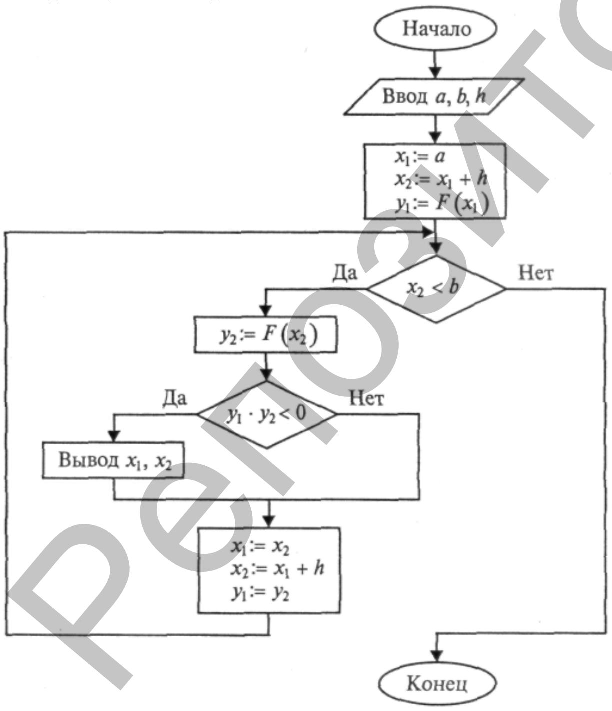
Вопрос 23

Укажите вид функции для квадратичного приближения:

- $y = dx^t$
- $y = px^2 + qx + r$
- $y = \dfrac{x}{mx + d}$
- $y = Ae^{Kx}$
- $y = sx + f$

Вопрос 24

На рисунке приведена блок-схема алгоритма



- метода половинного деления
- метода касательных
- метода хорд
- отделения корней уравнения
- метода простой итерации

Вопрос 25

В методе Симпсона количество интервалов разбиения должно быть…

- не менее 5
- кратным 2
- кратным 3
- кратным 4
- любым

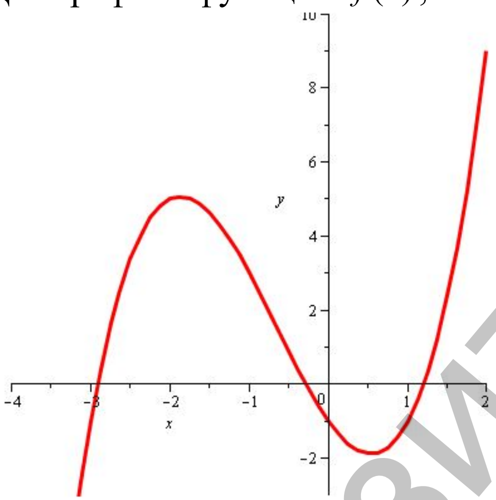
Вопрос 26

В каких случаях применяется численное дифференцирование функций?

- для таблично заданных функций
- для не дифференцируемых функций
- для функций, заданных дифференциалом
- для функций, заданных аналитически
- для параметрически заданных функций

Вопрос 27

Для графика функции $f(x)$, изображенной на рисунке,



результатом отделения корней уравнения $f(x)=0$ является

- *[–4; 2]*
- *[–1; 1]*
- *[–4; –2], [–1; 1], [1; 2]*
- *[1; 2]*
- *–3; –0.5; 1.2*

Вопрос 28

Как называются методы решения дифференциальных уравнений, которые используют для нахождения каждой следующей точки решения информацию только о нескольких предыдущих точках?

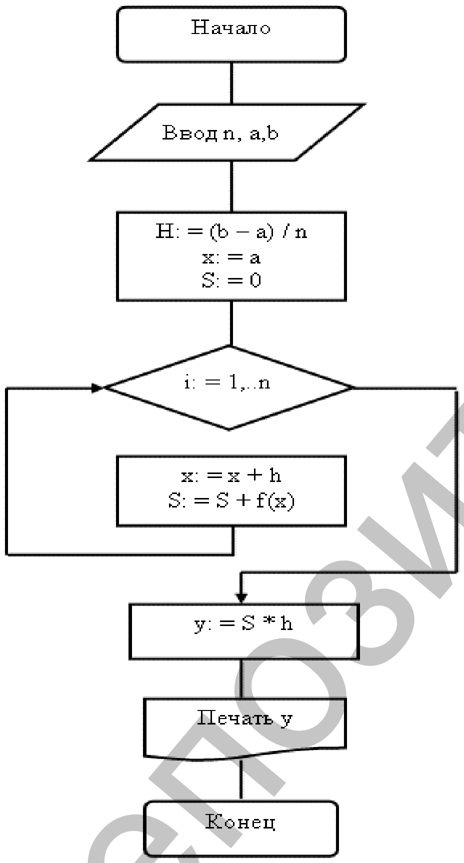
- одноступенчатые
- многоступенчатые
- точечные
- предшествующие
- корректирующие

Вопрос 29

Какая из формул соответствует методу секущих решения нелинейных уравнений?

- $x_{i+1} = \varphi(x_i)$
- $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$
- $x_i = \dfrac{a_i + b_i}{2}$
- $x_{i+1} = x_i - f(x_i)\dfrac{b - x_i}{f(b) - f(x_i)}$
- $x_{i+1} = x_i - \dfrac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i)$

Вопрос 30



На рисунке приведена блок-схема алгоритма

- метода правых прямоугольников
- метода левых прямоугольников
- метода средних прямоугольников
- метода трапеций
- метода Симпсона

Ответы на вопросы теста

1 – 3, **2** – 5, **3** – 2, **4** – 1, **5** – 4, **6** – 3, **7** – 5, **8** – 1, **9** – 2, **10** – 1, **11** – 5, **12** – 3, **13** – 1, **14** – 3, **15** – 2, **16** – 4, **17** – 1, **18** – 3, **19** – 3, **20** – 4, **21** – 1, **22** – 2, **23** – 2, **24** – 4, **25** – 2, **26** – 1, **27** – 3, **28** – 2, **29** – 5, **30** – 1.

ЛИТЕРАТУРА

1. Амосов, А. Вычислительные методы / А. Амосов, Ю. Дубинский, Н. Копченова. – М.: Лань, 2014. – 672 с.
2. Андреев, В.Б. Численные методы: учеб. пособие / В.Б. Андреев. – М.: МАКС Пресс, 2013. – 336 с.
3. Аристова, Е.Н. Практические занятия по вычислительной математике / Е.Н. Аристова, Н.А. Завьялова, А.И. Лобанов. – М.: МФТИ, 2014. – Ч. I. – 243 с.
4. Вабищевич, П.Н. Численные методы: вычислительный практикум / П.Н. Вабищевич. – М.: ЛИБРОКОМ, 2010. – 320 с.
5. Габбасов, Ф.Г. Численные методы. Примеры и задачи: учеб.-метод. пособие / Ф.Г. Габбасов, Л.Б. Ермолаева, И.В. Маланичев. – Казань: КГАСУ, 2017. – 107 с.
6. Демидович, Б.П. Основы вычислительной математики / Б.П. Демидович, И.А. Марон. – СПб.: Лань, 2009. – 672 с.
7. Зенько, С.И. Реализация современных образовательных технологий при обучении программированию будущих учителей информатики: пособие / С.И. Зенько, А.З. Кутыш. – Минск: БГПУ, 2019. – 320 с.
8. Киреев, В.И. Численные методы в примерах и задачах: учеб. пособие / В.И. Киреев. – М.: Высш. шк., 2008. – 480 с.
9. Лапчик, М.П. Численные методы / М.П. Лапчик, М.И. Рагулина, Е.К. Хеннер. – М.: Академия, 2005. – 384 с.
10. Маркова, Л.В. Вычислительные методы алгебры. Практикум: пособие / Л.В. Маркова, Е.А. Корчевская, А.Н. Красоткина. – Витебск: ВГУ, 2013. – 148 с.
11. Петров, И.Б. Лекции по вычислительной математике / И.Б. Петров, А.И. Лобанов. – М.: ИНТУИТ, 2013. – 528 с.
12. Рыжиков, Ю. Вычислительные методы / Ю. Рыжиков. – СПб.: БХВ, 2012. – 400 с.
13. Самарский, А.А. Задачи и упражнения по численным методам / А.А. Самарский, П.Н. Вабищевич, Е.А. Самарская. – М.: ЛИБРОКОМ, 2017. – 208 с.
14. Тарасевич, Ю.Ю. Использование пакетов Maple, Mathcad и LATEX2ε при решении математических задач и подготовке математических и естественно-научных текстов: учеб. пособие / Ю.Ю. Тарасевич. – Изд. 3-е. – М.: ЛИБРОКОМ, 2012. – 136 с.
15. Сборник задач по методам вычислений: учеб. пособие / под ред. П.И. Монастырного. – Минск: БГУ, 2007. – 376 с.

Учебное издание

АЛЕЙНИКОВА Татьяна Григорьевна
ШЕРБАФ Алмас Ибрагимовна

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

Практикум

Технический редактор	*Г.В. Разбоева*
Корректор	*Л.В. Моложавая*
Компьютерный дизайн	*Л.Р. Жигунова*

Подписано в печать 14.07.2020. Формат 60x84 $^{1}/_{16}$. Бумага офсетная.
Усл. печ. л. 5,75. Уч.-изд. л. 6,37. Тираж 100 экз. Заказ 83.

Издатель и полиграфическое исполнение – учреждение образования «Витебский государственный университет имени П.М. Машерова».

Свидетельство о государственной регистрации в качестве издателя, изготовителя, распространителя печатных изданий
№ 1/255 от 31.03.2014.

Отпечатано на ризографе учреждения образования «Витебский государственный университет имени П.М. Машерова».
210038, г. Витебск, Московский проспект, 33.