Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский Государственный университет
(национально исследовательский университет)»
Филиал ФГАОУ ВО «ЮУрГУ (НИУ)» в г. Златоусте
Факультет «Техники и технологии»
Кафедра «Математика и вычислительная техника»
Факультет Техники и технологии

Решения задач

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине вычислительная математика

ЮУрГУ – 231000.2020.230.00 ПЗ КР

Руководитель
Коннов С.В.
___________________2018г.

Автор проекта
студент группы ФТТ-307
Б.А. Мурашов
__________________2018г.
Проект защищен с оценкой
___________________2018г.

Златоуст 2018г

СОДЕРЖАНИЕ

1. ТЕОРИЯ ПОГРЕШНОСТИ ВЫЧИСЛЕНИЙ

Задание 1.1.1–1

Определить, какое равенство точнее. $\sqrt{44} = 6.63$; $\frac{9}{41} = 0{,}219$.

Решение:

$$\sqrt{44} = 6.63;\ \frac{9}{41} = 0.219$$

$$\sqrt{44} = 6.6332;\ \frac{9}{41} = 0.2195$$

$$\Delta_{a_1} = |6.63 - 6.6332| = 0.0032 \qquad \Delta_{a_2} = |0.219 - 0.2195| = 0.0005$$

$$\delta_{a_1} = \frac{0{,}0032}{6.63} * 100\% \approx 0.048\% \qquad \delta_{a_2} = \frac{0.0005}{0.2195} * 100\% \approx 0.22\%$$

Ответ: $\sqrt{44} = 6.63$ точнее.

Задание 1.1.2–1

Округлить сомнительные цифры числа, оставив верные знаки: а) в узком смысле; б) в широком смысле.

а) 22.553 ± 0.016

б) 2.8546; δ = 0,3%.

Решение:

а) $0{,}016 < 0{,}05 => 0{,}5 * 10^{-1}$

$$-1 = 1 - 1 - n + 1 => n = 3 => 22.5\ \pm 0.016$$

Ответ: $22.5\ \pm 0.016$

б) $2.8546, \delta = 0.3\%$

$$\delta = \frac{\Delta a}{A} =>\ \Delta_a = 0.003 * 2.8546 = 0.008638$$

$$0.008639 < 0{,}01 = 1 * 10^{-2} =>\ -2 = 0 - n + 1 => n = 3$$

Ответ: 2.85 δ = 0.3%.

Задание 1.1.3–1

Найти предельные абсолютные и относительные погрешности чисел, если они имеют только верные цифры: а) в узком смысле; б) в широком смысле.

а) 0.2387

б) 42.884

Решение:

а) При m = –1, n = 4 получим $\Delta_a = 0{,}5^{m-n+1}$= 0,00005

$$\delta_a \frac{0{,}5}{2} * 10^{1-4} = 0.00025 = 0.025\%$$

б) При m = 1, n = 5 получим $\Delta a = 1 * 10^{1-5+1} = 10^{-3} = 0.001$

$$\delta_a = \frac{1}{4} * 10^{1-5} = 0.000025 = 0.0025\%$$

Ответ: а) $\Delta_a = 0.00005, \delta_a = 0.025\%;$ б) $\Delta_a = 0.001\ \delta_a = 0.0025\%$

Задание 1.2.1–1

Вычислить значение выражения, принимая значения аргументов с четырьмя верными знаками в узком смысле. Оценить погрешность результата двумя способами.

$$y = \frac{\ln(tg(20^\circ))}{\sqrt{\pi} * \lg(\sqrt{5})} + \sqrt[3]{e}$$

Решение:

<u>Первый способ:</u>

$$X1 = tg(20) = 0.363 * \Delta X1 => 0.5 * 10^{-3}$$

$$X2 = \ln\big(tg(20)\big) = -1.010 * \Delta X2 => 0.5 * 10^{-4}$$

$$X3 = \pi = 3.141 * \Delta X3 => 0.5 * 10^{-4}$$

$$X4 = \sqrt{\pi} = 1.141 * \Delta X4 => 0.5 * 10^{-4}$$

$$X5 = \sqrt{5} = 2.236 * \Delta X5 => 0.5 * 10^{-4}$$

$$X6 = \lg(\sqrt{5}) = 2.349 * \Delta X6 => 0.5 * 10^{-3}$$

$$X7 = e = 2.718 * \Delta X7 => 0.5 * 10^{-4}$$

$$X8 = \sqrt[3]{e} = 1.395 * \Delta X8 => 0.5 * 10^{-4}$$

$$\delta_{a+b} = \frac{|a|}{|a+b|} + \frac{|b|}{|a+b|}$$

$$\delta_a = \delta_c + \delta_d => \ \delta_c = \frac{0.0005}{cos^2(20^\circ)} = 0.566 * 10^{-4}$$

$$\delta_d = \delta_g + \delta_k => \delta_g = \frac{0.0005}{2 * \sqrt{5}} = 0.111 * 10^{-4}$$

$$\delta_k = \frac{0.0005}{\sqrt{5} * \ln(10)} = 0.0971 * 10^{-4}$$

$$\delta_d = 0.2071 * 10^{-4};\ \delta_a = 0.134 * 10^{-3}$$

$$\delta_b = \frac{0.0005}{3 * \sqrt[3]{e}} = 0.0119 * 10^{-3}$$

$$a = -1.631;\ b = 1.395$$

$$\delta_{a+b} = \left(\frac{|-1.631| * 0.7731}{|-1.631 + 1.395|} + \frac{|1.395| * 0.0119}{|-1.631 + 1.395|}\right) * 10^{-4} = 6.007 * 10^{-4}$$

$$\Delta = \frac{6.007 * 10^{-4}}{0.225} = 0.00266$$

<u>Второй способ:</u>

$$\Delta y = \left|\frac{\ln(tg(X_1))}{\sqrt{X_2} * \lg(X_3)} + \sqrt[3]{X_4}\right|'_{X_1} * \Delta X_1 + \left|\frac{\ln(tg(X_1))}{\sqrt{X_2} * \lg(X_3)} + \sqrt[3]{X_4}\right|'_{X_2} * \Delta X_2$$
$$+ \left|\frac{\ln(tg(X_1))}{\sqrt{X_2} * \lg(X_3)} + \sqrt[3]{X_4}\right|'_{X_3} * \Delta X_3 + \left|\frac{\ln(tg(X_1))}{\sqrt{X_2} * \lg(X_3)} + \sqrt[3]{X_4}\right|'_{X_4} * \Delta X_4$$

$$\frac{cos^2X_1}{\sqrt{X_2}*\lg(X_3)}*\Delta X_1-\frac{\ln(tgX_1)*\Delta X_2}{\sqrt{X_2^3}*\sqrt{X_2}}+$$

$$+\frac{X_3*\ln(10)*\ln\left(tg(X_1)\right)*\Delta X_3}{\sqrt{X_2}}+\frac{\Delta X_4}{3*\sqrt[3]{X_4^2}}=1.42*10^{-2}+0.081*10^{-5}-$$

$$-6.564*10^{-3}+0.171*10^{-4}=0.003653$$

Ответ:$\Delta_1=0.00266;\ \Delta_2=0.003653$

Задание 1.2.2–1

С каким числом верных знаков следует взять значения аргументов функции, чтобы значение этой функции имело четыре верных знака?

Решение:

$$y=\frac{\ln(tg(20^{\circ}))}{\sqrt{\pi}*\lg(\sqrt{5})}+\sqrt[3]{e}$$

$$x_1=tg(20^{\circ})\approx 0.36$$

$$x_2=\pi\approx 3.1$$

$$x_3=\sqrt{5}\approx 2.2$$

$$x_4=e\approx 2.7$$

$$y=0.255;\ \Delta y\le 0.5*10^{-1-4+1}=0.00005$$

$$\frac{\partial y}{\partial x_1}=\frac{1}{x_1*\lg(x_3)*\sqrt{x_2}}=4.61;\ \frac{\partial y}{\partial x_2}=\frac{\ln(x_1)}{\lg(x_3)*2*\sqrt[2]{x_3^2}}=-0.273$$

$$\frac{\partial y}{\partial x_3}=\frac{\ln(x_1)*x_3*\ln(10)}{\sqrt{x_2}}=-2.93;\ \frac{\partial y}{\partial x_4}=\frac{1}{3*\sqrt[3]{x_4^2}}=0.171$$

$$\Delta x_i\le\frac{\Delta y}{4*\left|\frac{\partial y}{\partial x_i}\right|}$$

$$\Delta x_1\le\frac{0.00005}{4*4.61}=>0.000002=>0.000005$$

$$\Delta x_2 \leq \frac{0.00005}{4 * 0.273} => 0.00004 => 0.00005$$

$$\Delta x_3 \leq \frac{0.00005}{4 * 2.93} => 0.000002 => 0.000005$$

$$\Delta x_4 \leq \frac{0.00005}{4 * 0.171} => 0.00007 => 0.00005$$

Ответ:

$$\Delta x_1 = 0.36 \pm 0.000005;\ \Delta x_2 = 3.1 \pm 0.00005$$

$$\Delta x_3 = 2.2 \pm 0.000005;\ \Delta x_4 = 2.7 \pm 0.00005$$

2. МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Задание 2.1.1–1

Отделить корни аналитически $5^x + 3x = 0$:

Решение:

$$y = 5^x + 3x$$

$$y' = 5^x \ln(5) + 3 => \ 5^x \ln(5) + 3 = 0; \quad 5^x > 0, y' \neq 0.$$

Найти корень уравнениея возможно табличным методом (табл. 1).

Таблица 1

x	y(x)
-1,00	-2,80
-0,90	-2,47
-0,80	-2,12
-0,70	-1,78
-0,60	-1,42
-0,50	-1,05
-0,40	-0,67
-0,30	-0,28
-0,20	0,12
-0,10	0,55
0,00	1,00
0,10	1,47
0,20	1,98

Ответ: Корень уравнения находится в интервале$[-0.3; -0.2]$.

Задание 2.1.2–1

Отделить корни многочлена табличным способом, используя средства пакета MathCAD и уточнить один из них методом бисекции с точностью до 0.01. Сделать проверку.

Решение:

Отделим корни уравнения табличным методом. Для этого создадим программу на языке Python, код находится в приложении **А**. Решение табличным методом:

Решение:

$y(x) = x^4 - x - 1$, интервал$[-3; 3]$ (табл. 2)

Таблица 2

x	y(x)
-3,00	83,00
-2,00	17,00
-1,00	1,00
0,00	-1,00
1,00	-1,00
2,00	13,00
3,00	77,00

Из вычислений видно два отрезка с корнями уравнения $[-1; 0]$ и $[1; 2]$.

Для уточнения корня методом бисекции напишем алгоритм решения, код находится в приложении **В.**

$y(x) = x^4 - x - 1$, интервал$[-1; 0]$, с точностью 0.01.

Ответ: Корень уравнения $x = -0.73,$ проверка $=> y(x) = 0.01$.

Задание 2.1.3–1

Отделить уравнения графически $x^2 - 2 + 0.5^x = 0$.

Решение:

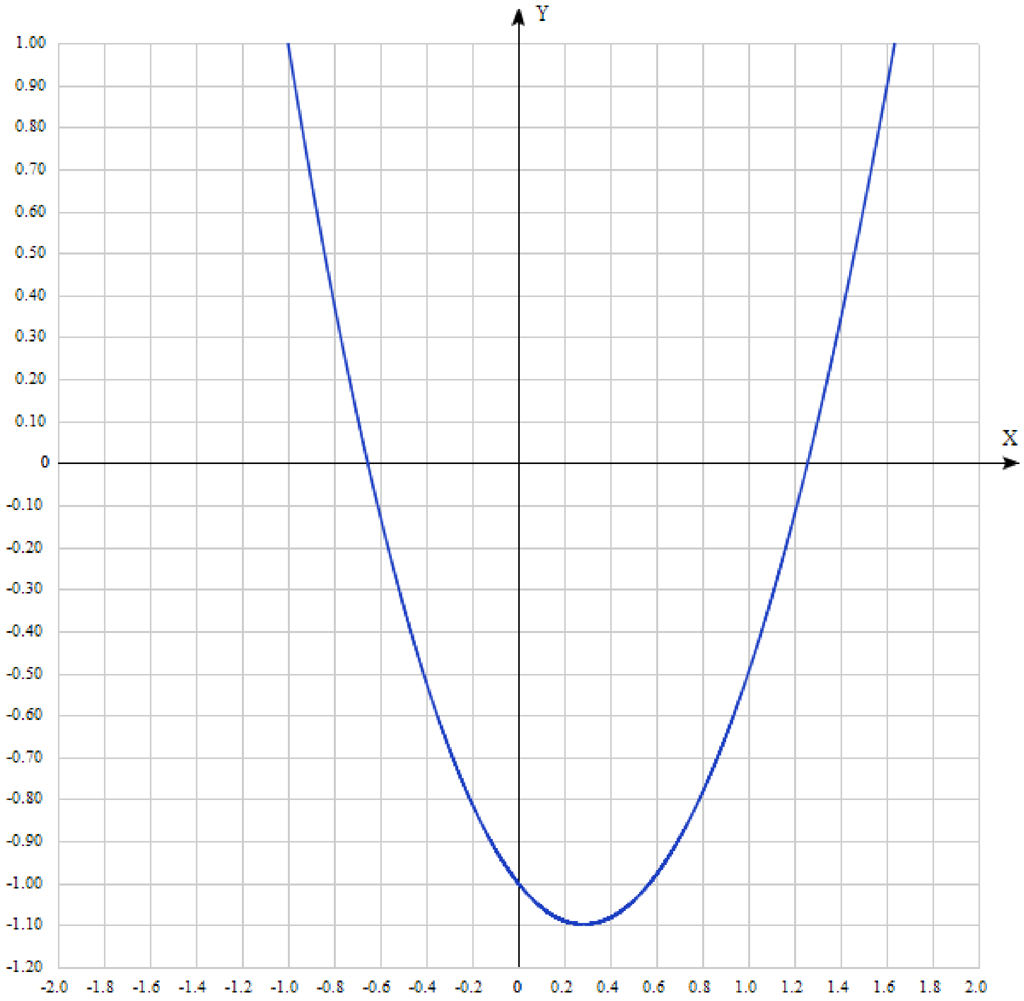


Рис. 2.1.3.1. График функции

По графику (рис. 2.1.3.1.), корни уравнения находятся в отрезках $[-0.8; -0.6]$ и $[1.2; 1.4]$.

Ответ: $[-0.8; -0.6] \cup [1.2; 1.4]$.

Задание 2.1.4–1

Отделить корни уравнения графически и уточнить один из них методом бисекции с точностью 0.05. Сделать проверку.

$$y(x) = (x-1)^2 * \lg(x+11) - 1.$$

Решение:

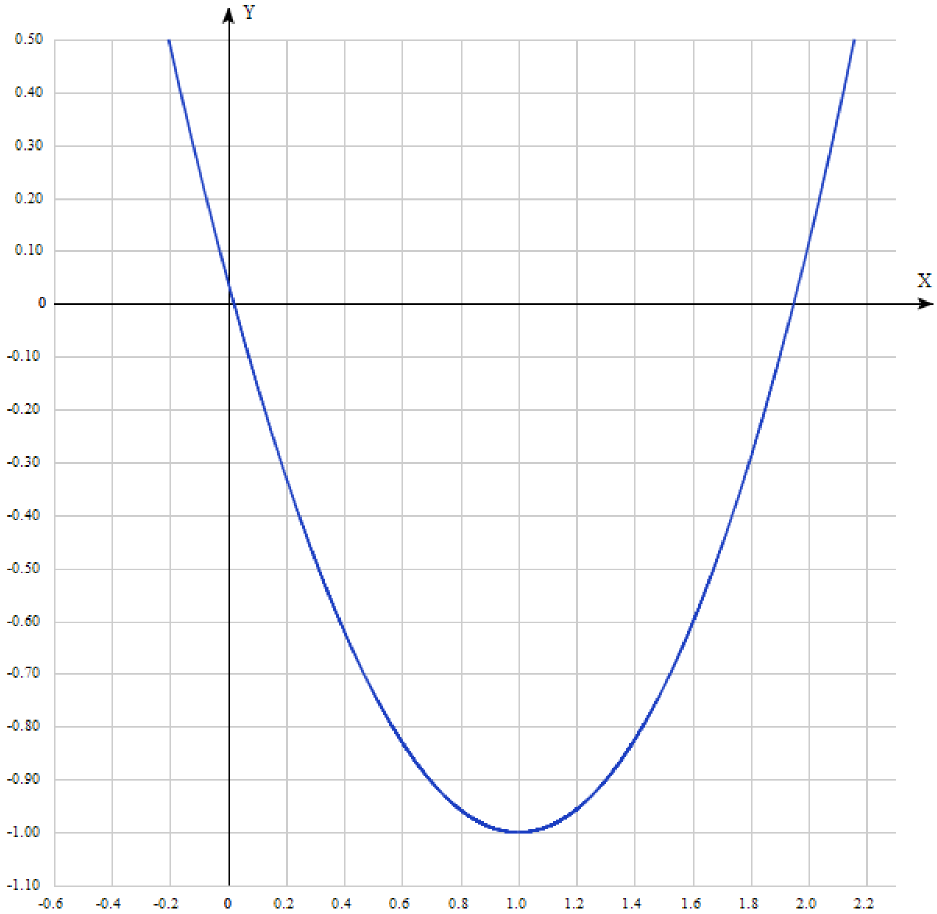


Рис. 2.1.4.1. График функции

По графику (рис. 2.1.4.1.), корни уравнения находятся в отрезках$[-0.2; 0.2]$ и $[1.8; 2]$.

Для уточнения корня методом бисекции напишем алгоритм решения, код находится в приложении **С.**

Для решении возьмем отрезок $[-0.2; 0.2]$ с точностью 0.05

Ответ: Корень уравнения $x = 0.025,$ проверка $=> y(x) = -0.0091.$

Задание 2.2.1–1

Отделить корни уравнения графически и уточнить один из них методом касательных с точностью до 0.001. Точность приближения установить двумя способами, сравнив результат, сделать вывод.

Уравнение $y(x) = x - \sin(x) - 0.25$.

Решение:

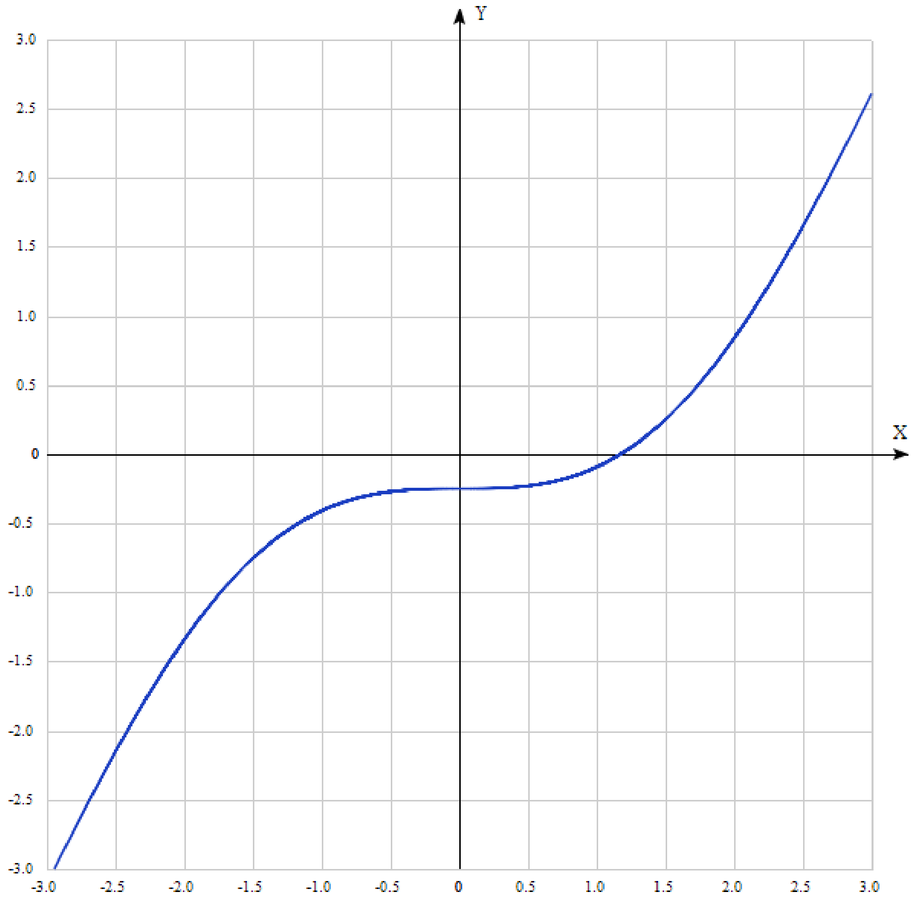


Рис. 2.2.1.1. График функции

По графику функции (рис 2.2.1.1.) видно, что корень находится в отрезки $[1.0; 1.5]$. Реализуем алгоритм касательных, код алгоритма находится в приложении **D**. Согласно методу Ньютона, если нам известно, что функция y(x) на отрезке [a, b] непрерывна и дважды дифференцируема, и имеет ровно один корень, тогда можно взять за нулевое приближение значение одного из концов отрезка [a, b] в зависимости от знака второй производной, иначе при первом же приближении можно попасть за пределы отрезка [a, b].

$$y'(x) = 1 - \cos(x) => \ y''(x) = \sin(x): \ y''(1) > 0 \text{ и } \ y''(1.5) > 0$$

Вторая производная положительна на концах отрезка, значит можно выбрать в качестве начального приближения $x_0 = 1$. Результаты расчета программы:

$x = 1.171,$ проверка $y(x) = 0$

Ответ: $x = 1.171$

Задание 2.2.2–1

Отделить корни уравнения аналитически и уточнить один из них методом хорд: а) с точность до 0.005; б) оценить точность k-го приближения.

Решение:
Заданное уравнение: $\mathrm{y(x)} = \mathrm{x}^3 - 3\mathrm{x}^2 + 9\mathrm{x} - 8$

Согласно теоремам о корнях нелинейных уравнений:

- Уравнение 3 степени имеет 3 корня действительных и комплексных;
- Уравнение нечетной степени имеет хотя бы один действительный корень;
- Коэффициенты действительные, значит комплексные корни уравнения комплексно сопряженные.

Найдем корни уравнения аналитически:

$$\mathrm{y(x)} = \mathrm{x}^3 - 3\mathrm{x}^2 + 9\mathrm{x} - 8$$

$$\mathrm{y}' = 3\mathrm{x}^2 - 6\mathrm{x} + 9$$

$$3\mathrm{x}^2 - 6\mathrm{x} + 9 = 0 => \mathrm{x}^2 - 2\mathrm{x} + 3 = 0$$

$$\mathrm{D} = 4 - 12 = -8$$

$$\mathrm{x}_{1,2} = \frac{2 \pm \sqrt{-8}}{2} => 1 \pm 1.41i$$

Найдем действительный корень уравнения через грубую оценку модулей корней

$$\frac{1}{1+\frac{B}{|a_0|}} \leq |x_k| \leq 1 + \frac{A}{|a_n|}$$

$$A = \max\{|-8|, |-3|\} => 8$$

$$B = \max\{|9|, |-3|\} => 9$$

$$\frac{1}{1+\frac{9}{|3|}} \leq |x_k| \leq 1 + \frac{8}{|8|}$$

$$0.25 \leq |x_k| \leq 2$$

Корень находится в отрезке $\{0.25; 2\}$. Реализуем метод хорд, код алгоритма находится в приложении **E**.

Ответ:

а) Корень $x = 1.166, => y(x) = 0.004$

б) Точность k = 1 приближения 0.1.

Задание 2.3.1–1

Отделить корни уравнения аналитически и уточнить один из них с точностью до 0.0005 комбинированным методом хорд и касательных.

Уравнение $y(x) = 2x^3 - 3x^2 - 12x - 5$.

Отделим корни уравнения аналитически:

$$y' = 6x^2 - 6x - 12 => x^2 - x - 2 = 0$$

$$D = 1 + 8 = 9$$

$$x_{1,2} = \frac{1 \pm 3}{2} = 2; -1$$

$$(-\infty; -1) \cup (-1; 2) \cup (2; +\infty)$$

Уточним корень из интервала $(-1; 2)$, комбинированным методом хорд и касательных, код программы в приложении **F**.

Ответ: Корень $x = -0.4999, => y(x) = 0.0005$

Задание 2.4.1–1

Отделить корни уравнения графически и уточнить один из них методом простой итерации с точностью до 0,001.

Уравнение $y(x) = \ln(x) + (x + 1)^3$

Решение:

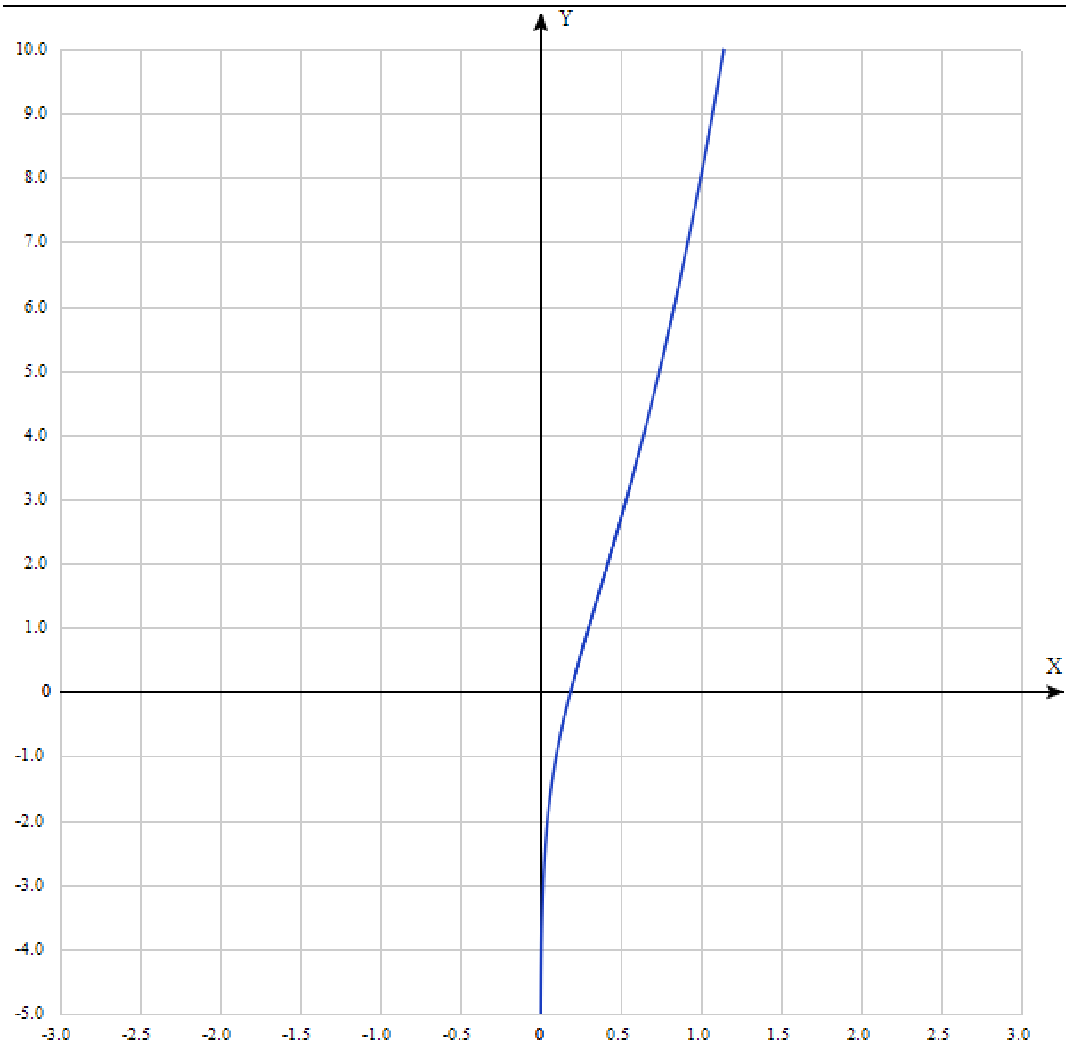


Рис. 2.4.1.1. График функции

По графику функции (рис 2.4.1.1.) видно, что корень находится в отрезке $[0.1; 0.5]$.

Приведем уравнение $y(x)$ к виду $x = \varphi(x)$.

$$\varphi(x) = x - \lambda(x) f(x), \text{где}\, \lambda(x) = \frac{1}{f^{/}(x)}$$

$$f^{'}(x) = \frac{1}{x} + 3 * (x+1)^2$$

$$\varphi(x) = x - \frac{\ln(x) + (x+1)^3}{\frac{1}{x} + 3 * (x+1)^2}$$

Уточним корень из отрезка $[0.1; 0.5]$, итерационным методом, код программы в приложении **G**.

Ответ: Корень $\mathrm{x} = 0.1874, => \mathrm{y}(\mathrm{x}) = -0.0003$

Задание 2.4.2–1

Отделить корни уравнения табличным способом и уточнить один из них с точностью до 0.005 методом простой итерации, приведя уравнение к итерационному виду, используя метод приведения.

Уравнение $y(x) = x^3 + 2 * x^2 + 2$

Решение:

Найти корень уравнениея возможно табличным методом (табл. 3).

Таблица 3

x	f(x)
-4	-30
-3,5	-16,375
-3	-7
-2,5	-1,125
-2	2
-1,5	3,125
-1	3
-0,5	2,375
0	2
0,5	2,625
1	5
1,5	9,875

Из таблицы (табл. 3) видно что корень уравнения находится в отрезке $[-2.5; -2]$.

Приведем уравнение $y(x)$ к виду $x = \varphi(x)$.

$$\varphi(x) = x - \lambda(x)f(x), \text{где}\, \lambda(x) = \frac{1}{f^{/}(x)}$$

$$f^{'}(x) = \ 3 * x^2 + 4 * x$$

$$\varphi(x) = x - \frac{x^3 + 2 * x^2 + 2}{3 * x^2 + 4 * x}$$

Уточним корень из отрезка $[-2.5; -2]$, итерационным методом, код программы в приложении **Н**.

Ответ: Корень $\mathrm{x} = -2.359, => \mathrm{y}(\mathrm{x}) = 0.00$

3. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ УРАВНЕНИЙ

Задание 3.1.1–1

Решить систему линейных алгебраических уравнений (СЛАУ) методом итераций, с точностью до $\varepsilon = 0.01$, для этого:

a) привести систему к итерационному виду используя обобщённый метод;
b) проверить условие сходимости и определить параметр q;
c) вычислить процесс организовать, используя средства MathCAD;
d) сделать проверку с помощью функции islove.

Решение:
a)

$$\begin{cases} 4.4x_1 - 2.5x_2 + 19.2x_3 - 10.8x_4 = 4.3 \\ 5.5x_1 - 9.3x_2 - 14.2x_3 + 13.2x_4 = 6.8 \\ 7.1x_1 - 11.5x_2 + 5.3x_3 - 6.7x_4 = -1.8 \\ 14.2x_1 + 3.4x_2 - 1.8x_3 + 5.3x_4 = 7.2 \end{cases}$$

$$A = \begin{bmatrix} 4.4 & -2.5 & 19.2 & -10.8 \\ 5.5 & -9.3 & -14.2 & 13.2 \\ 7.1 & -11.5 & 5.3 & -6.7 \\ 14.2 & 3.4 & -1.8 & 5.3 \end{bmatrix}$$

$$B = \begin{bmatrix} 4.3 \\ 6.8 \\ -1.8 \\ 7.2 \end{bmatrix}$$

$$A = \frac{A^T * A}{500} = \begin{bmatrix} 0.603 & -0.191 & 0.037 & 0.106 \\ -0.191 & 0.473 & 0.034 & -0.001 \\ 0.037 & 0.034 & 1.203 & -0.88 \\ 0.106 & -0.001 & 0.728 & -0.88 \end{bmatrix}$$

Ответ: Итерационный вид матрицы $A = \begin{bmatrix} 0.603 & -0.191 & 0.037 & 0.106 \\ -0.191 & 0.473 & 0.034 & -0.001 \\ 0.037 & 0.034 & 1.203 & -0.88 \\ 0.106 & -0.001 & 0.728 & -0.88 \end{bmatrix}$

$$B = \frac{B^T * B}{500} = \begin{bmatrix} 0.292 \\ -0.058 \\ -0.078 \\ 0.187 \end{bmatrix}$$

$$0.603 > 0.191 + 0.037 + 0.106 => 0.603 > 0.334$$

$$0.473 > 0.191 + 0.034 + 0.001 => 0.473 > 0.226$$

$$1.203 > 0.037 + 0.034 + 0.88 => 1.203 > 0.951$$

$$0.88 > 0.728 + 0.001 + 0.106 => 0.88 > 0.836$$

$$\begin{cases} 0.603x_1 - 0.191x_2 + 0.037x_3 + 0.106x_4 = 0.292 \\ -0.191x_1 + 0.473x_2 + 0.034x_3 - 0.031x_4 = -0.058 \\ 0.037x_1 + 0.034x_2 + 1.203x_3 - 0.88x_4 = -0.073 \\ 0.106x_1 - 0.001x_2 + 0.728x_3 - 0.88x_4 = 0.187 \end{cases}$$

$$\begin{cases} x_1 = 0.317x_2 + 0.061x_3 - 0.175x_4 + 0.484 \\ x_2 = 0.404x_1 - 0.072x_3 + 0.003x_4 - 0.122 \\ x_3 = -0.031x_1 - 0.028x_2 + 0.731x_4 - 0.061 \\ x_4 = 0.120x_1 - 0.001x_2 + 0.827x_3 + 0.212 \end{cases}$$

1 условие

$$\sum_{j=1}^{4} |a_{1j}| = 0.317 + 0.061 + 0.175 = 0.553 < 1$$

$$\sum_{j=1}^{4} |a_{2j}| = 0.404 + 0.072 + 0.003 = 0.479 < 1$$

$$\sum_{j=1}^{4} |a_{3j}| = 0.031 + 0.028 + 0.731 = 0.790 < 1$$

$$\sum_{j=1}^{4} |a_{4j}| = 0.120 + 0.001 + 0.827 = 0.948 < 1$$

$$q = ||a||_\infty = max\{0.553; 0.479; 0.790; 0.948\} = 0.948$$

2 условие

$$\sum_{i=1}^{4} |a_{i1}| = 0.404 + 0.031 + 0.120 = 0.555 < 1$$

$$\sum_{i=1}^{4}|a_{i1}| = 0.317 + 0.028 + 0.001 = 0.346 < 1$$

$$\sum_{i=1}^{4}|a_{i1}| = 0.061 + 0.072 + 0.827 = 0.960 < 1$$

$$\sum_{i=1}^{4}|a_{i1}| = 0.175 + 0.003 + 0.731 = 0.909 < 1$$

$$q = \left||a|\right|_1 = max\{0.555; 0.346; 0.960; 0.909\} = 0.960$$

3 условие

$$\left||a|\right|_E =$$

$$= \sqrt{0.317^2 + 0.061^2 + 0.175^2 + 0.404^2 + 0.072^2 + 0.003^2 + 0.031^2 + 0.028^2}$$

$$\sqrt{0.731^2 + 0.120^2 + 0.001^2 + 0.827^2} = \sqrt{0.815895} \approx 0.903 < 1; q = 0.9$$

Ответ: Условия сходимости выполняются, q = 0.9

c) Решим итерационным методом, код программы находится в приложении **I**.

Ответ: $X = \begin{bmatrix} 0.136 \\ -0.130 \\ 0.962 \\ 1.413 \end{bmatrix}$

d) Проверка (рис. 3.1.1.1):

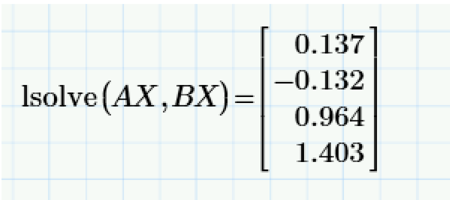


Рис. 3.1.1.1. Проверка решения средствами MathCAD

Ответ: Проверка потвердила решение

Задание 3.1.2–1

Решить СЛАУ с точностью до 0.001 методом Зейделя, для этого:

a) привести СЛАУ виду, удовлетворяющему условию сходимости;
b) преобразовать СЛАУ к итерационному виду, проверить условия сходимости;
c) вычислить процесс организовать, используя средства MathCAD;
d) сделать проверку с помощью функции islove.

Решение:

a)

$$\begin{cases} 1.7x_1 + 2.8x_2 + 1.9x_3 = 0.7 \\ 2.1x_1 + 3.4x_2 + 1.8x_3 = 1.1 \\ 4.2x_1 - 1.7x_2 + 1.3x_3 = 2.8 \end{cases}$$

$$A = \left[\begin{matrix} 1.7 & 2.8 & 1.9 \\ 2.1 & 3.4 & 1.8 \\ 4.2 & -1.7 & 1.3 \end{matrix}\middle|\begin{matrix} 0.7 \\ 1.1 \\ 2.8 \end{matrix}\right]$$

$$A = \left[\begin{matrix} 4.2 & -1.7 & 1.3 \\ 0.4 & 0.6 & -0.1 \\ 1.7 & 2.8 & 1.9 \end{matrix}\middle|\begin{matrix} 2.8 \\ 0.4 \\ 0.7 \end{matrix}\right] => -1 * [1] + [2]$$

$$A = \left[\begin{matrix} 4.2 & -1.7 & 1.3 \\ 0.4 & 0.6 & -0.1 \\ 0.3 & 0.2 & 2.4 \end{matrix}\middle|\begin{matrix} 2.8 \\ 0.4 \\ -1.3 \end{matrix}\right] => -5 * [2] + [3]$$

$$4.2 > 1.7 + 1.3 => 3 > 4.2$$

$$0.6 > 0.1 + 0.4 => 0.6 > 0.5$$

$$2.4 > 0.3 + 0.2 => 2.4 > 0.5$$

$$\begin{cases} 4.2x_1 - 1.7x_2 + 1.3x_3 = 2.8 \\ 0.4x_1 + 0.6x_2 - 0.1x_3 = 0.4 \\ 0.3x_1 + 0.2x_2 + 2.4x_3 = -1.3 \end{cases}$$

$$\begin{cases} x_1 = 0.667 + 0.405x_2 - 0.310x_3 \\ x_2 = 0.667 - 0.667x_1 - 0.167x_3 \\ x_3 = -0.542 - 0.125x_1 - 0.089x_2 \end{cases}$$

1 условие

$$\sum_{j=1}^{4}\left|a_{1j}\right| = 0.405 + 0.310 = 0.715 < 1$$

$$\sum_{j=1}^{4}\left|a_{2j}\right| = 0.667 + 0.167 = 0.834 < 1$$

$$\sum_{j=1}^{4}\left|a_{3j}\right| = 0.125 + 0.089 = 0.214 < 1$$

$$q = \left||a|\right|_{\infty} = max\{0.715; 0.834; 0.214\} = 0.834$$

2 условие

$$\sum_{i=1}^{4}|a_{i1}| = 0.667 + 0.125 = 0.792 < 1$$

$$\sum_{i=1}^{4}|a_{i1}| = 0.405 + 0.089 = 0.494 < 1$$

$$\sum_{i=1}^{4}|a_{i1}| = 0.310 + 0.167 = 0.477 < 1$$

$$q = \left||a|\right|_{1} = max\{0.792; 0.494; 0.477\} = 0.792$$

3 условие

$$\left||a|\right|_{E} = \sqrt{0.405^2 + 0.310^2 + 0.667^2 + 0.167^2 + 0.125^2 + 0.089^2}$$

$$\left||a|\right|_{E} = \sqrt{0.756} = 0.87$$

Ответ: Условия сходимости выполняются.

c) Решим методом Зейделя, код программы находится в приложении **J**.

Ответ: $X = \begin{bmatrix} 0.875 \\ -0.018 \\ -0.649 \end{bmatrix}$

d) Проверка (рис. 3.1.2.1):

$$\mathrm{lsolve}(A, B) = \begin{bmatrix} 0.861 \\ -0.015 \\ -0.648 \end{bmatrix}$$

Рис. 3.1.2.1. Проверка решения средствами MathCAD

Ответ: Проверка подтвердила решение

Задание 3.2.1–1

Используя метод итераций, решить систему нелинейных уравнений с точностью до 0.005.

Решение:

$$\begin{cases} \sin(x+1) - y = 1.2 \\ 2x + \cos(y) = 2 \end{cases}$$

Приведем к виду: $\begin{cases} x = \varphi_1(x, y) \\ y = \varphi_2(x, y) \end{cases}$

$$\begin{cases} x = 1 - \dfrac{\cos(y)}{2} \\ y = \sin(x+1) - 1.2 \end{cases}$$

Выберем начальное значение x, y по графику (рис 3.2.1.1.):

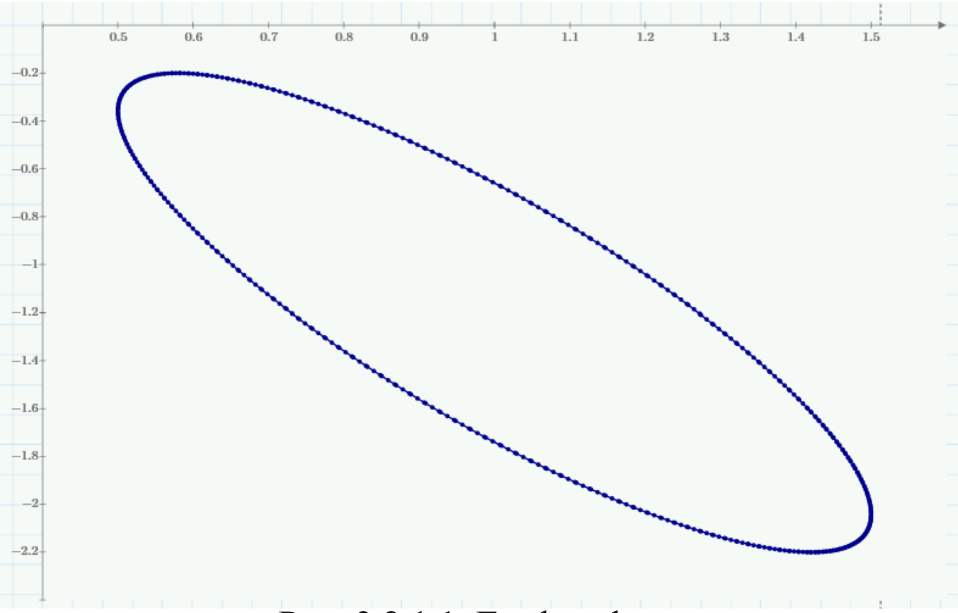


Рис. 3.2.1.1. График функции

$$x = 1, y = -1$$

Код реализованного метода итераций находится в приложении **К**.

Ответ: Решение: $x = 0.510, y = -0.209$.

Задание 3.2.2–1

Привести систему нелинейных уравнений к виду, удовлетворяющему условиям сходимости метода итераций, и найти одно из решений этим методом с точностью до 0.0001.

$$\begin{cases} x^2 + y^2 = 1 \\ x^3 - y = 0 \end{cases}$$

$$\begin{cases} F(x,y) = x^2 + y^2 - 1 \\ G(x,y) = x^3 - y \end{cases}$$

$$\frac{\partial F(x,y)}{\partial x} = 2x; \ \frac{\partial F(x,y)}{\partial y} = 2y;$$

$$\frac{\partial G(x,y)}{\partial x} = 3x^2; \ \frac{\partial G(x,y)}{\partial y} = -1;$$

Составим матрицы Якоби и другие:

$$J(x,y) = \begin{bmatrix} 2x & 2y \\ 3x^2 & -1 \end{bmatrix}$$

$$\Delta_h = \begin{bmatrix} x^2+y^2-1 & 2y \\ x^3-y & -1 \end{bmatrix} \Delta_k = \begin{bmatrix} 2x & x^2+y^2-1 \\ 3x^2 & x^3-y \end{bmatrix}$$

Программа итерации находится в приложении **М.**

Выберем начальное значение x, y по графику (рис 3.2.2.1.):

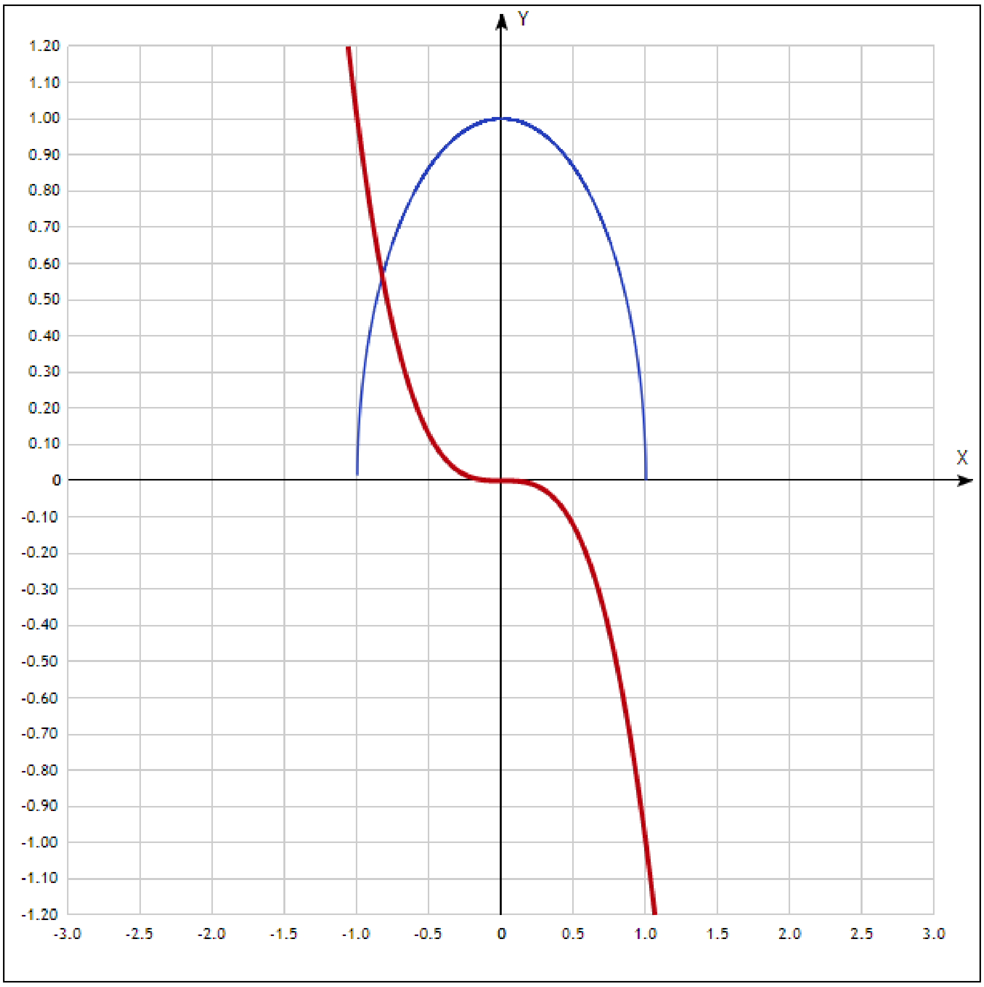


Рис. 3.2.2.1. График функции

$$x = -1, y = 0.5$$

Ответ: $x = 0.5101$ $y = -0.2018$

Задание 3.2.3–1

Найти приближенное решение системы уравнений графическим способом, используя пакет MathCAD, и уточнить одно из них методом Ньютона с точностью до 0.001.

Решение:

$$\begin{cases} tg(xy+0.4)=x^2 \\ 0.6x^2+2y^2=1 \end{cases}$$

$$\begin{cases} F(x,y)=tg(xy+0.4)-x^2 \\ G(x,y)=0.6x^2+2y^2-1 \end{cases}$$

$$\frac{\partial F(x,y)}{\partial x}=\frac{y}{cos^2(xy+0.4)}-2x;\ \frac{\partial F(x,y)}{\partial y}=\frac{x}{cos^2(xy+0.4)};$$

$$\frac{\partial G(x,y)}{\partial x}=1.2x;\ \frac{\partial G(x,y)}{\partial y}=4y;$$

Составим матрицы Якоби и другие:

$$J(x,y)=\begin{bmatrix} \frac{y}{cos^2(xy+0.4)}-2x & \frac{x}{cos^2(xy+0.4)} \\ 1.2x & 4y \end{bmatrix}$$

$$\Delta_h=\begin{bmatrix} tg(xy+0.4)-x^2 & \frac{x}{cos^2(xy+0.4)} \\ 0.6x^2+2y^2-1 & 4y \end{bmatrix}$$

$$\Delta_k=\begin{bmatrix} \frac{y}{cos^2(xy+0.4)}-2x & tg(xy+0.4)-x^2 \\ 1.2x & 0.6x^2+2y^2-1 \end{bmatrix}$$

Программа итерации находится в приложении **L.**

Выберем начальное значение x, y по графику (рис 3.2.3.1.):

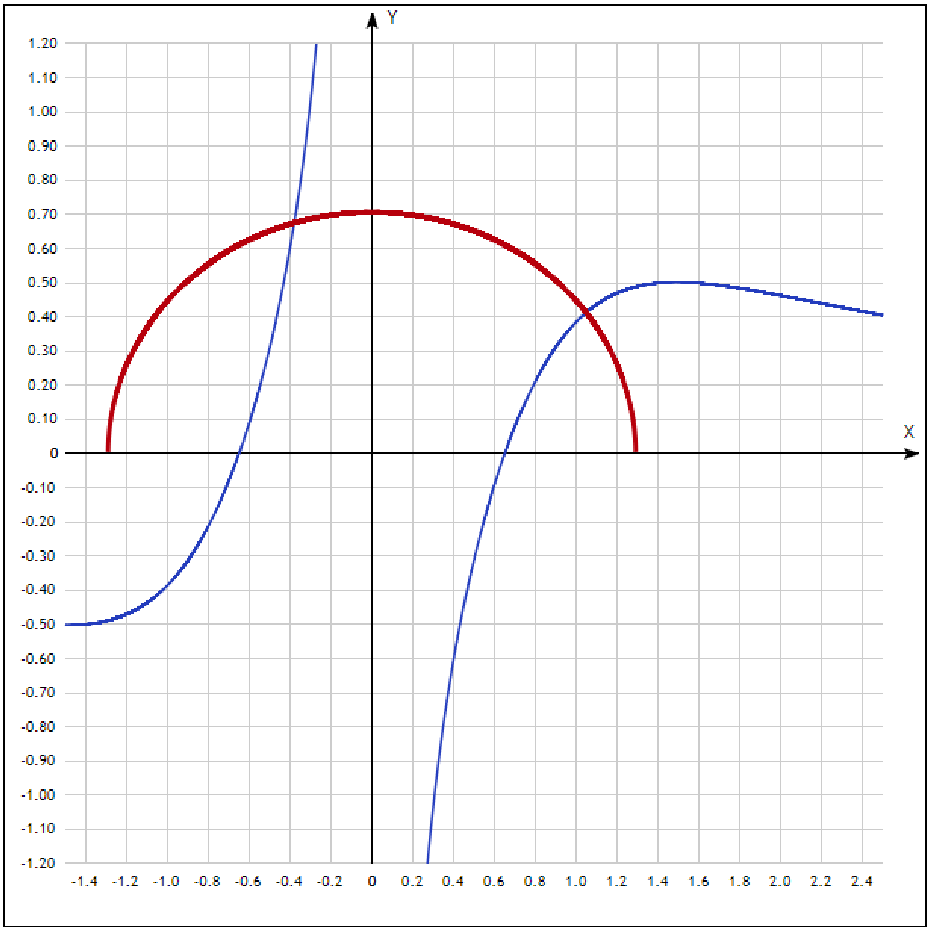


Рис. 3.2.3.1. График функции

$$x = 1, y = 0.5$$

Ответ: Решение $x = 1.048$ $y = 0.413$

4. ПРИБЛИЖЕНИЕ ФУНКЦИЙ

Задание 4.1-1

Для функции, заданной таблично, записать:

a) интерполяционный многочлен Лагранжа;

b) интерполяционный многочлен Ньютона.

Вычислить с помощью полученных многочленов значение функции в заданной точке $(x^*; f(x^*))$.

Начальные данные $f(x^*) = f(1.63)$:

x_i	1.62	1.64	1.65	1.67	1.68
$f(x_i)$	1.172	1.179	1.182	1.186	1.189

Решение:

a) Для вычисления интерполяционного многочлена Лагранжа написана программа, текст программы находится в приложении **M**. С помощью программы вычислено значение функции в точке $f(1.63) = 1.175$.

b) Для вычисления интерполяционного многочлена Ньютона написана программа, текст программы находится в приложении **M**. С помощью программы вычислено значение функции в точке $f(1.63) = 1.176$.

Ответ: а) $f(1.63) = 1.175$; б) $f(1.63) = 1.176$

Задание 4.2-1

Функция задана аналитически $y = (x^2 + 1)e^{-2x}$.

1. Вычислить значение функции в точке 1.7 с помощью интерполяционной формулы Лангранжа и определить точность приближения, если известны значения данной функции в узловых точках: $0,1.4,2.6,4$.
2. Определить шаг для таблице в данной функции на отрезке $[0; 4]$, чтобы с точностью $\varepsilon = 5 * 10^{-3}$ она допускала:
   a. Линейную;
   b. Квадратичную интерполяцию для равнодействующих узлов.

Результат проверить.

Решение.

1) Найдем значение функции в узловых точках, таблица 4.

Таблица 4

x	$(x^2+1)e^{-2x}$
0	1
1.4	0.180
2.6	0.043
4	0.006

Для вычисления функции в точке 1.7 воспользуемся многочленом Лагранжа для не равностоящих узлов, программа представленная в приложении **N**. $f(1.7)$ =0.134.

Определим погрешность метода используя формулу:

$$|R_3^{max}(x^*)| \le \frac{M_4}{4!}|(x^*-x_0)(x^*-x_1)(x^*-x_2)(x^*-x_3)|,$$

$$\text{где } M_4 = \frac{max}{0\le x\le 4}\left|y^{(4)}(x)\right|$$

Вычислим производную 4 порядка для функции $(x^2+1)e^{-2x}$:

$$f^{'}(x) = -2e^{-2x} * (x^2 - x + 1)$$

$$f^{''}(x) = 2e^{-2x} * (2x^2 - 4x + 3)$$

$$f^{'''}(x) = -4e^{-2x} * (2x^2 - 6x + 5)$$

$$f^{\langle 4\rangle}(x) = 16e^{-2x} * (x-2)^2$$

$$M_4 = \frac{max}{0\le x\le 4}\left|f^{\langle 4\rangle}(x)\right| = 16e^0 * (0-2)^2 \to |R_3^{max}(1.7)| \le \frac{64}{24}|(1.7-0)(1.7-1.4)(1.7-2.6)(1.7-4)| = 2.813$$

Учитывая погрешность, можно записать $f(1.7) \approx 2.947$.

2) а) Определим шаг для таблицы для данной функции на отрезке $[0;4]$,

чтобы с точностью $\varepsilon = 5 * 10^{-3}$ она допускала линейную интерполяцию многочленом Ньютона для равностоящих узлов по формуле

$$h \leq \sqrt{\frac{8 * \varepsilon}{M_2}}, \text{где } M_2 \geq \frac{max}{0 \leq x \leq 4} \left| f^{\langle 2 \rangle}(x) \right|$$

$$f''(x) = 2e^{-2x} * (2x^2 - 4x + 3)$$

$$\frac{max}{0 \leq x \leq 4} \left| f^{\langle 2 \rangle}(0) \right| = 2e^0 * (0 + 3) = 6$$

Вычислим шаг таблицы для данной функции: $h \leq \sqrt{\frac{8 * \varepsilon}{M_2}} = \sqrt{\frac{0{,}04}{6}} = 0{,}082.$

$$h = 0.08$$

Составим интерполяционный многочлен Ньютона первого порядка, рисунок 4.2.2.1.

Запишем интерполяционный многочлен Ньютона 1 порядка:

i	x	f(x)	deltaY	deltaY^2
0	0,000	1,000	-0,142	0,030
19	1,520	0,158	-0,013	0,001
20	1,600	0,145	-0,012	0,001
21	1,680	0,133	-0,011	0,001
22	**1,760**	**0,121**	**-0,011**	**0,001**
23	1,840	0,111	-0,010	0,001
24	1,920	0,101	-0,009	0,001
25	2,000	0,092	-0,008	0,001
26	2,080	0,083	-0,008	0,001
44	3,520	0,012	-0,001	0,000
45	3,600	0,010	-0,001	0,000
46	3,680	0,009	-0,001	0,000
47	3,760	0,008	-0,001	0,000
48	3,840	0,007	-0,001	0,000
49	3,920	0,006	-0,001	-0,005
50	4,000	0,006	-0,006	0,006

Рис. 4.2.2.1 Таблица конечных разностей, h=0.08

$$f(1.7) \approx N_1(1.7) = 0.121 + \frac{-0.011}{0.08}(1.7 - 1.76) = 0.129$$

Точное значение функции $f(1.7) \approx 0.130$

2) b) Определим шаг для таблицы для данной функции на отрезке [0;4], чтобы с точностью $\varepsilon = 5 * 10^{-3}$ она допускала квадратичную интерполяцию многочленом Ньютона для равностоящих узлов по формуле

$$h \le \sqrt[3]{\frac{9\sqrt{3} * \varepsilon}{M_3}}, \text{где } M_3 \ge \max_{0 \le x \le 4} \left| f^{\langle 3 \rangle}(x) \right|$$

$$f^{'''}(x) = -4e^{-2x} * (2x^2 - 6x + 5)$$

$$\max_{0 \le x \le 4} \left| f^{\langle 3 \rangle}(0) \right| = -4e^0 * (0 + 5) = 20$$

Вычислим шаг таблицы для данной функции: $h \le \sqrt[3]{\frac{9\sqrt{3}*\varepsilon}{M_3}} = \sqrt{\frac{0.078}{20}} = 0.078.$

$$h = 0.07$$

Составим интерполяционный многочлен Ньютона первого порядка, рисунок 4.2.2.2.

i	x	f(x)	deltaY	deltaY^2	deltaY^3
0	0,000	1,00000000	-0,12638191	0,02336092	-0,00491445
19	1,330	0,19367963	-0,01368185	0,00078734	-0,00003989
20	1,400	0,17999779	-0,01289450	0,00074745	-0,00003406
21	1,470	0,16710328	-0,01214705	0,00071339	-0,00003023
22	1,540	0,15495623	-0,01143366	0,00068315	-0,00002783
23	1,610	0,14352256	-0,01075051	0,00065533	-0,00002641
24	1,680	0,13277205	-0,01009518	0,00062892	-0,00002564
25	**1,750**	**0,12267687**	**-0,00946626**	**0,00060328**	**-0,00002528**
26	1,820	0,11321061	-0,00886298	0,00057800	-0,00002516
27	1,890	0,10434763	-0,00828498	0,00055284	-0,00002513
28	1,960	0,09606264	-0,00773214	0,00052771	-0,00002511
29	2,030	0,08833050	-0,00720443	0,00050260	-0,00002505
30	2,100	0,08112607	-0,00670183	0,00047755	-0,00002490
31	2,170	0,07442424	-0,00622429	0,00045264	-0,00002465
58	4,060	0,00520187	-0,00520187	0,00520187	-0,00520187

Рис. 4.2.2.2 Таблица конечных разностей, h=0.07

Запишем интерполяционный многочлен Ньютона 2 порядка:

$$f(1.7) \approx N_2(1.7) = 0.12268 + \frac{-0.00947}{0.07}(1.7 - 1.75) + \frac{0.000603}{2 * 0.07^2}(1.7 - 1.75)(1.7 - 1.82) = 0.130$$

Задание 4.3-1

Функция $y = f(x)$ задана таблично (таблица 4.3.1).

Табл. 4.3.1

x	f(x)
1.2	3.32
1.25	3.491
1.3	3.669
1.35	3.857
1.4	4.055
1.45	4.263
1.5	4.282
1.55	4.712
1.6	4.953
1.65	5.203

1. Построить первый и второй интерполяционные многочлены Ньютона и с их помощью найти значения функции в точках 1.22 и 1.58 с погрешностью не более, чем 0.005, двумя способами.

2. Вычислить значения функции в точках 1.33 и 1.62, используя интерполяционную формулу Ньютона второго порядка, и оценить погрешность.

Решение. 1. Решение выполнено в программе код которой находиться в приложении **О**.

1. Первый интерполяционный многочлен Ньютона:

$$f(1.22) = 3.388, f(1.58) = 4.85$$

2. Второй интерполяционный многочлен Ньютона:

$$f(1.22) = 3.388, f(1.58) = 4.85$$

2. Вычислим значение функции в точках $x_1 = 1.33$ и $x_1 = 1.62$. Запишем интерполяционный многочлен Ньютона второго порядка используя вторую запись формулы Ньютона (рисунок 4.3.2.1.):

i	x	y	dy	dy2	dy3
0,000	1,200	3,320	0,171	0,007	0,003
1,000	1,250	3,491	0,178	0,010	0,000
2,000	1,300	3,669	0,188	0,010	0,000
3,000	1,350	3,857	0,198	0,010	-0,199
4,000	1,400	4,055	0,208	-0,189	0,600
5,000	1,450	4,263	0,019	0,411	-0,600
6,000	1,500	4,282	0,430	-0,189	0,198
7,000	1,550	4,712	0,241	0,009	
8,000	1,600	4,953	0,250		

Рис. 4.3.2.1 Таблица конечных разностей, h=0.05

$$q = \frac{1.33 - 1.3}{0.05} = 0.6$$

$$N_2(x_1) = 3.320 + 0.6 * 0.171 + \frac{0.6(0.6-1)}{2}0.007 +$$

$$+\frac{0.6(0.6-1)(0.6-2)}{6}0.003 = 3.422$$

Погрешность $|R_2(x)| \leq \frac{0.6}{6}|0.6(0.6-1)(0.6-2)| \approx 0.034$

$$f(x_1) = 3.422 \pm 0.034$$

$$q = \frac{1.62 - 1.6}{0.05} = 0.4$$

$$N_2(x_1) = 3.320 + 0.4 * 0.171 + \frac{0.4(0.4-1)}{2}0.007 +$$

$$+\frac{0.4(0.4-1)(0.4-2)}{6}0.003 = 3.388$$

Погрешность $|R_2(x)| \leq \frac{0.6}{6}8 * 6 * 7 \approx 0.038.$

$$f(x_1) = 3.388 \pm 0.038$$

Ответ: 1) $f(x_1) = 3.422 \pm 0.034;$ 2) $f(x_1) = 3.388 \pm 0.038$

Задание 4.4-1

Функция $y = \sin(x) + \frac{\ln(x)}{x}$ задана аналитически на $[1; 4]$.

1. Приблизить данную функцию на заданном отрезке интерполяционным многочленом Ньютона 2, 3 и 4 степени и оценить максимальную погрешность приближения на заданном отрезке:
   a) по системе узлов включая концы;
   b) по методу Чебышева.
2. На одном чертеже построить график приближающего многочлена и данную функцию с узловыми точками.
3. Сравнить полученные результаты в п. a и b, вычислить отклонения значений многочлена от точных значений функции в не узловых точках и сопоставить их с оценкой максимальной погрешностью.
4. Вычислить значения функции в точке 1.113, с помощью полученных многочленов, оценить погрешность. Найти точное значение функции в данной точке и сравнить с приближенным.
5. Сделать вывод, проведя сравнения качества построенных интерполяционных многочленов в зависимости от их степени и расположения узлов интерполяции.

Решение:

1. Разобьём отрезок $[1; 4]$ на две равные части, шаг:

$$h = \frac{4-1}{2} = 1.5$$

Составим таблицу разделенных разностей (рисунок 4.4.1).

i	x	y	dy1	dy2
0	1	0,84147	0,12352	-1,49873
1	2,5	0,96499	-1,37522	
2	4	-0,41023		

Рис. 4.4.1 Таблица конечных разностей, h=1.5

Запишем интерполяционный член Ньютона 2 степени.

$$N_2(x) = 0.841147 + \frac{0.12352}{1.5^1}(x - x_0) - \frac{1.49873}{2 * 1.5^2}(x - x_0)(x - x_1)$$

Максимальная погрешность приближения при n = 2 для равностоящих узлов:

$$|R_2^{max}(x)| \leq \frac{M_3 h^3}{3!}|q(q-1)(q-2)|, \text{где } q = \frac{x - x_0}{h}, x \in [1; 4]$$

По графику функции $y(q) = |q(q-1)(q-2)|$ определим ее наибольшее значение на интервале $(0; 2)$, так как, если $n = 2$, то $q \in (0; 2)$ (рисунок 4.4.2).

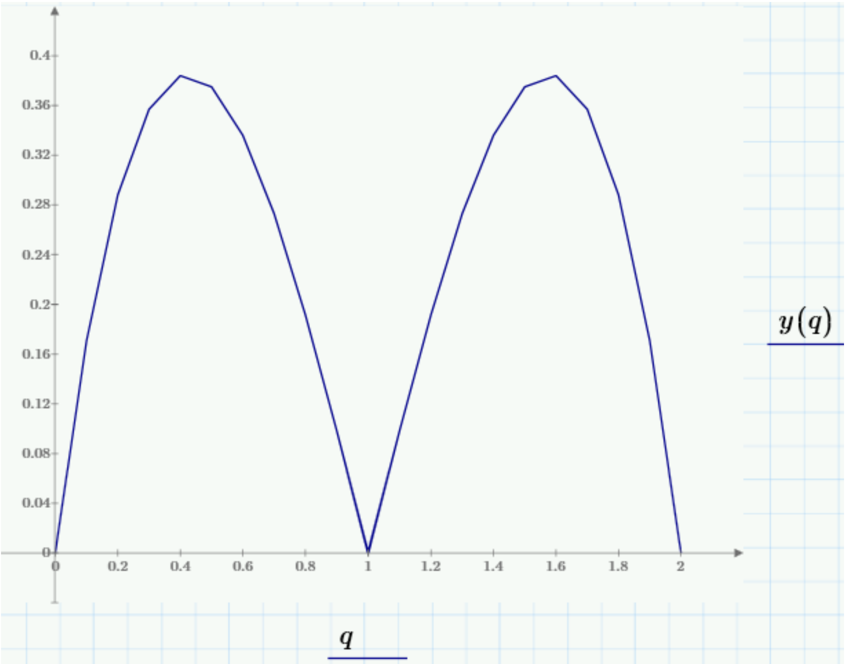


Рис. 4.4.2 График функции $y(q) = |q(q-1)(q-2)|$

Получим что $y(q) < 0.4$.

$M_3 = \max\limits_{1 \le x \le 4} \left| y^{\langle 3 \rangle}(x) \right| \le 4$ определим по графику производной $d3(x) = y^{\langle 3 \rangle}(x)$ (рисунок 4.4.3).

$$f(x) := \sin(x) + \frac{\ln(x)}{x}$$

$$x := 1, 1.1 .. 4$$

$$d3(x) := \frac{d^3}{dx^3} f(x)$$

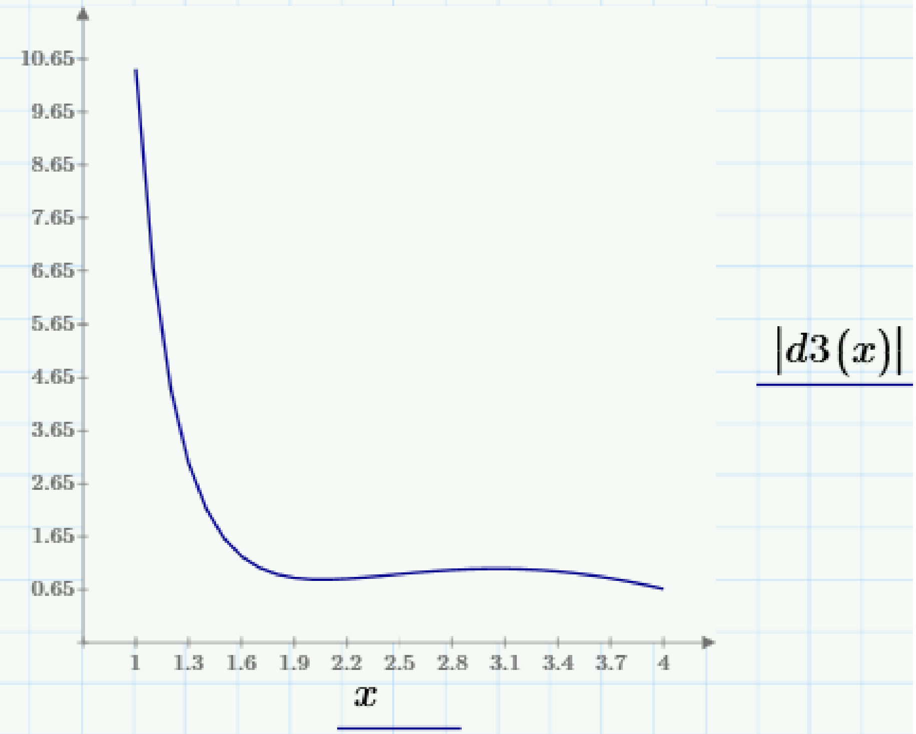


Рис. 4.4.3 График производной $y^{\langle 3 \rangle}(x)$

$$|R_3^{max}(x)| \le \frac{10.46 * 1.5^3 * 0.4}{6} = 1.569$$

Следовательно, максимальная погрешность интерполяции данной функции на промежутке $[1; 4]$ не превосходит 1.569.

Минимизирую погрешность интерполяции путем специального выбора узлов интерполяции (метод Чебышева).

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{(2i+1)\pi}{6}\right)$$

Найдем значение функции в полученных точках (рисунок 4.4.4).

i	x	y
0	3,799	-0,26
1	2,5	0,965
2	1,201	1,0849

Рис. 4.4.4 Узлы интерполяции

Составим таблицу разделенных разностей первого и второго порядка (рисунок 4.4.5):

i	x	y	y,i+1	y,i+2
0	1,201	1,0849	-0,092	-0,327
1	2,5	0,965	-0,943	
2	3,799	-0,26		

Рис. 4.4.5 Таблицу разделенных разностей

Запишем многочлен Ньютона 2 порядка по узлам Чебышева:

$$T_2(x) = 1{,}0849 - 0{,}092(x - x_0) - 0.327(x - x_0)(x - x_1)$$

Максимальную погрешность приближения при $n = 2$ определим по формуле:

$$|E_2^{max}(x)| \le \frac{10.46 * (4-1)^3}{2^5 * 6} = 1.471$$

Следовательно, погрешность интерполирования данной функции на $[1;4]$ многочленом $T_2(x)$ не превосходит 1.471.

Построим графики с приближающими многочленами и значениями функции (рисунок 4.4.6 и рисунок 4.4.7).

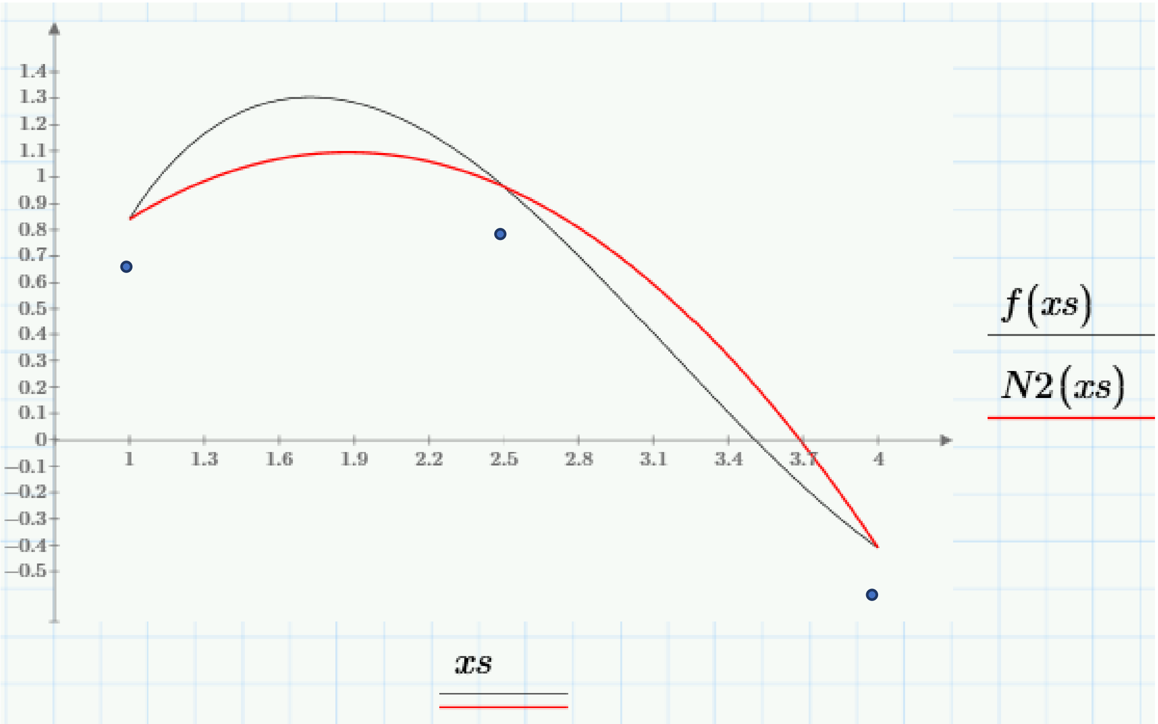


Рис. 4.4.6

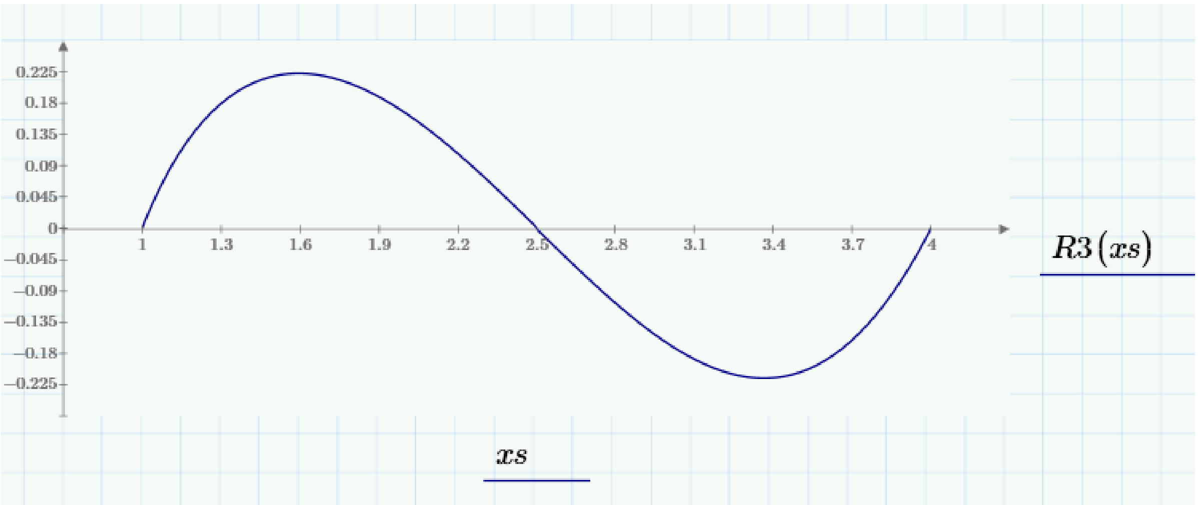
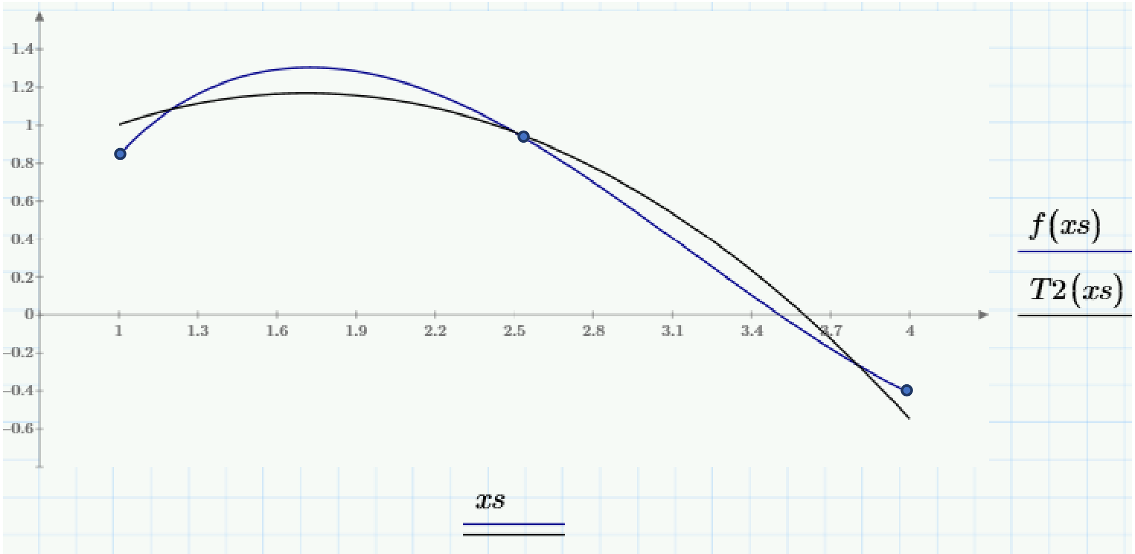


График функции f(x) и приближающего многочлена Ньютона 2 порядка $R2(x) = f(x) - N2(x)$.

По графику R2 определим $|R_2(x)| \le 0.225$ что почти в 2 раза меньше по сравнению с максимальной погрешностью$R_2^{max}(x)$ на [1; 4].



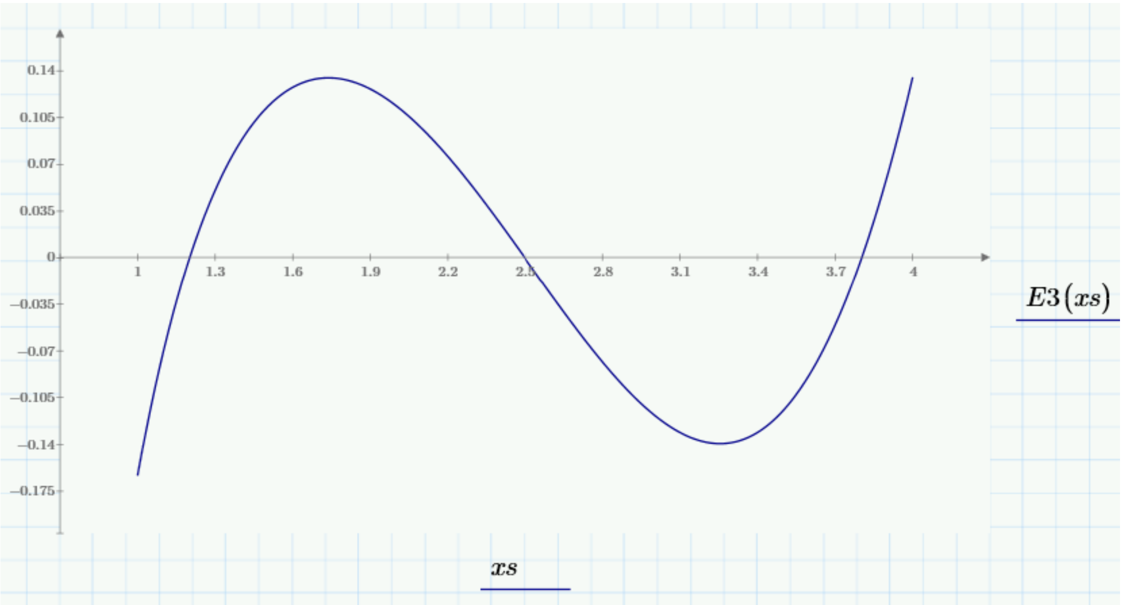


Рис. 4.4.7 График функции f(x) и приближающего многочлена Чебышева 2 порядка и функция ошибки $E2(x) = f(x) - T2(x)$

По графику E2 определим $|E_2(x)| \le 0.14$ что почти в 2 раза меньше по сравнению с максимальной погрешностью$E_2^{max}(x)$ на [1; 4].

Для построения многочлена 3 порядка разобьём отрезок $[1;4]$ на 3 равные части, шаг:

$$h = \frac{4-1}{3} = 1$$

Составим таблицу разделенных разностей (рисунок 4.4.8).

i	x	y	dy1	dy2	dy3
0	1	0,84147	0,41440	-1,16295	0,99394
1	2	1,25587	-0,74855	-0,16901	
2	3	0,50732	-0,91755		
3	4	-0,41023			

Рис. 4.4.8 Таблица конечных разностей, h=1

Запишем интерполяционный член Ньютона 3 степени.

$$N_3(x) = 0.841147 + \frac{0,41440}{1^1}(x-x_0) - \frac{1.16295}{2*1^2}(x-x_0)(x-x_1) + \frac{0,99394}{6*1^3}(x-x_0)(x-x_1)(x-x_2)$$

Максимальная погрешность приближения при n = 3 для равностоящих узлов:

$$|R_3^{max}(x)| \leq \frac{M_4 h^4}{4!}|q(q-1)(q-2)(q-3)|, \text{где } q = \frac{x-x_0}{h}, x \in [1;4]$$

По графику функции $y(q) = |q(q-1)(q-2)(q-3)|$ определим ее наибольшее значение на интервале $(0; 3)$, так как, если $n = 3$, то $q \in (0; 3)$ (рисунок 4.4.9).

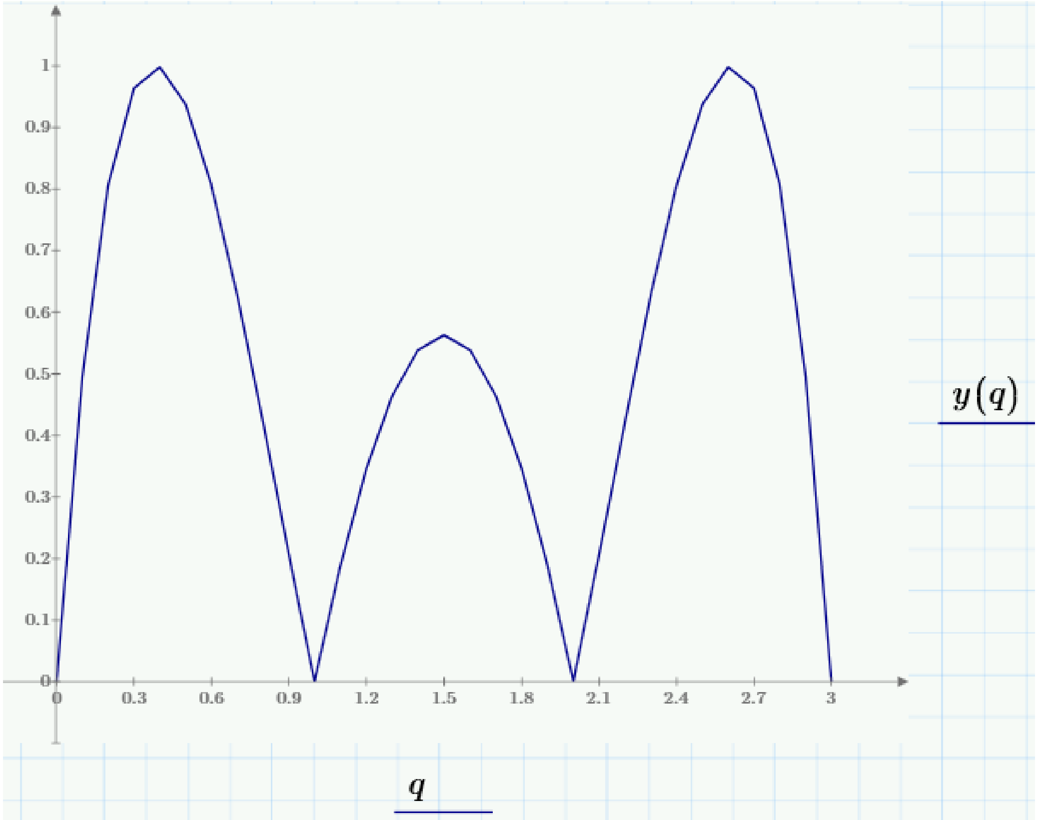


Рис. 4.4.9 График функции $y(q) = |q(q-1)(q-2)(q-3)|$

Получим что $y(q) < 1$

$M_4 = \max\limits_{1 \le x \le 4} \left|y^{\langle 4 \rangle}(x)\right| \le 49.159$ определим по графику производной $d4(x) = y^{\langle 4 \rangle}(x)$ (рисунок 4.4.10).

$$f(x) := \sin(x) + \frac{\ln(x)}{x}$$

$$x := 1, 1.1 .. 4$$

$$d4(x) := \frac{d^4}{dx^4} f(x)$$

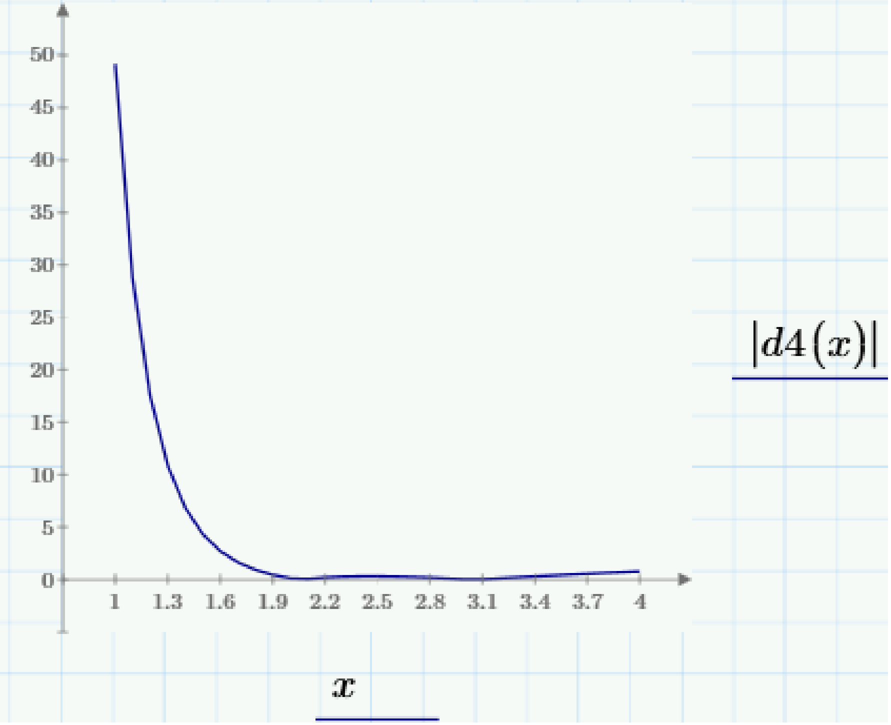


Рис. 4.4.10 График производной $y^{\langle 4 \rangle}(x)$

$$|R_4^{max}(x)| \le \frac{49.159 * 1^4 * 1}{24} = 0.8048$$

Следовательно, максимальная погрешность интерполяции данной функции на промежутке $[1; 4]$ не превосходит 0.8048.

Минимизирую погрешность интерполяции путем специального выбора узлов интерполяции (метод Чебышева).

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{(2i+1)\pi}{8})$$

Найдем значение функции в полученных точках (рисунок 4.4.11).

i	x	y
0	3,8858	-0,328
1	3,074	0,4328
2	1,926	1,2779
3	1,1142	0,9946

Рис. 4.4.11 Узлы интерполяции

Составим таблицу разделенных разностей первого и второго порядка (рисунок 4.4.12):

i	x	y	y,i+1	y,i+2	y,i+3
0	1,1142	0,9946	0,34895	-0,554	0,1627
0	1,926	1,2779	-0,73609	-0,103	
1	3,074	0,4329	-0,93735		
2	3,8858	-0,328			

Рис. 4.4.12 Таблицу разделенных разностей

Запишем многочлен Ньютона 3 порядка по узлам Чебышева:

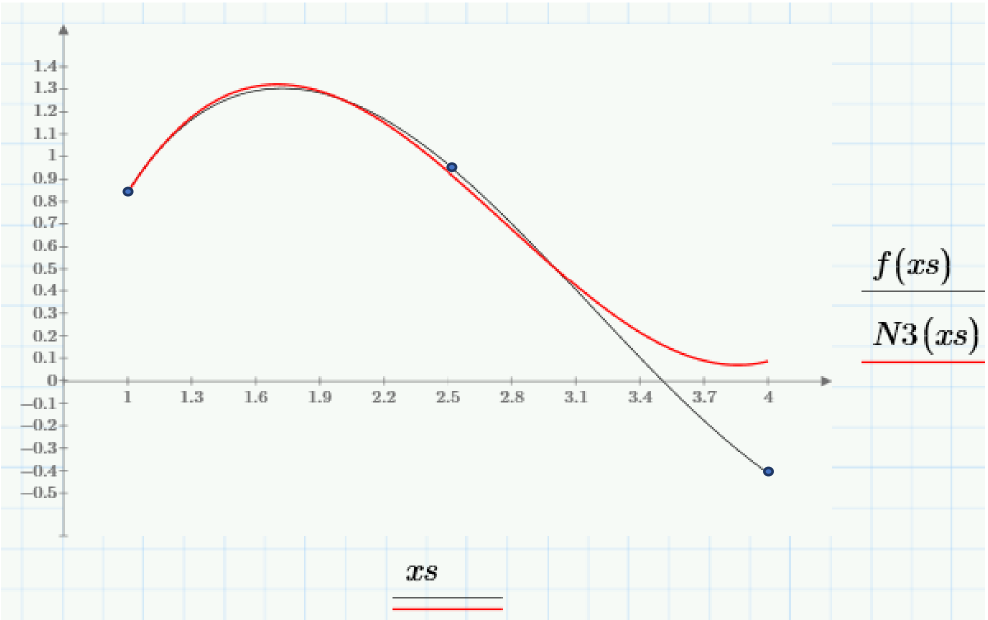
$$T_3(x) = 0.9946 + 0.34895(x - x_0) - 0.554(x - x_0)(x - x_1) + 0.163(x - x_0)(x - x_1)(x - x_2)$$

Максимальную погрешность приближения при $n = 3$ определим по формуле:

$$|E_3^{max}(x)| \leq \frac{49.159 * (4 - 1)^3}{2^5 * 24} = 0,3728$$

Следовательно, погрешность интерполирования данной функции на $[1; 4]$ многочленом $T_3(x)$ не превосходит 0.3728.

Построим графики с приближающими многочленами и значениями функции (рисунок 4.4.13 и рисунок 4.4.14).



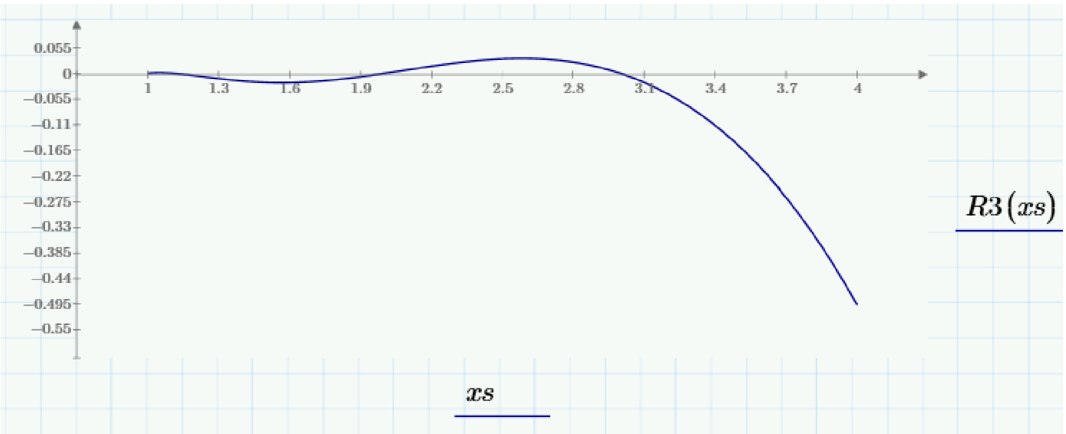
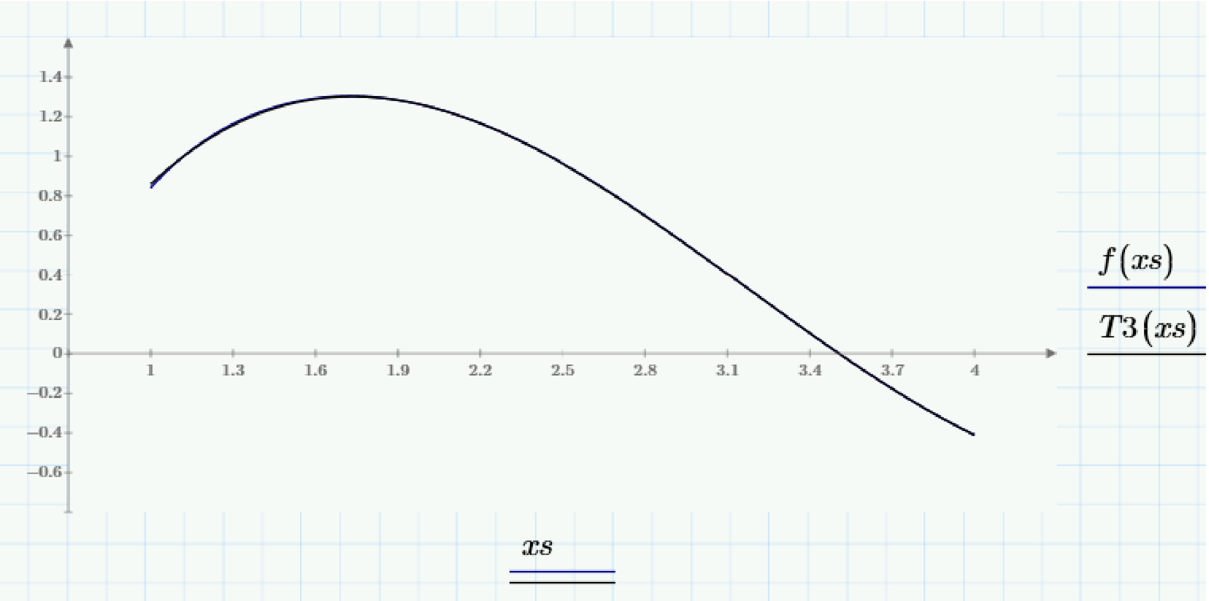
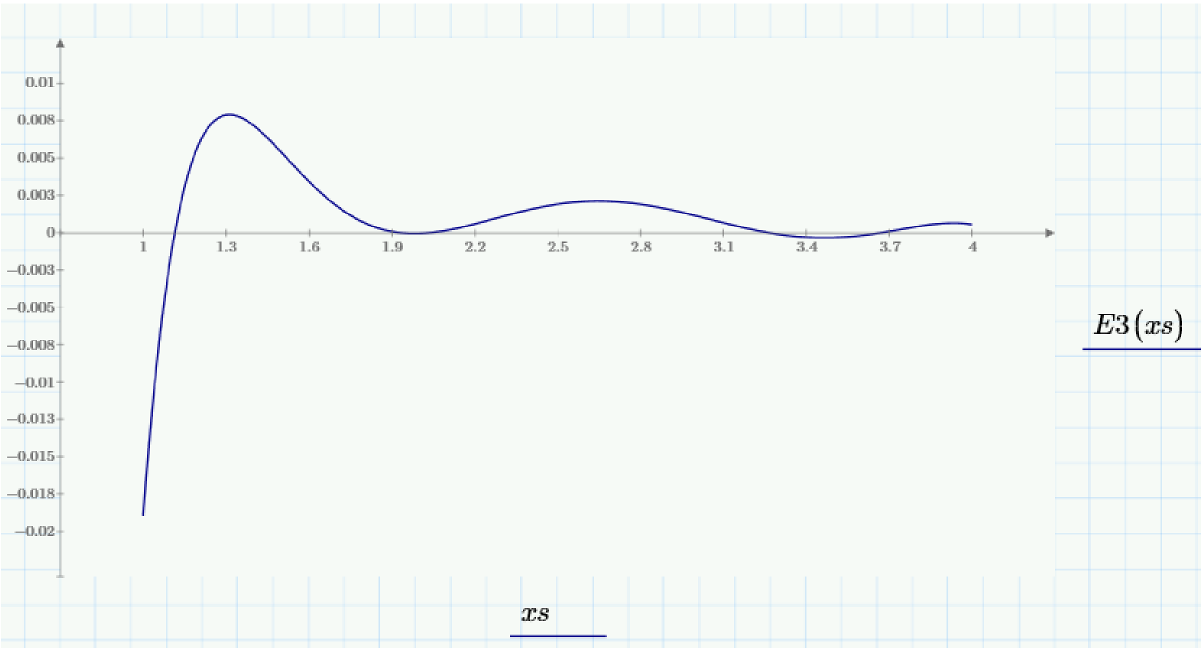


Рис. 4.4.13 График функции f(x) и приближающего многочлена Ньютона 3 порядка $R3(x) = f(x) - N3(x)$.

По графику R3 определим $|R_3(x)| \leq 0.5$ что почти в 2 раза меньше по сравнению с максимальной погрешностью$R_3^{max}(x)$ на [1; 4].

Рис.





4.4.14 График функции f(x) и приближающего многочлена Чебышева 2 порядка и функция ошибки $E3(x) = f(x) - T3(x)$

По графику E3 определим $|E_3(x)| \leq 0{,}018$ что почти в 2 раза меньше по сравнению с максимальной погрешностью$E_3^{max}(x)$ на [1; 4].

Для построения многочлена 4 порядка разобьём отрезок $[1; 4]$ на 4 равные части, шаг:

$$h = \frac{4-1}{4} = 0{,}75$$

Составим таблицу разделенных разностей (рисунок 4.4.15).

i	x	y	dy1	dy2	dy3	dy4
0	1	0,84147	0,46230	-0,80107	0,42933	-0,01176
1	1,75	1,30377	-0,33878	-0,37174	0,41757	
2	2,5	0,96499	-0,71052	0,04582		
3	3,25	0,25447	-0,66470			
4	4	-0,41023				

Рис. 4.4.15 Таблица конечных разностей, h=0,75

Запишем интерполяционный член Ньютона 4 степени.

$$N_3(x) = 0.841147 + \frac{0,46230}{0,75^1}(x - x_0) - \frac{0,80107}{2 * 0,75^2}(x - x_0)(x - x_1) + \frac{0,42933}{6 * 0,75^3}(x - x_0)(x - x_1)(x - x_2) - \frac{0,01176}{24 * 0,75^4}(x - x_0)(x - x_1)(x - x_2)(x - x_3)$$

Максимальная погрешность приближения при n = 4 для равностоящих узлов:

$$|R_4^{max}(x)| \leq \frac{M_5 h^5}{5!}|q(q-1)(q-2)(q-3)(q-4)|, \text{где } q = \frac{x - x_0}{h}, x \in [1; 4]$$

По графику функции $y(q) = |q(q-1)(q-2)(q-3)(q-4)|$ определим ее наибольшее значение на интервале $(0; 4)$, так как, если $n = 4$, то $q \in (0; 4)$ (рисунок 4.4.16).

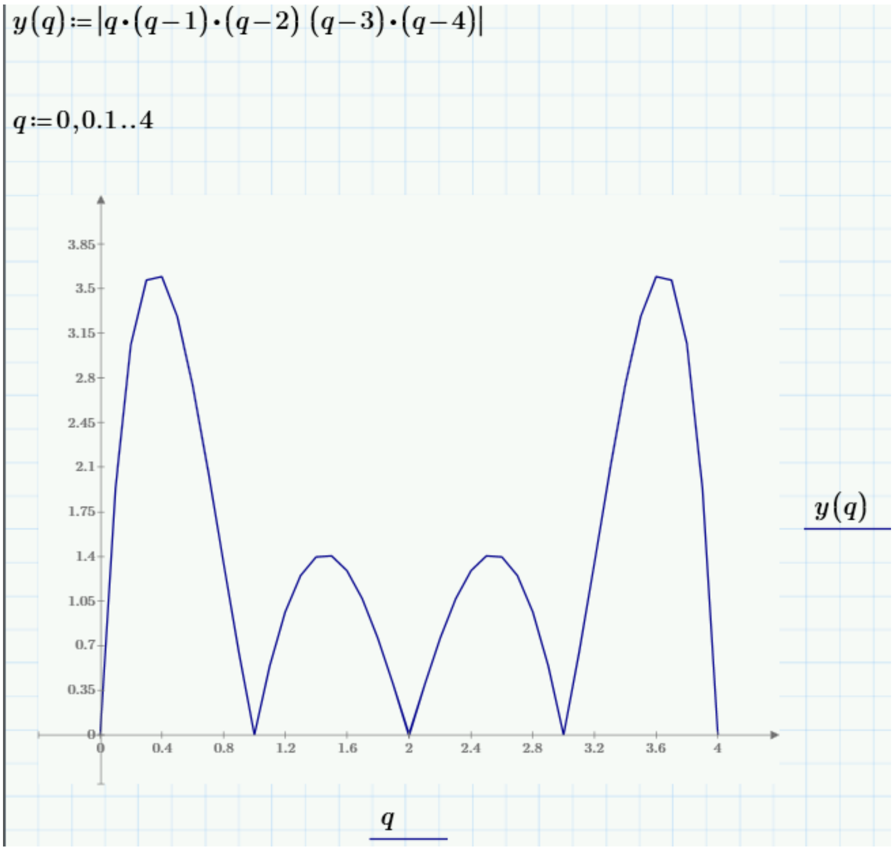


Рис. 4.4.16 График функции $y(q) = |q(q-1)(q-2)(q-3)(q-4)|$

Получим что $y(q) < 3{,}5$

$M_5 = \max\limits_{1 \le x \le 4} \left|y^{\langle 5 \rangle}(x)\right| \le 274{,}54$ определим по графику производной $d5(x) = y^{\langle 5 \rangle}(x)$ (рисунок 4.4.17).

$$f(x) := \sin(x) + \frac{\ln(x)}{x}$$

$$x := 1, 1.1 .. 4$$

$$d5(x) := \frac{d^5}{dx^5} f(x)$$

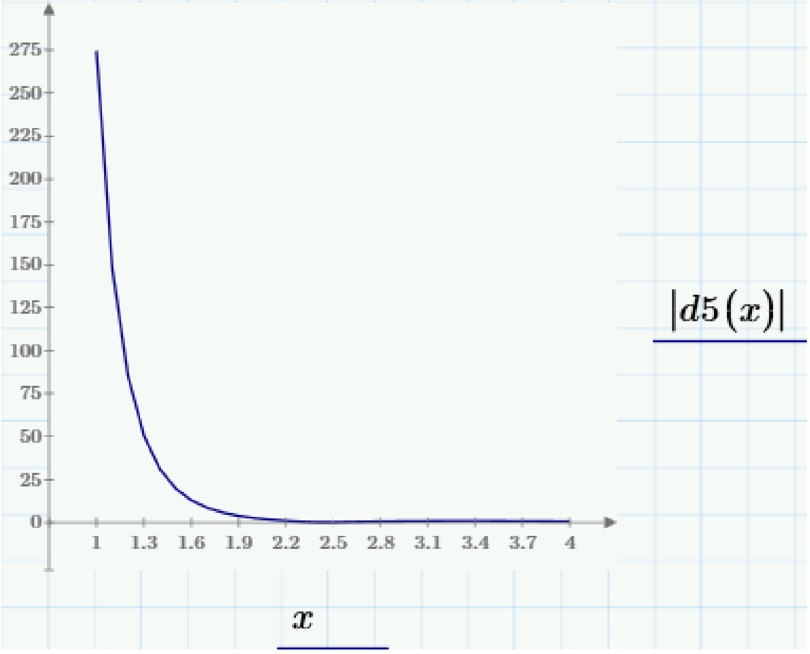


Рис. 4.4.17 График производной $y^{\langle 5 \rangle}(x)$

$$|R_4^{max}(x)| \le \frac{274{,}54 * 0{,}75^5 * 3{,}5}{120} = 0{,}19$$

Следовательно, максимальная погрешность интерполяции данной функции на промежутке $[1; 4]$ не превосходит 0,19.

Минимизирую погрешность интерполяции путем специального выбора узлов интерполяции (метод Чебышева).

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2i+1)\pi}{10}\right)$$

Найдем значение функции в полученных точках (рисунок 4.4.18).

i	x	y
0	3,9266	-0,358
1	3,3817	0,1225
2	2,5	0,965
3	1,6183	1,2963
4	1,0734	0,9448

Рис. 4.4.18 Узлы интерполяции

Составим таблицу разделенных разностей первого и второго порядка (рисунок 4.4.19):

i	x	y	y,i+1	y,i+2	y,i+3	y,i+4
0	1,0734	0,9448	0,6451	-0,716	0,1676	-0,001
1	1,6183	1,2963	-0,376	-0,329	0,1646	
2	2,5	0,965	-0,956	0,0511		
3	3,3817	0,1225	-0,883			
4	3,9266	-0,358				

Рис.

4.4.19 Таблицу разделенных разностей

Запишем многочлен Ньютона 4 порядка по узлам Чебышева:

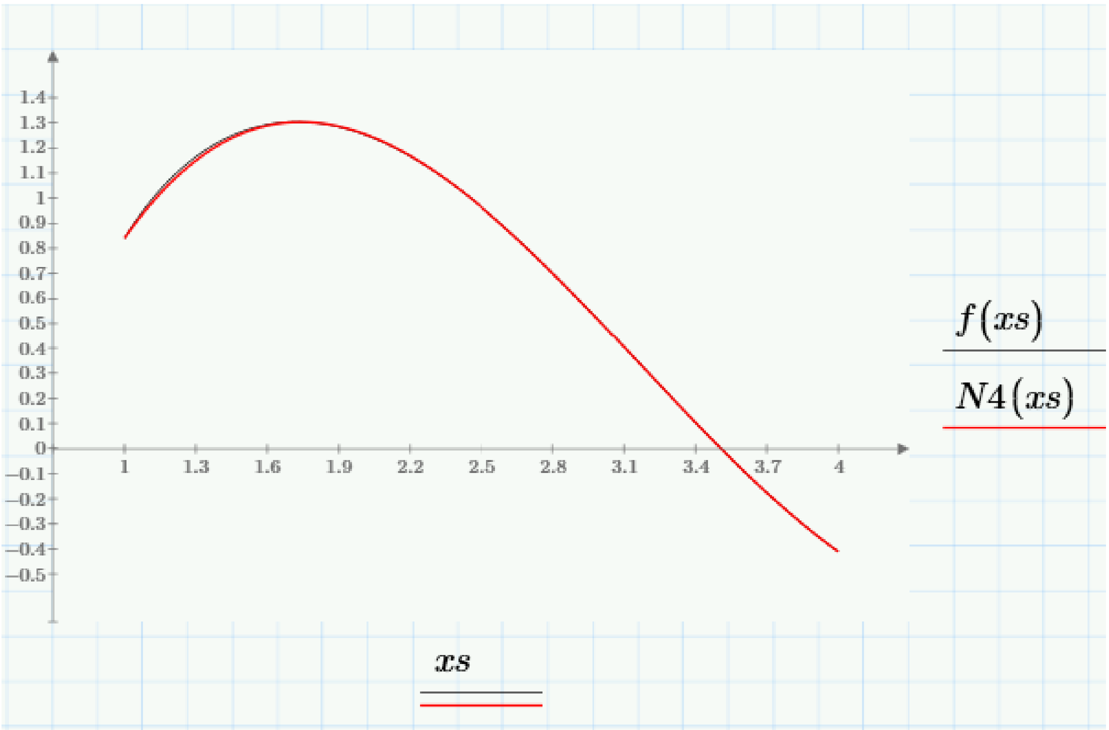
$$T_4(x) = 0.9448 + 0.645(x - x_0) - 0,716(x - x_0)(x - x_1) + 0.1676(x - x_0)(x - x_1)(x - x_2) - 0.001(x - x_0)(x - x_1)(x - x_2)(x - x_3)$$

Максимальную погрешность приближения при $n = 4$ определим по формуле:

$$|E_4^{max}(x)| \leq \frac{274,54 * (4-1)^5}{2^9 * 120} = 0,109$$

Следовательно, погрешность интерполирования данной функции на $[1;4]$ многочленом $T_4(x)$ не превосходит 0,109.

Построим графики с приближающими многочленами и значениями функции (рисунок 4.4.20 и рисунок 4.4.21).



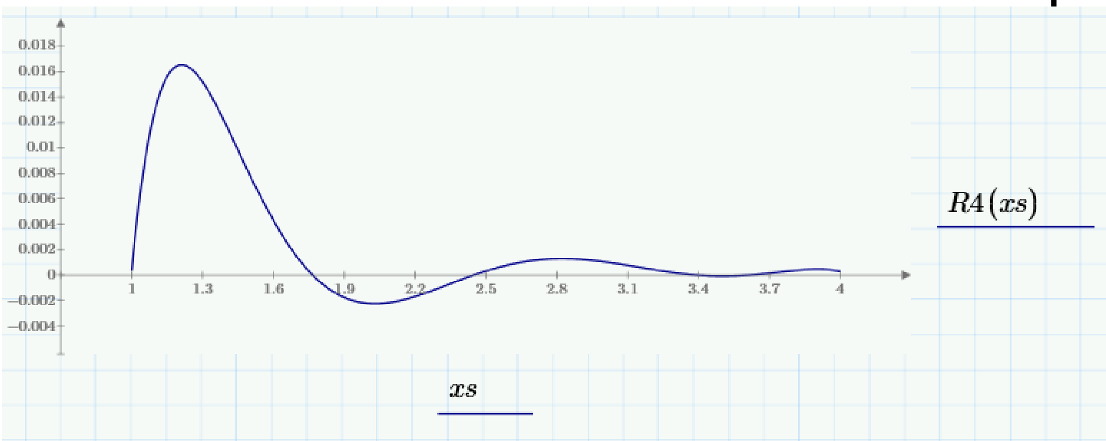
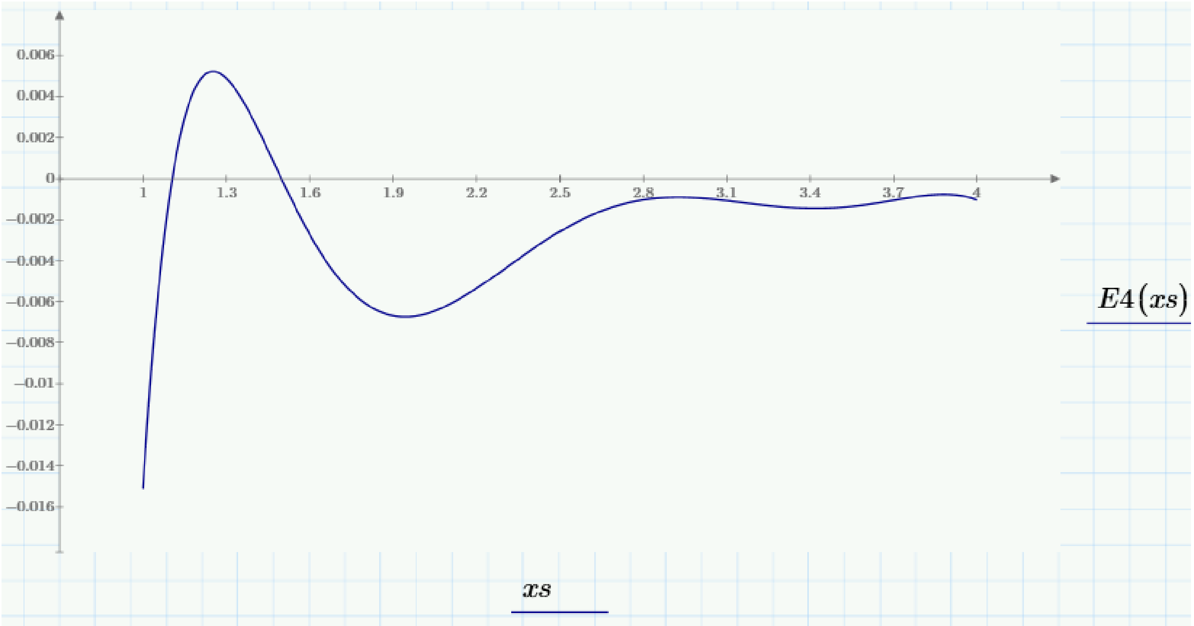


Рис. 4.4.20 График функции f(x) и приближающего многочлена Ньютона 3 порядка $R4(x) = f(x) - N4(x)$.

По графику R4 определим $|R_3(x)| \leq 0.016$ что почти в 2 раза меньше по сравнению с максимальной погрешностью$R_4^{max}(x)$ на [1; 4].

Рис.



4.4.21 График функции f(x) и приближающего многочлена Чебышева 2 порядка и функция ошибки $E4(x) = f(x) - T4(x)$

По графику E4 определим $|E_4(x)| \leq 0{,}014$ что почти в 2 раза больше по сравнению с максимальной погрешностью$E_4^{max}(x)$ на [1; 4].

2. Значение функции в точке $x = 1.113$ равно $f(x) = 0.99322$.

Значение функции, вычисленное с помощью построенных многочленов:

$$N2(x) = 0.90365$$

$$T2(x) = 1.05308$$

$$\Delta_{N_2} = |0.99322 - 0.90365| = 0.089 \qquad \Delta_{T_2} = |0.99322 - 1.05308| = 0.059$$

$$\delta_{N_2} = \frac{0,089}{0.90322} * 100\% \approx 0.099\% \qquad \delta_{T_2} = \frac{0,059}{1.05308} * 100\% \approx 0.057\%$$

$$N3(x) = 0.99325$$

$$T3(x) = 0.99333$$

$$\Delta_{N_3} = |0.99322 - 0.99325| = 0.3 * 10^{-4} \quad \Delta_{T_3} = |0.99322 - 0.99333| = 0.11 * 10^{-3}$$

$$\delta_{N_3} = \frac{0,00003}{0.903325} * 100\% \approx 0.003\% \qquad \delta_{T_3} = \frac{0,00011}{0.99333} * 100\% \approx 0.0001\%$$

$$N4(x) = 0.97932$$

$$T4(x) = 0.99258$$

$$\Delta_{N_4} = |0.99322 - 0.97932| = 0.0139 \quad \Delta_{T_2} = |0.99322 - 0.99258| = 0.64 * 10^{-3}$$

$$\delta_{N_4} = \frac{0,0139}{0.97932} * 100\% \approx 1.41\% \qquad \delta_{T_4} = \frac{0,0064}{0.99258} * 100\% \approx 0.64\%$$

Ответ:

$$f(x) \approx N2(x) = 0.90365 \pm 0.089$$

$$f(x) \approx T2(x) = 1.05308 \pm 0.059$$

$$f(x) \approx N3(x) = 0.99325 \pm 0.00003$$

$$f(x) \approx T3(x) = 0.99333 \pm 0.00011$$

$$f(x) \approx N4(x) = 0.97932 \pm 0.00139$$

$$f(x) \approx T4(x) = 0.99258 \pm 0.0064$$

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ A

```
import FunctionTask
import math, sys
a = float(input("a = "))
b = float(input("b = "))
step = float(input("step = "))
"""[a,b] корня функции, step - шаг хода"""
print ("Табличный метод отделения корней")
while a <= b:
    print ("|x = " , round(a,3), "|f(x) = 
",round(FunctionTask.Function2(a),3),"|")
    a+=step
```

ПРИЛОЖЕНИЕ B

```
import math,FunctionTask
"""Метод бисекции"""
a = float(input("a = "))
b = float(input("b = "))
accuracy = float(input("accuracy = "))
x = (a+b)/2.0
while abs(a-b)/(2) >= accuracy:
    if FunctionTask.Function2(a) * 
FunctionTask.Function2(x) < 0:
        b = x
    if FunctionTask.Function2(b) * 
FunctionTask.Function2(x) < 0:
        a = x
    x = (a+b)/2.0
print ("x = ", round(x,2))
print ("f(x) = ", round(FunctionTask.Function2(x),2))
```

ПРИЛОЖЕНИЕ C

```
import math,FunctionTask
"""Метод бисекции"""
a = float(input("a = "))
b = float(input("b = "))
accuracy = float(input("accuracy = "))
x = (a+b)/2.0
while abs(a-b)/(2.0) >= accuracy:
```

```
    if FunctionTask.Function4(a) *
FunctionTask.Function4(x) < 0:
        b = x
    if FunctionTask.Function4(b) *
FunctionTask.Function4(x) < 0:
        a = x
    x = (a+b)/2.0
print ("x = ", round(x,4))
print ("f(x) = ", round(FunctionTask.Function4(x),4))
```

ПРИЛОЖЕНИЕ D

```
import math
#Базовая функция
def Function(x):
    return x - math.sin(x) - 0.25
#Первая производная
def Derivate(x):
    return 1 - math.cos(x)
#Вторая производная
def DerivateTwo(x):
    return 1 + math.sin(x)
#Рекурсионный метод касательных
def RecursionTanget(x,accuracy):
    newX = x - Function(x)/Derivate(x)
    if abs(newX - x) > accuracy:
        return RecursionTanget(newX,accuracy)
    else:
        return newX
#Вызовы программы
print("Метод касательных Ньютона")
x = 1.5
accuracy = 0.001
y = round(RecursionTanget(float(x),accuracy),4)
print("x =",y,"f(x) =",round(Function(y),4))
```

ПРИЛОЖЕНИЕ E

```
import math
#Исходная функция
def Function(_x):
    return _x**3-3*_x**2+9*_x-8
#Первая производная
```

```
def DerivativeFirst(_x):
    return 3*_x**2-6*_x+9
#Вторая производная
def DerivativeSecond(_x):
    return 6*_x-6
# Данные для вычесления точности k приблежения
k = 1
i = 0
m = DerivativeFirst(2)
#Рекурсионая функция, метод хорд
# _Funs - кортеж из функции, первой производной и
второй
# _a,_b - границы отрезка, содержащего корень
# _x - приблеженное значение, вычесленное аналитически
# _accur - точность вычислений
def Chords( _Funs,  _a,  _b,  _x, _accur):
    _newX = 0
    global i
    if _Funs[0](_b) * _Funs[2](_x) > 0:
        _newX = _x - ( ( _Funs[0](_x) * (b-_x) ) / (
_Funs[0](_b) - _Funs[0](_x) ) )
    else:
        _newX = _x - ( ( _Funs[0](_a) * (_x-_a) ) / (
_Funs[0](_x) - _Funs[0](_a) ) )
    if i == k:
        print("Точность приближения k=",k,":
",abs(_newX - _accur)," <= ",(abs(_Funs[0](_newX))/m))
    i+=1
    if abs(_newX - _x) > _accur:
        return Chords(_Funs,_a,_b,_newX,_accur)
    else:
        return _newX
#Вычисление конкретной функции
accuracy = 0.005
x0 = 1
a = 0.25
b = 2
```

```
xn =
Chords((Function,DerivativeFirst,DerivativeSecond),a,b
,x0,accuracy)
y = Function(xn)
print("x =",round(xn,3),"y(x) =",round(y,3))
```

ПРИЛОЖЕНИЕ F

```
import math
#Базовая функция
def Function(_x):
    return 2*_x**3-3*_x**2-12*_x-5
#Первая производная
def DerivateFirst(_x):
    return 6*_x**2 - 6*_x - 12
#Вторая производная
def DerivateSecond(_x):
    return 12*_x - 6
#Комбинированый метод касательных и хорд
# _Funs - кортеж из функции, первой производной и
второй
# _a,_b - границы отрезка, содержащего корень
# _accur - точность вычислений
def ChAndTangMethod(_Func,_a,_b,_accur):
    _newA = 0
    _newB = 0
    _x = (_a  + _b) / 2.0
    if _Func[1](_x) * _Func[2](_x) < 0:
        _newA = _a - _Func[0](_a) / _Func[1](_a)
        _newB = _b - (_Func[0](_b) * (_b - _a) ) /
(_Func[0](_b) - _Func[0](_a) )
    else:
        _newA = _a - (_Func[0](_a) * (_b - _a) ) /
(_Func[0](_b) - _Func[0](_a) )
        _newB = _b - _Func[0](_b) / _Func[1](_b)
    if abs(_newA - _newB) > _accur:
        return
ChAndTangMethod(_Func,_newA,_newB,_accur)
    else:
        return (_newA  + _newB) / 2.0
#Работа приложения
```

```
a = -1
b = 1.9
accuraty = 0.0005
x =
ChAndTangMethod((Function,DerivateFirst,DerivateSecond
),a,b,accuraty)
y = Function(x)
print("x =",round( x,5),"y(x) =",round(y,5))
```

ПРИЛОЖЕНИЕ G

```
import math
#Базовая функция
def Function(_x):
    return math.log(_x,math.e) + (_x + 1) ** 3
#Модифицированая функция для итерации
def IterModFunc(_x):
    return _x - ( Function( _x ) ) / ( ( 1 / _x ) + 3
* ( x + 1 ) ** 2)
#Итерациный метод
def IterRecursion(_func,_x,_accur):
    _newX = IterModFunc(_x)
    if abs(_newX - _x) > _accur:
        return IterRecursion(_func,_newX,_accur)
    else:
        return _newX
#Решение задание
x = 0.1
accuracy = 0.001
newX = IterRecursion(Function,x,accuracy)
print("x =",round( newX,4),"f(x)
=",round(Function(newX),4))
```

ПРИЛОЖЕНИЕ H

```
import math
#Базовая функция
def Function(_x):
    return _x**3+2*_x**2+2
#Модифицированая функция для итерации
def IterModFunc(_x):
    return _x - ( Function( _x ) ) / (3*_x**2+4*_x)
```

```
#Итерациный метод
def IterRecursion(_func,_x,_accur):
    _newX = IterModFunc(_x)
    if abs(_newX - _x) > _accur:
        return IterRecursion(_func,_newX,_accur)
    else:
        return _newX
#Решение задание
x = -2
accuracy = 0.005
newX = IterRecursion(Function,x,accuracy)

print("x =",round( newX,4),"f(x)
=",round(Function(newX),4))
```

ПРИЛОЖЕНИЕ I

```
import math, Matrix
import numpy as np
#Итерационный метод решения СЛАУ
def MethodDeterminatFunc(cBase,fBase,xNumber,accur):
    xNumberNext =
Matrix.MatrixMove(Matrix.Multiplication(cBase,xNumber)
,fBase,lambda x1, x2 : x1+x2)
    forAccuracy =
Matrix.MatrixMove(xNumberNext,xNumber,lambda
x1,x2:abs(x2 - x1))

    for j in range(0,forAccuracy.shape[0]):
        if forAccuracy[j][0] > accur:
            xNumberNext =
MethodDeterminatFunc(cBase,fBase,xNumberNext,accur)
            break

    return xNumberNext
def Result():
    A = np.array([[0,0.317,-0.061,-0.175],
                  [0.404,0,-0.072,0.003],
                  [-0.031,-0.028,0,0.731],
                  [-0.145,0.002,1.208,0]])
    B = np.array([[0.484], [-0.122], [-0.061],
```

```
                  [0.257]])
    xBase = np.array([[0.484], [-0.122], [-0.061],
                      [0.257]])
    accuracy = 0.01
    print("Матрица коофициентов:\n",A)
    print("Матрица свободных членов:\n",B)
    print("Начальное приближение:\n",xBase)
    result = MethodDeterminatFunc(A,B,xBase,accuracy)
    print("Решение:\n",result)
```

ПРИЛОЖЕНИЕ J

```
import math, Matrix,IterationSLY
import numpy as np
#Решение СЛАУ методом Зейделя
def Zeildel(aBase,bBase,xNumber,accur):
    xNumberNext =
np.zeros((xNumber.shape[0],xNumber.shape[1]))
    for i in range(0,xNumberNext.shape[0]):
        xNumberNext[i][0] = xNumber[i][0]

    for i in range(0,aBase.shape[0]):
        temp = 0
        for j in range(0,aBase.shape[1]):
            temp+=xNumberNext[j][0] * aBase[i][j]
        xNumberNext[i][0] = temp
    forAccuracy =
Matrix.MatrixMove(xNumberNext,xNumber,lambda
x1,x2:abs(x2 - x1))

    for j in range(0,forAccuracy.shape[0]):
        if forAccuracy[j][0] > accur:
            xNumberNext =
IterationSLY.MethodDeterminatFunc(aBase,bBase,xNumberN
ext,accur)
            break
    return xNumberNext
A = np.array([[0,0.405,-0.310],
                       [-0.667,0,-0.167],
                       [-0.125,-0.083,0]])
B = np.array([[0.667],
```

```
                       [0.667],
                       [-0.542]])
xBase = np.array([[0.667],
                       [0.667],
                       [-0.542]])
accuracy = 0.001
print("Матрица коофициентов:\n",A)
print("Матрица свободных членов:\n",B)
print("Начальное приближение:\n",xBase)
result = Zeildel(A,B,xBase,accuracy)
print("Решение:\n",result)
```

ПРИЛОЖЕНИЕ K

```
import numpy as np
import Matrix,math
#Итерационный метод
def IterSNAY(function,x,y,accur):
    xNew = function[0](x,y)
    yNew = function[1](x,y)
    newValue = (xNew,yNew)
    if abs(xNew - x) > accur or abs(yNew - y) > accur:
        newValue = IterSNAY(function,xNew,yNew,accur)

    return newValue
#Решение
x0 = 1
y0 = 1

func  = (lambda x,y: 1 - math.cos(y)/2.0, lambda x,y:
math.sin(x + 1) - 1.2)
accuracy = 0.0001
print("Решение СНАУ: ",IterSNAY(func,x0,y0,accuracy))
```

ПРИЛОЖЕНИЕ L

```
import math, Matrix
import numpy as np
#Вычисления массива функций
def DeterminatFunc(baseFunc,baseValue):
    newValue =
np.zeros((baseFunc.shape[0],baseFunc.shape[1]))
    for i in range(0,newValue.shape[0]):
```

```
        for j in range(0,newValue.shape[1]):
            newValue[i][j] =
baseFunc[i][j](baseValue[0],baseValue[1])
    return Matrix.Determinat(newValue)
#Метод задания матриц для задания 3.2
def Decision1(baseX, accur):
    ikobi = np.array([[lambda x,y: 2*x,     lambda
x,y:  2*y],
                              [lambda x,y: 3*x**2,
lambda x,y:  -1   ]])
    delta1 = np.array([[lambda x,y: x**2 + y**2 - 1,
lambda x,y:  2*y],
                                [lambda x,y: x**3 - y,
lambda x,y:  -1   ]])
    delta2 = np.array([[lambda x,y: 2*x,     lambda
x,y:  x**2 + y**2 - 1],
                                 [lambda x,y: 3*x**2,
lambda x,y:  x**3 - y  ]])
    return
RecursionDesicion(ikobi,delta1,delta2,baseX,accur)
RecursionDesicion(ikobi,delta1,delta2,baseX,accur)
#Рекурсионный метод итерационнго вычисления
def
RecursionDesicion(ikobi,delta1,delta2,baseX,accur):

    ikobiD = DeterminatFunc(ikobi,baseX)
    delta1D = DeterminatFunc(delta1,baseX)
    delta2D = DeterminatFunc(delta2,baseX)

    newX = (baseX[0]-delta1D/ikobiD,baseX[1]-
delta2D/ikobiD)
    if abs(newX[0] - baseX[0]) > accur or abs(newX[1]
- baseX[1]) > accur:
        newX =
RecursionDesicion(ikobi,delta1,delta2,newX,accur)
    return newX
x = 1
y = 1
ccuracy = 0.0001
```

```
print("Решения",Decision1((1,1),accuracy))
```

ПРИЛОЖЕНИЕ М

```
import numpy as np
import math as mt
# Интерполяционный многочлен Лангранжа
def Langrange(arrayX,arrayY,arrayNewX):
    arrayNew =
np.zeros((arrayX.shape[0],arrayX.shape[0]))
    arrayD = np.zeros((arrayX.shape[0]))
    arrayL = np.zeros((arrayX.shape[0]))
    arrayNewY = np.zeros((arrayNewX.shape[0]))
    for k in range(0,arrayNewX.shape[0]):
        for i in range(0,arrayNew.shape[1]):
            arrayD[i] = 1
            tempD = 1
            for j in range(0,arrayNew.shape[1]):
                if j == i:
                    arrayNew[i][j] = 0
                else:
                    arrayNew[i][j] = arrayX[i]-
arrayX[j]
                    arrayD[i] *= arrayNew[i][j]
                    tempD *= arrayNewX[k]-arrayX[j]
            arrayL[i] = arrayY[i] / arrayD[i]
            arrayNewY[k] += arrayL[i]*tempD
    return arrayNewY
# Интерполяционнный многочлен Ньютона второго порядка
def NewtonMethodTwo(arrayX,arrayY,valueX):
    h = arrayX[1]-arrayX[0]
    temp = arrayY[arrayY.shape[0]-1]
    q = (valueX - arrayX[arrayX.shape[0]-1])/h
    difTab =
np.zeros((arrayY.shape[0],arrayY.shape[0]))
    for i in range(0,arrayY.shape[0]):
        difTab[i][0] = arrayY[i]
    for j in range(1,difTab.shape[1]):
        for i in range(0,difTab.shape[0]-j):
```

```
            difTab[i][j] = difTab[i+1][j-1]-
difTab[i][j-1]
    def factorial (_n):
        if _n == 1:
            return 1
        else:
            return _n * factorial(_n-1)

    def coof  (_q,_n):
        result = 1.0/factorial(_n)
        for i in range(1,_n+1):
            result *= _q + i - 1
        return result
    i = difTab.shape[0] - 2
    j = 1
    while i >= 0:
        temp += coof(q,j) * difTab[i][j]
        j+=1
        i-=1
    return temp
if __name__ == "__main__":
   arrayY = np.array([1.172,1.179,1.182,1.186,1.189])
   arrayX = np.array([1.62,1.64,1.65,1.67,1.68])
   arrayNewX = np.array([1.63])
   resultArray = Langrange(arrayX,arrayY,arrayNewX)
   print(resultArray)
   resultArray =
NewtonMethodTwo(arrayX,arrayY,arrayNewX[0])
   print(resultArray)
```

ПРИЛОЖЕНИЕ N

```
import numpy as np
import math as mt
# Интерполяционнный многочлен Ньютона второго порядка
def NewtonMethodTwo(arrayX,arrayY,valueX):
    h = arrayX[1]-arrayX[0]
    temp = arrayY[arrayY.shape[0]-1]
    q = (valueX - arrayX[arrayX.shape[0]-1])/h
    difTab =
np.zeros((arrayY.shape[0],arrayY.shape[0]))
```

```
    for i in range(0,arrayY.shape[0]):
        difTab[i][0] = arrayY[i]
    for j in range(1,difTab.shape[1]):
        for i in range(0,difTab.shape[0]-j):
            difTab[i][j] = difTab[i+1][j-1]-
difTab[i][j-1]
    def factorial (_n):
        if _n == 1:
            return 1
        else:
            return _n * factorial(_n-1)
    def coof  (_q,_n):
        result = 1.0/factorial(_n)
        for i in range(1,_n+1):
            result *= _q + i - 1
        return result
    i = difTab.shape[0] - 2
    j = 1
    while i >= 0:
        temp += coof(q,j) * difTab[i][j]
        j+=1
        i-=1
    return temp
# Интерполяционнный многочлен Ньютона первого порядка
def NewtonMethodFirst(arrayX,arrayY,valueX):
    h = arrayX[1]-arrayX[0]
    temp = arrayY[0]
    q = (valueX - arrayX[0])/h
    difTab =
np.zeros((arrayY.shape[0],arrayY.shape[0]))
    for i in range(0,arrayY.shape[0]):
        difTab[i][0] = arrayY[i]
    for j in range(1,difTab.shape[1]):
        for i in range(0,difTab.shape[0]-j):
            difTab[i][j] = difTab[i+1][j-1]-
difTab[i][j-1]
    def factorial (_n):
        if _n == 1:
            return 1
```

```
        else:
            return _n * factorial(_n-1)
    def coof  (_q,_n):
        result = 1.0/factorial(_n)
        for i in range(1,_n+1):
            result *= _q - i + 1
        return result

    j = 1
    for i in range(0,difTab.shape[0]-1):
        temp += coof(q,j) * difTab[0][j]
        j+=1
    return temp
if __name__ == "__main__":
   arrayX = np.array([0,1.4,2.6,4])
   arrayY = np.array([1,0.180,0.043,0.006])
   arrayNewX = np.array([1.7])

   resultArray = Langrange(arrayX,arrayY,arrayNewX)
   print(resultArray)
```

ПРИЛОЖЕНИЕ О

```
import numpy as np
import math as mt
# Интерполяционнный многочлен Ньютона второго порядка
def NewtonMethodTwo(arrayX,arrayY,valueX):
    h = arrayX[1]-arrayX[0]
    temp = arrayY[arrayY.shape[0]-1]
    q = (valueX - arrayX[arrayX.shape[0]-1])/h
    difTab = 
np.zeros((arrayY.shape[0],arrayY.shape[0]))
    for i in range(0,arrayY.shape[0]):
        difTab[i][0] = arrayY[i]
    for j in range(1,difTab.shape[1]):
        for i in range(0,difTab.shape[0]-j):
            difTab[i][j] = difTab[i+1][j-1]-
difTab[i][j-1]
    def factorial (_n):
        if _n == 1:
            return 1
```

```
        else:
            return _n * factorial(_n-1)
    def coof  (_q,_n):
        result = 1.0/factorial(_n)
        for i in range(1,_n+1):
            result *= _q + i - 1
        return result
    i = difTab.shape[0] - 2
    j = 1
    while i >= 0:
        temp += coof(q,j) * difTab[i][j]
        j+=1
        i-=1
    return temp
# Интерполяционнный многочлен Ньютона первого порядка
def NewtonMethodFirst(arrayX,arrayY,valueX):
    h = arrayX[1]-arrayX[0]
    temp = arrayY[0]
    q = (valueX - arrayX[0])/h
    difTab =
np.zeros((arrayY.shape[0],arrayY.shape[0]))
    for i in range(0,arrayY.shape[0]):
        difTab[i][0] = arrayY[i]
    for j in range(1,difTab.shape[1]):
        for i in range(0,difTab.shape[0]-j):
            difTab[i][j] = difTab[i+1][j-1]-
difTab[i][j-1]
    def factorial (_n):
        if _n == 1:
            return 1
        else:
            return _n * factorial(_n-1)
    def coof  (_q,_n):
        result = 1.0/factorial(_n)
        for i in range(1,_n+1):
            result *= _q - i + 1
        return result
    j = 1
    for i in range(0,difTab.shape[0]-1):
```

```
        temp += coof(q,j) * difTab[0][j]
        j+=1
    return temp
if __name__ == "__main__":
   arrayX =
np.array([1.2,1.25,1.3,1.35,1.4,1.45,1.5,1.55,1.6,1.65
])
   arrayY =
np.array([3.32,3.491,3.669,3.857,4.055,4.263,4.482,4.7
12,4.953,5.203])
   arrayNewX = np.array([1.22])

   resultArray =
NewtonMethodTwo(arrayX,arrayY,arrayNewX[0])
   print(resultArray)

   resultArray =
NewtonMethodFirst(arrayX,arrayY,arrayNewX[0])
   print(resultArray)
```