МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по практической работе №6

по дисциплине «Вычислительная математика»

Тема: Интерполяционные и аппроксимационные формулы для равноотстоящих узлов

Студент гр. 0304 Крицын Д. Р.

Преподаватель Попова Е. В.

Санкт-Петербург

2021

Цель работы: исследование методов интерполяции и аппроксимации для равноотстоящих узлов с последующей реализацией на одном из языков программирования.

Основные теоретические положения.

Значения функции $f(x_i)=y_i$ заданы в точках $x_i=x_0+nh \quad i\in[0,\ldots,n]$, необходимо найти промежуточные значения.

Интерполяционный многочлен Лагранжа:

$$P_n(x)=\sum_{i=0}^{n} y_i\cdot L_n(x) \quad ,$$

где $L_n(x)$ — множитель Лагранжа;

$$L_n(x)=\frac{(x-x_0)\ \ldots\ (x-x_{i-1})(x-x_{i+1})\ \ldots\ (x-x_n)}{(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ \ldots\ (x_i-x_n)}=\prod_{k=0,k\neq i}^{n}\frac{(x-x_k)}{(x_i-x_k)} \quad .$$

Следовательно

$$P_n(x)=\sum_{i=0}^{n} y_i\left(\prod_{k=0,k\neq i}^{n}\frac{x-x_k}{x_i-x_k}\right).$$

Первый интерполяционный многочлен Ньютона. Точка интерполирования находится в начале таблицы.

Интерполирующий полином ищется в виде

$$P_n(x)=a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+\ \ldots\ +a_n(x-x_0)\ldots(x-x_{n-1}) \quad .$$

Построение многочлена сводится к определнию коэффициентов a_i. При записи коэффициентов пользуются конечным разностями.

Конечные разности первого порядка запишутся в виде:

$$\Delta y_0=y_1-y_0;$$
$$\Delta y_1=y_2-y_1;$$
$$\ldots$$
$$\Delta y_{n-1}=y_n-y_{n-1},$$

где y_i — значения функции при соответствующих значениях x_i.

Конечные разности второго порядка:

$$\Delta^2 y_0 = \Delta y_1 - \Delta y_0;$$
$$\Delta^2 y_1 = \Delta y_2 - \Delta y_1;$$
$$\ldots$$
$$\Delta^2 y_{n-2} = \Delta y_{n_1} - \Delta y_{n-2}.$$

Конечные разности высших порядков найдутся аналогично:

$$\Delta^k y_0 = \Delta^{k-1} y_1 - \Delta^{k-1} y_0;$$
$$\Delta^k y_1 = \Delta^{k-1} y_2 - \Delta^{k-1} y_1;$$
$$\ldots$$
$$\Delta^k y_{n-2} = \Delta^{k-1} y_{n-1} - \Delta^{k-1} y_{n-2}.$$

Общая формула для нахождения всех коэффициентов имеет вид:

$$a_i = \frac{\Delta^i y_0}{i!h^i},$$

где $i = 1 \ldots n$.

В результате

$$P_n(x) = y_0 + \frac{\Delta y_0}{1!h}(x - x_0) + \frac{\Delta^2 y_0}{2!h^2}(x - x_0)(x - x_1) + \ldots + \frac{\Delta^n y_0}{n!h^n}(x - x_0) \ldots (x - x_{n-1}).$$

Данный многочлен называют первым полиномом Ньютона.

<u>Второй интерполяционный многочлен Ньютона.</u>

Для нахождения значений функции в конце интервала интерполирования интерполяционный полином запишется в виде

$$P_n(x) = a_0 + a_1(x - x_n) + a_2(x - x_n)(x - x_{n-1}) + \ldots + a_n(x - x_n)(x - x_{n-1}) \ldots (x - x_1).$$

Формула для нахождения всех коэффициентов запишется как:

$$a_i = \frac{\Delta^i y_{n-1}}{i!h^i}.$$

Получим вторую интерполяционную формулу Ньютона:

$$P_n(X) = y_n + \frac{\Delta y_{n-1}}{h}(x - x_n) + \frac{\Delta^2 y_{n-2}}{2!h^2}(x - x_n)(x - x_{n-1}) + \frac{\Delta^3 y_{n-3}}{3!h^3}(x - x_n)(x - x_{n-1})(x - x_{n-2}) +$$
$$\ldots + \frac{\Delta^n y_0}{n!h^n}(x - x_n)(x - x_{n-1}) \ldots (x - x_1).$$

Аппроксимация функции. Необходимо найти эмпирическую формулу, значения которой при $x=x_i$ мало бы отличались от входных данных. Будет использоваться линейная аппроксимация, при которой данные описываются линейной зависимостью

$$P(x)=ax+b \quad .$$

Формулы для расчёта коэффициентов a и b определяются по методу наименьших квадратов:

$$F=\sum_{i=1}^{n}(y_i-ax_i-b)^2 \rightarrow min \quad .$$

$$\begin{cases} \dfrac{dF}{db}=-2\sum_{i=1}^{n}(y_i-a\,x_i-b)\cdot 1=0\,, \ \dfrac{dF}{da}=-2\sum_{i=1}^{n}(y_i-ax_i-b)\cdot x_i=0. \end{cases}$$

$$a=\frac{n\sum_{i=1}^{n}(x_i\,y_i)-\sum_{i=1}^{n}x_i\cdot\sum_{i=1}^{n}y_i}{n\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2},$$

$$b=\frac{\sum_{i=1}^{n}y_i-a\sum_{i=1}^{n}x_i}{n}.$$

Порядок выполнения работы.

1. Выбрать три точки, не входящие таблицу данных в начале, конце, по середине таблицы.

2. Составить необходимые подпрограммы-функции.

3. Составить головную программу, содержащую обращение к соответствующим подпрограммам и осуществляющую печать результатов.

4. Используя многочлен Лагранжа, найти приближенные значения функции. Подсчитать точные значения, абсолютную погрешность, относительную погрешность, верные и значащие цифры.

5. Используя многочлены Ньютона найти все, что перечислено в предыдущем пункте, дополнительно – в процентном соотношении точку приемлемого использования первого многочлена по сравнению со вторым многочленом при работе с точкой в начале таблицы и наоборот.

6. Составить таблицу конечных разностей до второго порядка. К одному из значений y_i прибавить погрешность (α), превосходящую допустимую. Показать, как при этом изменятся конечные разности большего порядка.

x_0	y_0	Δy_0	$\Delta^2 y_0$
x_1	y_1	Δy_1	$\Delta^2 y_1$
x_2	$y_2 + \alpha$	Δy_2	$\Delta^2 y_2$
x_3	y_3	Δy_3	$\Delta^2 y_3$
x_4	y_4	Δy_4	$\Delta^2 y_4$
x_5	y_5	Δy_5	—
x_6	y_6	—	—

7. При аппроксимировании функции использовать для нахождения новых значений те же три точки. Рассчитать все, что указано в пункте 4.

8. Составить сводную таблицу по результатам исследования.

Выполнение работы.

Функция $y = ctg\left(\sqrt{\sin x}\right)$.

x	**y**
1.5	0.64
1.7	0.65
1.9	0.68
2.1	0.75
2.3	0.85
2.5	1.02

Точки для проведения вычислений: 1.62, 2.03, 2.47.

1. *Теорема*. Для любой элементарной функции существует единственный интерполяционный многочлен.

   Доказательство. Интерполяционный многочлен имеет вид:

   $P_N(x) = c_0 + c_1 x + \ldots + c_{N-1} x^{N-1} + c_n x^N$. Используя таблицу с исходными данными, составим систему линейных уравнений:

$$c_0 + c_1 x_0 + \ldots + c_N x_0^N = P_N(x_0) = y_0,$$
$$\ldots$$
$$c_0 + c_1 x_N + \ldots + c_N x_N^N = P_N(x_N) = y_N.$$

В матричном представлении система имеет вид $A\boldsymbol{c} = \boldsymbol{y}$:

$$A = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^N \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_N & x_N^2 & \ldots & x_N^N \end{pmatrix},$$
$$\boldsymbol{c} = (c_0, \ldots, c_N)^T, \boldsymbol{y} = (y_0, \ldots, y_N)^T.$$

Система имеет ровно одно решение тогда и только тогда, когда $|A| \neq 0$. Определитель матрицы A является определителем Вандермонда: $|A| = \prod_{0 \le j \le i \le N} (x_i - x_j)$, который не равен нулю, когда $x_i \neq x_j \forall i \neq j$. Таким образом, коэффициенты c_0, …, c_N находятся однозначно, а значит, интерполяционный многочлен существует и является единственным.

2. Вычислим приближённое значение функции в первой точке x = 1.62, используя интерполяцию при помощи многочленов Лагранжа, Ньютона (с начала), Ньютона (с конца), а также аппроксимацию методом наименьших квадратов. Также найдём точное значение функции в данной точке, абсолютную и относительную погрешности для вычислений всеми этими четырьмя методами, значащие и верные цифры в полученных значениях. Результаты работы программы приведены на рисунке 1.

метод	x	точное y	прибл. y	Δy	δy	знач. цифры	верные цифры
мн. Лагранжа	1.62	0.642947819476	0.647555840000	0.004608	0.007167	64755583	64
мн. Ньютона с нач.	1.62	0.642947819476	0.647555840000	0.004608	0.007167	64755584	64
мн. Ньютона с кнц.	1.62	0.642947819476	0.647555840000	0.004608	0.007167	64755584	64
аппр. мет. наим. кв.	1.62	0.642947819476	0.625485714286	0.017462	0.027159	62548571	6

Рис. 1 — вычисление значения функции в точке x = 1.62

Видно, что точность всех методов интерполяции на порядок больше, чем точность метода линейной аппроксимации (0,7% относительной погрешности интерполяции против 2% аппроксимации), и поэтому верных цифр в результате аппроксимации функции в данной точке на одну меньше.

3. Аналогично пункту 2, вычислим значение функции в точке x = 2.03 всеми 3 методами интерполяции и методом линейной аппроксимации. Реузльтаты работы программы представлены на рисунке 2.

метод	x	точное y	прибл. y	Δy	δy	знач. цифры	верные цифры
мн. Лагранжа	2.03	0.719979063927	0.721837449062	0.001858	0.002581	72183744	72
мн. Ньютона с нач.	2.03	0.719979063927	0.721837449062	0.001858	0.002581	72183744	72
мн. Ньютона с кнц.	2.03	0.719979063927	0.721837449062	0.001858	0.002581	72183744	72
аппр. мет. наим. кв.	2.03	0.719979063927	0.776014285714	0.056035	0.077829	77601428	7

Рис. 2 — вычисление значения функции в точке x = 2.03

Количество верных цифр осталось таким же. Относительная погрешность линейной аппроксимации функции в данной точке снова более чем в 10 раз больше, чем относительная погрешность любого метода интерполяции (0,2% во всех случаях интерполяции, 7% в результате аппроксимации).

4. Аналогично пункту 3, вычислим значение функции в точке x = 2.47. Результаты вычислений даны на рисунке 3.

метод	x	точное y	прибл. y	Δy	δy	знач. цифры	верные цифры
мн. Лагранжа	2.47	0.993183965980	0.985339066563	0.007845	0.007899	98533906	98
мн. Ньютона с нач.	2.47	0.993183965980	0.985339066563	0.007845	0.007899	98533906	98
мн. Ньютона с кнц.	2.47	0.993183965980	0.985339066563	0.007845	0.007899	98533906	98
аппр. мет. наим. кв.	2.47	0.993183965980	0.937557142857	0.055627	0.056009	93755714	9

Рис. 3 — вычисление значения функции в точке x = 2.47

Как и в пункте 3, относительные погрешности всех методов интерполяции совпадают и примерно раны 0,79%, относительная погрешность линейной аппроксимации — 5%. Из-за этого верных цифр в результате аппроксимации только 1, а в методах интерполяции — 2.

5. Вычислим конечные разности первого и второго порядков до и после внесения погрешности в y_2. Результаты данных вычислений приведены на рисунке 4.

Конечные разности:

x	y*	Δy*	Δ^2y*
1.5	0.64	0.01	0.02
1.7	0.65	0.03	0.04
1.9	0.68	0.07	0.03
2.1	0.75	0.1	0.07
2.3	0.85	0.17	--
2.5	1.02	--	--

Конечные разности с добавленной погрешностью:

x	y*	Δy*	Δ^2y*
1.5	0.64	0.01	0.564
1.7	0.65	0.574	-1.048
1.9	1.224	-0.474	0.574
2.1	0.75	0.1	0.07
2.3	0.85	0.17	--
2.5	1.02	--	--

Рис. 4 — конечные разности до и после внесения погрешности α в y_2

Видно, что при изменении значения y_i изменяются на некоторые значения и разности $\Delta y, \Delta y_{i-1}$, а также и данные разности, но больших порядков. При этом чем больше порядок конечной разности, тем сильнее на разности этого порядка отразится внесённое изменение.

В данном случае внесение погрешности $\alpha = 0{,}8\, y_3^* = 0{,}544$ в значение функции при x = 1.9 изменяет конечные разности:

$$\Delta(\Delta y_3^*) = (y_4^* - (y_3^* + \alpha)) - (y_4^* - y_3^*) = -\alpha,$$

$$\Delta(\Delta y_2^*) = ((y_3^* + \alpha) - y_2^*) - (y_3^* - y_2^*) = \alpha;$$

$$\textit{для второй степени}:$$

$$\Delta(\Delta^2 y_3^*) = (\Delta y_4^* - (\Delta y_3^* - \alpha)) - (\Delta y_4^* - \Delta y_3^*) = \alpha,$$

$$\Delta(\Delta^2 y_2^*) = ((\Delta y_3^* - \alpha) - (\Delta y_2^* + \alpha)) - (\Delta y_3^* - \Delta y_2^*) = -2\alpha,$$

$$\Delta(\Delta^2 y_1^*) = ((\Delta y_2^* + \alpha) - \Delta y_1^*) - (\Delta y_2^* - \Delta y_1^*) = \alpha.$$

6. Составим сводную таблицу по результатам вычисления значений функции в точках в различных точках таблицы. Сравним результаты вычислений методами интерполяции многочленом Лагранжа и аппроксимации наименьших квадратов. Результаты приведены на рисунке 5.

```
              метод|   x|         точное y|        прибл. y|         Δy|          δy| знач. цифры|верные цифры
        мн. Лагранжа|1.62| 0.642947819476| 0.647555840000|  0.004608|  0.007167|    64755583|           64
        мн. Лагранжа|2.03| 0.719979063927| 0.721837449062|  0.001858|  0.002581|    72183744|           72
        мн. Лагранжа|2.47| 0.993183965980| 0.985339066563|  0.007845|  0.007899|    98533906|           98
аппр. мет. наим. кв.|1.62| 0.642947819476| 0.625485714286|  0.017462|  0.027159|    62548571|            6
аппр. мет. наим. кв.|2.03| 0.719979063927| 0.776014285714|  0.056035|  0.077829|    77601428|            7
аппр. мет. наим. кв.|2.47| 0.993183965980| 0.937557142857|  0.055627|  0.056009|    93755714|            9
```

Рис. 5 — сравнение методов интерполяции и аппроксимации

Видно, что интерполяция методом многочлена Лагранжа во всех выбранных точках даёт лучшее приближение функции, нежели чем линейная аппроксимация. Результат вычисления значения функции многочленом Лагранжа всё время имеет 2 верные цифры против одной у аппроксимации, и погрешность 0,18% - 0,78% против 0,5-1,7% у аппроксимации.

7. Модифицируем программу для рассчёта многочлена Лагранжа, а также коэффициентов многочленов Ньютона с конца и с начала:

```
Многочлен Лагранжа:
 ((x - 1.700000) / -0.200000) * ((x - 1.900000) / -0.400000) * ((x - 2.100000) / -0.600000) * ((x - 2.300000) / -0.800000) * ((x - 2.500000) / -1.000000)  * 0.640000 +
 ((x - 1.500000) / 0.200000) * ((x - 1.900000) / -0.200000) * ((x - 2.100000) / -0.400000) * ((x - 2.300000) / -0.600000) * ((x - 2.500000) / -0.800000)  * 0.650000 +
 ((x - 1.500000) / 0.400000) * ((x - 1.700000) / 0.200000) * ((x - 2.100000) / -0.200000) * ((x - 2.300000) / -0.400000) * ((x - 2.500000) / -0.600000)  * 0.680000 +
 ((x - 1.500000) / 0.600000) * ((x - 1.700000) / 0.400000) * ((x - 1.900000) / 0.200000) * ((x - 2.300000) / -0.200000) * ((x - 2.500000) / -0.400000)  * 0.750000 +
 ((x - 1.500000) / 0.800000) * ((x - 1.700000) / 0.600000) * ((x - 1.900000) / 0.400000) * ((x - 2.100000) / 0.200000) * ((x - 2.500000) / -0.200000)  * 0.850000 +
 ((x - 1.500000) / 1.000000) * ((x - 1.700000) / 0.800000) * ((x - 1.900000) / 0.600000) * ((x - 2.100000) / 0.400000) * ((x - 2.300000) / 0.200000)  * 1.020000


Коэффициенты многочлена Ньютона с начала:
0.640000 0.050000 0.250000 0.416667 -0.781250 2.083333

Коэффициенты многочлена Ньютона с конца:
0.640000 0.850000 0.875000 0.833333 1.302083 2.083333
```

Рис. 6 — рассчёт общего вида интерполяционных многочленов для данной функции и таблицы данных

Таким образом, многочлен Лагранжа имеет следующий вид:

$$
\begin{aligned}
f(x) = {} & \frac{x-1.7}{-0.2}\cdot\frac{x-1.9}{-0.4}\cdot\frac{x-2.1}{-0.6}\cdot\frac{x-2.3}{-0.8}\cdot\frac{x-2.5}{-1.0}\cdot 0.64 + \\
& \frac{x-1.5}{0.2}\cdot\frac{x-1.9}{-0.2}\cdot\frac{x-2.1}{-0.4}\cdot\frac{x-2.3}{-0.6}\cdot\frac{x-2.5}{-0.8}\cdot 0.65 + \\
& \frac{x-1.5}{0.4}\cdot\frac{x-1.7}{0.2}\cdot\frac{x-2.1}{-0.2}\cdot\frac{x-2.3}{-0.4}\cdot\frac{x-2.5}{-0.6}\cdot 0.68 + \\
& \frac{x-1.5}{0.6}\cdot\frac{x-1.7}{0.4}\cdot\frac{x-1.9}{0.2}\cdot\frac{x-2.3}{-0.2}\cdot\frac{x-2.5}{-0.4}\cdot 0.75 + \\
& \frac{x-1.5}{0.8}\cdot\frac{x-1.7}{0.6}\cdot\frac{x-1.9}{0.4}\cdot\frac{x-2.1}{0.2}\cdot\frac{x-2.5}{-0.2}\cdot 0.85 + \\
& \frac{x-1.5}{1.0}\cdot\frac{x-1.7}{0.8}\cdot\frac{x-1.9}{0.6}\cdot\frac{x-2.1}{0.4}\cdot\frac{x-2.3}{0.2}\cdot 1.02.
\end{aligned}
$$

Многочлен Ньютона с начала:

$$f(x)=0.64+0.05\cdot(x-1.5)+0.25\cdot(x-1.5)(x-1.7)$$
$$+0.416667\cdot(x-1.5)(x-1.7)(x-1.9)$$
$$-0.78125\cdot(x-1.5)(x-1.7)(x-1.9)(x-2.1)$$
$$+2.083333\cdot(x-1.5)(x-1.7)(x-1.9)(x-2.1)(x-2.3).$$

Многочлен Ньютона с конца:

$$f(x)=1.02+0.85\cdot(x-2.3)+0.875\cdot(x-2.3)(x-2.1)$$
$$+0.83333\cdot(x-2.3)(x-2.1)(x-1.9)$$
$$+1.302083\cdot(x-2.3)(x-2.1)(x-1.9)(x-1.7)$$
$$+2.083333\cdot(x-2.3)(x-2.1)(x-1.9)(x-1.7)(x-1.5).$$

8. Прямая в методе линейной аппроксимации также была рассчитана при помощи модификации программы:

```
Прямая в методе линейной аппроксимации:
0.367143x + 0.030714
```

$$f(x)=0.367143\,x+0.030714.$$

9. По входным данным и полученным коэффициентам прямой в методе линейной аппроксимации построим график (изображён на Рисунке 8). Видно, что из-за того, что прямая приближается к графику только в двух точках (по сравнению с 5 входными точками в методах интерполяции), линейная аппроксимация даёт точность хуже, чем любой метод интерполяции.

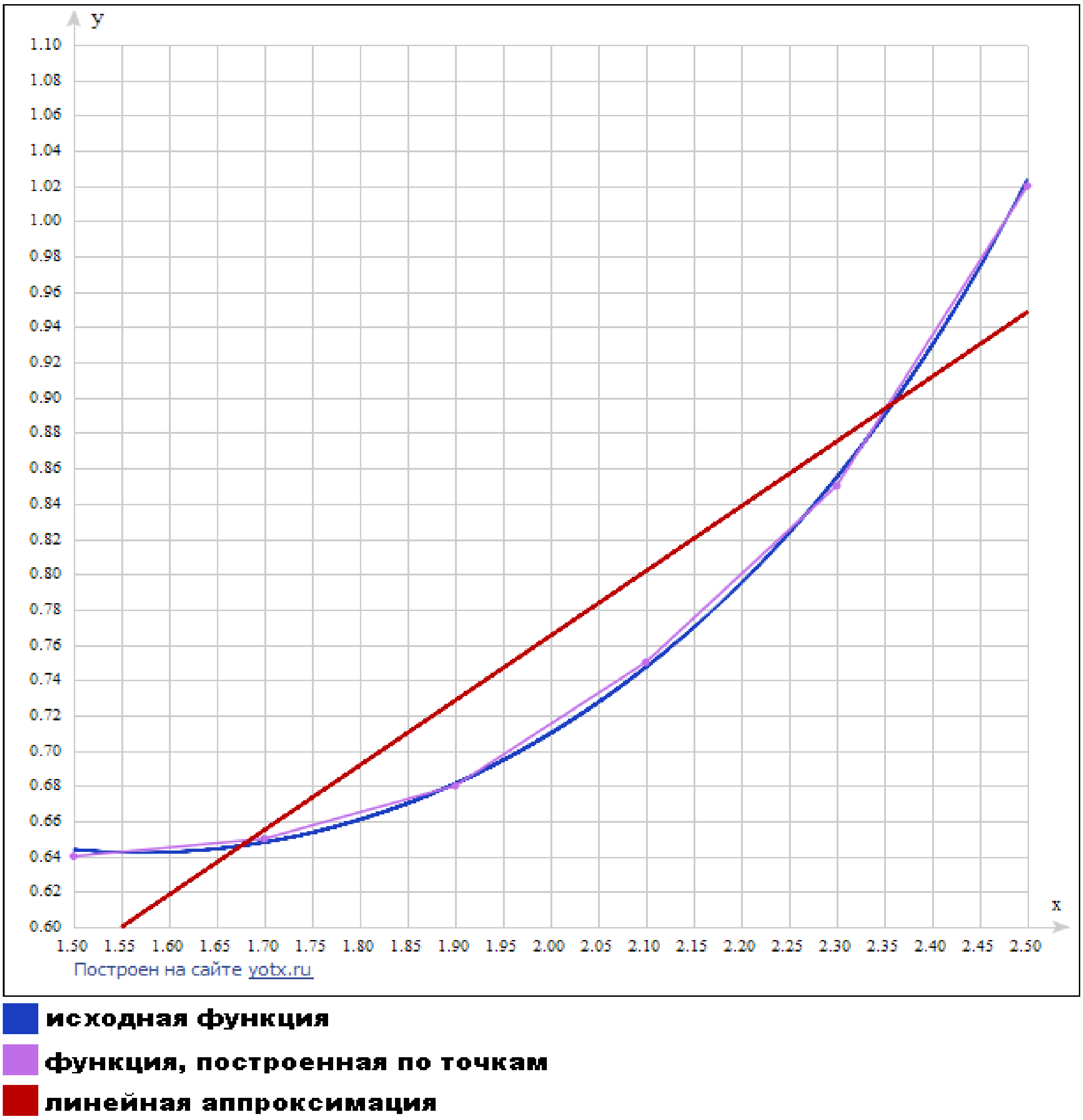

Рис. 8 — график функции, функции построенной по входным точкам, и аппроксимационной прямой

Выводы.

Были исследованы три методы интерполяции и один метод линейной аппроксимации функции для равноотстоящих узлов. Была написана программа для вычисления значения функции в точках, не принадлежащих таблице, используя интерполяционные многочлены Лагранжа и Ньютона, а также линейную аппроксимацию.

В ходе работы было доказано, что для функции, заданной в виде таблицы, все интерполяционные многочлены будут совпадать. Также было проверено, что с помощью интерполяционного многочлена функцию можно приблизить в целом точнее, чем с помощью линейной аппроксимации, полученной методом наименьших квадратов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

// Многочлен Лагранжа
double lagrange_mult(double x, int i,
              double* arr_x, size_t arr_x_sz)
{
    double ret = 1;
    double x_i = arr_x[i];
    for(size_t k = 0; k < arr_x_sz; ++k)
    {
         if(k == i) continue;
         ret *= (x - arr_x[k]) / (x_i - arr_x[k]);
    }
    return ret;
}

double interp_lagrange(double x,
              double* arr_x, double* arr_y, size_t arr_xy_sz)
{
    double y = 0;
    for(size_t i = 0; i < arr_xy_sz; ++i)
         y += arr_y[i] * lagrange_mult(x, i, arr_x, arr_xy_sz);
    return y;
}
void print_lagrange(double* arr_x, double* arr_y, size_t arr_xy_sz)
{
    for(size_t i = 0; i < arr_xy_sz; ++i){
         double x_i = arr_x[i];
         for(size_t k = 0; k < arr_xy_sz; ++k)
         {
              if(k == i) continue;
```

```
            printf("  ((x  -  %lf)  /  %lf)  %c",  arr_x[k],  (x_i  -
arr_x[k]),  (k  ==  arr_xy_sz  -  1)  ||  (i  ==  arr_xy_sz  -  1  &&  k  ==
arr_xy_sz - 2) ? ' ' : '*');
        }
        printf("* %lf %c \n", arr_y[i], i == arr_xy_sz - 1 ? ' ' :
'+');
    }
    puts("");
}

// Конечные разности
double** get_fdiffs(double* arr_y, size_t arr_y_sz)
{
    double** ret = malloc(sizeof(double*) * arr_y_sz);
    for(size_t i = 0; i < arr_y_sz; ++i)
        ret[i] = malloc(sizeof(double) * arr_y_sz);

    for(size_t n = 0; n < arr_y_sz; ++n)
        ret[0][n] = arr_y[n];
    for(size_t i = 1; i < arr_y_sz; ++i)
        for(size_t n = 0; n < arr_y_sz - i; ++n)
            ret[i][n] = ret[i-1][n+1] - ret[i-1][n];

    return ret;
}

// Многочлен Ньютона
// с начала:
double interp_newton_fw(double x, double** fdiffs,
            double* arr_x, double* arr_y, size_t arr_xy_sz)
{
    double y = arr_y[0];
    double denom = 1;
    double mult = 1;
    double h = arr_x[1] - arr_x[0];
    for(size_t i = 1; i < arr_xy_sz; ++i)
```

```
    {
        denom *= i * h;
        mult *= x - arr_x[i-1];
        y += fdiffs[i][0] * mult / denom;
    }
    return y;
}
void print_newton_fw(double** fdiffs,
              double* arr_x, double* arr_y, size_t arr_xy_sz)
{
    double denom = 1;
    double h = arr_x[1] - arr_x[0];
    printf("%lf ", fdiffs[0][0]);
    for(size_t i = 1; i < arr_xy_sz; ++i)
    {
        denom *= i * h;
        printf("%lf ", fdiffs[i][0] / denom);
    }
    puts("");
}
// с конца:
double interp_newton_bw(double x, double** fdiffs,
              double* arr_x, double* arr_y, size_t arr_xy_sz)
{
    double y = arr_y[arr_xy_sz-1];
    double denom = 1;
    double mult = 1;
    double h = arr_x[1] - arr_x[0];
    for(int i = arr_xy_sz - 2; i > -1; --i)
    {
        denom *= (arr_xy_sz - i - 1) * h;
        mult *= x - arr_x[i+1];
        y += fdiffs[arr_xy_sz - i - 1][i] * mult / denom;
    }
    return y;
}
```

```c
void print_newton_bw(double** fdiffs,
              double* arr_x, double* arr_y, size_t arr_xy_sz)
{
    double denom = 1;
    double h = arr_x[1] - arr_x[0];
    printf("%lf ", fdiffs[0][arr_xy_sz-1]);
    for(int i = arr_xy_sz - 2; i > -1; --i)
    {
         denom *= (arr_xy_sz - i - 1) * h;
         printf("%lf ", fdiffs[arr_xy_sz - i - 1][i] / denom);
    }
    puts("");
}

// Аппроксимация методом наименьших квадратов
void approx_coeffs(double* arr_x, double* arr_y, size_t arr_xy_sz,
              double* a, double* b)
{
    double x_sum = 0, xpow2_sum = 0, y_sum = 0, xy_sum = 0;
    size_t n = arr_xy_sz;

    for(size_t i = 0; i < arr_xy_sz; ++i)
    {
         x_sum += arr_x[i];
         y_sum += arr_y[i];
         xy_sum += arr_x[i] * arr_y[i];
         xpow2_sum += arr_x[i]*arr_x[i];
    }
    *a = (n * xy_sum - x_sum * y_sum) /
         (n * xpow2_sum - x_sum * x_sum);
    *b = (y_sum - (*a) * x_sum) / n;
}
double approx_result(double x, double a, double b)
{
    return a*x + b;
}
```

```
// Значащие и верные цифры
void sign_corr_digits(double approx_y, double exact_y,
              int* sign, int* corr)
{
    int isign, icorr;
    size_t digits = 8;
    int correct_cnt = 0;
    double inacc = fabs(exact_y - approx_y);
    for(size_t i = 0; i < digits; ++i)
    {
         if(correct_cnt == 0 && inacc < 1)
              inacc *= 10;
         else if(correct_cnt == 0)
              correct_cnt = i;

         approx_y *= 10;
         exact_y *= 10;
    }

    isign = (int)approx_y; icorr = (int)exact_y;
    int digit_cnt = 0;
    { int _isign = isign; while(_isign > 0) { ++digit_cnt; _isign /=
10; } }
    int ch_c = isign / pow(10, digit_cnt - correct_cnt);
    int ch_e = icorr / pow(10, digit_cnt - correct_cnt);
    if(ch_c == ch_e)
         icorr = ch_c;
    else
         icorr = isign / pow(10, digit_cnt - correct_cnt + 1);
    *sign = isign;
    *corr = icorr;
}

double f(double x) { return 1 / tan( pow(sin(x), 1./2) ); }
```

```
int main()
{
    const int arr_xy_sz = 6;
    double arr_x[6] = {1.5, 1.7, 1.9, 2.1, 2.3, 2.5};
    double arr_y[6] = {0.64, 0.65, 0.68, 0.75, 0.85, 1.02};

    double** fdiffs = get_fdiffs(arr_y, arr_xy_sz);
    double approx_a, approx_b;
    approx_coeffs(arr_x, arr_y, arr_xy_sz, &approx_a, &approx_b);

    double calc_x[3] = {1.62, 2.03, 2.47};

    // Вычисления значений для 4 методов в 3 точках
    printf("%25s|%4s|%21s|%20s|%11s|%11s|%21s|%12s\n",
        "метод", "x", "точное y", "прибл. y", "Δy", "δy", "знач.
цифры", "верные цифры");
    for(size_t i = 0; i < 3; ++ i)
    {
        double exact = f(calc_x[i]);
        double y = interp_lagrange(calc_x[i], arr_x, arr_y,
arr_xy_sz);
        double dy_abs = fabs(exact - y);
        double dy_rel = fabs(dy_abs / exact);
        int sign, corr; sign_corr_digits(y, exact, &sign, &corr);
        printf("%30s|%4lg|%15.12lf|%15.12lf|%10lf|%10lf|%12d|%12d\
n",
            "мн. Лагранжа",
            calc_x[i], exact, y, dy_abs, dy_rel, sign, corr);

        y = interp_newton_fw(calc_x[i], fdiffs, arr_x, arr_y,
arr_xy_sz);
        dy_abs = fabs(exact - y);
        dy_rel = fabs(dy_abs / exact);
        sign_corr_digits(y, exact, &sign, &corr);
        printf("%33s|%4lg|%15.12lf|%15.12lf|%10lf|%10lf|%12d|%12d\
n",
```

```
            "мн. Ньютона с нач.",
            calc_x[i], exact, y, dy_abs, dy_rel, sign, corr);

        y = interp_newton_bw(calc_x[i], fdiffs, arr_x, arr_y,
arr_xy_sz);
        dy_abs = fabs(exact - y);
        dy_rel = fabs(dy_abs / exact);
        sign_corr_digits(y, exact, &sign, &corr);
        printf("%33s|%4lg|%15.12lf|%15.12lf|%10lf|%10lf|%12d|%12d\
n",
            "мн. Ньютона с кнц.",
            calc_x[i], exact, y, dy_abs, dy_rel, sign, corr);

        y = approx_result(calc_x[i], approx_a, approx_b);
        dy_abs = fabs(exact - y);
        dy_rel = fabs(dy_abs / exact);
        sign_corr_digits(y, exact, &sign, &corr);
        printf("%30s|%4lg|%15.12lf|%15.12lf|%10lf|%10lf|%12d|%12d\
n",
            "аппр. мет. наим. кв.",
            calc_x[i], exact, y, dy_abs, dy_rel, sign, corr);
    }

    // Таблицы конечных разностей
    printf("\nКонечные разности:\n");
    printf("%10s|%10s|%11s|%14s\n", "x", "y*", "Δy*", "Δ^2y*");
    for(size_t i = 0; i < arr_xy_sz; ++i)
    {
        if(arr_xy_sz - i > 2)
            printf("%10.5lg|%10.5lg|%10.5lg|%13.5lg\n",
                arr_x[i], fdiffs[0][i], fdiffs[1][i], fdiffs[2]
[i]);
        else if(arr_xy_sz - i == 1)
            printf("%10.5lg|%10.5lg|%10s|%13s\n",
                arr_x[i], fdiffs[0][i], "--", "--");
        else
```

```
            printf("%10.5lg|%10.5lg|%10.5lg|%13s\n",
                arr_x[i], fdiffs[0][i], fdiffs[1][i], "--");
    }

    double prev_arr_y2 = arr_y[2];
    arr_y[2] *= 1.8;
    double** _fdiffs = get_fdiffs(arr_y, arr_xy_sz);
    printf("\nКонечные разности с добавленной погрешностью:\n");
    printf("%10s|%10s|%11s|%14s\n", "x", "y*", "Δy*", "Δ^2y*");
    for(size_t i = 0; i < arr_xy_sz; ++i)
    {
        if(arr_xy_sz - i > 2)
            printf("%10.5lg|%10.5lg|%10.5lg|%13.5lg\n",
                arr_x[i], _fdiffs[0][i], _fdiffs[1][i], _fdiffs[2]
[i]);
        else if(arr_xy_sz - i == 1)
            printf("%10.5lg|%10.5lg|%10s|%13s\n",
                arr_x[i], _fdiffs[0][i], "--", "--");
        else
            printf("%10.5lg|%10.5lg|%10.5lg|%13s\n",
                arr_x[i], _fdiffs[0][i], _fdiffs[1][i], "--");
    }
    arr_y[2] = prev_arr_y2;

    // Сводная таблица (сравнение интерполяции и аппроксимации)
    printf("\n%25s|%4s|%21s|%20s|%11s|%11s|%21s|%12s\n",
        "метод", "x", "точное y", "прибл. y", "Δy", "δy", "знач.
цифры", "верные цифры");
    for(size_t i = 0; i < 3; ++i)
    {
        double exact = f(calc_x[i]);
        double y = interp_lagrange(calc_x[i], arr_x, arr_y,
arr_xy_sz);
        double dy_abs = fabs(exact - y);
        double dy_rel = fabs(dy_abs / exact);
        int sign, corr; sign_corr_digits(y, exact, &sign, &corr);
```

```
        printf("%30s|%4lg|%15.12lf|%15.12lf|%10lf|%10lf|%12d|%12d\
n",
            "мн. Лагранжа",
            calc_x[i], exact, y, dy_abs, dy_rel, sign, corr);
    }
    for(size_t i = 0; i < 3; ++i)
    {
        double exact = f(calc_x[i]);
        double y = approx_result(calc_x[i], approx_a, approx_b);
        double dy_abs = fabs(exact - y);
        double dy_rel = fabs(dy_abs / exact);
        int sign, corr; sign_corr_digits(y, exact, &sign, &corr);
        printf("%30s|%4lg|%15.12lf|%15.12lf|%10lf|%10lf|%12d|%12d\
n",
            "аппр. мет. наим. кв.",
            calc_x[i], exact, y, dy_abs, dy_rel, sign, corr);
    }

    // вывод коэффициентов:
    printf("\n\nМногочлен Лагранжа:\n");
    print_lagrange(arr_x, arr_y, arr_xy_sz);
    printf("\nКоэффициенты многочлена Ньютона с начала:\n");
    print_newton_fw(fdiffs, arr_x, arr_y, arr_xy_sz);
    printf("\nКоэффициенты многочлена Ньютона с конца:\n");
    print_newton_bw(fdiffs, arr_x, arr_y, arr_xy_sz);
    printf("\nПрямая в методе линейной аппроксимации:\n");
    printf("%lfx + %lf\n", approx_a, approx_b);
}
```