МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и прикладной математики

Кафедра прикладной математики и экономико-математических методов

КУРСОВАЯ РАБОТА

по дисциплине:

«Численные методы»

Тема: «МЕТОД НАИМЕНЬШИХ КВАДРАТОВ»

Направление: 01.03.02 Прикладная математика и информатика

Студент: Попова Софья Ивановна

Группа: ПМ-1901 Подпись:__________________

Проверил: Хазанов Владимир Борисович

Должность: д. ф-м. н., профессор

Оценка:____________________________ Дата:___________________

Подпись:_______________________

Санкт-Петербург

2021

Оглавление

ВВЕДЕНИЕ

Данная курсовая работа содержит описание методов решения некоторых задач численного анализа, их программную реализацию, решение тестовых задач и анализ полученных результатов.

Цели работы:

- Изучить и продемонстрировать принцип работы определённых численных методов алгебры и анализа на языках программирования Wolfram Mathematica.
- Изучить и сделать выводы о целесообразности использования метода наименьших квадратов для построения наилучшего приближения.

Задачи:

- Определить необходимую теоретическую основу метода наименьших квадратов и других методов.
- Реализовать метод наименьших квадратов и прочие методы на языке программированияWolfram Mathematica.
- Проверить корректность работы алгоритмов.

1. МЕТОД КВАДРАТНЫХ КОРНЕЙ

1.1 Описание метода

Пусть дана симметричная система линейных уравнений в матричном виде: Ax=b

К её решению может быть применена идея разложения матрицы А в произведение двух матриц специального вида. Основанием для этого служит следующая теорема:

Теорема. Какова бы ни была матрица А с отличными от нуля главными минорами:

$$a_{11} \neq 0,\ \begin{vmatrix} a_{11}a_{12} \\ a_{21}a_{22} \end{vmatrix} \neq 0,\ \dots,\ \begin{vmatrix} a_{11}\dots a_{1n-1} \\ \dots\dots\dots\dots \\ a_{n-11}\dots a_{n-1n-1} \end{vmatrix} \neq 0$$

ее всегда можно разложить в произведение двух треугольных матриц ВС,

где В - левая треугольная матрица:

$$B=\begin{bmatrix} \beta_{11} & 0 & \dots & 0 \\ \beta_{21} & \beta_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \beta_{n1} & \beta_{n2} & \dots & \beta_{nn} \end{bmatrix}$$

С - правая треугольная матрица:

$$C=\begin{bmatrix} \gamma_{11} & \gamma_{12} & \dots & \gamma_{1n} \\ 0 & \gamma_{22} & \dots & \gamma_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \gamma_{nn} \end{bmatrix}$$

Так как данная матрица симметрична, то она раскладывается на произведение двух взаимно транспонированных треугольных матриц:

A = T′ T

$$где \quad T' = \begin{pmatrix} t_{11} & 0 & \dots & 0 \\ t_{12} & t_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ t_{1n} & t_{2n} & \dots & t_{nn} \end{pmatrix}, T = \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ 0 & t_{22} & \dots & t_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & t_{nn} \end{pmatrix}$$

Найдем элементы tij матрицы T. Для этого перемножим T и T’ между собой и приравняем полученное к исходной матрице:

Получим следующие формулы для определения tij:

$$\begin{cases} t_{11} = \sqrt{a_{11}}, \\ t_{1j} = \dfrac{a_{1j}}{t_{11}}, при(j > 1), \\ t_{ii} = \sqrt{a_{ii} - \sum\limits_{k=1}^{i-1} t_{ki}^2}, при(1 < i \le n), \\ t_{ij} = \dfrac{a_{ij} - \sum\limits_{k=1}^{i-1} t_{ki} t_{kj}}{t_{ij}}, при(i < j), \\ t_{ij} = 0, при(i > j). \end{cases}$$

Далее, решение системы сводится к решению двух треугольных систем. Действительно, равенство равносильно двум равенствам:

y=b и Tx=y.

Запишем в развернутом виде системы:

$$\begin{cases} t_{11} y_1 = b_1, \\ t_{12} y_1 + t_{22} y_2 = b_2, \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots \\ t_{1n} y_1 + t_{2n} y_2 + \dots + t_{nn} y_n = b_n, \end{cases}$$

$$\begin{cases} t_{11}x_1 + t_{12}x_2 + \ldots + t_{1n}x_n = y_1, \\ t_{22}x_2 + \ldots + t_{2n}x_n = y_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ t_{nn}x_n = y_n. \end{cases}$$

И из этих систем последовательно находим:

$y_1 = \dfrac{b_1}{t_{11}}$, $y_i = \dfrac{b_i - \sum_{k=1}^{i-1} t_{ki} y_k}{t_{ii}}$, при (i>1)

$x_1 = \dfrac{y_n}{t_{nn}}$, $x_i = \dfrac{y_i - \sum_{k=i+1}^{n} t_{ik} x_k}{t_{ii}}$, при (i<n).

1.2 Программная реализация

```
Clear[A, b];
очистить

metodHolechkogo[A_, b_] := Module[{x, A21 = A, y , s1, s2}, {s1, s2} = Dimensions[A];
                           программный модуль                  размеры массива

  Do[A21[[i, j]] = 0;
  оператор цикла

   Which[ i > j, A21[[i, j]] = (A[[i, j]] - Sum[A21[[j, o]] * A21[[i, o]], {o, 1, j-1}]) / A21[[j, j]], i == j, A21[[i, j]] = Sqrt[(A[[i, j]] - Sum[A21[[i, o]] * A21[[i, o]], {o, 1, j-1}])]],
   условный оператор с множественными ветвями

   {i, 1, s1}, {j, 1, s2}];

  y = LinearSolve[A21, b ]; x = LinearSolve[Transpose[A21], y]; x // MatrixForm]
      решить линейные уравнения  решить лине···  транспозиция         матричная форма

metodHolechkogo[{{1.00, 0.42, 0.54, 0.66}, {0.42, 1.00, 0.32, 0.44}, {0.54, 0.32, 1.00, 0.22}, {0.66, 0.44, 0.22, 1.00}},
 {{0.3}, {0.5}, {0.7}, {0.9}}]
```

$$\begin{pmatrix} -1.25779 \\ 0.0434873 \\ 1.03917 \\ 1.48239 \end{pmatrix}$$

Рисунок 1 — Реализация метода Холецкого

1.3 Анализ результатов

Входные представлены на рисунке. (см. рисунок 2)

№	
6	$\mathbf{f} = \begin{pmatrix} 0.3 \\ 0.5 \\ 0.7 \\ 0.9 \end{pmatrix}$
3	$A = \begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix}$

Рисунок 2 — Входные данные

Результат работы программы представлен на рисунке. (см. рисунки 3,4)

```
x = metodHolechkogo[A, f]
{{-1.25779}, {0.0434873}, {1.03917}, {1.48239}}

Print["Вектор невязки =", f - A.x]
печатать
Вектор невязки ={{-1.66533×10^-16}, {0.}, {0.}, {-1.11022×10^-16}}
```

Рисунок 3 — Проверка работы программы 1

Проверка:

Ответ:

$$x_1 = \frac{-83315}{66239}$$
$$x_2 = \frac{25925}{596151}$$
$$x_3 = \frac{206500}{198717}$$
$$x_4 = \frac{883730}{596151}$$

Система уравнений:

$$\begin{cases} 1\,x_1 + 0{,}42\,x_2 + 0{,}54\,x_3 + 0{,}66\,x_4 = 0{,}3 \\ 0{,}42\,x_1 + 1\,x_2 + 0{,}32\,x_3 + 0{,}44\,x_4 = 0{,}5 \\ 0{,}54\,x_1 + 0{,}32\,x_2 + 1\,x_3 + 0{,}22\,x_4 = 0{,}7 \\ 0{,}66\,x_1 + 0{,}44\,x_2 + 0{,}22\,x_3 + 1\,x_4 = 0{,}9 \end{cases}$$

Общее решение: $X = \begin{pmatrix} \frac{-83315}{66239} \\ \frac{25925}{596151} \\ \frac{206500}{198717} \\ \frac{883730}{596151} \end{pmatrix}$

Рисунок 4 — Проверка работы программы 2

2 РЕШЕНИЕ СЛАУ МЕТОДОМ (ПОСЛЕДОВАТЕЛЬНЫХ ПРИБЛИЖЕНИЙ) ПРОСТЫХ ИТЕРАЦИЙ

2.1 Описание метода

При большом числе уравнений прямые методы решения СЛАУ (за исключением метода прогонки) становятся труднореализуемыми на ЭВМ прежде всего из-за сложности хранения и обработки матриц большой размерности. В то же время характерной особенностью ряда часто встречающихся в прикладных задачах СЛАУ является разреженность матриц.

Число ненулевых элементов таких матриц мало по сравнению с их размерностью. Для решения СЛАУ с разреженными матрицами предпочтительнее использовать итерационные методы.

Методы последовательных приближений, в которых при вычислении

последующего приближения решения используются предыдущие, уже известные приближенные решения, называются итерационными.

Рассмотрим СЛАУ (1) с невырожденной матрицей (det A≠0). Приведем СЛАУ к эквивалентному виду

$$\begin{cases} x_1 = \beta_1 + \alpha_{11}x_1 + \alpha_{12}x_2 + \ldots + \alpha_{1n}x_n \\ x_2 = \beta_2 + \alpha_{21}x_1 + \alpha_{22}x_2 + \ldots + \alpha_{2n}x_n \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = \beta_n + \alpha_{n1}x_1 + \alpha_{n2}x_2 + \ldots + \alpha_{nn}x_n \end{cases} \qquad (2)$$

Такое приведение может быть выполнено различными способами. Одним из наиболее распространенных является следующий:

Разрешим систему (1) относительно неизвестных при ненулевых диагональных элементах, $a_{ii} \neq 0, i=1,n$ (если какой-либо коэффициент на главной диагонали равен нулю, достаточно соответствующее уравнение поменять местами с любым другим уравнением). Получим следующие выражения для компонентов: вектора β и матрицы α эквивалентной системы:

$$\beta_1 = \frac{b_1}{a_n};\ \alpha_{ij} = -\frac{a_{ij}}{a_{ii}}, i,j = \overline{1,n}, i \neq j; a_{ij} = 0, i = j, i = \overline{1,n} \quad (3)$$

При таком способе приведения исходной СЛАУ к эквивалентному виду метод простых итераций носит название метода Якоби. В качестве нулевого приближения вектора неизвестных примем вектор правых частей $x_0 = \beta$ или $(x_1^0\ x_{2...}^0 x_n^0)^T = (\beta_1\ \beta_2\ ...\beta_n)^T$

Тогда метод простых итераций примет вид:

$$\begin{cases} x^0 = \beta \\ x^1 = \beta + \alpha x^0 \\ x^2 = \beta + \alpha x^1 \\ \dots\dots\dots\dots\dots\dots \\ x^k = \beta + \alpha x^{(k-1)} \end{cases} \quad (4)$$

Из (4) видно преимущество итерационных методов по сравнению, например, с рассмотренным выше методом Гаусса. В вычислительном процессе участвуют только произведения матрицы на вектор, что позволяет работать только с ненулевыми элементами матрицы, значительно упрощая процесс хранения и обработки матриц. Имеет место следующее достаточное условие сходимости метода простых итераций [1]. Метод простых итераций (4) сходится к единственному решению СЛАУ (2) (а следовательно и к решению исходной СЛАУ (1)) при любом начальном приближении x^0, если какая-либо норма матрицы α эквивалентной системы меньше единицы. $|a| < 1$. Если используется метод Якоби (выражения (3) для эквивалентной СЛАУ), то достаточным условием сходимости

$$|a_{ii}| > \sum_{j=1, i\neq j}^{n} |a_{ij}|$$

является диагональное преобладание матрицы А, т.е. (для каждой строки матрицы А модули элементов, стоящих на главной диагонали, больше суммы модулей недиагональных элементов). Очевидно, что в этом случае $|\alpha|$ ε меньше единицы и, следовательно, итерационный процесс (4) сходится. Приведем также необходимое и достаточное условие сходимости метода простых итераций**.** Для сходимости итерационного процесса необходимо и достаточно, чтобы спектр матрицы α эквивалентной системы лежал внутри круга с радиусом, равным единице.

При выполнении достаточного условия сходимости оценка погрешности решения на k - ой итерации дается выражением:

$$\left|x^k - x^*\right| \le \varepsilon^k = \frac{|\alpha|}{1-|\alpha|}\left|x^k - x^{k-1}\right|, \quad (5)$$

Где x* - точное решение СЛАУ. *

Процесс итераций останавливается при выполнении условия $\varepsilon^{(k)} \le \varepsilon$, где ε задаваемая вычислителем точность. Принимая во внимание, что из (5) следует неравенство

$$\left|x^k - x^*\right| \le \frac{|\alpha|^k}{1-|\alpha|}\left|x^1 - x^0\right|$$

можно получить априорную оценку необходимого для достижения заданной точности числа итераций. При использовании в качестве начального приближения вектора β такая оценка определится неравенством: $\frac{|\alpha|^{k+1}}{1-|\alpha|}|\beta| \le \varepsilon$ откуда получаем априорную оценку числа итераций k при $|\alpha| < 1$

$$k + 1 \ge \frac{lg\varepsilon - lg|\beta| + lg\,(1-|\alpha|)}{lg|\alpha|}$$

Следует подчеркнуть, что это неравенство дает завышенное число итераций k, поэтому редко используется на практике.

Замечание. Поскольку $\|a\| < 1$ является только достаточным (не необходимым) условием сходимости метода простых итераций, то итерационный процесс может сходиться и в случае, если оно не выполнено. Тогда критерием окончания итераций может служить неравенство:

$$\left|x^k + x^{k-1}\right| \le \varepsilon.$$

2.2 Программная реализация

```
Clear[MetodPI];
MetodPI[A_ , f_ , k_] := Module[{x, i1, i = Abs@Eigenvalues[A]}, x = ConstantArray[0, Length[A]];
  i1 = (Max[i] + Min[i]) / 2;
  Do[If[i[[p]] ≥ 1, A = i1 * A; f = i1 * f; Break[]], {p, Length@A}];
  Do[Do[x[[j]] = (f[[j, 1]] - Sum[A[[j, i]] * x[[i]], {i, Length[A]}] + A[[j, j]] * x[[j]]) / A[[j, j]], {j, 4}],
   {u, k}];
  x // MatrixForm]
```

Рисунок 5 — Реализация метода простых итераций

2.3 Анализ результатов

Входные данные представлены на рисунке 6.

```
A = {{1.00, 0.42, 0.54, 0.66}, {0.42, 1.00, 0.32, 0.44}, {0.54, 0.32, 1.00, 0.22},
   {0.66, 0.44, 0.22, 1.00}} // MatrixForm
                               матричная форма
f = {{0.3}, {0.5}, {0.7}, {0.9}} // MatrixForm
                                    матричная форма
```

$$\begin{pmatrix} 1. & 0.42 & 0.54 & 0.66 \\ 0.42 & 1. & 0.32 & 0.44 \\ 0.54 & 0.32 & 1. & 0.22 \\ 0.66 & 0.44 & 0.22 & 1. \end{pmatrix}$$

$$\begin{pmatrix} 0.3 \\ 0.5 \\ 0.7 \\ 0.9 \end{pmatrix}$$

Рисунок 6 — Входные данные

Результат работы программы представлен на рисунках (см. рисунки 7-10)

```
MetodPI[A, f, 10]
```

$$\begin{pmatrix} -1.25283 \\ 0.0458626 \\ 1.03741 \\ 1.47846 \end{pmatrix}$$

Рисунок 7 — Пример работы программы

```
Grid[{{"A", , "x", , "f"}, {MatrixForm@A, "*", MatrixForm@MetodPI[A, f, 10], "=", MatrixForm@f}, {"", "", "", "", ""},
таблица                      матричная форма      матричная форма                   матричная форма
  {"Количество итераций", "", "Приближение", "", ""}, {"10", "", "10^-3", "", ""}}]
```

$$\begin{array}{ccccc} A & & x & & f \\ \begin{pmatrix} 1. & 0.42 & 0.54 & 0.66 \\ 0.42 & 1. & 0.32 & 0.44 \\ 0.54 & 0.32 & 1. & 0.22 \\ 0.66 & 0.44 & 0.22 & 1. \end{pmatrix} & * & \begin{pmatrix} -1.25283 \\ 0.0458626 \\ 1.03741 \\ 1.47846 \end{pmatrix} & = & \begin{pmatrix} 0.3 \\ 0.5 \\ 0.7 \\ 0.9 \end{pmatrix} \end{array}$$

Количество итераций — 10

Приближение — 10^{-3}

Рисунок 8 — Пример работы программы

```
LinearSolve[A, f] // MatrixForm
решить линейные уравн⋯ матричная форма
```

$$\begin{pmatrix} -1.25779 \\ 0.0434873 \\ 1.03917 \\ 1.48239 \end{pmatrix}$$

Рисунок 9 — Пример работы встроенной функции

```
x = MetodPI[A, f, 10]

{-1.25283, 0.0458626, 1.03741, 1.47846}

f - A.x

{{-0.00241692}, {-0.0021663}, {-0.000816035}, {0.}}
```

Рисунок 10 — Вектор невязки

Полученный результат совпадает с встроенной функцией Wolfram Mathematica LinearSolve[A,f]

3 РЕШЕНИЕ НЕЛИНЕЙНОГО УРАВНЕНИЯ С ПОМОЩЬЮ ПАДЕ-АППРОКСИМАЦИИ

3.1 Описание метода

Аппроксимация Паде представляет собой рациональную функцию вида:

$$[L/M] = \frac{a_0 + a_1 z + \ldots + a_L z^L}{b_0 + b_1 z + \ldots + b_M z^M},$$

Нелинейное уравнение решалось с помощью формул:

Методы Паде-аппроксимации

В основе методов лежат два вида Паде-аппроксимации функции $x = g(y)$ обратной к функции $y = f(x)$.

2.1.10. $x = [0/1]_g(y)$ **2.1.11. Метод Галлея (Хэлли)** $x = [1/1]_g(y)$

x_0 – начальное приближение

$f_k = f(x_k),\ f'_k = f'(x_k),\ f''_k = f''(x_k)$

$t_k = -\dfrac{f_k}{f'_k}$ $\quad r_k = \dfrac{f''_k}{f'_k} t_k^2$

$$x_{k+1} = \frac{x_k^2}{x_k - t_k},\ k = 1,2,3,\ldots$$

$$x_{k+1} = x_k + \frac{t_k^2}{t_k + \frac{1}{2} r_k},\ k = 1,2,3,\ldots$$

Рисунок 11 — Необходимые формулы

3.2 Программная реализация

На рисунках представлена программная реализация метода для системы нелинейных уравнений. (см. рисунок 12)

```
Clear[Programm];
очистить

Programm[f_, k_] := Module[{x = {k}, n = 100, i = 1}, For[i = 1, i ≤ n, i++, x = Append[x, {}]; x[[i + 1]] = x[[i]]^2/(x[[i]] - (10-10 x[[i]]+4 x[[i]]^2-x[[i]]^3)/(10-8 x[[i]]+3 x[[i]]^2));
                    программный модуль          цикл ДЛЯ                    добавить в конец

   If[x[[i]] == x[[i + 1]], Break[]]];
   условный оператор        прервать цикл

  Print["x=", x[[i]], ", кол-во инераций = ", i]]
  печатать
```

Рисунок 12 — Реализация метода паде-аппроксимации для системы нелинейных уравнений

3.3 Анализ результатов

Входные данные представлены на рисунке:

10	$f(x) = x^3 + 6x^2 + 9x - 4$

Рисунок 13 — Входные данные

Результат работы программы представлен на рисунках:

```
Programm[x^3 - 4 x^2 + 10 x - 10, 1208.]
x=1.62936, кол-во инераций = 29
```

Рисунок 14 – Пример работы программы

```
NSolve[0 == x^3 - 4 x^2 + 10 x - 10, {x}][[3]]
численное решение уравнений

{x → 1.62936}
```

Рисунок 15 – решение встроенной функции

Полученный результат совпадает с встроенной функцией Wolfram Mathematica.

```
r1 = Programm[x^3 - 4 x^2 + 10 x - 10, 1208.]

1.62936

r = NSolve[0 == x^3 - 4 x^2 + 10 x - 10][[3]]
     численное решение уравнений

{x → 1.62936}

r[[1, 2]] - r1

-2.88658 × 10^-15
```

Рисунок 16 – оценка точности

4. МЕТОДЫ КООРДИНАТНОЙ РЕЛАКСАЦИИ

4.1 Описание метода

Метод применим только в случае, если матрица A симметрична. В некотором роде он близок к итерационным методам для решения линейных систем, основанным на релаксации того или другого функционала. В данном случае роль такого функционала играет отношение Релея.

Вычислим прежде всего, как изменяется отношение Релея $\mu(X)=\dfrac{(AX,X)}{(X,X)}$ при изменении X в определённом направлении. Пусть

$$X' = X + \alpha Y, \tag{1.8}$$

где Y – некоторый фиксированный вектор, определяющий направление изменения вектора X. Тогда

$$(AX', X') = (AX + \alpha AY, X + \alpha Y) = (AX, X) + \alpha(AX, Y) + \alpha(AY, X) + \alpha^2(AY, Y) =$$

$$= (AX, X) + 2\alpha(AX, Y) + \alpha^2(AY, Y)$$

и, следовательно,

$$\mu(X') = \frac{(AX', X')}{(X', X')} = \frac{(AX, X) + 2\alpha(AX, Y) + \alpha^2(AY, Y)}{(X, X) + 2\alpha(AX, Y) + \alpha^2(Y, Y)}. \tag{1.9}$$

Подберём теперь множитель α так, чтобы отношение $\mu(X')$достигало наибольшего значения. Вычисляя производную $\mu(X')$ по α, получим

$$\frac{d\mu(X')}{d\alpha}=\frac{\left[2(AX,Y)+2\alpha(AY,Y)\right]\left[(X,X)+2\alpha(X,Y)+\alpha^2(Y,Y)\right]}{\left[(X,X)+2\alpha(X,Y)+\alpha^2(Y,Y)\right]^2}-$$

$$-\frac{\left[2(X,Y)+2\alpha(Y,Y)\right]\left[(AX,X)+2\alpha(AX,Y)+\alpha^2(AY,Y)\right]}{\left[(X,X)+2\alpha(X,Y)+\alpha^2(Y,Y)\right]^2}$$

и поэтому для определения α получим уравнение

$$\begin{aligned}&\left[(AX,Y)(Y,Y)-(AY,Y)(X,Y)\right]\alpha^2+\left[(AX,X)(Y,Y)-(AY,Y)(X,X)\right]\alpha+\\&+(AX,X)(X,Y)-(AX,Y)(X,X)=a\alpha^2+b\alpha+c=0,\end{aligned}\qquad(1.10)$$

где $a=(AX,Y)(Y,Y)-(AY,Y)(X,Y)$, $b=(AX,X)(Y,Y)-(AY,Y)(X,X)$,

$c=(AX,X)(X,Y)-(AX,Y)(X,X)$.

Исследование уравнения (1.10) показывает, что его корни всегда вещественны и различны.

Координатный релаксационный метод заключается в том, что за вектор Y на каждом шагу процесса берётся один из координатных векторов e_i.

Выбрав соответствующий корень, определим следующее приближение

$$X' = X + \alpha e_i. \tag{1.14}$$

Для этого приближения будем иметь

$$F' = AX' = AX + \alpha A e_i = F + \alpha A_i, \tag{1.15}$$

где A_i есть i-й столбец матрицы A. Далее

$$\begin{aligned} p' &= (AX', X') = (F', X') = p + 2\alpha f_i + \alpha^2 a_{ii}, \\ q' &= (X', X') = q + 2\alpha x_i + \alpha^2. \end{aligned} \tag{1.16}$$

Таким образом, величины p', q' и компоненты вектора F', нужные для проведения следующего шага, легко вычисляются.

Если на следующем шаге менять j-ю компоненту, $j \neq i$, то в квадратном уравнении нужно заменить a_{ii} на a_{jj}, x_i на x_j', f_i на f_j', q на q' и p на p'. Выбор номеров изменяемых компонент может осуществляться различно. Простейшей возможностью являются циклическое чередование индексов. Ввиду возможного накопления ошибок округления время от времени нужно вычислять величины q, p и f_i $(i = 1, 2, ..., n)$ непосредственно по определяющим их формулам.

Отметим также соотношение

$$\mu' = \frac{p - f_i x_i + a\alpha}{q - x_i^2} = \frac{p'}{q'}, \tag{1.17}$$

которое можно использовать для контроля вычислений.

4.2 Программная реализация

На рисунках представлена программная реализация координатного релаксационного метода. (см. рисунок 17). Критерием остановки итерационного процесса является количество итераций и близость двух приближений.

```
m[A_] := For[i1 = 1, i1 ≤ 100, i1++, If[Mod[i1, 4] ≠ 0, j = Mod[i1, 4], j = 4]; res = Reduce[
           цикл ДЛЯ                     остаток от деления   остаток от деления        привести
    x^2 (F[[j]] - A[[j, j]] * X[[j]]) + x (F.X - A[[j, j]] * X.X) + F.X * X[[j]] - F[[j]] * X.X == 0, x]; res = N@Table[res[[i, 2]], {i, 1, 2}];
                                                                                                   таблица значений
  α = Max@res;
      максимум
  X = X + α * e[[j]]; F = F + α * A[[j]]; u[[Mod[j, 2] + 1]] = ((A.X).X)/(X.X);
                                            остаток от деления
  If[Round[u[[1]], 0.000001] == Round[u[[2]], 0.000001],
     округлить                 округлить
   Print["Собственное число:  ", u[[1]], " получено за ", i1, " итераций", " ,собственный вектор:", X];
   печатать
   Break[]]];
   прервать цикл
```

Рисунок 17 — Реализация координатного релаксационного метода

4.3 Анализ результатов

Входные данные представлены на рисунке. (см. рисунок 18).

```
A = {{1.00, 0.42, 0.54, 0.66}, {0.42, 1.00, 0.32, 0.44}, {0.54, 0.32, 1.00, 0.22}, {0.66, 0.44, 0.22, 1.00}};
X = {0.3, 0.2, 0.9, 0.75};
F = A.X
p = (A.X).X
e = IdentityMatrix@4;
      единичная матрица
u = {0, 0};

{1.365, 0.944, 1.291, 1.234}

2.6857
```

Рисунок 18 — Входные данные

```
m[A]

Собственное число:  2.32275 получено за 11 итераций ,собственный вектор:{0.965945, 0.766907, 0.722513, 0.857647}
```

Рисунок 19 – решение

```
S = Eigenvalues@A
      собственные числ

{2.32275, 0.796707, 0.638284, 0.242261}
```

Рисунок 20 – решение встроенной функции

Полученный результат совпадает с встроенной функцией Wolfram Mathematica.

```
S[[1]] - u[[1]]

2.44259 × 10^-7
```

Рисунок 21 – оценка точности

5. ИНТЕРПОЛИРОВАНИЕ ФУНКЦИИ ОДНОЙ ИЛИ ДВУХ ПЕРЕМЕННЫХ. ИНТЕРПОЛЯЦИОННАЯ ФОРМУЛА ЛАГРАНЖА

5.1 Описание метода

В общем виде интерполяционный многочлен в форме Лагранжа записывается в следующем виде:

$$L(x) = \sum_{i=0}^{n} \left(f(x_i) \cdot \prod_{j=0,\, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \right)$$

где n - степень полинома $L(x)$;

$f(x_i)$ - значение значения интерполирующей функции $f(x)$ в точке x_i;

$l_i(x)$ - базисные полиномы (множитель Лагранжа), которые определяются по формуле:

$$l_i(x) = \prod_{j=0,\, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_i - x_0} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n}$$

Так, например, интерполяционный многочлена в форме Лагранжа, проходящий через три заданных точки $\left[x_1, f(x_1);\ \ x_2, f(x_2);\ \ x_3, f(x_3) \right]$, будет записываться в следующем виде:

$$L(x) = f(x_0) \cdot \frac{(x - x_1) \cdot (x - x_2)}{(x_0 - x_1) \cdot (x_0 - x_2)} + f(x_1) \cdot \frac{(x - x_0) \cdot (x - x_2)}{(x_1 - x_0) \cdot (x_1 - x_2)} +$$
$$+ f(x_2) \cdot \frac{(x - x_0) \cdot (x - x_1)}{(x_2 - x_0) \cdot (x_2 - x_1)}$$

Многочлен в форме Лагранжа в явном виде содержит значения функций в узлах интерполяции, поэтому он удобен, когда значения функций меняются, а узлы интерполяции неизменны.

5.2 Программная реализация

```
P[j_, x_, X_] := Product[If[j ≠ i, (x - X[[i]])/(X[[j]] - X[[i]]), 1], {i, 1, Length@X}]

P2[i_, j_, x_, y_, X_, Y_] := Times[P[i, x, X], P[j, y, Y]]

PAll[f_, x_, y_, X_, Y_] := Sum[Sum[f[X[[i]], Y[[j]]] * P2[i, j, x, y, X, Y], {j, 1, Length@Y}], {i, 1, Length@X}]
```

Рисунок 22 — Программная реализация вывода интерполяционного полинома

5.3 Анализ результатов

Входные данные представлены на рисунке. (см.рисунок 23)

```
f1[x_, y_] := x^2 y + Cos[y / 2] / 2 + y
                      косинус

f2[x_, y_] := x^3 + Sin[y^4 / 2]
                    синус
```

Рисунок 23 — Входные данные

```
Plot3D[{PAll[f1, x, y, {-5, -1, 2, 5}, {-5, -1, 2, 5}], f1[x, y]}, {x, -5, 5}, {y, -5, 5}, PlotLegends → "Expressions"]
график функции 2-х переменных                                                                                    легенды графика
```

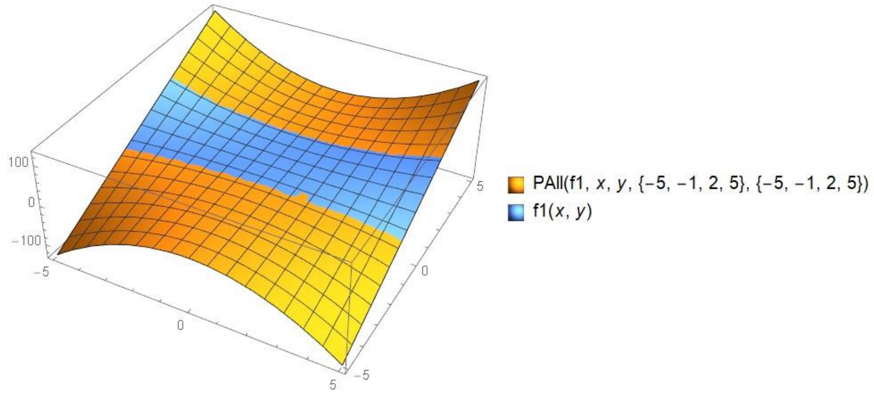

Рисунок 24 - Сравнение результата интерполирования со значениями функции

```
Plot3D[{PAll[f2, x, y, {-9, -3, 0, 5, 10, 25}, {-9, -3, 0, 3, 7, 20}], f2[x, y]}, {x, -3, 6}, {y, -3, 10},
график функции 2-х переменных
 PlotLegends → "Expressions"]
 легенды графика
```

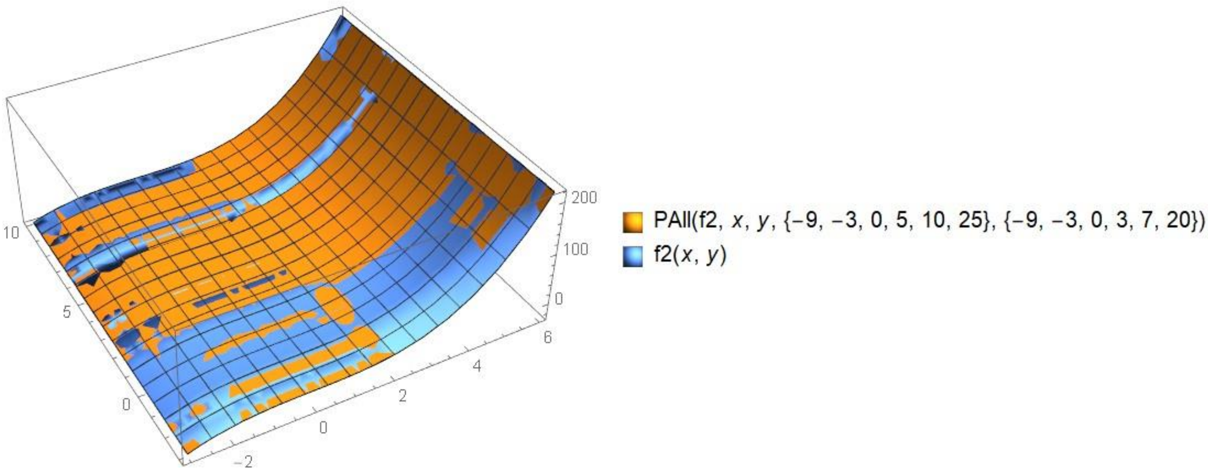

Рисунок 25 - Сравнение результата интерполирования со значениями функции

6. ПОСТРОЕНИЕ НАИЛУЧШЕГО ПРИБЛИЖЕНИЯ ФУНКЦИИ. МЕТОД НАИМЕНЬШИХ КВАДРАТОВ.

6.1 Описание метода

Метод наименьших квадратов - математический метод, основанный на определении аппроксимирующей функции, которая строится в ближайшей близости от точек из заданного массива экспериментальных данных. Близость исходной и аппроксимирующей функции F(x) определяется числовой мерой, а именно: сумма квадратов отклонений экспериментальных данных от аппроксимирующей кривой F(x) должна быть наименьшей.

Аппроксимирующая функция по методу наименьших квадратов определяется из условия минимума суммы квадратов отклонений (ξ_i) расчетной аппроксимирующей функции от заданного массива экспериментальных данных. Данный критерий метода наименьших квадратов записывается в виде следующего выражения:

$$\sum_{i=1}^{N} \xi_i^{\,2} = \sum_{i=1}^{N} \left(F(x_i) - y_i\right)^2 \to \min$$

$F(x_i)$ - значения расчетной аппроксимирующей функции в узловых точках x_i,

y_i - заданный массив экспериментальных данных в узловых точках x_i.

Квадратичный критерий обладает рядом "хороших" свойств, таких, как дифференцируемость, обеспечение единственного решения задачи аппроксимации при полиномиальных аппроксимирующих функциях.

В зависимости от условий задачи аппроксимирующая функция представляет собой многочлен степени m

$$F_m(x) = a_0 + a_1 \cdot x + \ldots + a_{m-1} \cdot x^{m-1} + a_m \cdot x^m$$

Степень аппроксимирующей функции (m) не зависит от числа узловых точек, но ее размерность должна быть всегда меньше размерности (количества точек) заданного массива экспериментальных данных.

$$1 \le m \le N-1$$

В случае если степень аппроксимирующей функции m=1, то мы аппроксимируем табличную функцию прямой линией (линейная регрессия).

В случае если степень аппроксимирующей функции m=2, то мы аппроксимируем табличную функцию квадратичной параболой (квадратичная аппроксимация).

В случае если степень аппроксимирующей функции m=3, то мы аппроксимируем табличную функцию кубической параболой (кубическая аппроксимация).

В общем случае, когда требуется построить аппроксимирующий многочлен степени m для заданных табличных значений, условие минимума суммы квадратов отклонений по всем узловым точкам переписывается в следующем виде:

$$S = \sum_{i=1}^{N} \xi_i^{\,2} = \sum_{i=1}^{N} \left(a_0 + a_1 \cdot x_i + \ldots + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i\right)^2 \to \min$$

x_i, y_i - координаты узловых точек таблицы;

$a_j, (j = 0, \ldots, m)$ - неизвестные коэффициенты аппроксимирующего многочлена степени m;

N - количество заданных табличных значений.

Необходимым условием существования минимума функции является равенству нулю ее частных производных по неизвестным переменным $a_j, (j = 0, \ldots, m)$.

В результате получим следующую систему уравнений:

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2 \cdot \sum_{i=1}^{N} \left(a_0 + a_1 \cdot x_i + \ldots + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i \right) = 0 \\ \dfrac{\partial S}{\partial a_1} = 2 \cdot \sum_{i=1}^{N} \left(a_0 + a_1 \cdot x_i + \ldots + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i \right) \cdot x_i = 0 \\ \ldots \\ \dfrac{\partial S}{\partial a_m} = 2 \cdot \sum_{i=1}^{N} \left(a_0 + a_1 \cdot x_i + \ldots + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i \right) \cdot x_i^m = 0 \end{cases}$$

Преобразуем полученную линейную систему уравнений: раскроем скобки и перенесем свободные слагаемые в правую часть выражения. В результате полученная система линейных алгебраических выражений будет записываться в следующем виде:

$$\begin{cases} a_0 \cdot N + a_1 \cdot \sum_{i=1}^{N} x_i + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^{m-1} + a_m \cdot \sum_{i=1}^{N} x_i^m = \sum_{i=1}^{N} y_i \\ a_0 \cdot \sum_{i=1}^{N} x_i + a_1 \cdot \sum_{i=1}^{N} x_i^2 + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^m + a_m \cdot \sum_{i=1}^{N} x_i^{m+1} = \sum_{i=1}^{N} y_i \cdot x_i \\ \ldots \\ a_0 \cdot \sum_{i=1}^{N} x_i^m + a_1 \cdot \sum_{i=1}^{N} x_i^{m+1} + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^{2 \cdot m-1} + a_m \cdot \sum_{i=1}^{N} x_i^{2 \cdot m} = \sum_{i=1}^{N} y_i \cdot x_i^m \end{cases}$$

Данная система линейных алгебраических выражений может быть переписана в матричном виде:

$$\begin{vmatrix} N & \sum_{i=1}^{N} x_i & \cdots & \sum_{i=1}^{N} x_i^m \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \cdots & \sum_{i=1}^{N} x_i^{m+1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} x_i^m & \sum_{i=1}^{N} x_i^{m+1} & \cdots & \sum_{i=1}^{N} x_i^{2 \cdot m} \end{vmatrix} \cdot \begin{vmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{vmatrix} = \begin{vmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} y_i \cdot x_i \\ \vdots \\ \sum_{i=1}^{N} y_i \cdot x_i^m \end{vmatrix}$$

В результате была получена система линейных уравнений размерностью m+1, которая состоит из m+1 неизвестных. Данная система может быть решена с помощью любого метода решения линейных алгебраических уравнений (например, методом Гаусса). В результате решения будут найдены неизвестные параметры аппроксимирующей функции, обеспечивающие минимальную сумму квадратов отклонений аппроксимирующей функции от исходных данных, т.е. наилучшее возможное квадратичное приближение. Следует помнить, что при изменении даже одного значения исходных данных все коэффициенты изменят свои значения, так как они полностью определяются исходными данными.

6.2 Программная реализация

Функция Coef возвращает коэффициенты по множеству точек – points и базису – functions, функция Model возвращает готовую функцию по базису – functions, к которому мы добавляем t^0 –свободный член и набору коэффициентов из функции Coef.

```
Clear[Coef]
очистить
Coef[points_, functions_, var_] := Module[{l = Length@points, x = points[[All, 1]], X, y = points[[All, 2]]},
                                   программны… длина                 всё                      всё
  X = Join[ConstantArray[{1}, l], Transpose@Table[f1 /. var -> x, {f1, functions}], 2];
      сое… постоянный массив      транспози… таблица значений
  Inverse[X^T.X].X^T.y]
  обратная матрица

Clear[Model]
очистить
Model[functions_, coefs_] := coefs.Prepend[functions, t^0]
                                   добавить в начало
```

Рисунок 26 – реализация

6.3 Анализ результатов

Я рассматриваю функции $x^4 + 2x$, $\mathrm{Cosh}[x] - x$^3, $\mathrm{Sin}[8x]$, $\mathrm{Log}[3x] + 2x$^3.

функция $x^4 + 2x$:

```
Plot[{newf}, {t, 0, 2}, AspectRatio → 1, PlotRange → Full, AxesLabel → {"x", "y"}, Epilog → {PointSize[0.02], Point@points},
|график функции          |аспектное отноше···  |отображае···  |в по···  |обозначения на осях    |эпилог    |размер точки       |точка
 PlotLabels → "Expressions"]
 |пометки на графике
```

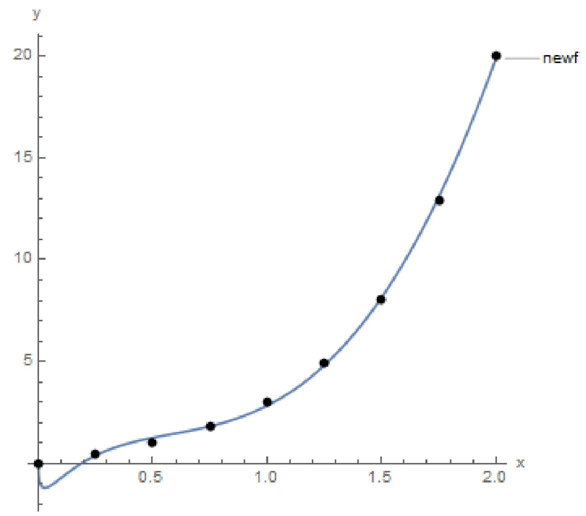

Рисунок 27 – пример №1

Функция $\mathrm{Cosh}[x] - x$^3:

```
newf = Model[functions, Coef[points, functions, t]]
Plot[{newf, Cosh[t] - t^3}, {t, 0, 2}, AspectRatio → 1, PlotRange → Full, AxesLabel → {"x", "y"}, PlotLabels → "Expressions"]
|график функ··· |гиперболический косинус   |аспектное отноше··· |отображае··· |в по··· |обозначения на осях    |пометки на графике
```

$1.02456 - 4.4703\sqrt{t} + 12.3654\,t - 8.31405\,t^{3/2}$

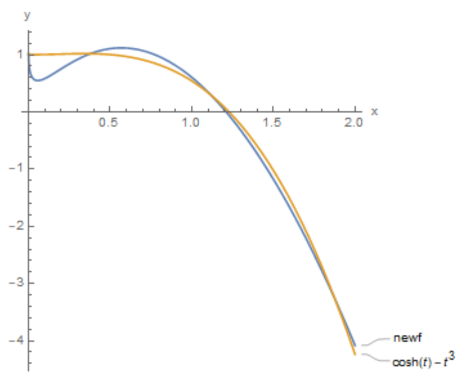

Рисунок 28 – пример №2

```
newf = Model[functions, Coef[points, functions, t]]
Plot[{newf}, {t, 0, 2}, AspectRatio → 1, PlotRange → Full, AxesLabel → {"x", "y"},
|график функции         |аспектное отноше··· |отображае··· |в по··· |обозначения на осях
 Epilog → {PointSize[0.02], Point@points}, PlotLabels → "Expressions"]
 |эпилог    |размер точки     |точка          |пометки на графике
```

$1.02456 - 4.4703\sqrt{t} + 12.3654\,t - 8.31405\,t^{3/2}$

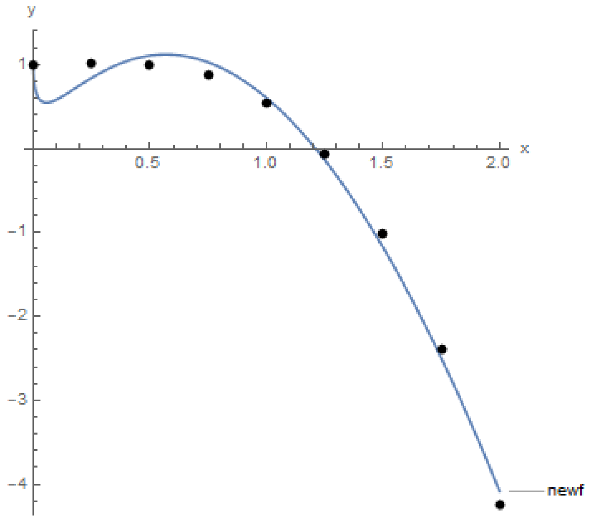

Рисунок 29 – пример №3

Функция $\mathrm{Sin}[8x]$:

```
newf = Model[functions, Coef[points, functions, t]]
Plot[{newf, Sin[8 t]}, {t, 0.1, 2}, AspectRatio → 1, PlotRange → Full, AxesLabel → {"x", "y"}, PlotLabels → "Expressions"]
|график функ··· |синус             |аспектное отноше··· |отображае··· |в по··· |обозначения на осях    |пометки на графике
```

```
{{0., 0.}, {0.0714286, 0.540834}, {0.142857, 0.909823}, {0.214286, 0.989723}, {0.285714, 0.755147}, {0.357143, 0.280629}, {0
 {0.714286, -0.538705}, {0.785714, 0.00252898}, {0.857143, 0.54296}, {0.928571, 0.91087}, {1., 0.989358}, {1.07143, 0.753487
 {1.35714, -0.990434}, {1.42857, -0.907712}, {1.5, -0.536573}, {1.57143, 0.00505794}, {1.64286, 0.545082}, {1.71429, 0.9119
```

$\{\sqrt{t},\ t,\ t^{3/2},\ t^2,\ t^{5/2},\ t^3,\ t^{7/2},\ t^4\}$

$0.000962036 + 189.351\sqrt{t} - 2271.72\,t + 10993.1\,t^{3/2} - 27509.1\,t^2 + 38607.7\,t^{5/2} - 30636.4\,t^3 + 12831.4\,t^{7/2} - 2203.72\,t^4$

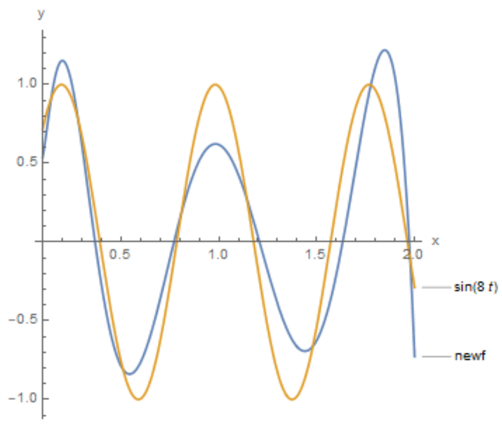

Рисунок 30 – пример №4

```
newf = Model[functions, Coef[points, functions, t]]
Plot[{newf}, {t, 0.1, 2}, AspectRatio → 1, PlotRange → Full, AxesLabel → {"x", "y"},
|график функции          |аспектное отноше… |отображае… |в по… |обозначения на осях
 Epilog → {PointSize[0.02], Point@points}, PlotLabels → "Expressions"]
 |эпилог   |размер точки  |точка          |пометки на графике
```

$0.000962036 + 189.351\sqrt{t} - 2271.72\,t + 10993.1\,t^{3/2} - 27509.1\,t^{2} + 38607.7\,t^{5/2} - 30636.4\,t^{3} + 12831.4\,t^{7/2} - 2203.72\,t^{4}$

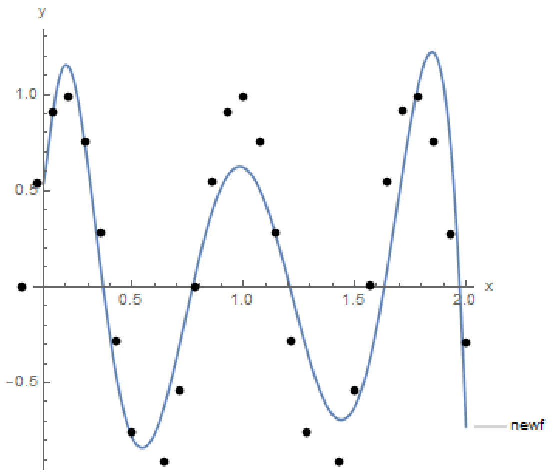

Рисунок 31 – пример №5

функция $\mathrm{Log}[3x] + 2x$^3:

```
newf = Model[functions, Coef[points, functions, t]]
Plot[{newf, Log[3 t] + 2 t^3}, {t, 1, 6}, AspectRatio → 1, PlotRange → Full, AxesLabel → {"x", "y"}, PlotLabels → "Expressions"]
|график функ… |натуральный логарифм        |аспектное отноше… |отображае… |в по… |обозначения на осях       |пометки на графике
```

$535.245 - 827.696\sqrt{t} + 316.643\,t$

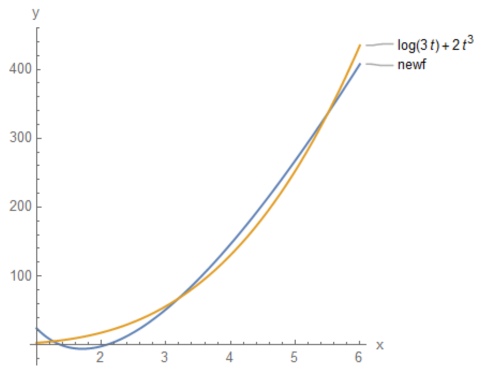

Рисунок 32 – пример№6

```
Plot[{newf}, {t, 1, 6}, AspectRatio → 1, PlotRange → Full, AxesLabel → {"x", "y"},
Epilog → {PointSize[0.02], Point@points}, PlotLabels → "Expressions"]
```

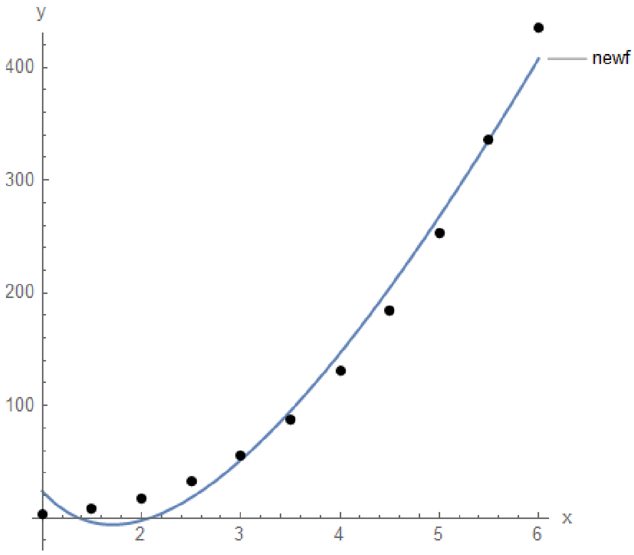

Рисунок 33 – пример №7

Полученный результат совпадает с встроенной функцией Wolfram Mathematica .

7. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ. КВАДРАТУРНЫЕ ФОРМУЛЫ ТИПА ГАУССА-КРИСТОФФЕЛЯ.

7.1 Описание метода

Рассмотрим общую задачу численного интегрирования с весовой функцией $\mu(x) > 0, \quad x \in [a, b]$.

При построении квадратурных формул интерполяционного типа на бесконечных интервалах необходимо ввести дополнительно условие на весовую функцию:

$$\left|\int_a^b \mu(x) x^i dx\right| < \infty, \quad i = 0, 1, \ldots \quad (1)$$

Запишем квадратурную формулу для произвольного, но фиксированного распределения узлов

$a \le x_1^{(n)} < x_2^{(n)} < \ldots < x_n^{(n)} \le b$:

$$J(f) = \int_a^b f(x)\mu(x)dx = \sum_{k=1}^n C_k^{(n)} \cdot f(x_k^{(n)}) + R_n(f) \quad (18)$$

7.2 Программная реализация

```
Clear[x, gaussCrist];
очистить

gaussCrist[f_, n_] := Module[{Integr, x1 = {}, y, a = {}, xi, Ai, p, a1 = -1, b1 = 1, s = {}, ylocal, q, I}, p = D[(x^2 - 1)^n, {x, n}] / (2^n * n!);
                      программный модуль                                                                  мнимая единица

  y = #[[1, 2]] & /@ Solve[p == 0, x];
                     решить уравнения

  xi = 1/2 * (a1 + b1) + 1/2 (b1 - a1) * ylocal;

  AppendTo[x1, N[xi /. ylocal :> #]] & /@ y;
  добавить в ко···  численное приближение

  a = 2 / ((1 - #^2) * (D[p, x] /. x :> #)^2) & /@ x1;

  Integr = ∑_{k=1}^{n} a[[k]] * (f /. x :> x1[[k]]);

  Print["∫_{-1}^{1} ", f, "= ", Integr, "  Узлы - ", x1, " Веса - ", a]]
  печатать
```

Рисунок 34 — Программная реализация расчёта интеграла

7.3 Анализ результатов

Интегрирование функций: $x^2, x^3, e^3, \mathrm{Cos}[x] + Sin[x], \frac{1}{x+2}, \sqrt[3]{-1+x}, \frac{1}{\sqrt{2-x+x^3}}$

```
gaussCrist[x^2, 5]
N@Integrate[x^2, {x, -1, 1}]
```

$\int_{-1}^{1} x^2 = 0.666667$ Узлы – {0., –0.538469, 0.538469, –0.90618, 0.90618} Веса – {0.568889, 0.478629, 0.478629, 0.236927, 0.236927}

0.666667

```
gaussCrist[x^3, 5]
N@Integrate[x^3, {x, -1, 1}]
```

$\int_{-1}^{1} x^3 = 0.$ Узлы – {0., –0.538469, 0.538469, –0.90618, 0.90618} Веса – {0.568889, 0.478629, 0.478629, 0.236927, 0.236927}

0.

```
gaussCrist[Exp[3], 5]
N@Integrate[Exp[3], {x, -1, 1}]
```

$\int_{-1}^{1} e^3 = 40.1711$ Узлы – {0., –0.538469, 0.538469, –0.90618, 0.90618} Веса – {0.568889, 0.478629, 0.478629, 0.236927, 0.236927}

40.1711

Рисунок 35 – пример вычисления интеграла №1

```
gaussCrist[Cos[x] + Sin[x], 4]
N@Integrate[Cos[x] + Sin[x], {x, -1, 1}]
```

$\int_{-1}^{1} \mathrm{Cos}[x] + \mathrm{Sin}[x] = 1.68294$ Узлы – {–0.339981, 0.339981, –0.861136, 0.861136} Веса – {0.652145, 0.652145, 0.347855, 0.347855}

1.68294

$\mathrm{gaussCrist}\left[\frac{1}{x+2}, 5\right]$

$\mathrm{N@Integrate}\left[\frac{1}{x+2}, \{x, -1, 1\}\right]$

$\int_{-1}^{1} \frac{1}{2+x} = 1.09861$ Узлы – {0., –0.538469, 0.538469, –0.90618, 0.90618} Веса – {0.568889, 0.478629, 0.478629, 0.236927, 0.236927}

1.09861

```
gaussCrist[CubeRoot[x - 1], 5]
N@Integrate[CubeRoot[x - 1], {x, -1, 1}]
```

$\int_{-1}^{1} \sqrt[3]{-1+x} = -1.89273$ Узлы – {0., –0.538469, 0.538469, –0.90618, 0.90618} Веса – {0.568889, 0.478629, 0.478629, 0.236927, 0.236927}

–1.88988

Рисунок 36 – пример вычисления интеграла №2

```
gaussCrist[1/Sqrt[x^3 - x + 2], 10]
N@Integrate[1/Sqrt[x^3 - x + 2], {x, -1, 1}]
```
[⋯ [интегрировать

$\int_{-1}^{1} \frac{1}{\sqrt{2 - x + x^3}} = $ 1.42453   Узлы - {-0.148874, 0.148874, -0.433395, 0.433395, -0.67941, 0.67941, -0.865063, 0.865063, -0.973907, 0.973907}

Веса - {0.295524, 0.295524, 0.269267, 0.269267, 0.219086, 0.219086, 0.149451, 0.149451, 0.0666713, 0.0666713}

1.42453

Рисунок 37 – пример вычисления интеграла №3

Полученный результат совпадает с встроенной функцией Wolfram Mathematica.

ЗАКЛЮЧЕНИЕ

Были выполнены поставленные задачи, а именно:

- изучить теорию по данным методам;
- написать программную реализацию методов с использованием теоретических данных;
- проверить правильность работы методов;
- проанализировать полученные результаты.

В дальнейшей перспективе целесообразно освоение других вычислительных методов и применение их на практике.

СПИСОК ЛИТЕРАТУРЫ

1. Юхно Л. Х. Модификация некоторых методов типа сопряжённых направлений для решения систем линейных алгебраических уравнений / Л. Х. Юхно; Ж. вычил. матем. и матем. физ.., 2007, том 47, номер 11, 1811-1818

2. В. Б. Хазанов. Конспект лекций «Численные методы алгебры»

3. В. Б. Хазанов. Конспект лекций «Численные методы анализа»

4. В. В. Воеводин. Численные методы алгебры. Теория и алгоритмы Изд-во «Наука», Москва, 1966

5. Гончаров В. Л. Теория интерполирования и приближения функций. – М.: Гос. изд. технико-теорет. лит., 1954.

6. Никольский С. М. Приближение функций многих переменных и теоремы вложения. – М.: Наука, 1969.