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: «Реализация метода явной интерполяции функции. Восстановление функции по заданному набору точек. Интерполяционная формула Лагранжа. Тригонометрическое интерполирование»

Направление: 01.03.02 Прикладная математика и информатика

Студент: Иксанов Марат Васильевич

Группа: ПМ-2001 Подпись _____

Проверил: Хазанов Владимир Борисович

Должность: д. ф-м. н., профессор

Оценка: _____ Дата: _____

Подпись: _____

Санкт-Петербург

2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель: изучить и реализовать метод явной интерполяции с помощью тригонометрических многочленов.

Задачи:

- Изучить теоретическую составляющую и продемонстрировать принцип работы определённых численных методов линейной алгебры и математического анализа
- Реализовать данные методы при помощи системы Wolfram Mathematica
- Проверить корректность и точность работы реализаций
- Сделать выводы о работе программной реализации реализованных методов

В работе рассматриваются методы решения некоторых задач численных методов в линейной алгебре и математическом анализе, решение тестовых задач с разными входными данными, результаты работы программных реализаций решений поставленных задач.

В курсовой работе содержатся теоретические описания методов решения задач численного анализа в областях линейной алгебры и математического анализа, решения тестовых задач с разными входными данными, результаты работы программных реализаций решений поставленных задач.

1. РЕШЕНИЕ МАТРИЧНОГО УРАВНЕНИЯ МЕТОДОМ LU-, LR-РАЗЛОЖЕНИЯ

1.1. Описание метода и алгоритм решения

Для решения матричного уравнения $Ax = f$ матрица A представляется в виде произведения двух матриц: $LUx = f$, где L – нижняя треугольная матрица, а U – верхняя треугольная матрица, где

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, a_{11} \neq 0, \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0, \dots, |\mathrm{A}| \neq 0,$$

$$L = \begin{pmatrix} l_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ l_{n1} & \cdots & l_{nn} \end{pmatrix}, R = \begin{pmatrix} 1 & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix},$$

$$f = \begin{pmatrix} f_{11} & \cdots & f_{1m} \\ \vdots & \ddots & \vdots \\ f_{n1} & \cdots & f_{nm} \end{pmatrix}, x = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$$

Данный метод является разновидностью метода Гаусса, поскольку можно разбить решение на 2 части, каждая из которых решается прямым и обратным ходом метода Гаусса.

Как было описано выше, мы представляем матричное уравнение $Ax = f$ как $LRx = f$. Далее можно представить это уравнение как $Lg = f$, где $Rx = g$, где $g = \begin{pmatrix} g_{11} & \cdots & g_{1m} \\ \vdots & \ddots & \vdots \\ g_{n1} & \cdots & g_{nm} \end{pmatrix}$

Поскольку L, R – нижняя и верхняя треугольные матрицы, на первом шаге решается уравнение $Lg = f$ прямым ходом. На втором шаге решается уравнение $Rx = g$ обратным ходом. В итоге, матрица x – является искомым решением матричного уравнения $Ax = f$

1.1.1. Построение L, R, g матриц и прямой ход решения матричного уравнения:

Первый столбец L матрицы равен первому столбцу матрицы A, в то время, как первая строка R матрицы равна первой строке матрицы A, а первая строка

матрицы g равна произведению l_{11}^{-1} на каждый элемент 1ой строки матрицы f. В дальнейшем, обратный элементом a^{-1} будем считать $\frac{1}{a}$.

Остальные столбцы и строки строятся следующим образом:

Последовательно строятся столбец k матрицы L и строка матрицы k матрицы R. Столбец L равен разности соответствующего столбца матрицы A и суммы всех $k-1$ столбцов матрицы L до, каждый из которых соответственно умножен на r_{ik}, где i – индекс соответствующего столбца матрицы L, который меньше искомого индекса k. Строка R равна произведению l_{11}^{-1} на разность соответствующей строки k и суммы всех i строк матрицы R, умноженных соответственно на l_{ki} , где $i < k$. Описанный в этом абзаце процесс повторяется (n-k) раз.

Правая часть матричного уравнения, матрица g строится следующим образом: k строка матрицы g равна произведению l_{11}^{-1} на разность k строки матрицы f и суммы $k-1$ предыдущих строк, соответственно умноженных на l_{ki}, где i индекс одной из этих строк.

В данном пункте k идет от 2 до n последовательно.

1.1.2. Обратный ход решения матричного уравнения:

В данном пункте k идет от $n-1$ до 1 последовательно.

В начале автоматически присваиваем n строке матрицы x строку матрицы g. Далее каждая k строка матрицы x равна разности k строки матрицы g и суммы всех строк, индекс которых между k и n, умноженных на r_{kj} соответственно, где i – индекс текущей строки матрицы x в сумме.

Полученная матрица x является искомым решением матричного уравнения.

1.2. Программная реализация

Метод LU-, LR- разложения реализован с помощью Wolfram Mathematica.

Входные данные: матрица A и матрица правой части f. Выходные данные: матрица x. На рисунке 1.1 представлена программная реализация метода.

```
input = {{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}};
f = Normal[SparseArray[{i_, i_} → 1, Dimensions[input]]];
n = Length[input];


g = ConstantArray[0, Dimensions[f]];
left = ConstantArray[0, {n, n}];
right = left;
x = ConstantArray[0, Dimensions[f]];

left[[All, 1]] = input[[All, 1]];
right[[1, All]] = left[[1, 1]]^-1 * input[[1, All]];
g[[1, All]] = left[[1, 1]]^-1 * f[[1, All]];

Do[
  Do[

    left[[All, k]] = input[[All, k]] - Sum[left[[All, j]] * right[[j, k]], {j, k - 1}];
    right[[k, All]] = left[[k, k]]^-1 * (input[[k, All]] - Sum[left[[k, j]] * right[[j, All]], {j, k - 1}]);

    g[[k, All]] = left[[k, k]]^-1 * (f[[k, All]] - Sum[left[[k, j]] * g[[j, All]], {j, k - 1}])
    , {i, k, n}];
  , {k, 2, n}];

x[[n, All]] = g[[n, All]];
Do[
  x[[k, All]] = g[[k, All]] - Sum[right[[k, j]] * x[[j, All]], {j, k + 1, n}]

  , {k, n - 1, 1, -1}];
x // MatrixForm
```

Рисунок 1.1 – программная реализация метода LU-,LR- разложения

1.3. Результаты вычислений и оценка точности решения

Для оценки точности решения рассмотрим входные данные, тип данных входных матриц – числа с плавающей точкой.

Входные данные: $A = \begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix}, f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Необходимо решить матричное уравнение $Ax = f$, иначе говоря найти обратную матрицу матрицы A

Решение с помощью собственной реализации представлено на рисунке 1.2:

$$\begin{pmatrix} 2.50759 & -0.123039 & -1.01149 & -1.37834 \\ -0.123039 & 1.33221 & -0.261427 & -0.447454 \\ -1.01149 & -0.261427 & 1.53183 & 0.445609 \\ -1.37834 & -0.447454 & 0.445609 & 2.00855 \end{pmatrix}$$

Рисунок 1.2 – решение, полученное собственной реализацией

Решение (нахождение обратной матрицы) с помощью встроенной функции $Inverse[A]$ представлено на рисунке 1.3 :

$$\begin{pmatrix} 2.50759 & -0.123039 & -1.01149 & -1.37834 \\ -0.123039 & 1.33221 & -0.261427 & -0.447454 \\ -1.01149 & -0.261427 & 1.53183 & 0.445609 \\ -1.37834 & -0.447454 & 0.445609 & 2.00855 \end{pmatrix}$$

Рисунок 1.3 – решение, полученное встроенной реализацией

Поскольку результатами вычислений являются матрицы, составленные из чисел с плавающей точкой, полученные значения могут отличаться на довольно малую величину, следовательно, для определения точности решения построим матрицу невязки и определим ее норму. Матрица невязки – разность левой и правой части матричного уравнения с найденной матрицей x, то есть $Ax - f$

Вычисленная норма матрицы невязки собственной функции представлена на рисунке 1.4:

```
Norm[input.x - f]
```

3.7904×10^{-16}

Рисунок 1.4 – норма невязка решения собственной реализации

Вычисленная норма матрицы невязки встроенной функции представлена на рисунке 1.5:

```
Norm[input.Inverse[input] - f]
```

3.4701×10^{-16}

Рисунок 1.5 – норма невязка решения собственной реализации

Можно заметить, что нормы практически совпадают и достаточно малы, отсюда следует, что обе функции вычисляют довольно точно.

2. РЕШЕНИЕ СЛАУ ИТЕРАЦИОННЫМ ГРАДИЕНТНЫМ МЕТОДОМ (МЕТОД НАИСКОРЕЙШЕГО СПУСКА)

2.1. Описание метода и алгоритм решения

Для решения уравнения $Ax = f$ методом наискорейшего спуска необходимо, чтобы матрица A была симметрична и положительно определена.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, f = \begin{pmatrix} f_{11} \\ \vdots \\ f_{n1} \end{pmatrix}, x = \begin{pmatrix} x_{11} \\ \vdots \\ x_{n1} \end{pmatrix}$$

Пусть x^* искомое решение СЛАУ. Введем функцию $H(x) = (Ax, x) - (f, x)$. Так как градиент функции $H(x)$ по некоторому направлению z имеет вид $-2(Ax - f)$, то это значит, что решение СЛАУ совпадает с точкой минимума самой функции $H(x)$. С этого момента задачу поиска решения x^* можно заменить равносильной задачей поиска минимума функции $H(x)$. Определим начальное приближение, например, $x^0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$, от которого будем опираться при нахождении дальнейших x_k приближений. Вектор невязки $r_k = f - Ax_k$ будем считать направлением спуска, а l_k выбирается из условия минимизации $H(x)$ вдоль направления спуска:

$H(x_{k+1}) = \cdots = H(x_k) - 2l_k(r_k, r_k) + l^2(Ar_k, r_k)$, где $r_{k+1} = f - Ax_{k+1} = r - l_k Ar_k$.

$$\min_{l_k} H(x_k)_1 \Rightarrow H(x_k)'_{l_k} = 0 \Rightarrow l_k = \frac{(r_k, r_k)}{(Ar_k, r_k)}$$

Таким образом, искомые приближения строятся по формуле: $x_{k+1} = x_k + l_k r_k$.

На данный момент не учтен один главный момент: мы не установили границы, на котором итерации заканчиваются.

Существуют 3 основных критерия прекращения итерационного процесса:

1) По числу итераций. $k < K_{max}$, где K_{max} задается пользователем
2) По близости к решению нормы разности $x_{k+1} - x_k$, которая должна быть меньше заданного δ

3) По малости невязки. Норма вектора невязки должна быть меньше некоторого ε

Также стоит отметить, что поскольку данный метод работает только для симметричных положительно определенных матриц, следовательно, данный метод не работает для несимметричных матриц. Для решения этой проблемы, чтобы использовать метод на любой матрице можно применить, например, 1 метод преобразования Гаусса, путем умножения левой и правой части на транспонированную матрицу A

2.2. Программная реализация метода

Метод наискорейшего спуска реализован с помощью Wolfram Mathematica. Входные данные: матрица A и матрица правой части f. Выходные данные: матрица x.

Для реализации данного метода были созданы 5 функций, которые выполняют данную задачу, однако в каждой из них используются, либо отличные от других методы, либо различные критерии прекращения, описанные выше.

На рисунке 2.1 приведен код функций, реализующих метод.

```
stDescentVer1[A_, f_, k_] := Module[
  {n = Length@f, x0, x1, x2, r1, l1},
  x0 = ConstantArray[0, {n, 1}];
  x1 = x0;
  Do[
   r1 = f - A.x1;
   l1 = (Transpose[r1].r1)/(Transpose[(A.r1)].r1)[[1, 1]];
   x2 = x1 + l1 * r1;
   x1 = x2
   , {i, 1, k}];
  x2
 ]

stDescentVer2[A_, f_, k_] := Nest[# + (Transpose[(f - A.#)].(f - A.#))/(Transpose[(A.(f - A.#))].(f - A.#))[[1, 1]] * (f - A.#) &, ConstantArray[0, {Length@f, 1}], k]

stDescentVer3[A_, f_, e_] := NestWhile[# + (Transpose[(f - A.#)].(f - A.#))/(Transpose[(A.(f - A.#))].(f - A.#))[[1, 1]] * (f - A.#) &, ConstantArray[0, {Length@f, 1}], Norm[f - A.#] >= e &]

stDescentVer4[A_, f_, e_] := Module[
  {n = Length@f, x0, x1, x2, r1 = 10, l1, k = 1},
  x0 = ConstantArray[0, {n, 1}];
  x1 = x0;
  While[Norm[r1] >= e,
   r1 = f - A.x1;
   l1 = (Transpose[r1].r1)/(Transpose[(A.r1)].r1)[[1, 1]];
   x2 = x1 + l1 * r1;
   x1 = x2;
   k++;
   ];
  {x1, k}
 ]

stDescentVer5[A_, f_, e_] := Module[
  {n = Length@f, x0, x1, x2, r1 = 10, l1, delta = 10, k = 1},
  x0 = ConstantArray[0, {n, 1}];
  x1 = x0;
  While[Norm[delta] >= e,
   r1 = f - A.x1;
   l1 = (Transpose[r1].r1)/(Transpose[(A.r1)].r1)[[1, 1]];
   x2 = x1 + l1 * r1;
   delta = Norm[x2 - x1];
   x1 = x2;
   k++;
   ];
  {x1, k}
 ]
```

Рисунок 2.1 – код функции, реализующей метод сопряжённых направлений

2.3. Результаты вычислений и оценка точности

Для оценки точности решения рассмотрим входные данные, тип данных входных матриц – числа с плавающей точкой.

Входные данные: $A1 = \begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix}$

$$A3 = \begin{pmatrix} 4.33 & -1.12 & -1.08 & 1.14 \\ -1.12 & 4.33 & 0.24 & -1.22 \\ -1.08 & 0.24 & 7.21 & -3.22 \\ 1.14 & -1.22 & -3.22 & 5.43 \end{pmatrix}, f = \begin{pmatrix} 0.3 \\ 0.5 \\ 0.7 \\ 0.9 \end{pmatrix}$$

Все 5 функций решают примерно с одинаковой нормой невязки и получают довольно точный ответ. Один из результатов, вычисленный с помощью собственной функции, например, функции 2 и для большей точности выберем количество итераций $k = 100$:

На рисунке 2.2 представлено решение, полученное 2 собственной функцией.

```
stDescentVer1[a1, f, 100]
{{-1.25779}, {0.0434873}, {1.03917}, {1.48239}}
```

Рисунок 2.2. – решение, полученное собственной функцией 2

Решение с помощью встроенной функции $LinearSolve$ представлено на рисунке 2.3:

```
LinearSolve[a1, f]
{{-1.25779}, {0.0434873}, {1.03917}, {1.48239}}
```

Рисунок 2.3. – решение, полученное встроенное функцией LinearSolve

На рисунке 2.4 представлены нормы векторов невязки, полученных с помощью собственной функции 2 и встроенной.

```
Norm[f - a1.stDescentVer1[a1, f, 100]]
5.16164×10^-15

Norm[f - a1.LinearSolve[a1, f]]
5.55112×10^-17
```

Рисунок 2.4. – нормы векторов невязки полученных решений

Нормы векторов невязки практически не отличаются.

Стоит отметить, что остальные функции работают также хорошо, поэтому предлагаю не сравнивать все остальные функции с встроенной, а сравнить их точность между собой в зависимости от выбранных данных для прекращения итерационного процесса.

Для начала следует описать особенности каждой из функций:

1, 2) Критерий прекращения K_{max}

3, 4) Критерий прекращения малость невязки

5)Критерий прекращения близость к решению

Предлагается далее сравнить схожие между собой функции на всех входных данных:

1) 1 – синяя и 2 – желтая функции:

Рассматривается промежуток $0 < k < 100$ на оси X и $0 < r < 0.0001$ на оси У. На рисунках 2.5 и 2.6 представлены сравнения 1 и 2 собственной функцией с помощью критерия прекращения итерационного процесса по числу итераций на входных данных $A1$ и $A2$ соответственно.

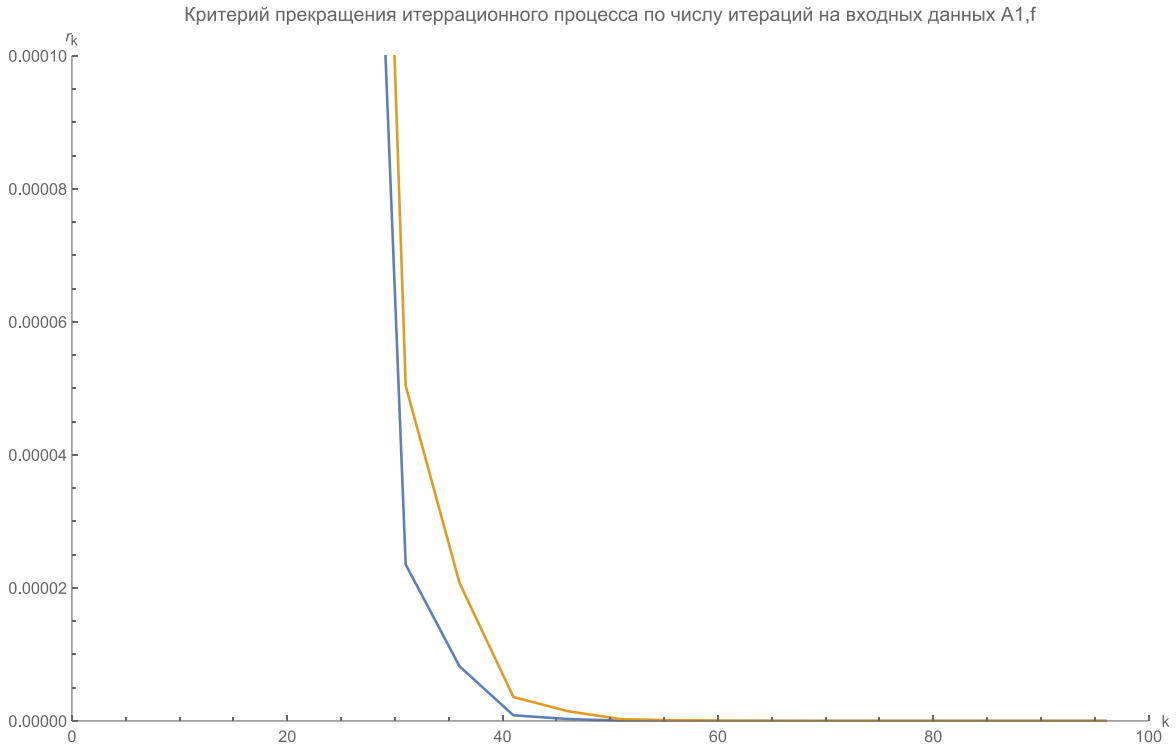

Рисунок 2.5. – сравнение 1 и 2 функции по входным данным A1 по числу итераций

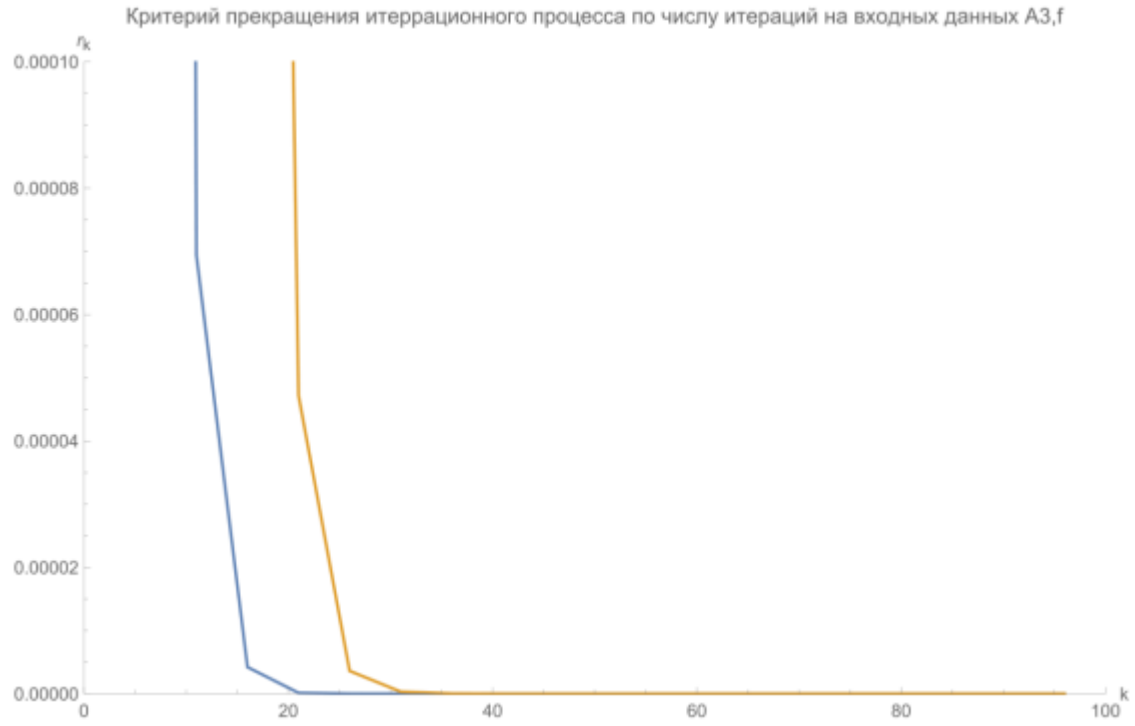

Рисунок 2.6. – сравнение 1 и 2 функции по входным данным А3 по числу итераций

Стоит подметить, что обе функции показывают себя по-разному при разных входных данных, однако в общем случае при $k > 50$ обе функции вычисляют достаточно точно и их норма вектора невязки практически равна нулю.

2) 3 - синяя и 4 - желтая функции:

Рассматривается промежуток $0 < \varepsilon < 1$ на оси X и $0 < r < 0.0001$ на оси У. На рисунках 2.7 и 2.8 представлены сравнения 3 и 4 функции при помощи критерия прекращения итерационного процесса по малости невязки на входных данных $A1$ и $A3$ соответственно.

Рисунок 2.7. – сравнение 3 и 4 функции по входным данным A1 по малости невязки

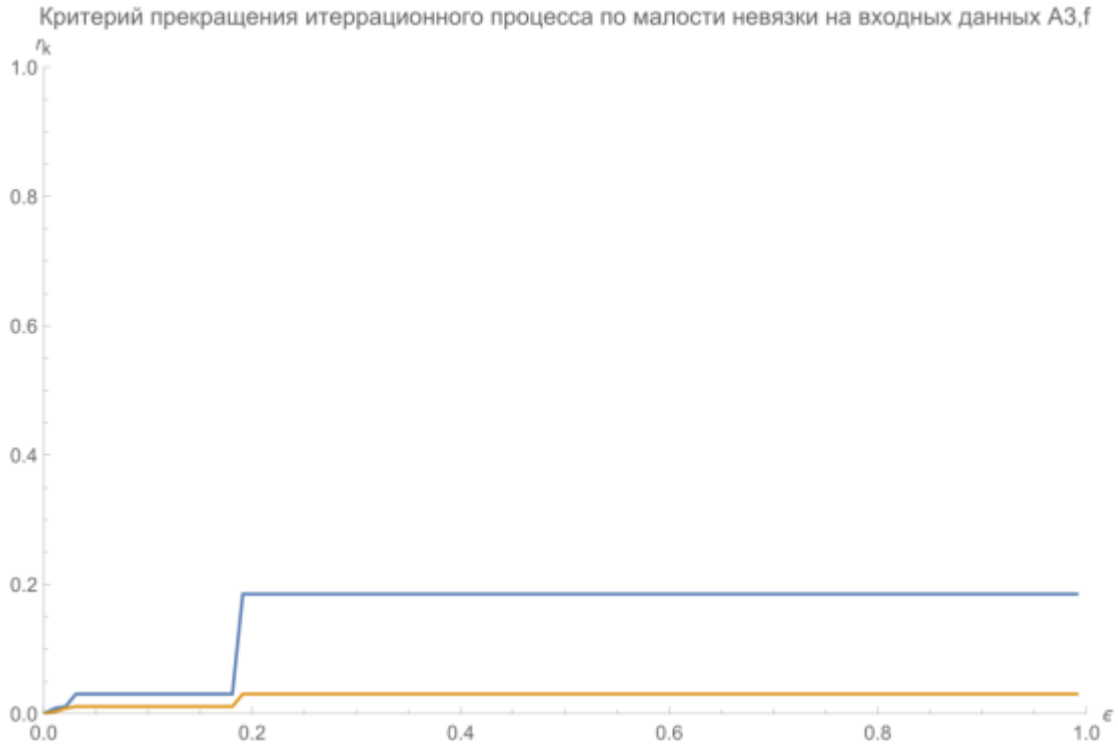

Рисунок 2.8. – сравнение 3 и 4 функции по входным данным A3 по малости невязки

По графикам видно, что при больших значениях ε нормы векторов невязки обоих функций могут отличаться, однако при малых значениях ε нормы

векторов обоих стремятся к нулю, что и логично и доказывает правильность итерационного метода.

3) Стоит также рассмотреть отдельно 4 функцию.

Она помимо вычисленного решения запоминают число итераций k, после которой норма невязки стала меньше заданного числа.

Для входных данных видно, что при уменьшении заданного числа в среднем при k = 9 норма вектора невязки практически равна 0, поскольку меньше очень малого числа.

4) Последняя функция, которую стоит рассмотреть – функция 5:

Рассматривается промежуток $0 < \delta < 1$ на оси X и $0 < k < 10$ на оси У. На рисунках 2.9, 2.10 и 2.11 представлены изменение функции 5 в зависимости от номера итерации, на котором итерационный процесс по малости невязки прекратился, критерия прекращения итерационного процесса по близости к решению на входных данных $A1$ и $A3$ соответственно.

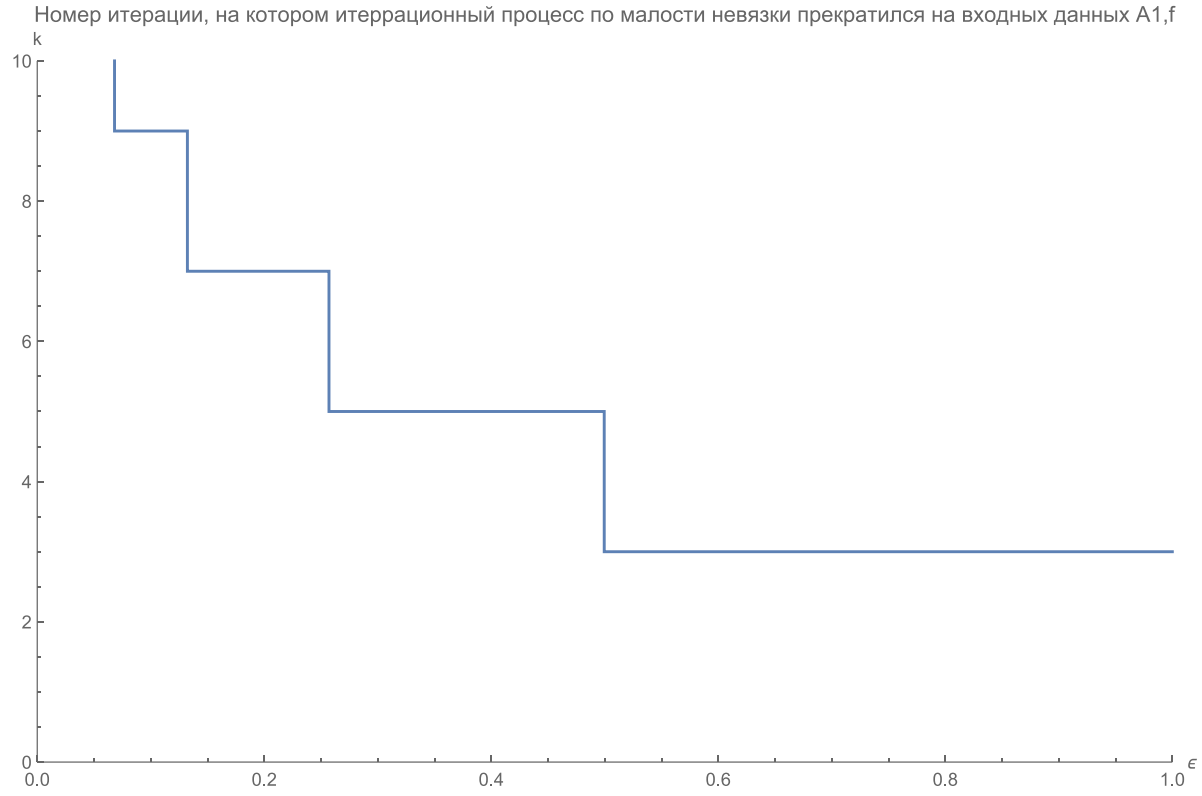

Рисунок 2.9. – зависимость 5 функции по входным данным A1 по номеру итераций, на котором невязка стала меньше заданной границы

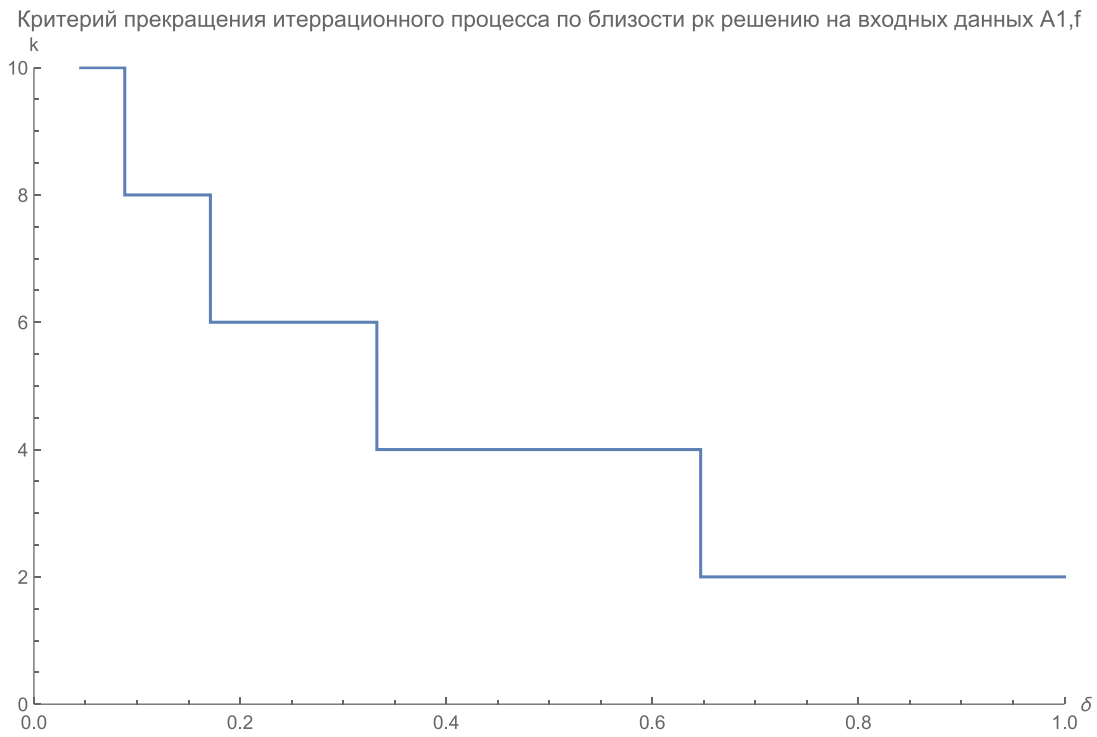

Рисунок 2.10. – зависимость 5 функции по входным данным A1 по критерию близости к решению

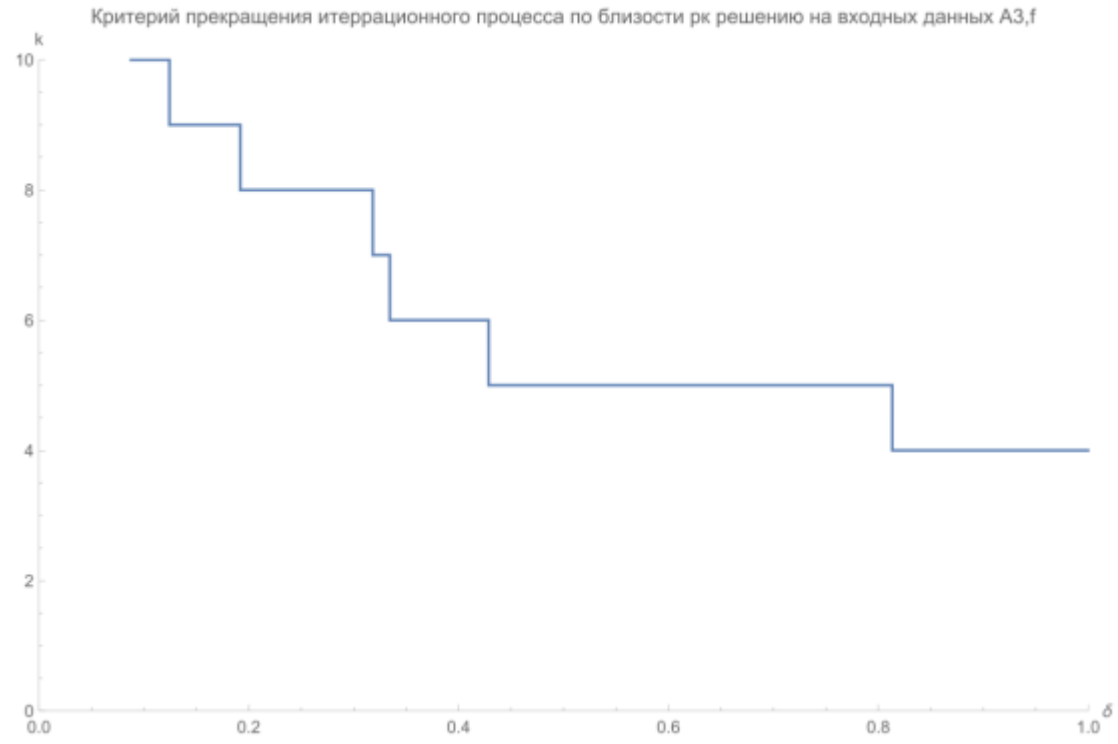

Рисунок 2.11. – зависимость 5 функции по входным данным A3по критерию близости к решению

По данным графикам видно, что при увеличении значения k, число, с которым сравнивается, норма соседних приближений уменьшается и стремится к 0, следовательно, это означает, что функция также работает корректно. Аналогично с графиком, где выбирается число, с которым сравнивается норма вектора невязки.

3. РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ ИТЕРАЦИОННЫМ МЕТОДОМ СЕКУЩИХ (ПРОСТЕЙШИЕ АППРОКСИМАЦИИ ЧАСТНЫХ ПРОИЗВОДНЫХ)

3.1. Описание метода и алгоритм поиска решения

1. Суть метода и алгоритм решения:

Дано уравнение $\mathbf{f}(\mathbf{x}) = \mathbf{0} \Leftrightarrow \begin{cases} f_1(x_1, \dots, x_s) = 0 \\ \dots \dots \dots \\ f_s(x_1, \dots, x_s) = 0 \end{cases}$

Искомым решением является вектор $x = (x_1, \dots, x_s)^T$, который строится с помощью итерационного метода.

Данный метод является дискретной модификацией метода Ньютона-Рафсона, в котором каждое новое итерационное решение $x_{k+1} = x_k - f_k'^{-1} f_k$, где $x_0 -$ начальное приближение, $f_k = f(x_k),\ f_k'^{-1} = f'^{-1}(x_k)$

Особенностью метода секущих является представление f_k' приблизительно равно $H_k^{-1}\Gamma_k$, а, следовательно, $f_k'^{-1}$ приблизительно равно $\Gamma_k^{-1} H_k$, где матрица $H_k -$ диагональная матрица приращений аргумента, каждый элемент h_{ii} можно принять за разность x_{k+1} и x_k

$+$ некоторое малая величина, чтобы избежать случая вырожденной матрицы

Матрица

Γ_k представляется матрицей приращений функций по каждой переменной,

То есть $\gamma_{ij} = f_i(x_k + h_j e_j) - f_i(x_k)$, где $e_j -$ j столбец единичной матрицы размера sxs

Таким образом, конечная формула нахождения следующего итерационного приближения: $x_{k+1} = x_k - \Gamma_k^{-1} H_k f_k$, где на каждом следующем шаге, Γ_k, H_k, f_k повторно вычисляются

Существуют 3 основных критерия прекращения итерационного процесса:

1) По числу итераций. $k < K_{max}$, где K_{max} задается пользователем
2) По близости к решению нормы разности $x_{k+1} - x_k$, которая должна быть меньше заданного δ

3) По малости нормы значения функции в текущем итерационном методе. Такая норма должна быть меньше некоторого ε, которое задается пользователем

Для корректности работы стоит использовать как минимум два критерия прекращения итерационного процесса. Мною выбран 1 и 3 критерий.

3.2. Программная реализация метода

Метод секущих: простейшие аппроксимации частных производных реализован с помощью Wolfram Mathematica:

Реализация представлена на рисунке 3.1

```
p = {};
e = 0.00000001;
k = 0;
kmax = 600;
While[k ≤ kmax ∧ Norm@f[Sequence @@ Flatten@x0] ≥ e,
 AppendTo[p, Flatten@x0];
 jacob  =  Table[D[f[q, w], i], {i, {q, w}}];
 jInv = Inverse[jacob] /. Thread[Rule[{q, w}, Flatten@x0]];
 x1 = x0 - jInv.f[Sequence @@ Flatten@x0];
 x0 = x1;
 k++;

]
```

Рисунок 3.1. – реализация метода секущих

Параметр $kmax$ задается пользователем и определяет максимальное количество итераций. Параметр e также задается пользователем и определяет число, норма значения исходной функции от текущего итерационного решения которого должна быть меньше этого числа, чтобы остановить итерационный процес.

3.3. Результаты вычислений и оценка точности

Для оценки точности решения рассмотрим входные данные, тип данных входящих систем – матрица, где каждая строка - нелинейное уравнение.

Входные данные:

$$f(x) = \begin{pmatrix} x_1^2 - x_2^2 - 1 \\ x_1 x_2^3 - x_2 - 1 \end{pmatrix}, x_0 = \begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix}$$

Рассмотрим полученное решение с помощью написанной мною функции и сравним ее точность со встроенной функцией.

Собственная реализация при $kmax = 600, \varepsilon = 1.\times 10^{-18}$. Полученные данные представлены на рисунке 3.2:

```
Flatten@x0

{1.50284, 1.12185}

Norm@f[Sequence @@ Flatten@x0]

8.00593 × 10^-16
```

Рисунок 3.2. – решение и норма невязки, полученные с помощью собственной реализации

Решение с помощью встроенной функции $NSolve$. Полученные данные представлены на рисунке 3.3:

```
x = {q, w} /. NSolve[f[q, w] == 0, {q, w}, Reals] // First

{1.50284, 1.12185}

Norm@f[Sequence @@ Flatten@x]

3.33067 × 10^-15
```

Рисунок 3.3. – решение и норма невязки, полученные с помощью встроенной реализации

Нормы векторов невязки практически не отличаются, что означает, что написанная собственная функция работает довольно таки точно, не хуже, чем встроенные функции системы $Wolfram\ Mathematica$.

Также для еще одного подтвержения правильности собственной реализации рассмотрим диаграмму разброса данных, представленную на рисунке 3.4.:

$$\text{ListPlot}[p\ [\![30;;130]\!] \rightarrow \text{Range}[\text{Length}[p\ [\![30;;130]\!]\ ]], \text{PlotStyle} \rightarrow \text{PointSize}[\text{Large}]]$$

На данном графике (рисунок 3.4.) видно, что с повышением числа k найденное решение на k итерации становится все ближе и ближе к искомому решению. Сами решения «кучкуются» вокруг искомого решения.

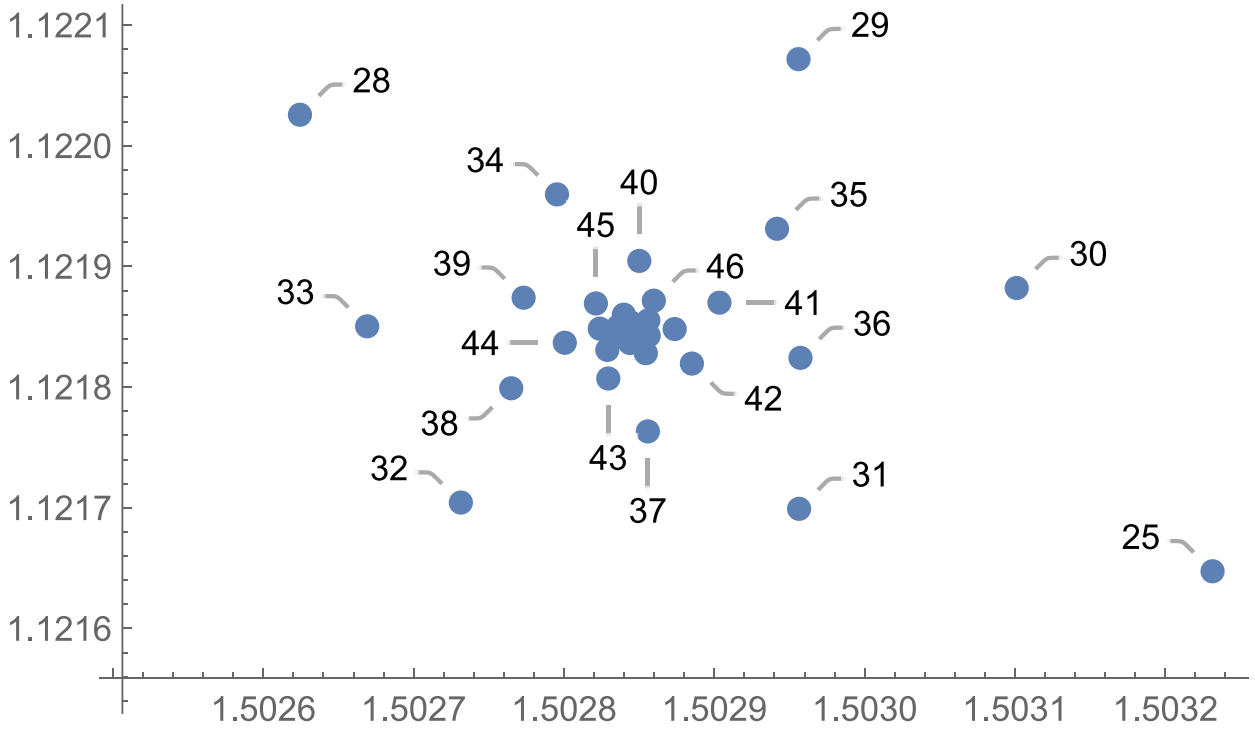

Рисунок 3.4. – диаграмма разброса данных

4. РЕШЕНИЕ ОБЫЧНОЙ ПРОБЛЕМЫ СОБСТВЕННЫХ ЗНАЧЕНИЙ ПРЯМЫМ МЕТОДОМ. ВЫЧИСЛЕНИЕ ХАРАКТЕРИСТИЧЕСКОГО ПОЛИНОМА ИЛИ ЕГО КОЭФФИЦИЕНТОВ. МЕТОД ЛЕВЕРЬЕ

4.1. Описание метода и алгоритм поиска решения

Дано квадратная матрица A размера $n_x n$. Необходимо найти характеристический полином матрицы A:

$$|A - \lambda E| = (-1)^n(\lambda^n - p_1\lambda^{n-1} - \cdots - p_n = (-1)^n\phi(\lambda)$$

Основной задачей является восстановление коэффициентов p_i характеристического полинома заданной матрицы A

Данный метод основан на формулах Ньютона для сумм степеней корней алгебраического уравнения.

Первым шагом вычисляются следы матрицы A^k, где $k = 1, \dots, n$. След матрицы A^k: $s_k = trA^k = \sum_{i=1}^{n} \lambda_i^k, k = 1, \dots, n$. Из данной формулы следует второй шаг. На этом шаге вычисляются искомые коэффициенты p_i искомого характеристического полинома.

Пусть $p_1 = s_1$. Тогда $kp_k = s_k - p_1 s_{k-1} - \cdots - p_{k-1}s_1$, $k = 1, 2, \dots, n$

Таким образом, $p_i = \frac{s_k - p_1 s_{k-1} - \cdots - p_{k-1}s_1}{k}, k = 1, 2, \dots, n$

4.2. Программная реализация метода

Метод Леверье реализован с помощью Wolfram Mathematica.

Входные данные: квадратная матрица A размера $n_x n$. Выходные данные: восстановленный характеристический полином входной матрицы.

На рисунке 4.1 представлена реализация метода Леверье.

```
leverrierMethod[a_] := Module[{n = Length@a, p, s, result},
  s = Table[Tr@Nest[#.a &, a, k], {k, 0, n - 1}];
  p = {s[[1]]};
  Do[
   AppendTo[p, (s[[k]] - Plus @@ Table[p[[i]] * s[[k - i]], {i, k - 1, 1, -1}])/k];
   , {k, 2, n}];
  result = (-1)^n (x^n - Plus @@ Table[p[[k]] * x^(n-k), {k, 1, n}]) // Expand
 ];
```

Рисунок 4.1. – код программной реализации метода

4.3. Результаты вычислений и оценка точности решения

Рассмотрим 5 различных матриц A. Будем оценивать точность и корректность решения путем сравнений, найденных с помощью собственной реализации и встроенной характеристических многочленов с помощью встроенной функции сравнения $Equal$.

В данном случае функция будет сравнивать соответствующие коэффициенты при соответствующих степенях λ. Если данные коэффициенты отличаются на малое число, а точнее на машинный эпсилон, тогда считается, что такие коэффициенты равны.

Вычисление с помощью встроенных функций будет осуществляться по определению: $|A - \lambda E|$

Соответственно из равенства многочленов следует и равенство полученных корней, однако все же после равенства сверим полученные корни, отсортировав список найденных собственных значений и попарно вычисляя модуль их разности и посчитаем норму полученного вектора

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

На рисунке 4.2 представлены полученные характеристические многочлены с помощью обоих функций на 1 входных данных.

Собственная	Встроенная
$0.286152 - 2.11186\,x + 4.752\,x^2 - 4.\,x^3 + x^4$	$0.286152 - 2.11186\,x + 4.752\,x^2 - 4.\,x^3 + x^4$

Рисунок 4.2. – полученные характеристические многочлена на 1 входных данных

Проверка на равенство полиномов представлена на рисунке 4.3:

```
leverrierMethod[a1] == Det[a1 - x * IdentityMatrix[Length@a1]]
True
```

Рисунок 4.3. – проверка полиномов на равенство

Проверка нормы разности попарно найденных собственных чисел представлена на рисунке 4.4:

```
(x /. Sort@Solve[leverrierMethod[a1] == 0, x]) - (x /. Solve[Det[a1 - x * IdentityMatrix[Length@a1]] == 0, x])
% // Norm
{6.10623 × 10^-16, -4.10783 × 10^-15, 4.44089 × 10^-15, -8.88178 × 10^-16}
6.14471 × 10^-15
```

Рисунок 4.4. – проверка нормы разности попарно найденных собственных чисел на 1 входных данных

2. $A = \begin{pmatrix} 4.33 & -1.12 & -1.08 & 1.14 \\ -1.12 & 4.33 & 0.24 & -1.22 \\ -1.08 & 0.24 & 7.21 & -3.22 \\ 1.14 & -1.22 & -3.22 & 5.43 \end{pmatrix}$

На рисунке 4.5 представлены полученные характеристические многочлены с помощью обоих функций на 2 входных данных.

Собственная	Встроенная
$445.432 - 439.773\,x + 151.727\,x^2 - 21.3\,x^3 + x^4$	$445.432 - 439.773\,x + 151.727\,x^2 - 21.3\,x^3 + x^4$

Рисунок 4.5. – полученные характеристические многочлена на 2 входных данных

Проверка на равенство полиномов представлена на рисунке 4.6:

```
leverrierMethod[a2] == Det[a2 - x * IdentityMatrix[Length@a2]]
True
```

Рисунок 4.6. – проверка полиномов на равенство

Проверка нормы разности попарно найденных собственных чисел представлена на рисунке 4.7:

```
(x /. Sort@Solve[leverrierMethod[a2] == 0, x]) - (x /. Solve[Det[a2 - x * IdentityMatrix[Length@a2]] == 0, x])
% // Norm
{4.52971 × 10^-14, -9.54792 × 10^-14, 5.50671 × 10^-14, -1.06581 × 10^-14}
1.19641 × 10^-13
```

Рисунок 4.7. – полученные характеристические многочлена на 3 входных данных

3. $A = \begin{pmatrix} 1. & 0.17 & -0.25 & 0.54 \\ 0.47 & 1. & 0.67 & -0.32 \\ -0.11 & 0.35 & 1. & -0.74 \\ 0.55 & 0.43 & 0.36 & 1. \end{pmatrix}$

На рисунке 4.8 представлены полученные характеристические многочлены с помощью обоих функций на 3 входных данных.

Собственная	Встроенная
$0.638638 - 3.38261\,x + 5.7651\,x^2 - 4.\,x^3 + x^4$	$0.638638 - 3.38261\,x + 5.7651\,x^2 - 4.\,x^3 + x^4$

Рисунок 4.8. – проверка нормы разности попарно найденных собственных чисел на 3 входных данных

Проверка на равенство полиномов представлена на рисунке 4.8:

```
leverrierMethod[a3] == Det[a3 - x * IdentityMatrix[Length@a3]]
True
```

Рисунок 4.9. – проверка полиномов на равенство

Проверка нормы разности попарно найденных собственных чисел представлена на рисунке 4.10:

```
(x /. Sort@Solve[leverrierMethod[a3] == 0, x]) - (x /. Solve[Det[a3 - x * IdentityMatrix[Length@a3]] == 0, x])
% // Norm
```

```
{0., 0., 0. + 0. i, 0. + 0. i}
```

```
0.
```

Рисунок 4.10. – проверка нормы разности попарно найденных собственных чисел на 3 входных данных

4. $A = \begin{pmatrix} b+c & b & 0 & 0 & 0 & 0 \\ b & c & b & 0 & 0 & 0 \\ 0 & b & c & b & 0 & 0 \\ 0 & 0 & b & b+c & b & 0 \\ 0 & 0 & 0 & b & c & b \\ 0 & 0 & 0 & 0 & b & c \end{pmatrix}$

На рисунке 4.11 представлены полученные характеристические многочлены с помощью обоих функций на 4 входных данных.

Собственная	Встроенная
$5\,b^5\,c + 4\,b^4\,c^2 - 7\,b^3\,c^3 - 4\,b^2\,c^4 + 2\,b\,c^5 + c^6 - 5\,b^5\,x - 8\,b^4\,c\,x + 21\,b^3\,c^2\,x + 16\,b^2\,c^3\,x - 10\,b\,c^4\,x - 6\,c^5\,x + 4\,b^4\,x^2 - 21\,b^3\,c\,x^2 - 24\,b^2\,c^2\,x^2 + 20\,b\,c^3\,x^2 + 15\,c^4\,x^2 + 7\,b^3\,x^3 + 16\,b^2\,c\,x^3 - 20\,b\,c^2\,x^3 - 20\,c^3\,x^3 - 4\,b^2\,x^4 + 10\,b\,c\,x^4 + 15\,c^2\,x^4 - 2\,b\,x^5 - 6\,c\,x^5 + x^6$	$5\,b^5\,c + 4\,b^4\,c^2 - 7\,b^3\,c^3 - 4\,b^2\,c^4 + 2\,b\,c^5 + c^6 - 5\,b^5\,x - 8\,b^4\,c\,x + 21\,b^3\,c^2\,x + 16\,b^2\,c^3\,x - 10\,b\,c^4\,x - 6\,c^5\,x + 4\,b^4\,x^2 - 21\,b^3\,c\,x^2 - 24\,b^2\,c^2\,x^2 + 20\,b\,c^3\,x^2 + 15\,c^4\,x^2 + 7\,b^3\,x^3 + 16\,b^2\,c\,x^3 - 20\,b\,c^2\,x^3 - 20\,c^3\,x^3 - 4\,b^2\,x^4 + 10\,b\,c\,x^4 + 15\,c^2\,x^4 - 2\,b\,x^5 - 6\,c\,x^5 + x^6$

Рисунок 4.11. – проверка нормы разности попарно найденных собственных чисел на 4 входных данных

Проверка на равенство полиномов представлена на рисунке 4.12:

```
leverrierMethod[a4] == Det[a4 - x * IdentityMatrix[Length@a4]]
```

```
True
```

Рисунок 4.12. – проверка полиномов на равенство

Проверка нормы разности попарно найденных собственных чисел представлена на рисунке 4.13:

```
(x /. Sort@Solve[leverrierMethod[a4] == 0, x]) - (x /. Solve[Det[a4 - x * IdentityMatrix[Length@a4]] == 0, x])
% // Norm
```

```
{0, 0, 0, 0, 0, 0}
```

```
0
```

Рисунок 4.13. – проверка нормы разности попарно найденных собственных чисел на 4 входных данных

5. $A = \begin{pmatrix} c & f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b & c & f & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & b & c & f & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b & c & f & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b & c & f & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & b & c & f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & b & c & f & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & b & c & f & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & c & f \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & c \end{pmatrix}$

На рисунке 4.14 представлены полученные характеристические многочлены с помощью обоих функций на 5 входных данных

Собственная	Встроенная
$c^{10} - 9\,b\,c^8\,f + 28\,b^2\,c^6\,f^2 - 35\,b^3\,c^4\,f^3 + 15\,b^4\,c^2\,f^4 - b^5\,f^5 - 10\,c^9\,x + 72\,b\,c^7\,f\,x - 168\,b^2\,c^5\,f^2\,x + 140\,b^3\,c^3\,f^3\,x - 30\,b^4\,c\,f^4\,x + 45\,c^8\,x^2 - 252\,b\,c^6\,f\,x^2 + 420\,b^2\,c^4\,f^2\,x^2 - 210\,b^3\,c^2\,f^3\,x^2 + 15\,b^4\,f^4\,x^2 - 120\,c^7\,x^3 + 504\,b\,c^5\,f\,x^3 - 560\,b^2\,c^3\,f^2\,x^3 + 140\,b^3\,c\,f^3\,x^3 + 210\,c^6\,x^4 - 630\,b\,c^4\,f\,x^4 + 420\,b^2\,c^2\,f^2\,x^4 - 35\,b^3\,f^3\,x^4 - 252\,c^5\,x^5 + 504\,b\,c^3\,f\,x^5 - 168\,b^2\,c\,f^2\,x^5 + 210\,c^4\,x^6 - 252\,b\,c^2\,f\,x^6 + 28\,b^2\,f^2\,x^6 - 120\,c^3\,x^7 + 72\,b\,c\,f\,x^7 + 45\,c^2\,x^8 - 9\,b\,f\,x^8 - 10\,c\,x^9 + x^{10}$	$c^{10} - 9\,b\,c^8\,f + 28\,b^2\,c^6\,f^2 - 35\,b^3\,c^4\,f^3 + 15\,b^4\,c^2\,f^4 - b^5\,f^5 - 10\,c^9\,x + 72\,b\,c^7\,f\,x - 168\,b^2\,c^5\,f^2\,x + 140\,b^3\,c^3\,f^3\,x - 30\,b^4\,c\,f^4\,x + 45\,c^8\,x^2 - 252\,b\,c^6\,f\,x^2 + 420\,b^2\,c^4\,f^2\,x^2 - 210\,b^3\,c^2\,f^3\,x^2 + 15\,b^4\,f^4\,x^2 - 120\,c^7\,x^3 + 504\,b\,c^5\,f\,x^3 - 560\,b^2\,c^3\,f^2\,x^3 + 140\,b^3\,c\,f^3\,x^3 + 210\,c^6\,x^4 - 630\,b\,c^4\,f\,x^4 + 420\,b^2\,c^2\,f^2\,x^4 - 35\,b^3\,f^3\,x^4 - 252\,c^5\,x^5 + 504\,b\,c^3\,f\,x^5 - 168\,b^2\,c\,f^2\,x^5 + 210\,c^4\,x^6 - 252\,b\,c^2\,f\,x^6 + 28\,b^2\,f^2\,x^6 - 120\,c^3\,x^7 + 72\,b\,c\,f\,x^7 + 45\,c^2\,x^8 - 9\,b\,f\,x^8 - 10\,c\,x^9 + x^{10}$

Проверка на равенство полиномов представлена на рисунке 4.15:

```
leverrierMethod[a5] == Det[a5 - x * IdentityMatrix[Length@a5]]

True
```

Рисунок 4.15. – проверка полиномов на равенство

Проверка нормы разности попарно найденных собственных чисел представлена на рисунке 4.16:

```
(x /. Sort@Solve[leverrierMethod[a4] == 0, x]) - (x /. Solve[Det[a4 - x * IdentityMatrix[Length@a4]] == 0, x])
 % // Norm

{0, 0, 0, 0, 0, 0}

0
```

Рисунок 4.16. – проверка нормы разности попарно найденных собственных чисел на 4 входных данных

По данным примерам видно, что значения полученных «норм» очень малы, следовательно, собственная реализация довольно хорошо решает поставленную задачу.

5. ИНТЕРПОЛЯЦИОННАЯ ФОРМУЛА ЛАГРАНЖА. ТРИГОНОМЕТРИЧЕСКОЕ ИНТЕРПОЛИРОВАНИЕ

5.1. Описание метода

Дан набор точек $(x_i, f(x_i))$, где f – некоторая периодическая функция.

Задача найти такую функцию g, что для любого x из D (в том числе для всех x_i из заданного набора), $g(x) = f(x)$(приблизительно)

Данный метод является модификацией алгебраического интерполирования, поскольку для периодических функций такой метод не подходит, так как алгебраические полиномы не являются периодическими функциями. Для решения данной проблемы используются тригонометрические функции.

Тригонометрическим полиномом от переменной x называется выражение вида: $g_n(x) = a_0 + \sum_{k=1}^{n}(a_k cos kx + b_k sin kx)$ при фиксированных постоянных $a_0, a_1, a_2, \dots, a_n, b_1, \dots, b_n$ – вещественных или комплексных. В данной задаче рассматриваем только вещественные. Если a_n или b_n отлично от нуля, то полином $g_n(x)$ имеет порядок n. В этом случае $a_0, \{a_j, b_j\}_{j=1}^{n}$ называются коэффициентами тригонометрического полинома, а числа a_n и b_n – старшими коэффициентами. Если все числа $\{b_j\}_{j=1}^{n}$ равны нулю, то о полиноме $g_n(x)$ можно говорить как о полиноме по косинусам, если же все числа $\{a_j\}_{j=0}^{n}$ равны нулю – то полином называют полиномом по синусам.

Теорема:

Пусть существуют 2n+1 узлов интерполяции $\{x_j\}_{j=1}^{2n+1}$ таких, что $g_n(x_j) = y_j$ при $j \in \{1, \dots, 2n+1\}$. В данной задаче предполагается, что узлы вещественны и расположены на интервале $[0; 2\pi)$.

Тогда функция $g_n(x) = y1 * \frac{\sin\left(\frac{x-x_2}{2}\right)*\dots*\sin\left(\frac{x-x_{2n+1}}{2}\right)}{\sin\left(\frac{x_1-x_2}{2}\right)*\dots*\sin\left(\frac{x_1-x_{2n+1}}{2}\right)} + \dots +$

$+y_{2n+1} * \frac{\sin\left(\frac{x-x_1}{2}\right)*\dots*\sin\left(\frac{x-x_{2n}}{2}\right)}{\sin\left(\frac{x_{2n+1}-x_1}{2}\right)*\dots*\sin\left(\frac{x_{2n+1}-x_{2n}}{2}\right)}$ удовлетворяет условиям:

$g_n(x_j) = y_j$ при $j \in \{1, \dots, 2n+1\}$.

Эта функция является тригонометрическим полиномом порядка не выше n. При таком ограничении на порядок, полином, удовлетворяющий заданным интерполяционным условиям, определяется единственным образом.

Доказательство:

Функция $g_n(x)$ является полным аналогом интерполяционного алгебраического полинома в форме Лагранжа. Каждое слагаемое $y_j W_j(x)$ в этой сумме «отвечает» исключительно только за свой узел интерполяции (т.е. обеспечивает в нем нужное значение) – при этом не обращает остальные узлы в ноль.

Далее необходимо показать, что функция $g_n(x)$ является тригонометрическим полиномом, т.е. допускает представление вида $a_0 + \sum_{k=1}^{n}(a_k coskx + b_k sinkx)$. Это достигается путем преобразования произведений синусов, стоящих в числителях слагаемых. В каждом таком произведении содержится четное число сомножителей, объединив их попарно и использовав формулы приведения получается следующее:

$$\sin\left(\frac{\mathrm{x}-\mathrm{x}_1}{2}\right)\sin\left(\frac{\mathrm{x}-\mathrm{x}_2}{2}\right) = \frac{1}{2}\left(\cos\left(\frac{\mathrm{x}_2-\mathrm{x}_1}{2}\right) - \cos\left(\mathrm{x} - \frac{x_1+x_2}{2}\right)\right)$$

$$= \frac{1}{2}\cos\left(\frac{x_2-x_1}{2}\right) - \frac{1}{2}\cos\left(\frac{x_1+x_2}{2}\right)\cos(\mathrm{x}) - \frac{1}{2}\sin\left(\frac{\mathrm{x}_1+x_2}{2}\right)\sin x$$

В результате происходит избавление от половинного аргумента под синусами. Дальнейшие преобразования аналогичны.

Ч.т.д.

Таким образом, общая формула явного интерполирования:

$$g_n(x) = \sum_{k=0}^{n} f_k \phi_k(x), f_i = f(x_i), \phi_k(x_i) = \delta_{ik} = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases} => g_n(x_i) = f(x_i),$$

$$i = 0, \dots, n$$

$$n - \text{количество точек в заданном наборе}$$

$$\phi_k(x) = \prod_{j \neq k} \frac{\sin\left(\frac{x - x_j}{2}\right)}{\sin\left(\frac{x_k - x_j}{2}\right)}$$

Или же можно представить иначе:

$$\text{Пусть } B(x) = \prod_{j=0}^{n} \sin\left(\frac{x - x_j}{2}\right), B'(x_k) = \frac{1}{2} \prod_{j \neq k}^{n} \sin\left(\frac{x_k - x_j}{2}\right)$$

Тогда $\phi_k(x)$ приобретает следующий вид:

$$\phi_k(x) = \frac{1}{2} \frac{B(x)}{B'(x) \sin\left(\frac{x_k - x_j}{2}\right)}$$

Набор входных точек определяется пользователем: выбирается область, на которой будет восстановлена функция. Далее выбирается шаг, через который будет выбираться каждая следующая точка. Например, для области $[a, b]$ пусть шаг будет равен $\frac{b-a}{n}$,

тогда исходных аргументов будет n штук , то есть n точек

$$\big(a, f(a)\big), \dots, (b, f(b))$$

5.2. Программная реализация метода

Данный метод реализован с помощью Wolfram Mathematica:

Входные данные: набор точек $(x_i, f(x_i))$. Количество и шаг определяется пользователем. Выходные данные: $g(x)$, такая что $f(x) = g(x)$ в любой точке (приблизительно).

Программная реализация данного метода представлена на рисунке 5.1.

```
ϕ[x_, k_, dots_] := Times @@ Table[If[dots[[k, 1]] == j[[1]], 1, Sin[(x-j[[1]])/2] / Sin[(dots[[k,1]]-j[[1]])/2]], {j, dots}]

g[x_, dts_] := Plus @@ Table[dts[[i, 2]] * ϕ[x, i, dts], {i, 1, Length@dts}]
```

Рисунок 5.1. – программная реализация данного метода

5.3. Визуализация результатов

Выберем несколько периодических функций, по котором построим набор исходных точек. Исследуем точность приближения относительно выбранного шага.

Для удобства будем строить все графики на одной координатной плоскости. Красные точки – точки исходного набора, по которым проводится интерполяция. Синяя кривая – обычный график $f(x)$, оранжевая – график полученной функции $g(x)$

а) Пусть $f(x) = \sin(x)\cos(x)$. Рассмотрим данную функцию на $[0, \pi]$. Для начала пусть шаг = $\frac{\pi}{3}$. На рисунке 5.2 представлен полученный результат.

```
dots = {#, Sin[#] * Cos[#]} & /@ Range[0, π, π/3];
p1 = Plot[Cos[x] Sin[x], {x, 0, Pi}, PlotStyle → Orange];
p2 = ListPlot[dots, PlotStyle → {Red, PointSize[Large]}];
p3 = Plot[g[x, dots], {x, 0, Pi}];
Show[p1, p2, p3]
```

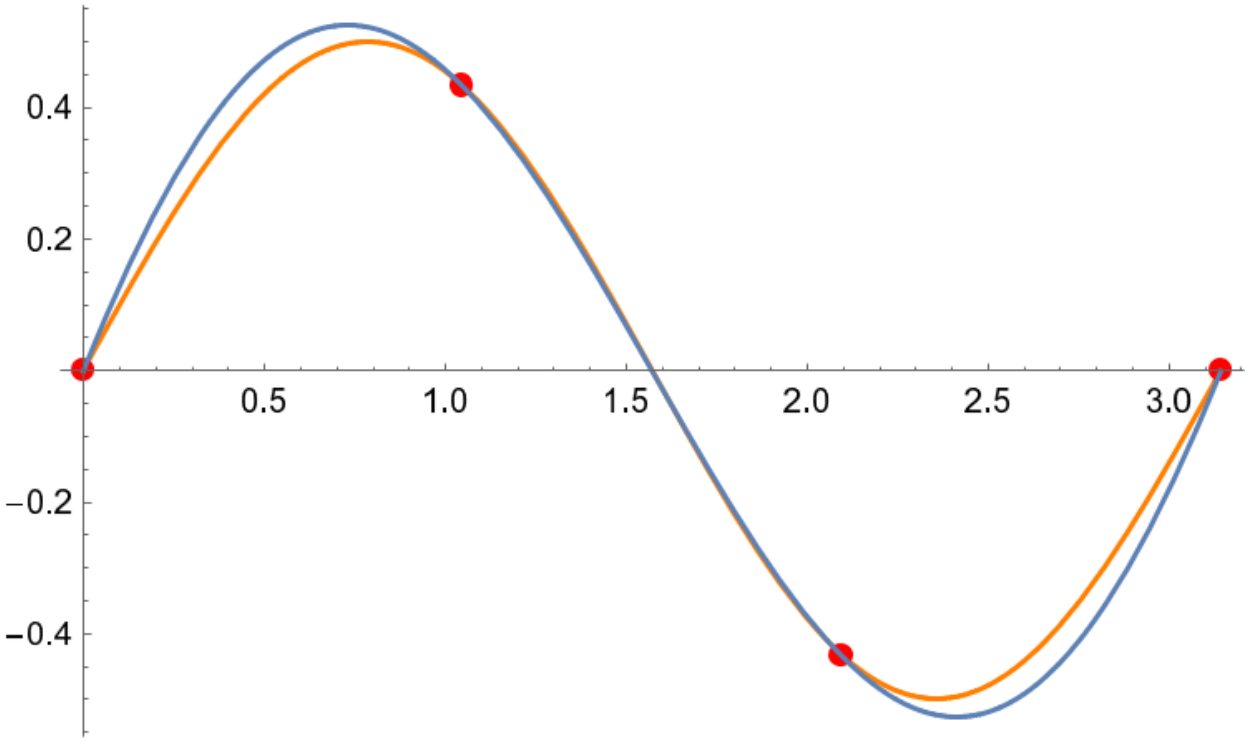

Рисунок 5.2. – график исходной функции и полученные с помощью интерполяции на входной функции а) с шагом $\frac{\pi}{3}$

Пусть шаг = $\pi/4$. Полученные результат представлен на рисунке 5.3:

```
dots = {#, Sin[#] * Cos[#]} & /@ Range[0, π, π/4];
p1 = Plot[Cos[x] Sin[x], {x, 0, Pi}, PlotStyle → Orange];
p2 = ListPlot[dots, PlotStyle → {Red, PointSize[Large]}];
p3 = Plot[g[x, dots], {x, 0, Pi}];
Show[p1, p2, p3]
```

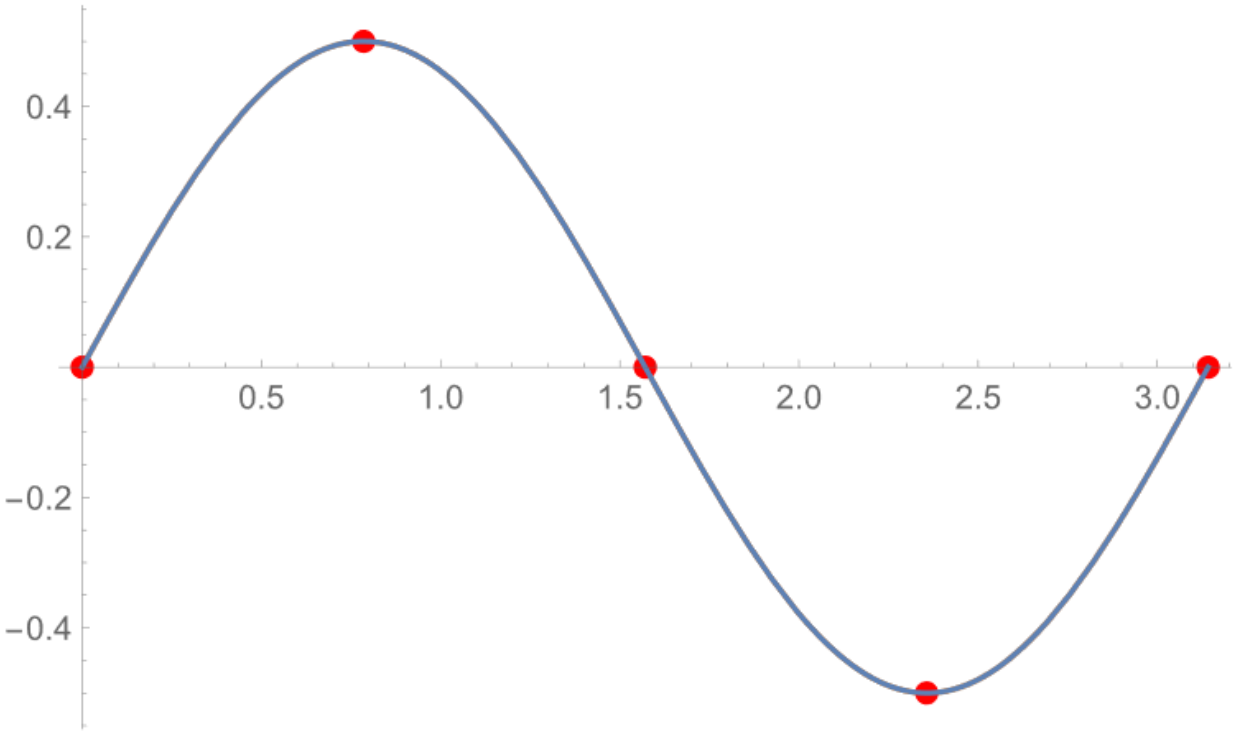

Рисунок 5.3. – график исходной функции и полученные с помощью интерполяции на входной функции а) с шагом $\frac{\pi}{4}$

Из рисунков видно, что при уменьшении шага, то есть при увеличении исходного количества точек, интерполяция проходит более точно, нежели с большим шагом, что логично и подтверждает правильность полученной

функции. И в данном случае, достаточно взять шаг $\leq \frac{\pi}{4}$, чтобы значения при равных точках обоих функций отличались на машинный эпсилон.

б) Стоит рассмотреть более значимый и наглядный пример.

Пусть $f(x) = \cos^3(x)\sin^2(x)$, которая рассматривается также на области $[0;\pi]$.

Начать рассмотрение стоит с шага = $\frac{\pi}{3}$, на рисунке 5.4 представлен полученный результат:

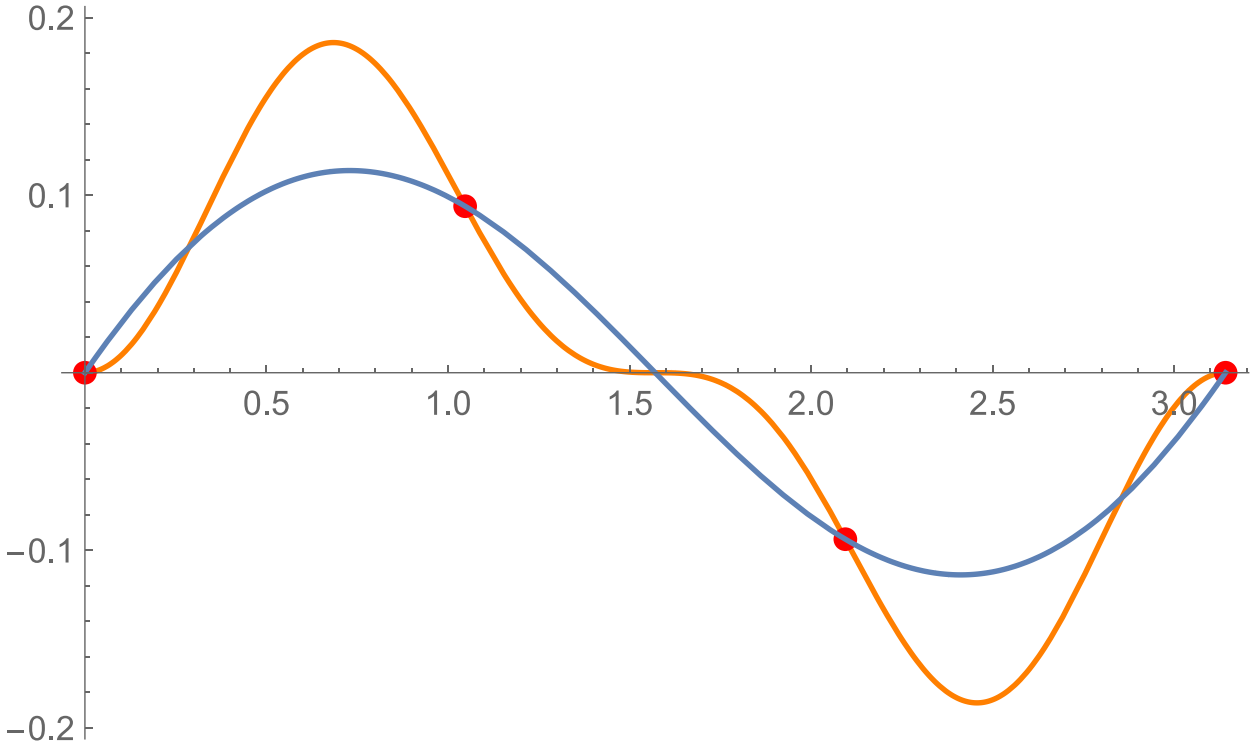

Рисунок 5.4. – график исходной функции и полученные с помощью интерполяции на входной функции б) с шагом $\frac{\pi}{3}$

Далее пусть шаг = $\frac{\pi}{4}$, полученный результат представлен на рисунке 5.5:

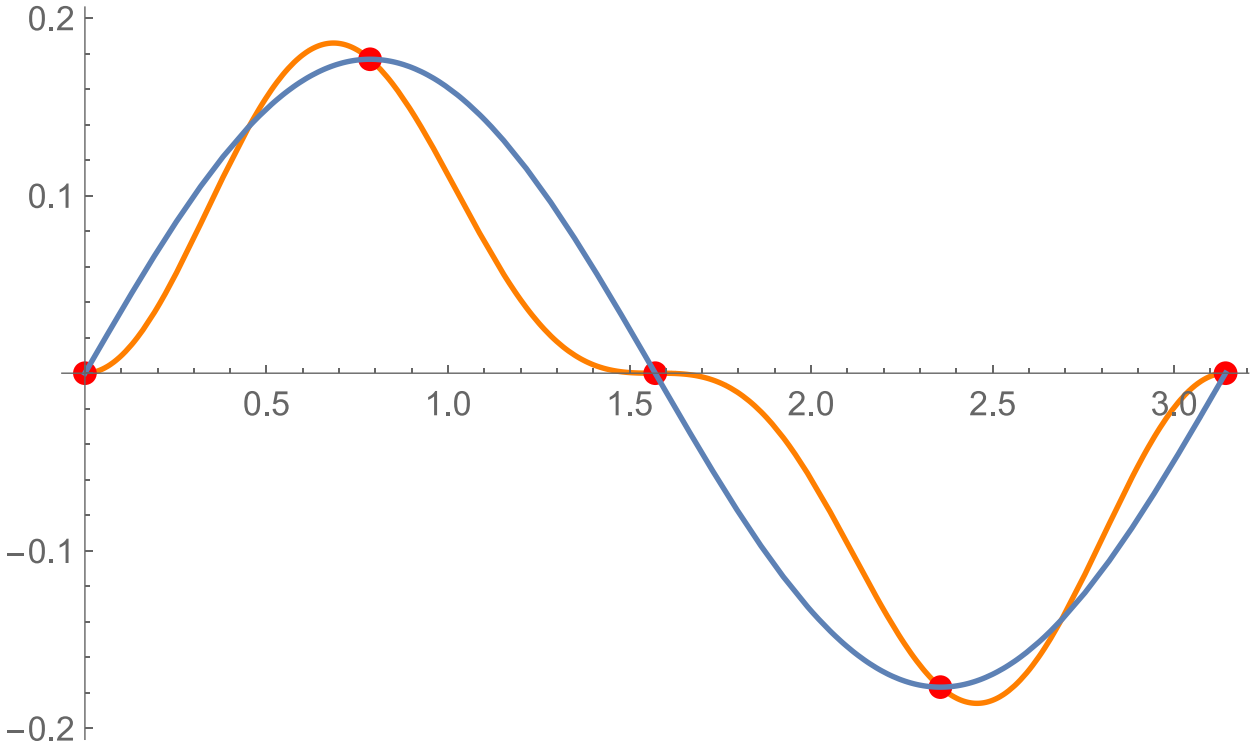

Рисунок 5.5. – график исходной функции и полученные с помощью интерполяции на входной функции б) с шагом $\frac{\pi}{4}$

Далее пусть шаг $=\frac{\pi}{5}$, полученный результат представлен на рисунке 5.6:

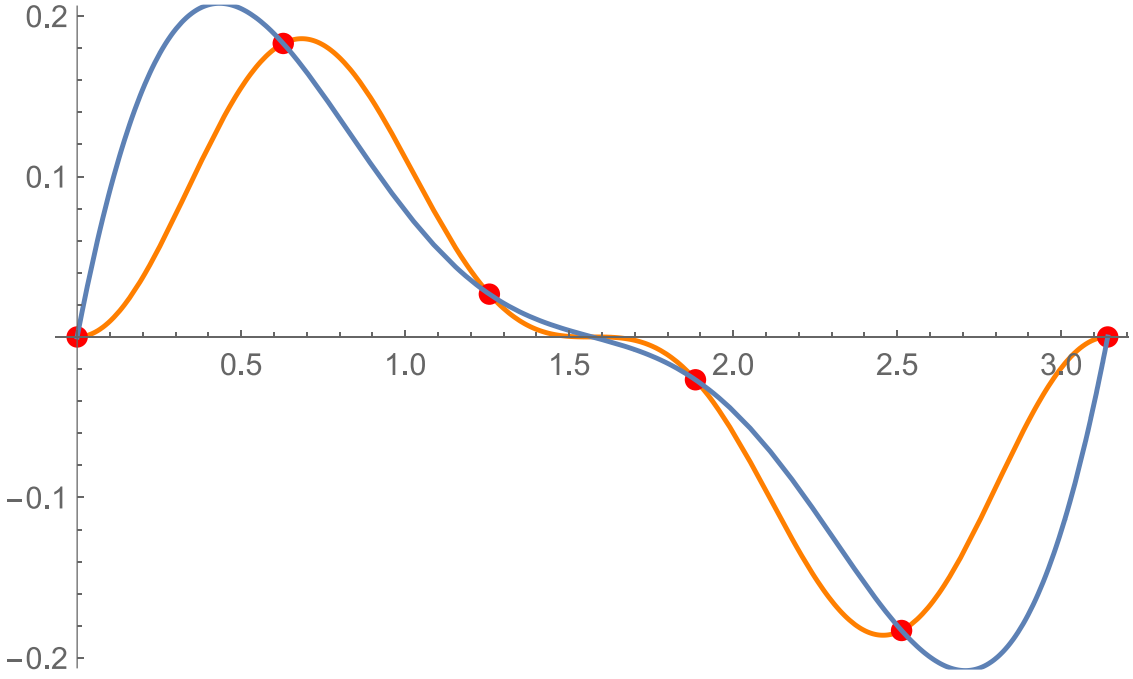

Рисунок 5.6. – график исходной функции и полученные с помощью интерполяции на входной функции б) с шагом $\frac{\pi}{5}$

Для ускорения процесса пусть шаг = $\frac{\pi}{7}$, результат представлен на рисунке 5.7:

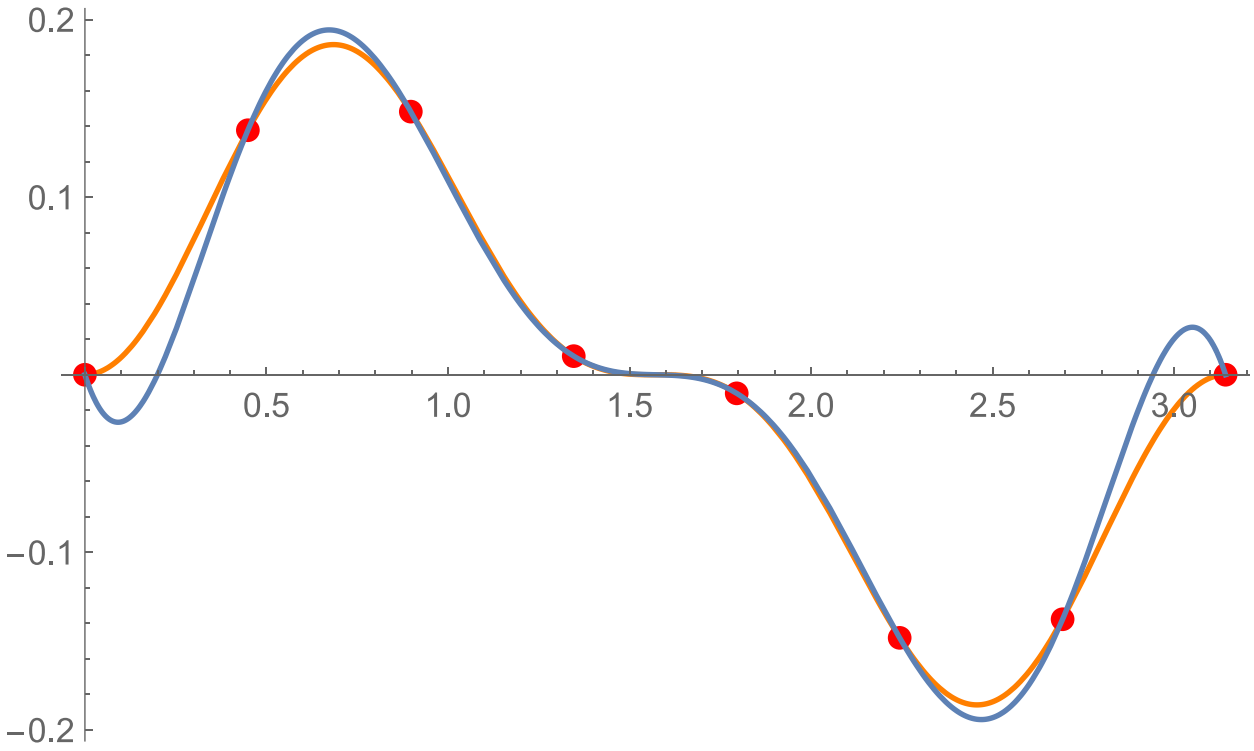

Рисунок 5.7. – график исходной функции и полученные с помощью интерполяции на входной функции б) с шагом $\frac{\pi}{7}$

Пусть завершающим шаг в сравнении = $\frac{\pi}{10}$, результат представлен нарисунке 5.8:

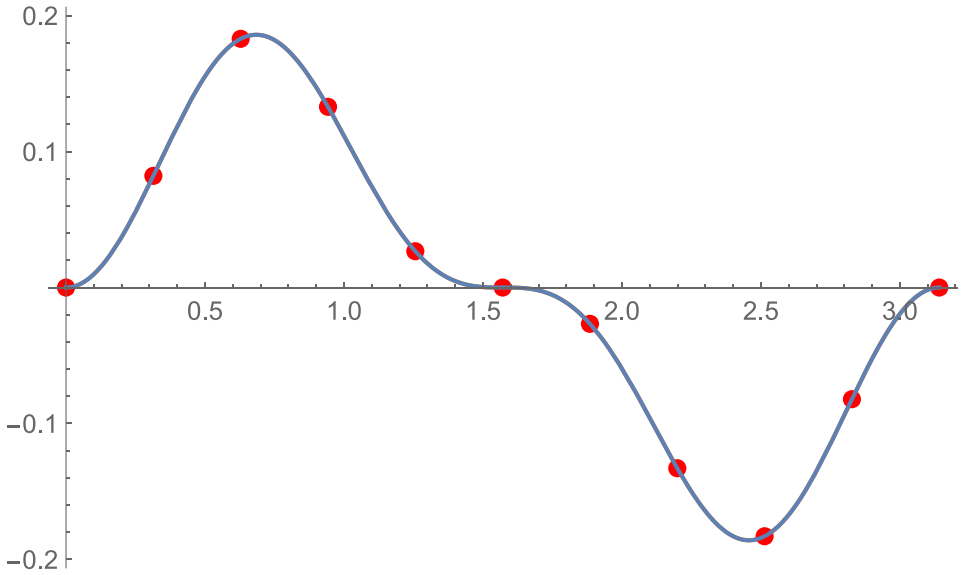

Рисунок 5.8. – график исходной функции и полученные с помощью интерполяции на входной функции б) с шагом $\frac{\pi}{10}$

Итак, можно наблюдать, что с уменьшением шага, найденная функция становилась все больше и больше приближенной к исходной функции. Это также подтверждает правильность собственной реализации, поскольку при увеличении количества точек в исходном наборе также растет и точность полученного решения.

в) Рассмотрим крайний пример, который является более наглядным, чем предыдущие два.

Пусть $f(x) = \cos^{10}(x)\sin^{7}(x)$. Также будем выбирать шаг и уменьшать его с каждым следующим сравнением. Далее будут выведены графики, с шагом $\frac{\pi}{k}$, где $k = 4, 8, \dots, 24, 28$, представленные на рисунках 5.9-5.15.

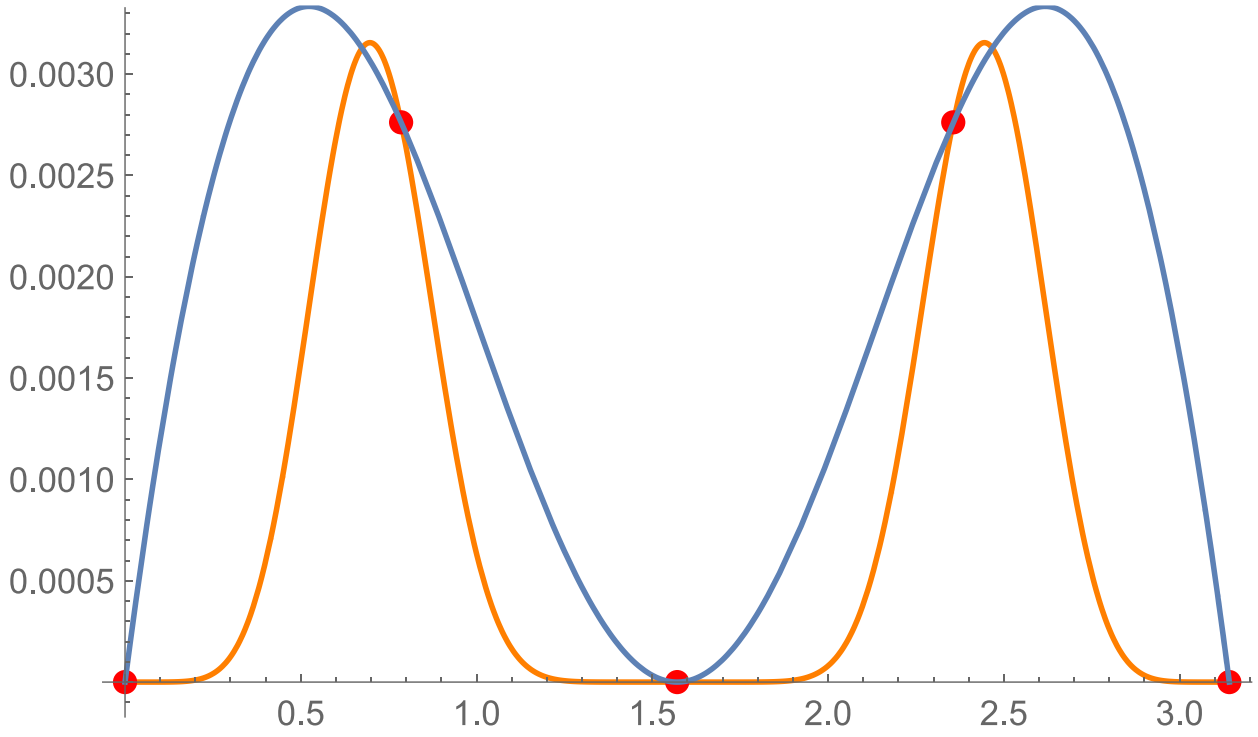

Рисунок 5.9. – график исходной функции и полученные с помощью интерполяции на входной функции в) с шагом $\frac{\pi}{4}$

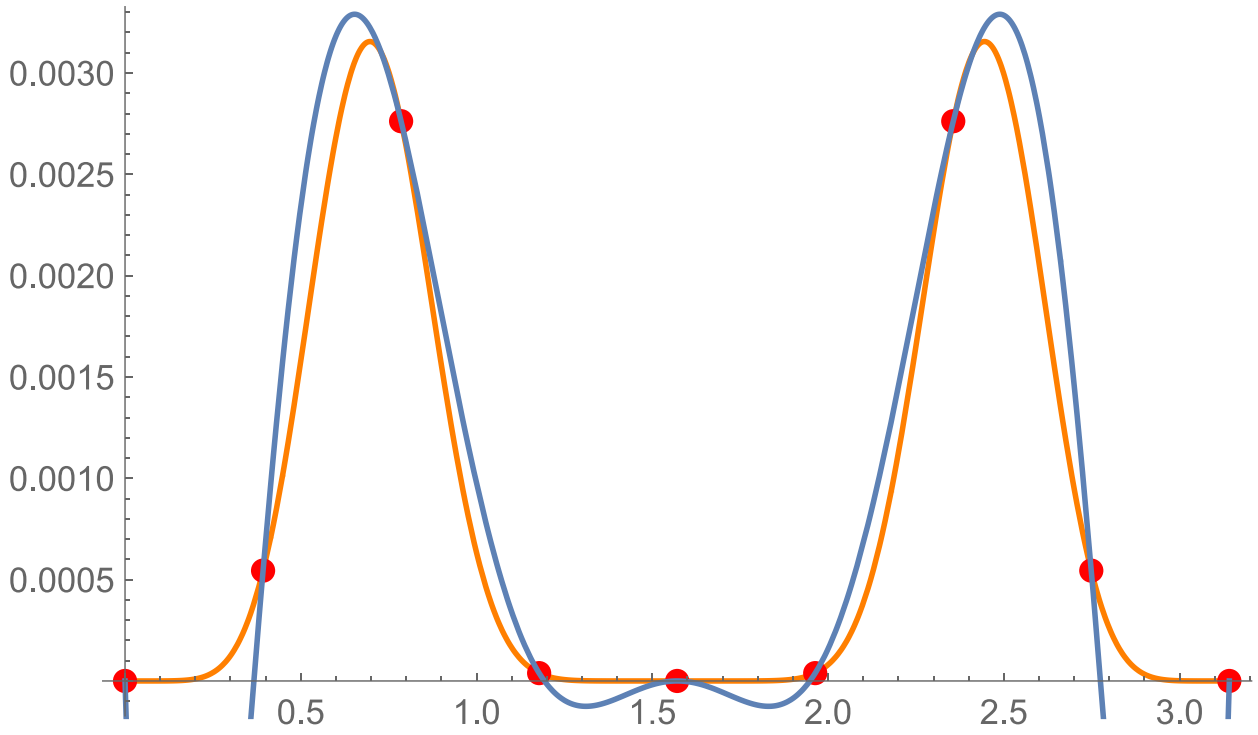

Рисунок 5.10. – график исходной функции и полученные с помощью интерполяции на входной функции в) с шагом $\frac{\pi}{8}$

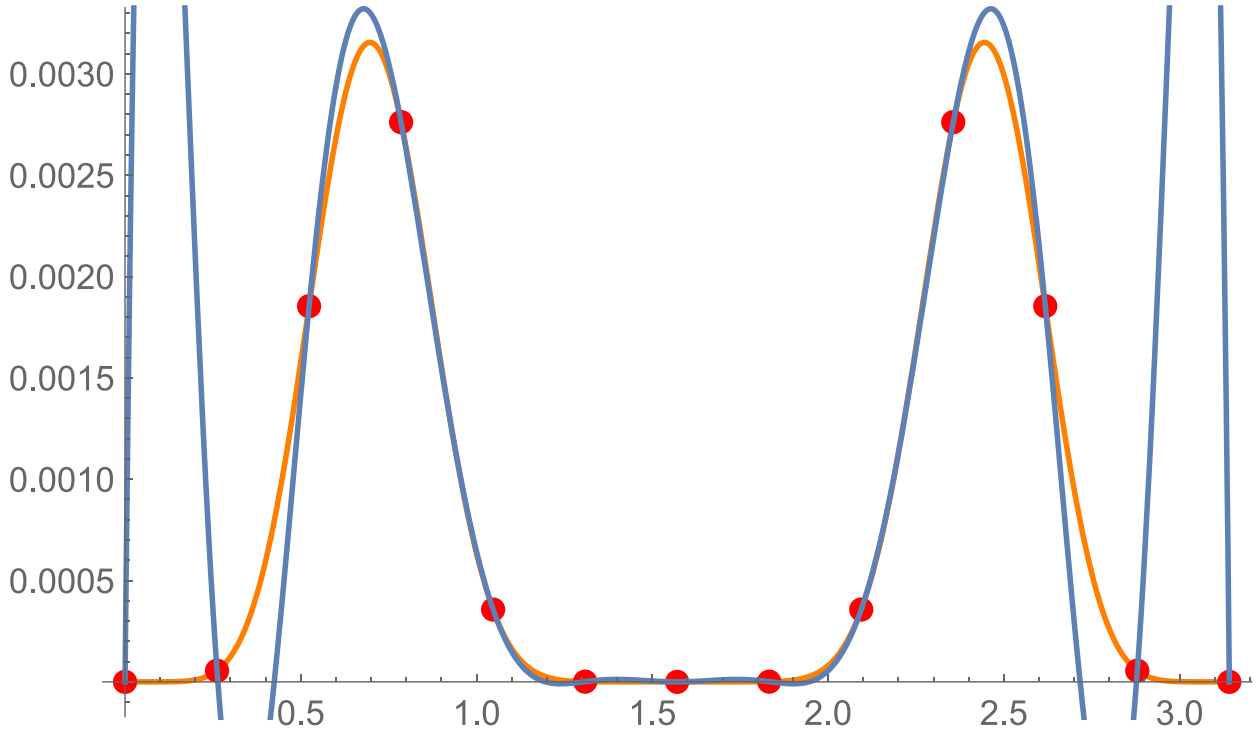

Рисунок 5.11. – график исходной функции и полученные с помощью интерполяции на входной функции в) с шагом $\frac{\pi}{12}$

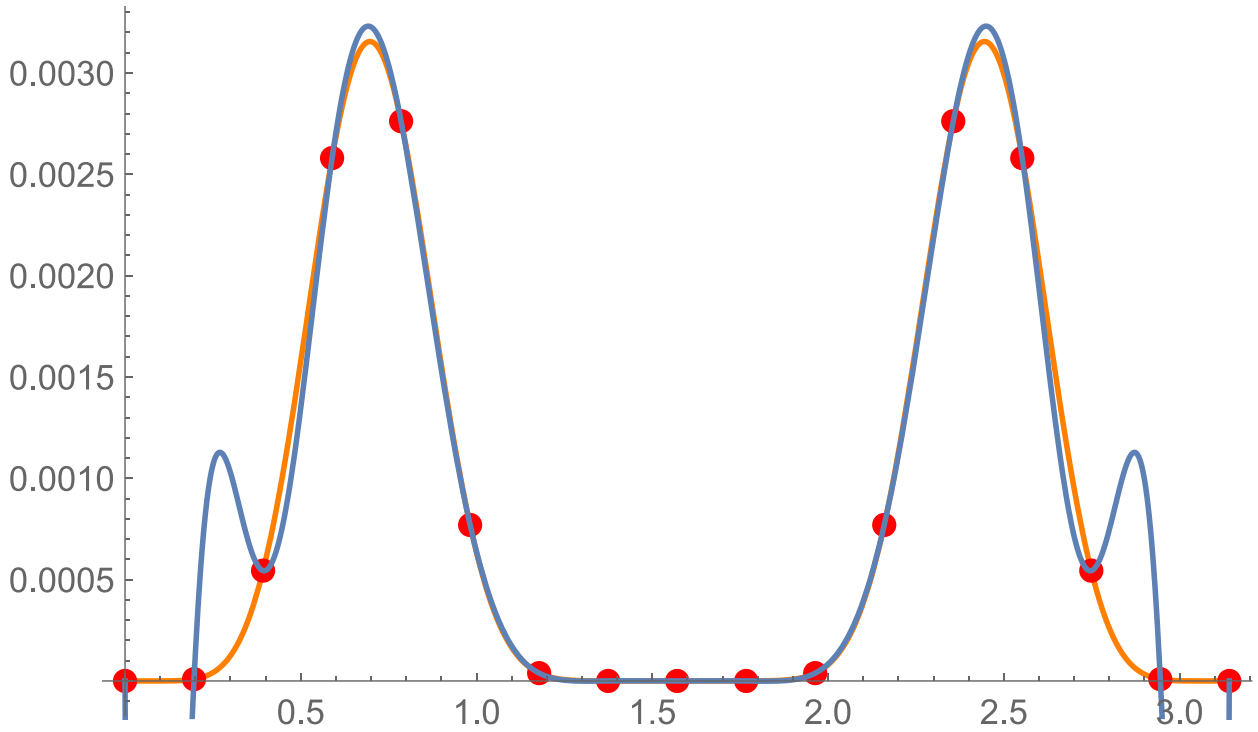

Рисунок 5.12. – график исходной функции и полученные с помощью интерполяции на входной функции в) с шагом $\frac{\pi}{16}$

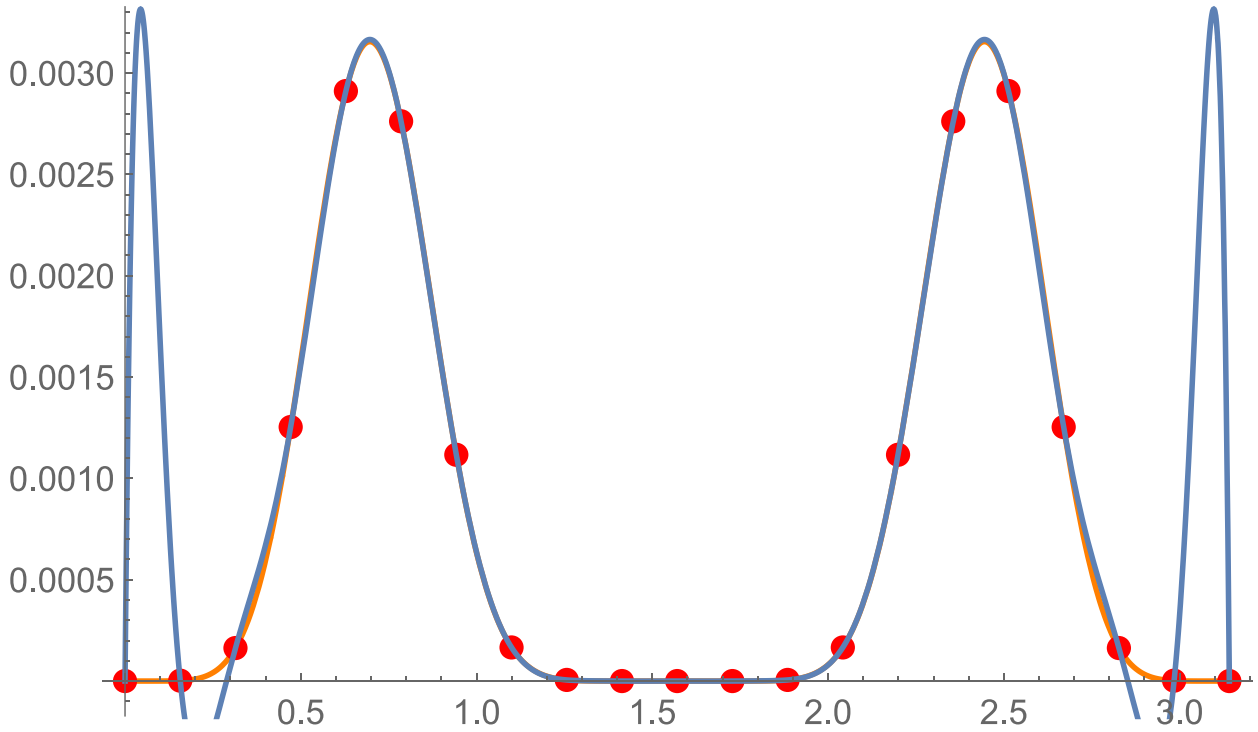

Рисунок 5.13. – график исходной функции и полученные с помощью интерполяции на входной функции в) с шагом $\frac{\pi}{20}$

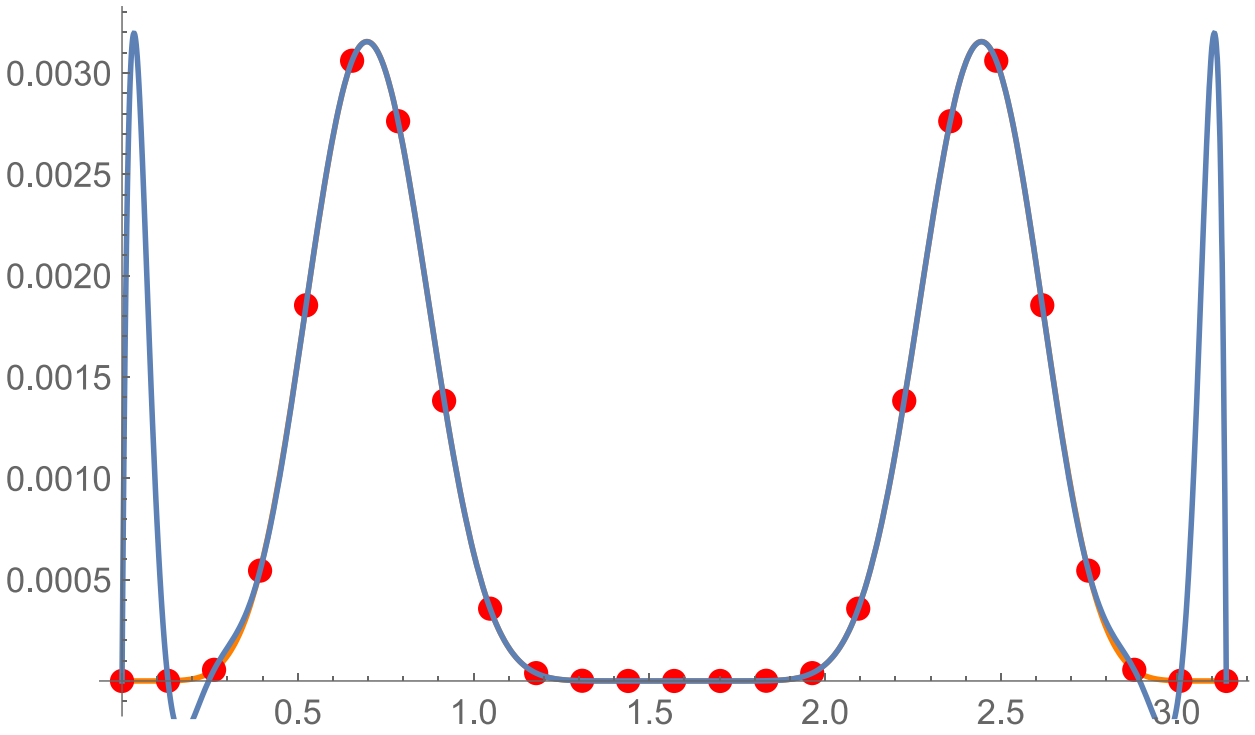

Рисунок 5.14. – график исходной функции и полученные с помощью интерполяции на входной функции в) с шагом $\frac{\pi}{24}$

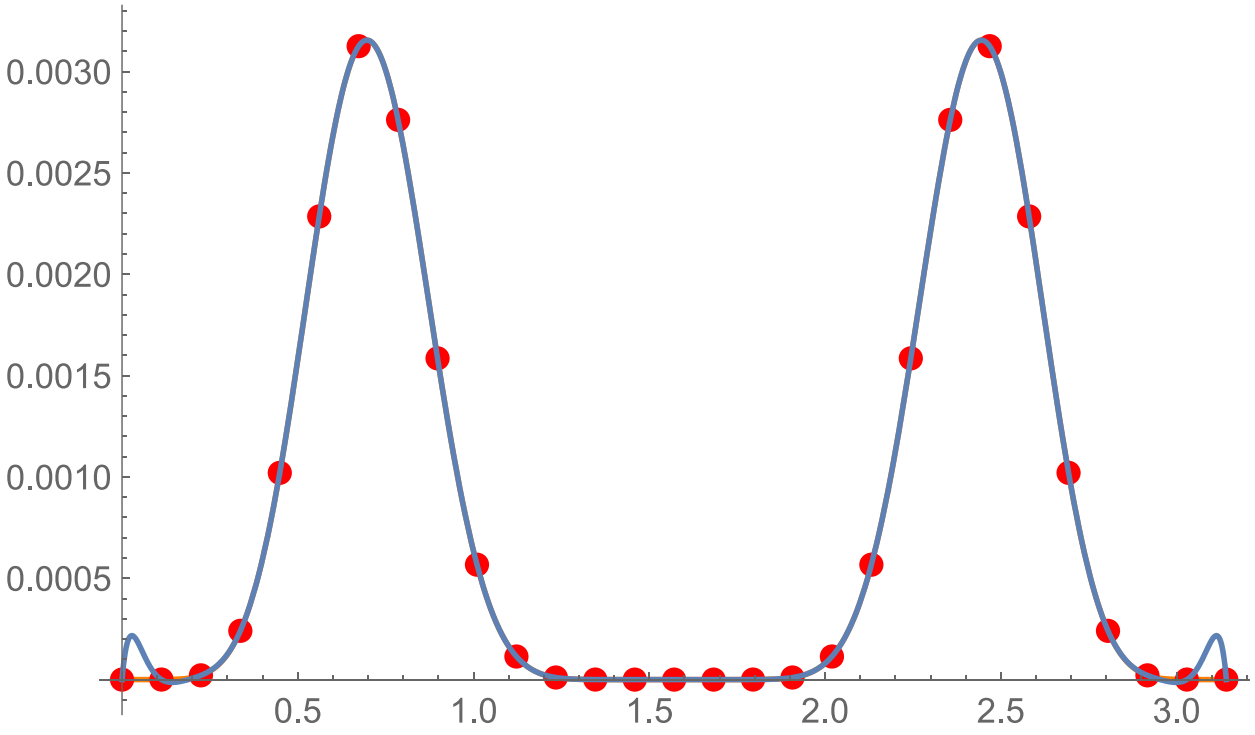

Рисунок 5.14. – график исходной функции и полученные с помощью интерполяции на входной функции в) с шагом $\frac{\pi}{28}$

Я считаю, что представленных мною примеров достаточно, для того, чтобы убедиться в корректности работы данного метода.

Подводя итоги, хочется сказать, что данный метод показал себя отлично и работает достаточно корректно.

6. ПОСТРОЕНИЕ НАИЛУЧШЕГО ПРИБЛИЖЕНИЯ ФУНКЦИИ МЕТОДОМ СРЕДНЕКВАДРАТИЧНОГО ПРИБЛИЖЕНИЯ. ИСПОЛЬЗОВАНИЕ ОРТОНОНАЛЬНЫХ ПОЛИНОМОВ ЛЕЖАНДРА

6.1. Описание метода и алгоритм поиска решения

Дано N точек $(x_i, f(x_i))$с некоторым отклонением, где $f -$ некоторая функция.

Задача построить ее наилучшее приближение. Также требуется, чтобы использовалась ортонормированная система базисных функций, построенная на основе ортогональных многочленов Лежандра.

Рассматривается некоторое евклидово пространство с нормой

$$(\phi, \psi) = \int_{-1}^{1} \rho(x)\phi(x)\psi(x)dx$$

Наилучшим среднеквадратичным приближением является:

$\sum_{k=0}^{n} a_k \phi_k(x)$, *где* $n -$ количество базисных функций. Для решения поставленной задачи достаточно взять $n = 5$

ϕ_j образуются с помощью нормированных ортогональных многочленов Лежандра, которые ортогональны на промежутке $[-1; 1]$

$$(n+1)L_{n+1}(x) = (2n+1)xL_n(x) - nL_{n-1}(x)$$

где $L_0 = 1, L_1 = x$

Базисные функции строятся следующим образом:

$$\phi_k(x) = \mu_k L_k(x)$$

В нашем случае $\mu_n = \sqrt{\frac{2n+1}{2}}$

Сами коэффициенты a_k находятся по следующей формуле:

$$\sum_{k=0}^{n} a_k \sum_{i=1}^{N} \rho_i \phi_k(x_i)\phi_i(x_i) = \sum_{i=1}^{N} \rho_i f(x_i)\phi_i(x_i)$$

Однако, поскольку мы работаем с ортонормированной системой, следовательно, из этой формулы получается следующее:

$$a_i = \sum_{i=1}^{N} \rho_i f(x_i)\phi_i(x_i), i = 0, \dots, n$$

Следует отметить, что в данном случае $\rho_i = 1$.

6.2. Программная реализация метода

Данный метод реализован с помощью Wolfram Mathematica.

Входные данные: набор N точек $(x_i, f(x_i))$с некоторым отклонением. Количество и шаг определяется пользователем. Выходные данные: Построенное наилучшее приближение. На рисунке 6.1 представлена сама реализация.

```
n = 5;
dots = Table[{x, f[x] + RandomReal[{-0.3, 0.3}]}, {x, -1, 1, 0.005}];
pol = Plus @@ Table[
    ((2 i + 1)/2)^1 * LegendreP[i, x] * (Plus @@ Table[k[[2]] * LegendreP[i, k[[1]]], {k, dots}])

    , {i, 0, n}];
```

Рисунок 6.1. – реализация метода

6.3. Визуализация результатов

Выберем следующую функцию: $f = 1.5\mathrm{Cos}[300x] + 4\mathrm{Sin}[4x] + \mathrm{Sin}[25x] + 40 - x^2/100$, которая определена на $[-1; 1]$

Для удобства будем строить все графики на одной координатной плоскости. Красные точки – точки исходного набора, по которым проводится аппроксимация. Синяя кривая – обычный график $f(x)$, оранжевые точки – набор заданных точек, красная кривая – построенное приближение.

Стоит отметить, что точки выбираются по следующему образу: выбирается шаг, через который будет выбираться новая точка x_i, лежащая в промежутке $[-1; 1]$. Далее вычисляется значение f в данной точке. После чего к полученному значению прибавляется случайное отклонение – случайно выбранное число из промежутка [-0.3 ; 0.3]. В данном случае шаг = 0.005.

На рисунке 6.2 представлена визуализация найденного наилучшего приближения для 1ой входной функций

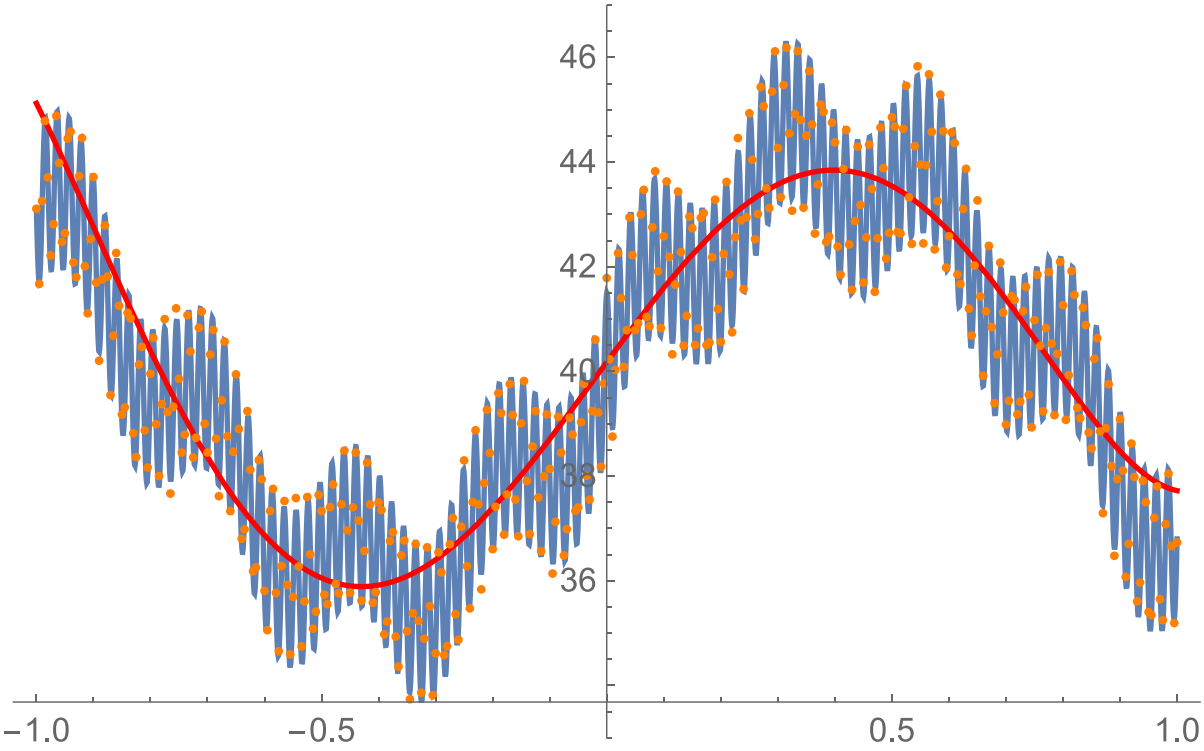

Рисунок 6.2. – визуализация наилучшего приближения для 1ой входной функции

Выберем еще одну функцию: $f = \sin[200x] + \cos[5x]$, также определенную на [-1 ; 1].

На рисунке 6.3. представлена визуализация найденного наилучшего приближения для 2ой входной функции.

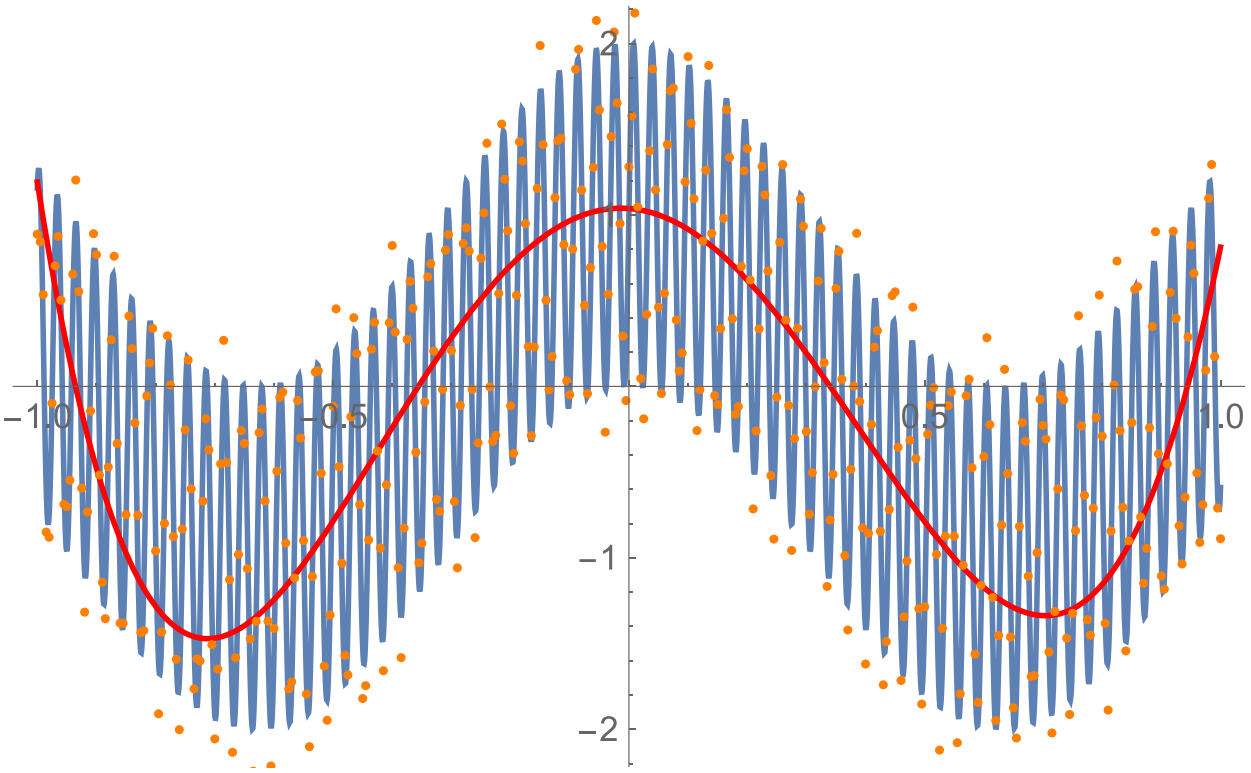

Рисунок 6.3. – визуализация наилучшего приближения для 2ой входной функции

Выберем еще одну функцию $f = \sin(50x) + 50x^2$, также определённую на [-1; 1].

На рисунке 6.4. представлена визуализация найденного наилучшего приближения для 3ой входной функции.

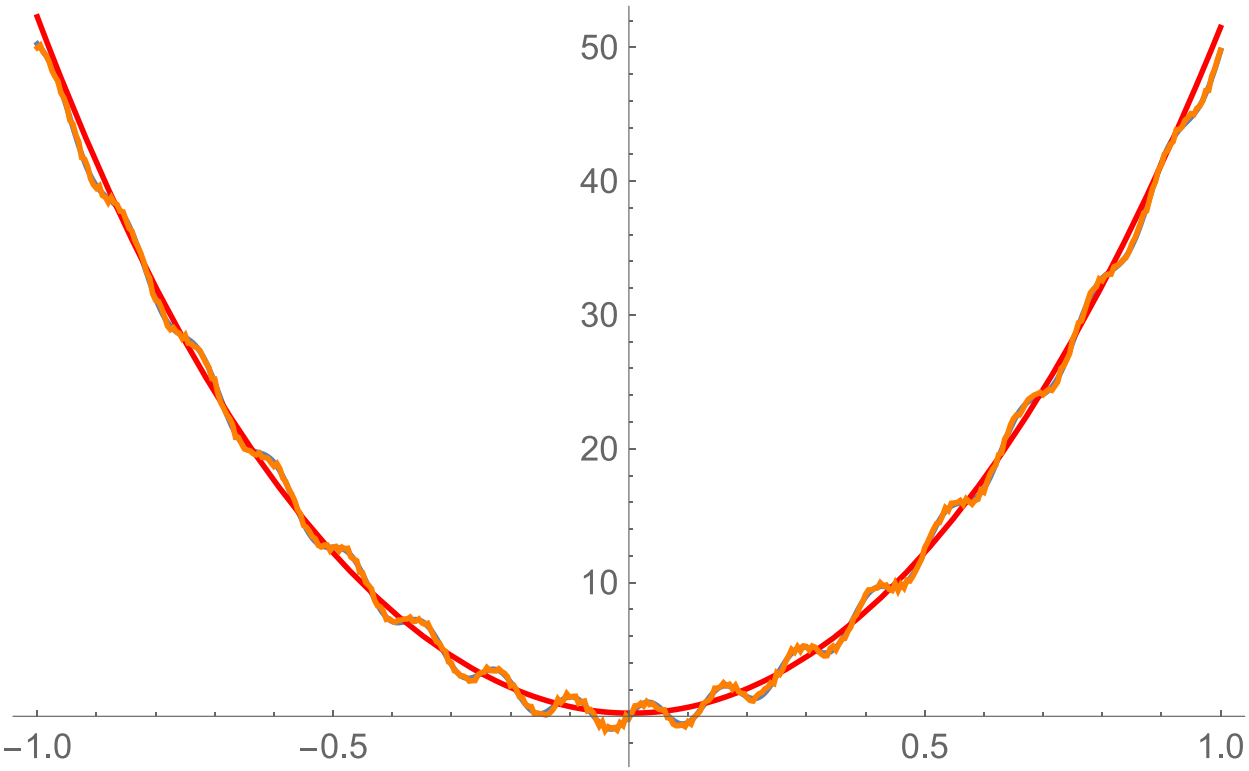

Рисунок 6.4. – визуализация наилучшего приближения для 3ой входной функции

По данным рисункам видно, что алгоритм работает корректно и достаточно точно. Стоит подметить, что из-за особенностей выбора ортонормированной системы функций, при попытке построить приближение за пределами области определения промежутка [-1; 1] будет некорректный результат. Это связано с тем, что полиномы Лежандра ортогональны только в данной области определения.

7. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ С ПОМОЩЬЮ КВАДРАТУРНОЙ ФОРМУЛЫ ГАУССА-КРИСТОФЕЛЛЯ – ИСПОЛЬЗОВАНИЕ ПОЛИНОМОВ ЧЕБЫШЁВА

7.1. Описание метода и алгоритм поиска решения

Дана непрерывная на некотором промежутке $[a; b]$ функция $f(x)$.

Пусть существует какая-то весовая функция $p(x) > 0$, которая непрерывна на интервале $(a; b)$.

Рассматривается задача поиска определенного интеграла вида:

$$I = \int_a^b f(x)p(x)dx$$

Данный метод необходим, когда выразить интеграл через элементарные функции в общем случае либо не удается, либо вызывает слишком много проблем. Поэтому обычно $f(x)$ заменяют на некоторую аппроксимирующую функцию $\phi(x) = f(x)$. Она подбирается таким образом, чтобы интеграл от нее легко считался в элементарных функциях. Стандартный пример $\phi(x) -$ некоторый обобщенный интерполяционный многочлен. При этом $f(x)$ заменяется линейным выражением со значениями в узлах в качестве некоторых коэффициентов:

$$f(x) = \sum_{k=1}^{n} f(x_k)\phi_k(x) + r(x),$$

где $r(x)$ – некоторый остаточный член аппроксимации, $n -$ количество узлов аппроксимации.

Подставив только что выраженную функцию $f(x)$ в интеграл получаем следующее:

$$I = \sum_{k=1}^{n} f(x_k)A_k + R,$$

где $A_k = \int_a^b \phi(x)p(x)dx$, $R = \int_a^b r(x)p(x)dx$.

В данном случае, $R=0$

Целью данной работы является реализация одного из частного вида данного метода: ортогональных полиномов Чебышёва 2 рода, определенных на $[-1; 1]$.

В этом случае $x_k = \cos\left(\frac{2k}{2n+1}\pi\right)$, $A_k = \frac{4\pi}{2n+1}\sin^2\frac{k\pi}{2n+1}$, где $\frac{1}{n}$ – шаг между k и $k+1$

$p(x) = \sqrt{1-x^2}$, $[a; b] = [-1; 1]$

7.2. Программная реализация метода

Данный метод реализован с помощью Wolfram Mathematica. Входные данные: функция $f(x)$ и количество узлов n. Выходные данные: найденное численное значение интеграла $\int_{-1}^{1} f(x)p(x)dx$. На рисунке 7.1 представлен код данной реализации.

```
gCr[f_, n_] := Module[{a = π/(n + 1) Sin[π * #/(n + 1)]^2 &, x = Cos[π * #/n] &, p = Sqrt[1 - #^2] &},
  f1 = Total@Table[a[k] * f /. d → x[k], {k, 1, n}] // N
]
```

Рисунок 7.1. – программная реализация метода

7.3. Тестирование и оценка точности

Рассмотрим несколько различных функций. Сравним точность вычислений со встроенными функциями $Wolfram\ Mathematica$ и рассмотрим, как влияет значение n на полученный результат.

1) $f(x) = 3\cos(5x) + 24\sin(30x) + x + 10$

Далее будут приводится графики, отображающее значение полученной невязки по оси ординат с значением n по оси абцисс.

На рисунке 7.2 представлена функция, строящая необходимые для сравнения норм графики, а на рисунке 7.3 представлен полученный график.

```
f2 = NIntegrate[f * p[d], {d, -1, 1}]
ints = Table[gCr[f, i], {i, 10, 3000, 5}];
ListLogPlot[Abs[ints - f2], PlotStyle → Red]
```

Рисунок 7.2. – программная реализация построения графиков роста нормы невязки

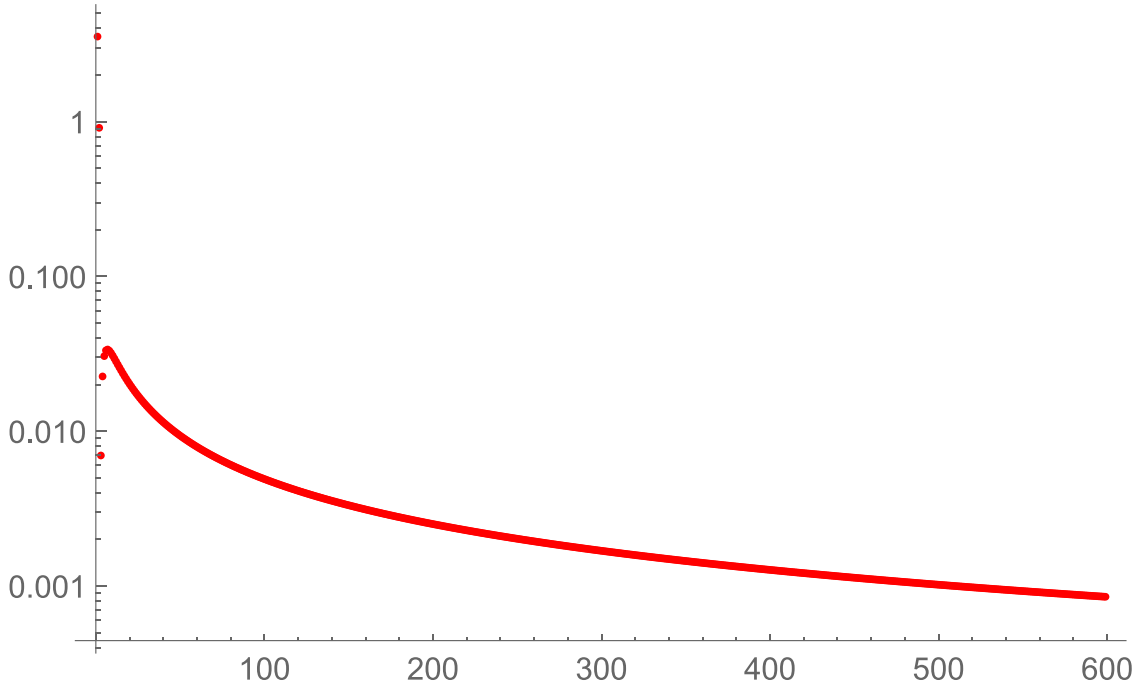

Рисунок 7.3. – изменение функции невязки с ростом числа n по 1 входной функции

2) $f(x) = x^5 - 66x^2 + x^{19}$.

На рисунке 7.4 представлен полученный график зависимости нормы невязки в зависимости от числа n.

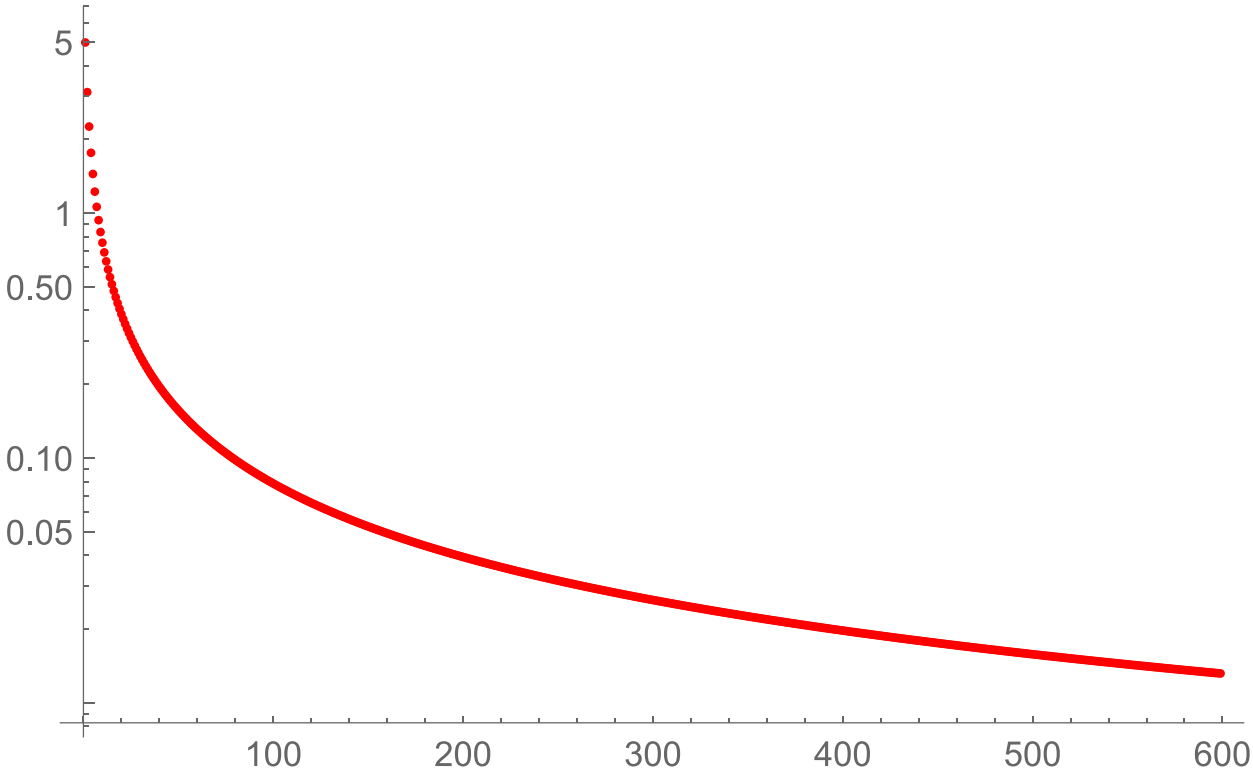

Рисунок 7.4. – изменение функции невязки с ростом числа n по 2 входной функции

3) $f(x) = 5\cos(350x) + 7\sin(4x) + \sin(24x) + (30 - \frac{x^2}{200})$.

На рисунке 7.5 представлен полученный график зависимости нормы невязки в зависимости от числа n.

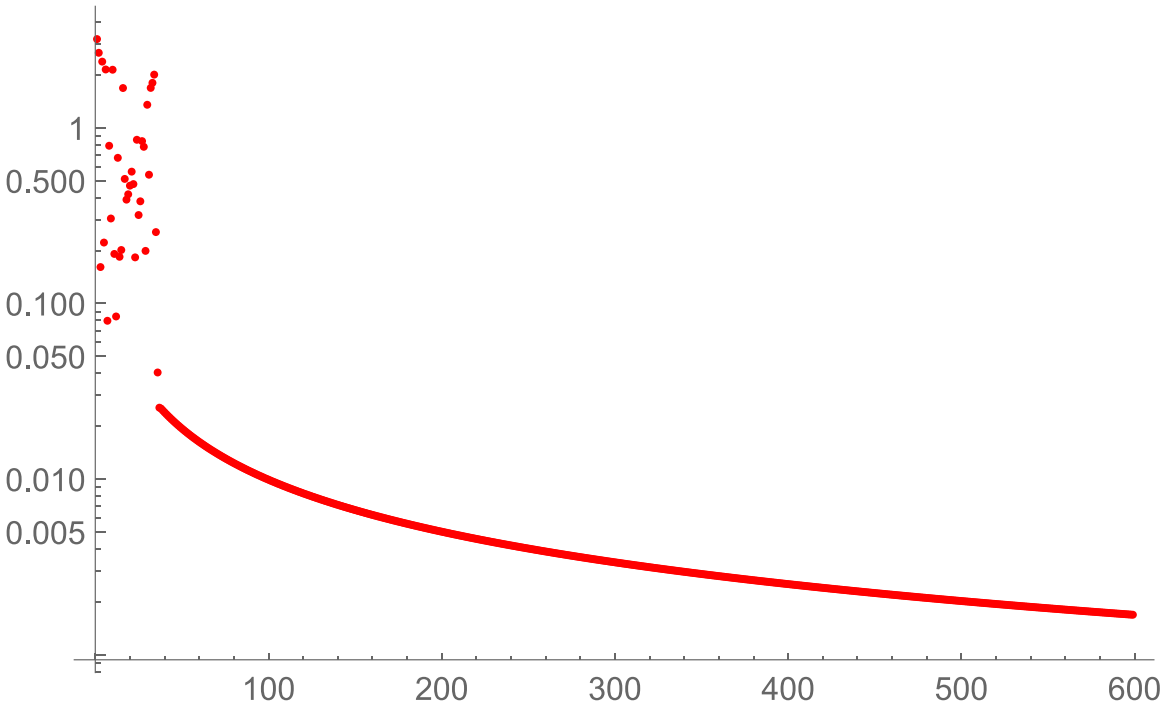

Рисунок 7.5. – изменение функции невязки с ростом числа n по 3 входной функции

По данным графикам наглядно видно, что при $n \to \infty$ норма невязки стремится к 0, что подтверждает правильность работы данного метода. Можно также наблюдать, как с ростом n норма невязки становится более монотонной, нежели при довольно малом n. Это также подтверждает корректность работы алгоритма.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы была рассмотрены некоторые задачи численных методов в математическом анализе и линейной алгебре, была изучена необходимая теория, были реализованы некоторые методы на языке Wolfram Language с использованием изученной ранее информации и проверены на правильность работы, также проведен глубокий анализ полученных конечных результатов на некотором количестве входных данных

В дальнейшем целесообразно освоение других методов вычислительной математики, а также и применение методов на практике для решения прикладных задач

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Березин И. С., Жидков Н. П. // Методы вычислений, Т.1. М.: ГИФМЛ, 1962. - 464 с.

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 – Москва: 1966. – 248 с.

5. Полиа Г., Сеге Г. //Задачи и теоремы из анализа, Т.2. М.Наука. 1978. – 432 с.

6. Турецкий А.Х. //Теория интерполирования в задачах – Минск. Вышэйшая школа. 1968. -320 с.

7. Wolfram Documentations Project [Электронный ресурс] / Documentation center. URL: https://reference.wolfram.com/language/ (дата обращения – 8.05.22)