|  |  |
| --- | --- |
| Изображение выглядит как текст, керамические изделия, фарфор  Автоматически созданное описание | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Факультет «Фундаментальные науки»

Кафедра «Техническая физика»

**Лабораторная работа №10**

по курсу «Вычислительная физика»

Выполнили: Коберник Т. Н., Плетенёв Б. А.

Группа: ФН4-71Б

Преподаватели: Хасаншин Р.Х., Ивлиев П.А.

Москва, 2022 г.

**Содержание**

[1. Теоретическая часть 3](#_Toc120016112)

[1.1. Метод прогонки 3](#_Toc120016113)

[1.2.Метод прогонки для дифференциального уравнения 5](#_Toc120016114)

[1.3.Минимизации невязки 6](#_Toc120016115)

[1.4.Метод коллокаций 7](#_Toc120016116)

[1.5. Метод наименьших квадратов 8](#_Toc120016117)

[1.6. Метод Галеркина 8](#_Toc120016118)

[2. Постановка задачи 9](#_Toc120016119)

[3. Программа 11](#_Toc120016122)

[4. Результаты 13](#_Toc120016123)

[5. Выводы 20](#_Toc120016124)

1. **Теоретическая часть** 
   1. **Метод прогонки**

Метод прогонки – модифицированный метод Гаусса для решения разреженных систем алгебраических уравнений.

Пускай дана система линейных алгебраических уравнений с трехдиагональной матрицей А.

Развернутая запись этой системы имеет вид

которому соответствует расширенная матрица

Если применить к этой матрице прямой метод Гаусса, то получим матрицу вида

Коэффициенты А формально взяты со знаком минус для упрощения записи выражений.

Тогда решение системы находится в виде

где – прогоночные коэффициенты

Запишем (2) для индекса :

и подставим в (1). Получим

Приводя эту формулу к виду (2) и сравнивая, получаем рекуррентные соотношения для коэффициентов

Для можем записать (1) и (2) соответственно:

Из рекуррентных формул (3) и начальных условий (4) мы можем найти все прогоночные коэффициенты, завершив ход прогонки.

Возьмем последнее уравнение системы (1):

Откуда

## **Метод прогонки дифференциального уравнения**

Пускай необходимо решить дифференциальное уравнение

L – линейный оператор

Введем на (a, b) равномерную сетку

Используем формулы численного дифференцирования для преобразования (5):

Преобразуем выражение (6)

Где

Решение будем искать в виде

Для получения рекуррентных формул для коэффициентов подставим в ур-ие (7) вместо

Возьмём аппроксимацию из (6.1) для

Откуда

Таким образом, из (8) и (9) модно выразить и завершить прямой ход прогонки.

## **Минимизации невязки**

Для того, чтобы краевая задача (10) имела единственное решение необходимо

Для нахождения приближенного решения рассмотрим систему дважды непрерывный линейно независимых функций При этом – удовлетворяет неоднородным граничным условиям (10), а остальные удовлетворяют однородным граничным условиям

*Рассмотрим линейную комбинацию функций в качестве приближенного решения (10)*

*Невязка*

Функцию можно рассматривать как некоторое отклонение приближенного решения от точного решения задачи (10), если она тождественно равна 0 по x, тогда .

## **Метод коллокаций**

Возьмем на [a, b] n узловых точек, тогда .

Потребуем, чтобы в этих точках невязка была равна 0.

Если эта система однозначно разрешима, тогда , получив все коэффициенты, можно построить приближенное распределение.

# **Метод наименьших квадратов**

В этом методе используется базис :

, где – скалярное произведение.

Если система линейно независимой, то система имеет одно решение.

В дискретном случае:

Тогда

# **Метод Галеркина**

Первым шагом в реализации метода Галеркина является выбор набора базисных функций, которые удовлетворяют граничным условиям и образуют полную систему в пределе бесконечного количества элементов базиса.

Конкретный вид функций определяется из специфики задачи и удобства работы. Часто применяются тригонометрические функции, ортогональные полиномы.

Зададим решение в виде разложения функции по выбранному базису:

Затем приближённое решение подставляется в исходное дифференциальное уравнение, и вычисляется его невязка. Для однородного уравнения:

Для неоднородного уравнения 𝐿[𝑢] = 𝑓(𝑥) невязка будет иметь вид 𝑁(𝑥) = 𝐿[𝑢] − 𝑓(𝑥).

C помощью требования ортогональности невязки к базисным функциям, получаем достаточное количество уравнений для единственного решения:

Отсюда получается однородная система уравнений для коэффициентов в разложении, и удаётся приближённо найти собственные значения задачи.

1. **Постановка задачи**

А) Решить краевую задачу, используя метод прогонки:

,  [0, 1],

, 

Б) Аппроксимировать функцию Рунге с помощью кубического сплайна при глобальном способе задания наклонов (используя метод прогонки для решения системы с целью определения наклонов сплайна).

В) Методы минимизации невязки

Пример 1. Решить, используя метод коллокации

**. (1)

,  (2)

Базисные функции 

Точки коллокации , хотя точек коллокации три, приближенное решение необходимо искать в виде



Решить краевую задачу (1), (2) используя интегральный и дискретный метод наименьших квадратов*.*

Пример 2. (решить методом Галеркина)

**.

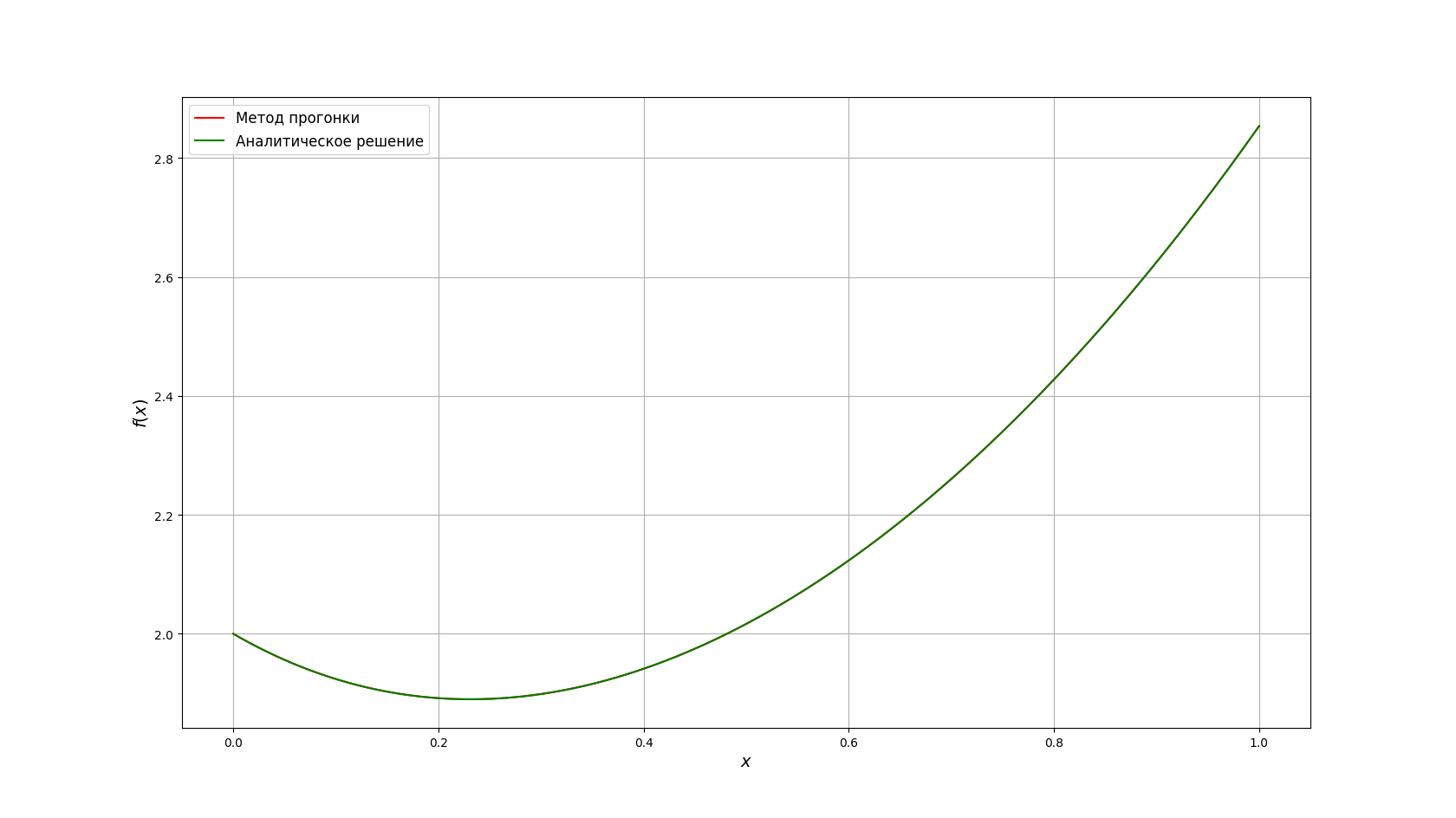
, 

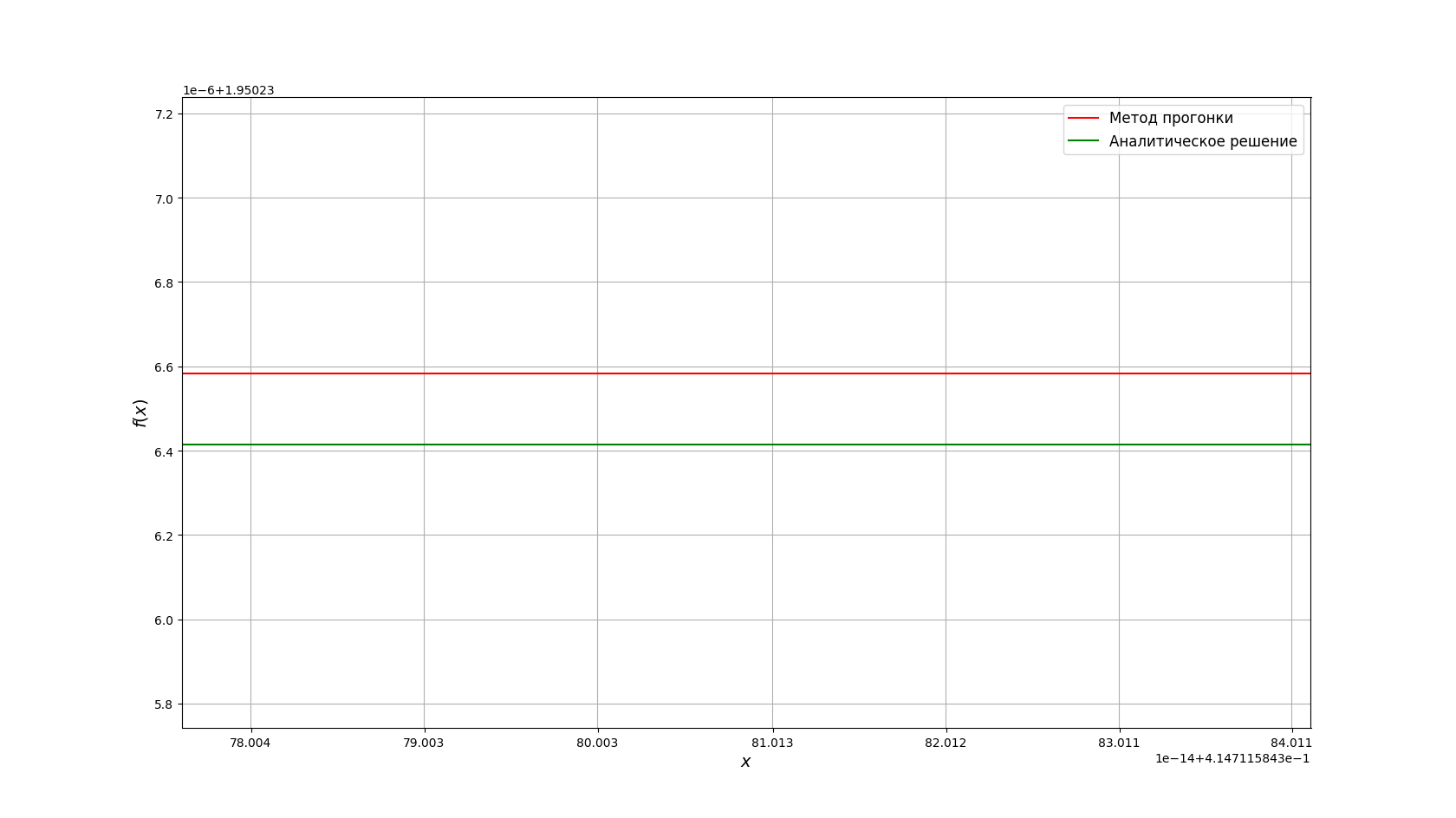
Базисные функции ,

Найти приближенные решения , 

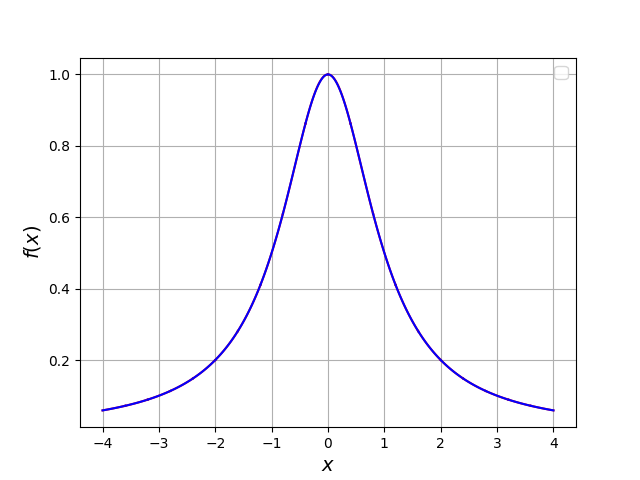
1. **Программа**
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
5. **def** answer(x):
6. **return** np.exp(**-** 2 **\*** x) **+** np.exp(x)
8. **def** runge(x):
9. **return** 1**/**(1 **+** x **\*\*** 2)
11. **def** runge\_diff(x):
12. **return** (**-**2 **\*** x) **/** ((1 **+** x **\*\*** 2) **\*\*** 2)
14. **def** num\_1():
15. x **=** np.linspace(0, 1, 501)
16. h **=** x[1] **-** x[0]
17. alpha **=** 1 **+** h **/** 2
18. beta **=** **-** 2 **\*** (h **\*\*** 2) **-** 2
19. gamma **=** 1 **-** h **/** 2
20. A, B **=** [0], [2]
21. y **=** [1 **for** j **in** range(len(x))]
22. y[0] **=** 2
23. y[**-**1] **=** (np.exp(**-**2) **+** np.exp(1))
25. **for** i **in** range(1, len(x) **-** 1):
26. B.append((**-**gamma **\*** B[**-**1]) **/** (beta **+** gamma **\*** A[**-**1]))
27. A.append(**-** alpha **/** (beta **+** gamma **\*** A[**-**1]))
29. **for** k **in** range(1, len(x) **-** 1):
30. y[**-**k**-**1] **=** y[**-**k]**\***A[**-**k] **+** B[**-**k]
32. plt.plot(x, y, color**=**'r', label**=**'Метод прогонки')
33. plt.plot(x, answer(x), color**=**'g', label**=**'Аналитическое решение')
34. plt.xlabel(r'$x$', fontsize**=**14)
35. plt.ylabel(r'$f(x)$', fontsize**=**14)
36. plt.grid(True)
37. plt.legend(loc**=**'best', fontsize**=**12)
39. plt.show()
41. **def** Spline\_sec(x, x1, m, kk):
42. h **=** x1[1] **-** x1[0]
43. m\_i **=** m[20**\***kk]
44. m\_i\_next **=** m[20**\***kk **+** 20]
45. S3 **=** ((x1[1] **-** x) **\*\*** 2) **\*** (2 **\*** (x **-** x1[0]) **+** h) **\*** runge(x1[0]) **/** (h **\*\*** 3) \
46. **+** (((x **-** x1[0]) **\*\*** 2) **\*** (2 **\*** (x1[1] **-** x) **+** h) **\*** runge(x1[1]) **/** (h **\*\*** 3)) \
47. **+** (((x1[1] **-** x) **\*\*** 2) **\*** (x **-** x1[0]) **\*** m\_i **/** (h **\*\*** 2)) \
48. **+** (((x **-** x1[0]) **\*\*** 2) **\*** (x **-** x1[1]) **\*** m\_i\_next **/** (h **\*\*** 2))
49. **return** S3
51. **def** num\_2():
52. x **=** np.linspace(**-**4, 4, 401)
53. x1 **=** np.linspace(**-**4, 4, 21)
54. h **=** x[1] **-** x[0]
55. m **=** [1 **for** j **in** range(len(x))]
56. m[0] **=** (**-**3 **\*** runge(x[0]) **+** 4 **\*** runge(x[1]) **-** runge(x[2])) **/** (2 **\*** h)
57. m[**-**1] **=** (3 **\*** runge(x[**-**1]) **-** 4 **\*** runge(x[**-**2]) **+** runge(x[**-**3])) **/** (2 **\*** h)
58. alpha **=** 1
59. beta **=** 4
60. gamma **=** 1
61. # C1 = 1, d1 = 1, C2 = 0, d2 = 0
62. A, B **=** [0], [runge(x[0])]
64. **for** i **in** range(1, len(x) **-** 1):
65. delta **=** (3 **/** h) **\*** (runge(x[i **+** 1]) **-** runge(x[i **-** 1]))
66. B.append((delta **-** gamma **\*** B[**-**1]) **/** (beta **+** gamma **\*** A[**-**1]))
67. A.append(**-** alpha **/** (beta **+** gamma **\*** A[**-**1]))
69. **for** k **in** range(1, len(x) **-** 1):
70. m[**-**k**-**1] **=** m[**-**k]**\***A[**-**k] **+** B[**-**k]
72. **for** kk **in** range(x1.size**-**1):
73. x **=** np.linspace(x1[kk], x1[kk **+** 1], round(500**/**x1.size))
74. plt.plot(x, np.array([runge(ii) **for** ii **in** x]), color**=**'r')
75. plt.plot(x, np.array([Spline\_sec(ii, [x1[kk], x1[kk **+** 1]], m, kk) **for** ii **in** x]), color**=**'b')
76. plt.xlabel(r'$x$', fontsize**=**14)
77. plt.ylabel(r'$f(x)$', fontsize**=**14)
78. plt.grid(True)
79. plt.legend(loc**=**'best', fontsize**=**12)
80. plt.show()
82. num\_1()
83. num\_2()
84. **Результаты**

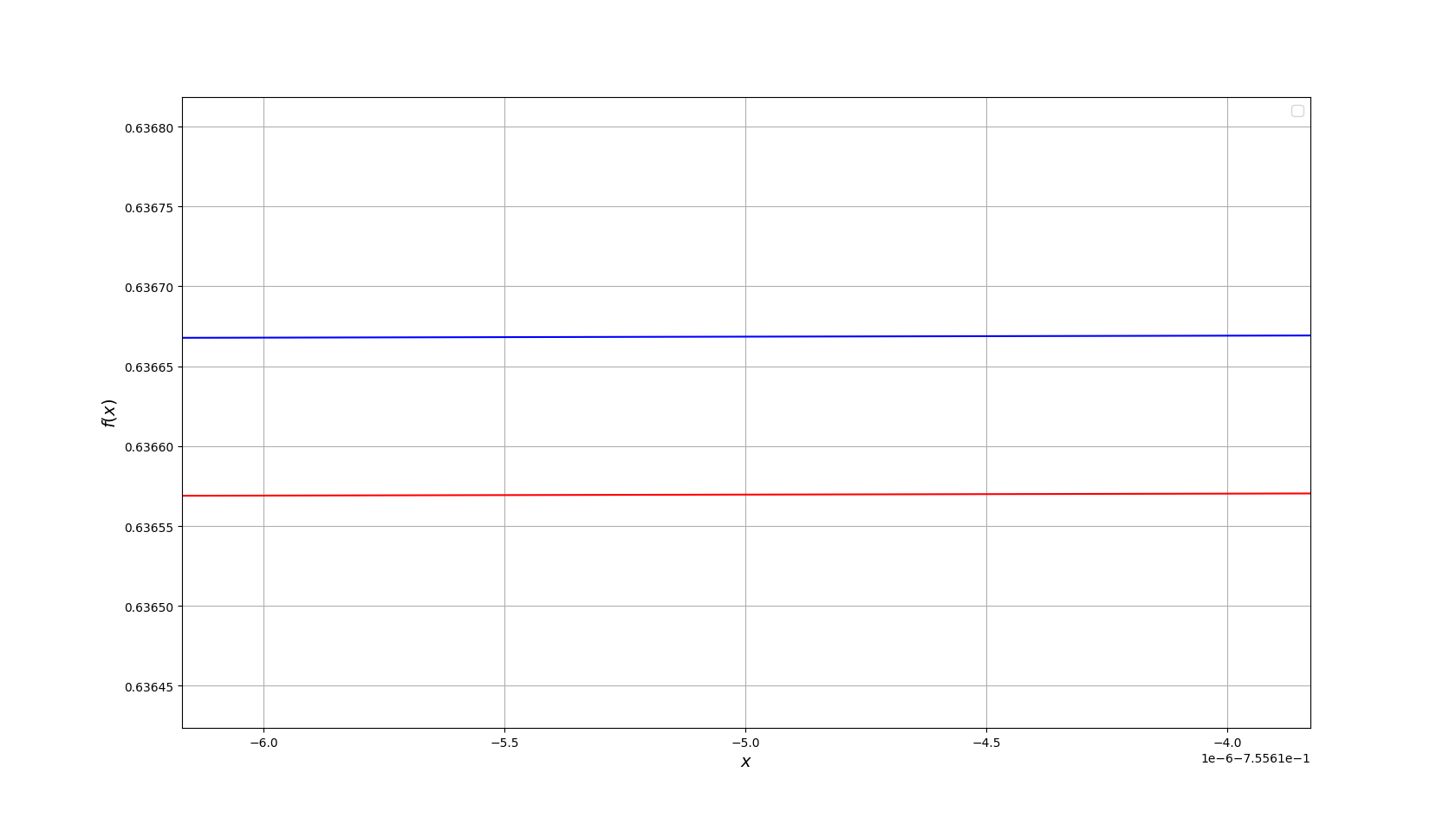
А) В результате работы программы было получено приближенное решение системы. График полученной функции:





Б) В результате работы программы была получена аппроксимация функции Рунге. График полученной функции:



В)

Пример 1

Базисные функции имеют вид

Точки коллокации

Приближенное решение будем искать в виде

Базисные функции линейно независимы, если

Найдем производные :

Тогда невязка

В точках коллокации невязку приравняем к нулю и найдем коэффициенты

Тогда решение:

Для того, чтобы интегральным методом наименьших квадратов решить эту задачу, рассмотрим интеграл

Этот интеграл должен быть постоянным относительно коэффициентов :

Найдем производные по параметрам

В таком случае имеем

Дискретный МНК:

Строим систему:

Пример 2

Задачу необходимо решить для методом Галеркина с базисными функциями

Для решения этим методом необходимо построить функцию вида

После чего найти , а далее решить систему

Построим для .

Тогда:

Для :

Теперь римем , тогда для

Таким образом

1. **Выводы**

Таким образом, в ходе работы был протестирован метод прогонки для решения краевой задачи и для построения аппроксимации функции Рунге. Решение краевой задачи, полученное данным способом, оказалось крайне точно, что без труда можно увидеть на графике (погрешность достигает порядка ). Аппроксимация функции Рунге с помощью кубического сплайна при глобальном способе задания наклонов (с использованием метода прогонки для решения системы с целью определения наклонов сплайна) так же дала хорошие результаты. Из выше сказанного можно заключить, что метод прогонки пригоден для решения широкого спектра задач. Так же в работе были рассмотрены методы минимизации невязки, в частности метод коллокаций (с применением интегрального и дискретного МНК) и метод Галеркина. Результаты, полученные различными методами, совпали с хорошей точностью.