|  |  |
| --- | --- |
| Московский авиационный институт  (Национальный исследовательский университет)  Институт №8 «Компьютерные науки и прикладная математика» | |
| **Курсовая работа**  по курсу «Численные методы»  по теме «Численное решение интегральных уравнений Фредгольма 2-го рода» | |
|  |  |
|  | **Выполнила студентка**: Бондарева Е. Е.  **Группа:** М8О-405Б-21 Преподаватель: Демидова О. Л. |
|  | **Оценка:**  **Дата**: 12.12.2024 |
|  |  |
| Москва, 2024 | |

**Содержание:**

**Введение** 3

**Методология** 3

**Процесс решения** 6

**Параметры задачи** 6

**Результат работы программы** 6

**Вывод** 10

**Приложение** 11

**Введение**

В этом разделе будут рассмотрены три численных метода для решения интегральных уравнений Фредгольма 2-го рода: метод прогонки, метод простых итераций и метод Зейделя. Каждый из этих методов имеет свои особенности и применяется для разных типов задач. Для численного решения задачи используется дискретизация уравнения и преобразование его в систему линейных уравнений или итерационных процедур, которые затем решаются с помощью соответствующих методов.

Интегральные уравнения Фредгольма 2-го рода имеют вид:

φ(x) = f (x) + λ x = [a, b],

где g(x) – заданная функция, λ – параметр, K (x, t) – ядро уравнения, φ(x) – функция, которую надо найти

## **Методология**

### **Дискретизация интегрального уравнения**

Для численного решения интегрального уравнения необходимо перейти от непрерывной задачи к дискретной. Это достигается путем дискретизации интервала и аппроксимации интеграла:

Интервал [*a*,*b*] разбивается на N равных частей с шагом h=(b−a)/N.

Узлы сетки определяются как:

xi = a + i \* h, где i = 0, 1, …, N

#### Интеграл в уравнении аппроксимируется с помощью квадратурной формулы, например, метода трапеций или метода Симпсона. Для простоты рассмотрим метод трапеций:

=

где *wj*​ — весовые коэффициенты (для метода трапеций *w*0​ = *wN* = 0.5, а остальныеw j = 1).

После дискретизации интегральное уравнение преобразуется в систему линейных уравнений:

, i = 0, 1, …, N

где: ui = u(xi) fi = f(xi) Kij = K(xi, xj)

Эта система может быть записана в матричной форме:

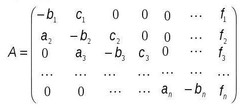
**u**=**f**+*λh***Ku**,

где **u** и **f** — векторы значений *ui* и fi , а **K** — матрица ядра.

Этот набор основных формул будет использоваться для реализации трех методов, рассмотренных ниже

**Метод прогонки**

Метод прогонки (или метод Томаса) является эффективным алгоритмом для решения систем линейных уравнений с трехдиагональной матрицей. В контексте интегральных уравнений Фредгольма 2-го рода метод прогонки может быть применен после приведения системы к трехдиагональному виду:



Метод прогонки состоит из двух этапов: прямого хода и обратного.

**Прямой ход заключается в вычислении прогоночных коэффициентов**:

|  |  |
| --- | --- |
|  |  |

гдеi, I ,***C*i – элементы трехдиагональной матрицы, а**  -**элементы правой части системы.**

**Обратный ход** заключается в нахождении неизвестных:

ui = ai ui + 1 + *βi*

Будем использовать этот набор формул для поставленной задачи

### **Метод простых итераций**

Метод простых итераций (или метод Якоби) является одним из самых простых итерационных методов решения систем линейных уравнений. Он основан на последовательном улучшении приближенного решения.

Исходная система уравнений:

u=f + λhKu

может быть переписана в виде:

uk+1= f + λhKuk , где uk — приближение на *k*-й итерации.

Метод простых итераций сходится, если спектральный радиус матрицы λhK меньше 1: *ρ*(*λh*K) < 1

#### **Алгоритм метода заключается в следующем:**

1. Задается начальное приближение **u** (0) (например, нулевое).
2. На каждой итерации вычисляется новое приближение:

Uk+1=f+ λhKuk

1. Итерации продолжаются до достижения заданной точности:

∥uk+1 − uk∥<*ε*

Алгоритм четко описывает план действий и стратегию для реализации данного метода.

### **Метод Зейделя**

Метод Зейделя является модификацией метода простых итераций, которая учитывает уже вычисленные значения на текущей итерации, что ускоряет сходимость. В данном методе новое приближение вычисляется с использованием уже обновленных значений:

#### **Алгоритм метода заключается в следующем:**

1. Задается начальное приближение **u**(0).
2. На каждой итерации последовательно обновляются значения **ui.**
3. Итерации продолжаются до достижения заданной точности.

### **Процесс решения**

1. **Дискретизация**: На интервале [a,b] строится сетка из N узлов, и вычисляются значения правой части f(x).
2. **Метод прогонки**: преобразуется система уравнений в трехдиагональную матрицу и решается методом прогонки.
3. **Методы итераций**: используются для последовательного улучшения приближенного решения, начиная с нулевого приближения.
4. **Метод Зейделя:** новое приближение вычисляется с использованием уже обновленных значений

В данной работе будет реализованы три метода и их сравнение.

### **Параметры задачи**

* **Границы интервала**: a=0, b=10.
* **Количество узлов (N)**: N=100.
* **Параметр λ**: λ=0.5.

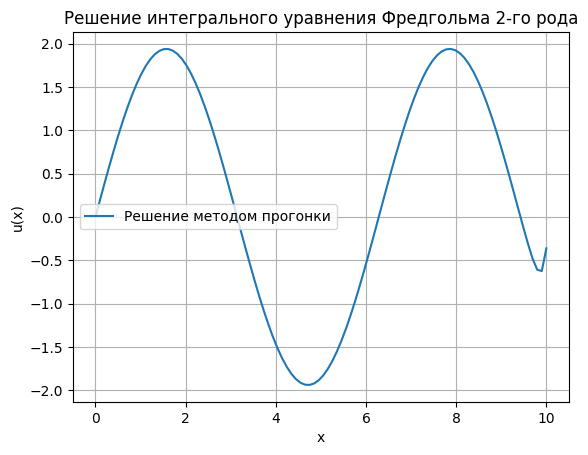
Задана правая часть уравнения f(x) = sin(x), что является типичным выбором для тестирования численных методов. Гауссово ядро используется для вычисления значений в интеграле.

**Результаты работы программы**

**Метод прогонки:**

x values:

1. [ 0. 0.1010101 0.2020202 0.3030303 0.4040404 0.50505051
2. 0.60606061 0.70707071 0.80808081 0.90909091 1.01010101 1.11111111
3. 1.21212121 1.31313131 1.41414141 1.51515152 1.61616162 1.71717172
4. 1.81818182 1.91919192 2.02020202 2.12121212 2.22222222 2.32323232
5. 2.42424242 2.52525253 2.62626263 2.72727273 2.82828283 2.92929293
6. 3.03030303 3.13131313 3.23232323 3.33333333 3.43434343 3.53535354
7. 3.63636364 3.73737374 3.83838384 3.93939394 4.04040404 4.14141414
8. 4.24242424 4.34343434 4.44444444 4.54545455 4.64646465 4.74747475
9. 4.84848485 4.94949495 5.05050505 5.15151515 5.25252525 5.35353535
10. 5.45454545 5.55555556 5.65656566 5.75757576 5.85858586 5.95959596
11. 6.06060606 6.16161616 6.26262626 6.36363636 6.46464646 6.56565657
12. 6.66666667 6.76767677 6.86868687 6.96969697 7.07070707 7.17171717
13. 7.27272727 7.37373737 7.47474747 7.57575758 7.67676768 7.77777778
14. 7.87878788 7.97979798 8.08080808 8.18181818 8.28282828 8.38383838
15. 8.48484848 8.58585859 8.68686869 8.78787879 8.88888889 8.98989899
16. 9.09090909 9.19191919 9.29292929 9.39393939 9.49494949 9.5959596
17. 9.6969697 9.7979798 9.8989899 10. ]
18. u(x) values:
19. [ 0. 0.19572804 0.38946075 0.57922315 0.76308075 0.93915921
20. 1.10566354 1.26089633 1.40327508 1.53134833 1.64381044 1.73951495
21. 1.8174862 1.87692933 1.91723835 1.93800233 1.93900961 1.9202499
22. 1.88191447 1.8243941 1.74827519 1.65433371 1.54352735 1.41698571
23. 1.27599879 1.12200386 0.95657082 0.78138614 0.59823573 0.40898668
24. 0.21556826 0.01995227 -0.17586713 -0.36989367 -0.56014938 -0.74469472
25. -0.92164836 -1.08920639 -1.24566064 -1.38941617 -1.51900748 -1.63311347
26. -1.7305709 -1.81038625 -1.87174586 -1.9140242 -1.93679027 -1.93981199
27. -1.92305855 -1.88670075 -1.83110922 -1.75685069 -1.66468218 -1.55554328
28. -1.43054661 -1.29096641 -1.13822563 -0.97388135 -0.79960897 -0.61718509
29. -0.42846939 -0.23538571 -0.03990243 0.15598763 0.3502875 0.54101641
30. 0.72623 0.90404013 1.07263414 1.23029333 1.37541045 1.50650613
31. 1.62224393 1.72144399 1.803095 1.86636461 1.9106078 1.93537356
32. 1.9404094 1.925664 1.89128767 1.83763087 1.76524059 1.67485485
33. 1.56739514 1.44395715 1.30579976 1.15433279 0.99110404 0.8177873
34. 0.63617536 0.44818849 0.25592581 0.06183229 -0.1308264 -0.31667048
35. -0.4847401 -0.60926288 -0.62511741 -0.36268074]



**Метод простых итераций:**

x values:

[ 0. 0.1010101 0.2020202 0.3030303 0.4040404 0.50505051

0.60606061 0.70707071 0.80808081 0.90909091 1.01010101 1.11111111

1.21212121 1.31313131 1.41414141 1.51515152 1.61616162 1.71717172

1.81818182 1.91919192 2.02020202 2.12121212 2.22222222 2.32323232

2.42424242 2.52525253 2.62626263 2.72727273 2.82828283 2.92929293

3.03030303 3.13131313 3.23232323 3.33333333 3.43434343 3.53535354

3.63636364 3.73737374 3.83838384 3.93939394 4.04040404 4.14141414

4.24242424 4.34343434 4.44444444 4.54545455 4.64646465 4.74747475

4.84848485 4.94949495 5.05050505 5.15151515 5.25252525 5.35353535

5.45454545 5.55555556 5.65656566 5.75757576 5.85858586 5.95959596

6.06060606 6.16161616 6.26262626 6.36363636 6.46464646 6.56565657

6.66666667 6.76767677 6.86868687 6.96969697 7.07070707 7.17171717

7.27272727 7.37373737 7.47474747 7.57575758 7.67676768 7.77777778

7.87878788 7.97979798 8.08080808 8.18181818 8.28282828 8.38383838

8.48484848 8.58585859 8.68686869 8.78787879 8.88888889 8.98989899

9.09090909 9.19191919 9.29292929 9.39393939 9.49494949 9.5959596

9.6969697 9.7979798 9.8989899 10. ]

u(x) values:

[ 1.00997932 1.25349275 1.50074544 1.74832197 1.99265562 2.23012698

2.45715936 2.67030554 2.86632214 3.04222935 3.19535597 3.32337057

3.42430099 3.49654447 3.53887097 3.55042181 3.53070526 3.47959017

3.39729812 3.28439432 3.14177699 2.97066495 2.77258324 2.54934631

2.30303875 2.0359935 1.75076761 1.45011569 1.13696137 0.81436689

0.48550142 0.15360811 -0.17802944 -0.5061212 -0.82740468 -1.13867865

-1.43683613 -1.71889642 -1.98203576 -2.22361638 -2.44121362 -2.63264086

-2.7959719 -2.92956079 -3.03205865 -3.10242754 -3.13995101 -3.14424145

-3.11524402 -3.05323713 -2.95882953 -2.83295398 -2.67685763 -2.49208906

-2.28048235 -2.04413812 -1.78540179 -1.50683943 -1.21121119 -0.9014428

-0.5805953 -0.25183334 0.08160759 0.41645469 0.7494319 1.07729393

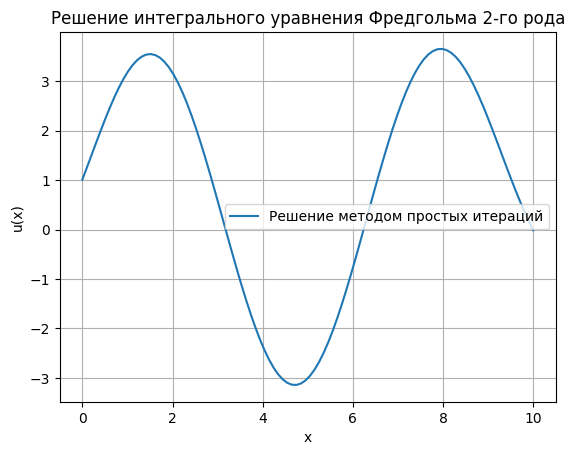
1.39686009 1.70504753 1.99890346 2.27563627 2.53264488 2.76754638

2.97820142 3.1627373 3.31956834 3.44741359 3.54531151 3.61263163

3.64908316 3.65472031 3.62994449 3.57550314 3.49248517 3.38231281

3.2467294 3.08778292 2.9078045 2.7093813 2.49532322 2.26862283

2.03240825 1.78988933 1.54429777 1.29882278 1.05654449 0.82036805  
 0.59296173 0.37670263 0.17363331 -0.01456786]

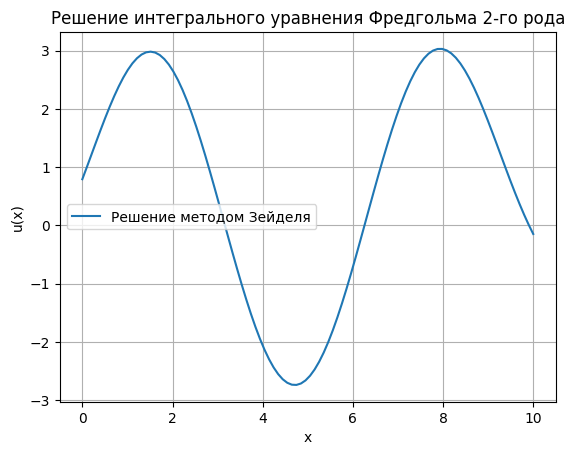
****

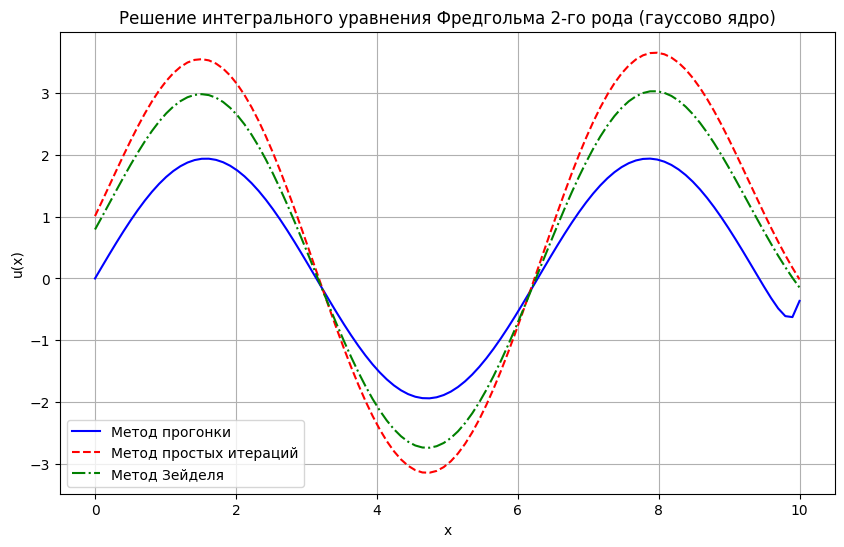
**Метод Зейделя:**

1. x values:

[ 0. 0.1010101 0.2020202 0.3030303 0.4040404 0.50505051

1. 0.60606061 0.70707071 0.80808081 0.90909091 1.01010101 1.11111111
2. 1.21212121 1.31313131 1.41414141 1.51515152 1.61616162 1.71717172
3. 1.81818182 1.91919192 2.02020202 2.12121212 2.22222222 2.32323232
4. 2.42424242 2.52525253 2.62626263 2.72727273 2.82828283 2.92929293
5. 3.03030303 3.13131313 3.23232323 3.33333333 3.43434343 3.53535354
6. 3.63636364 3.73737374 3.83838384 3.93939394 4.04040404 4.14141414
7. 4.24242424 4.34343434 4.44444444 4.54545455 4.64646465 4.74747475
8. 4.84848485 4.94949495 5.05050505 5.15151515 5.25252525 5.35353535
9. 5.45454545 5.55555556 5.65656566 5.75757576 5.85858586 5.95959596
10. 6.06060606 6.16161616 6.26262626 6.36363636 6.46464646 6.56565657
11. 6.66666667 6.76767677 6.86868687 6.96969697 7.07070707 7.17171717
12. 7.27272727 7.37373737 7.47474747 7.57575758 7.67676768 7.77777778
13. 7.87878788 7.97979798 8.08080808 8.18181818 8.28282828 8.38383838
14. 8.48484848 8.58585859 8.68686869 8.78787879 8.88888889 8.98989899
15. 9.09090909 9.19191919 9.29292929 9.39393939 9.49494949 9.5959596
16. 9.6969697 9.7979798 9.8989899 10. ]
17. u(x) values:
18. [ 0.79297498 1.0022182 1.21452588 1.4270393 1.63676893 1.84067353
19. 2.03573781 2.21904404 2.38783469 2.53956388 2.67193746 2.78294194
20. 2.87086392 2.93430153 2.97216985 2.98370197 2.96844684 2.92626506
21. 2.85732285 2.76208458 2.64130367 2.49601187 2.32750653 2.13733597
22. 1.92728252 1.69934359 1.4557105 1.19874556 0.93095725 0.65497404
23. 0.37351694 0.08937118 -0.1946428 -0.47569857 -0.75099312 -1.01777575
24. -1.27337663 -1.51523427 -1.74092204 -1.94817308 -2.13490373 -2.29923493
25. -2.43951156 -2.55431945 -2.6424999 -2.70316158 -2.73568967 -2.73975216
26. -2.71530319 -2.66258355 -2.58211813 -2.47471047 -2.34143449 -2.18362336
27. -2.00285575 -1.80093954 -1.5798931 -1.3419245 -1.08940863 -0.82486268
28. -0.55092003 -0.27030303 0.01420532 0.29979022 0.58363438 0.8629474
29. 1.13499492 1.39712731 1.64680748 1.88163768 2.09938499 2.29800514
30. 2.47566451 2.63076021 2.76193781 2.86810676 2.94845341 3.00245133
31. 3.02986913 3.03077551 3.00554164 2.95484063 2.8796441 2.78121536
32. 2.66109906 2.52110662 2.36329693 2.18995162 2.00354431 1.80670331
33. 1.60216789 1.39273824 1.18122034 0.97036733 0.7628199 0.56104842
34. 0.36730031 0.18355566 0.01149408 -0.14752516]



**Общий график:**

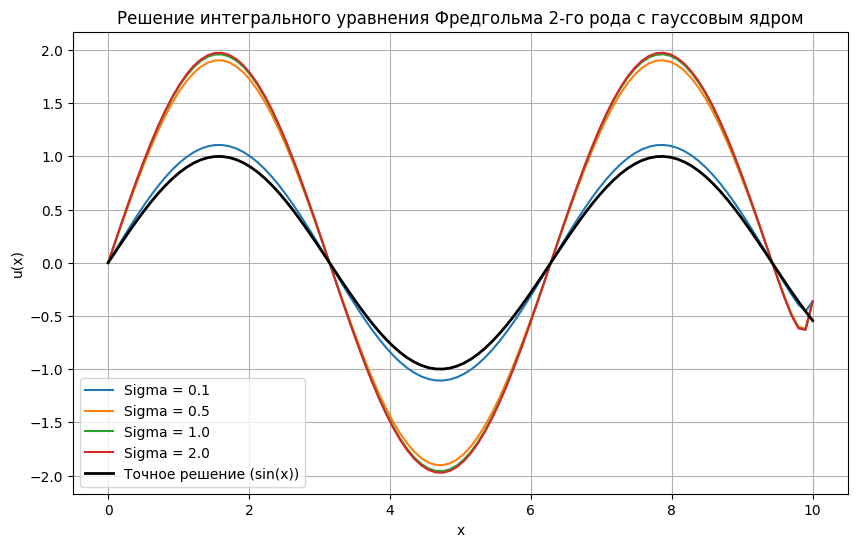
**Общий график с разным масштабом ядра:**

Точность для sigma = 0.1: 1.813404e-01

Точность для sigma = 0.5: 9.037819e-01

Точность для sigma = 1.0: 9.598748e-01

Точность для sigma = 2.0: 9.745591e-01

****

**Вывод**В рамках работы были использованы численные методы для решения интегрального уравнения Фредгольма 2-го рода с использованием **гауссовского ядра,** а также рассмотрен пример с модификацией этого ядра, включающей изменение масштаба (изменение параметра в ядре). Решение задачи было получено с использованием трех различных методов: **метод прогонки, метод простых итераций** и **метод Зейделя**. Каждый из этих методов был адаптирован для работы с гауссовым ядром, и в процессе исследования была проанализирована их эффективность.

**При заданных в начале работы параметрах (**a=0, b=10 – границы интервала; N=100 – количество узлов и λ=0.5) м**етод прогонки** продемонстрировал отличные результаты. При применении гауссовского ядра метод прогонки оказался быстрым и точным, так как его сложность для трехдиагональных систем минимальна. **Метод простых итераций** и **метод Зейделя** являются итерационными методами, которые также показали свою применимость. С увеличением масштаба (параметра λ) в гауссовом ядре увеличивается вклад соседних точек в интеграл, что может повлиять на сходимость этих методов. Несмотря на это, оба метода показали сходимость при достаточном числе итераций.

**Приложение:**#метод прогонки

import numpy as np

import matplotlib.pyplot as plt

# Метод прогонки для тридиагональной системы

def tridiagonal\_solver(a, b, c, d):

    """

    Решает систему линейных уравнений Ax = d для тридиагональной матрицы A.

    a, b, c - коэффициенты тридиагональной матрицы A, d - вектор правых частей.

    """

    n = len(d)

    alpha = np.zeros(n)

    beta = np.zeros(n)

    # Прямой ход прогонки

    alpha[0] = b[0]

    beta[0] = d[0] / alpha[0]

    for i in range(1, n):

        alpha[i] = b[i] - a[i-1] \* c[i-1] / alpha[i-1

        beta[i] = (d[i] - a[i-1] \* beta[i-1]) / alpha[i]

    # Обратный ход прогонки

    x = np.zeros(n)

    x[-1] = beta[-1]

    for i in range(n-2, -1, -1):

        x[i] = beta[i] - c[i] \* x[i+1] / alpha[i]

    return x

# Основная функция решения интегрального уравнения с методом прогонки

def solve\_integral\_equation(K, f, a, b, N, lambda\_val):

    """

    Решает интегральное уравнение Фредгольма 2-го рода с использованием метода прогонки.

    K - ядро интеграла

    f - правая часть уравнения

    a, b - границы интеграла

    N - количество разбиений (дискретизация)

    lambda\_val - параметр lambda

    """

    h = (b - a) / (N - 1)  # шаг

    x = np.linspace(a, b, N)  # сетка

    # Дискретизация правой части

    f\_values = f(x)

    # Формируем коэффициенты для тридиагональной системы

    a\_diag = np.zeros(N-1)  # Нижняя диагональ

    b\_diag = np.zeros(N)    # Основная диагональ

    c\_diag = np.zeros(N-1)  # Верхняя диагональ

    # Заполнение диагональных элементов

    for i in range(1, N-1):

        b\_diag[i] = 1 + lambda\_val \* K(x[i], x[i])

        a\_diag[i-1] = -lambda\_val \* K(x[i], x[i-1])

        c\_diag[i] = -lambda\_val \* K(x[i], x[i+1])

    # Граничные значения для основной диагонали

    b\_diag[0] = 1 + lambda\_val \* K(x[0], x[0])

    b\_diag[-1] = 1 + lambda\_val \* K(x[-1], x[-1])

    # Преобразуем систему к виду Ax = f

    d = f\_values.copy()

    # Решаем систему с помощью метода прогонки

    u = tridiagonal\_solver(a\_diag, b\_diag, c\_diag, d)

    return u, x

# Пример использования:

def K(x, t):

    """Пример ядра (гауссово ядро)"""

    return np.exp(-(x - t)\*\*2)

def f(x):

    """Пример правой части уравнения"""

    return np.sin(x)

# Параметры задачи

a, b = 0, 10  # границы

N = 100  # количество узлов

lambda\_val = 0.5  # параметр lambda

# Получаем решение

u, x = solve\_integral\_equation(K, f, a, b, N, lambda\_val)

# Вывод числовых решений

print("x values:")

print(x)

print("u(x) values:")

print(u)

# Отображение решения на графике

plt.plot(x, u, label="Решение методом прогонки")

plt.xlabel("x")

plt.ylabel("u(x)")

plt.title("Решение интегрального уравнения Фредгольма 2-го рода")

plt.legend()

plt.grid(True)

plt.show()

#метод простых итераций

import numpy as np

import matplotlib.pyplot as plt

# Метод простых итераций для решения интегрального уравнения

def simple\_iteration\_solver(K, f, a, b, N, lambda\_val, max\_iter=100, tol=1e-6):

    """

    Решает интегральное уравнение Фредгольма 2-го рода с использованием метода простых итераций.

    K - ядро интеграла

    f - правая часть уравнения

    a, b - границы интеграла

    N - количество разбиений (дискретизация)

    lambda\_val - параметр lambda

    max\_iter - максимальное количество итераций

    tol - точность сходимости

    """

    h = (b - a) / (N - 1)  # шаг

    x = np.linspace(a, b, N)  # сетка

    # Дискретизация правой части

    f\_values = f(x)

    # Инициализация приближения

    u = np.zeros(N)

    # Итерации метода простых итераций

    for k in range(max\_iter):

        u\_new = np.zeros(N)

        # Вычисление нового приближения для каждого узла

        for i in range(N):

            sum\_term = 0

            for j in range(N):

                sum\_term += K(x[i], x[j]) \* u[j] \* h  # интегрирование по сетке

            u\_new[i] = f\_values[i] + lambda\_val \* sum\_term

        # Проверка на сходимость

        if np.linalg.norm(u\_new - u, np.inf) < tol:

            print(f"Метод сходится за {k+1} итераций.")

            break

        u = u\_new

    return u, x

# Пример использования:

def K(x, t):

    """Пример ядра (гауссово ядро)"""

    return np.exp(-(x - t)\*\*2)

def f(x):

    """Пример правой части уравнения"""

    return np.sin(x)

# Параметры задачи

a, b = 0, 10  # границы

N = 100  # количество узлов

lambda\_val = 0.5  # параметр lambda

# Получаем решение

u, x = simple\_iteration\_solver(K, f, a, b, N, lambda\_val)

# Вывод числовых решений

print("x values:")

print(x)

print("u(x) values:")

print(u)

# Отображение решения на графике

plt.plot(x, u, label="Решение методом простых итераций")

plt.xlabel("x")

plt.ylabel("u(x)")

plt.title("Решение интегрального уравнения Фредгольма 2-го рода")

plt.legend()

plt.grid(True)

plt.show()

#метод Зейделя

import numpy as np

import matplotlib.pyplot as plt

# Метод Зейделя для решения интегрального уравнения

def seidel\_solver(K, f, a, b, N, lambda\_val, max\_iter=100, tol=1e-6):

    """

    Решает интегральное уравнение Фредгольма 2-го рода с использованием метода Зейделя.

    K - ядро интеграла

    f - правая часть уравнения

    a, b - границы интеграла

    N - количество разбиений (дискретизация)

    lambda\_val - параметр lambda

    max\_iter - максимальное количество итераций

    tol - точность сходимости

    """

    h = (b - a) / (N - 1)  # шаг

    x = np.linspace(a, b, N)  # сетка

    # Дискретизация правой части

    f\_values = f(x)

    # Инициализация приближения

    u = np.zeros(N)

    # Итерации метода Зейделя

    for k in range(max\_iter):

        u\_new = u.copy()

        # Обновление приближений для каждого узла

        for i in range(N):

            sum\_term = 0

            for j in range(N):

                if j != i:

                    sum\_term += K(x[i], x[j]) \* u[j] \* h  # интегрирование по сетке

            u\_new[i] = f\_values[i] + lambda\_val \* sum\_term

        # Проверка на сходимость

        if np.linalg.norm(u\_new - u, np.inf) < tol:

            print(f"Метод сходится за {k+1} итераций.")

            break

        u = u\_new

    return u, x

# Пример использования:

def K(x, t):

    """Пример ядра (гауссово ядро)"""

    return np.exp(-(x - t)\*\*2)

def f(x):

    """Пример правой части уравнения"""

    return np.sin(x)

# Параметры задачи

a, b = 0, 10  # границы

N = 100  # количество узлов

lambda\_val = 0.5  # параметр lambda

# Получаем решение

u, x = seidel\_solver(K, f, a, b, N, lambda\_val)

# Вывод числовых решений

print("x values:")

print(x)

print("u(x) values:")

print(u)

# Отображение решения на графике

plt.plot(x, u, label="Решение методом Зейделя")

plt.xlabel("x")

plt.ylabel("u(x)")

plt.title("Решение интегрального уравнения Фредгольма 2-го рода")

plt.legend()

plt.grid(True)

plt.show()

# общее решение интегрального уравнения Фредгольма 2-го рода (гауссово ядро)

import numpy as np

import matplotlib.pyplot as plt

# Метод прогонки для тридиагональной системы

def tridiagonal\_solver(a, b, c, d):

    n = len(d)

    alpha = np.zeros(n)

    beta = np.zeros(n)

    # Прямой ход прогонки

    alpha[0] = b[0]

    beta[0] = d[0] / alpha[0]

    for i in range(1, n):

        alpha[i] = b[i] - a[i-1] \* c[i-1] / alpha[i-1]

        beta[i] = (d[i] - a[i-1] \* beta[i-1]) / alpha[i]

    # Обратный ход прогонки

    x = np.zeros(n)

    x[-1] = beta[-1]

    for i in range(n-2, -1, -1):

        x[i] = beta[i] - c[i] \* x[i+1] / alpha[i]

    return x

# Метод простых итераций для решения интегрального уравнения

def simple\_iteration\_solver(K, f, a, b, N, lambda\_val, max\_iter=100, tol=1e-6):

    h = (b - a) / (N - 1)

    x = np.linspace(a, b, N)

    f\_values = f(x)

    u = np.zeros(N)

    for k in range(max\_iter):

        u\_new = np.zeros(N)

        for i in range(N):

            sum\_term = 0

            for j in range(N):

                sum\_term += K(x[i], x[j]) \* u[j] \* h

            u\_new[i] = f\_values[i] + lambda\_val \* sum\_term

        if np.linalg.norm(u\_new - u, np.inf) < tol:

            print(f"Метод сходится за {k+1} итераций.")

            break

        u = u\_new

    return u, x

# Метод Зейделя для решения интегрального уравнения

def seidel\_solver(K, f, a, b, N, lambda\_val, max\_iter=100, tol=1e-6):

    h = (b - a) / (N - 1)

    x = np.linspace(a, b, N)

    f\_values = f(x)

    u = np.zeros(N)

    for k in range(max\_iter):

        u\_new = u.copy()

        for i in range(N):

            sum\_term = 0

            for j in range(N):

                if j != i:

                    sum\_term += K(x[i], x[j]) \* u[j] \* h

            u\_new[i] = f\_values[i] + lambda\_val \* sum\_term

        if np.linalg.norm(u\_new - u, np.inf) < tol:

            break

        u = u\_new

    return u, x

# Гауссово ядро K(x, t)

def K\_gaussian(x, t):

    return np.exp(-(x - t)\*\*2)

# Правая часть f(x)

def f(x):

    return np.sin(x)  # Правая часть уравнения

# Параметры задачи

a, b = 0, 10  # Границы интервала

N = 100  # Количество узлов (разбиений)

lambda\_val = 0.5  # Параметр lambda

# Получаем решение с методом прогонки

h = (b - a) / (N - 1)

x = np.linspace(a, b, N)

f\_values = f(x)

# Заполнение коэффициентов для тридиагональной системы

a\_diag = np.zeros(N-1)  # Нижняя диагональ

b\_diag = np.zeros(N)    # Основная диагональ

c\_diag = np.zeros(N-1)  # Верхняя диагональ

for i in range(1, N-1):

    b\_diag[i] = 1 + lambda\_val \* K\_gaussian(x[i], x[i])

    a\_diag[i-1] = -lambda\_val \* K\_gaussian(x[i], x[i-1])

    c\_diag[i] = -lambda\_val \* K\_gaussian(x[i], x[i+1])

# Граничные значения для основной диагонали

b\_diag[0] = 1 + lambda\_val \* K\_gaussian(x[0], x[0])

b\_diag[-1] = 1 + lambda\_val \* K\_gaussian(x[-1], x[-1])

# Преобразуем систему к виду Ax = f

d = f\_values.copy()

# Решаем систему с помощью метода прогонки

u\_prog = tridiagonal\_solver(a\_diag, b\_diag, c\_diag, d)

# Решение методом простых итераций

u\_iter, x\_iter = simple\_iteration\_solver(K\_gaussian, f, a, b, N, lambda\_val)

# Решение методом Зейделя

u\_seidel, x\_seidel = seidel\_solver(K\_gaussian, f, a, b, N, lambda\_val)

# Отображение решения на графике

plt.figure(figsize=(10, 6))

plt.plot(x, u\_prog, label="Метод прогонки", linestyle='-', color='b')

plt.plot(x\_iter, u\_iter, label="Метод простых итераций", linestyle='--', color='r')

plt.plot(x\_seidel, u\_seidel, label="Метод Зейделя", linestyle='-.', color='g')

plt.xlabel("x")

plt.ylabel("u(x)")

plt.title("Решение интегрального уравнения Фредгольма 2-го рода (гауссово ядро)")

plt.legend()

plt.grid(True)

plt.show()

# общее решение интегрального уравнения Фредгольма 2-го рода с гауссовым ядром (с разным масштабом)

import numpy as np

import matplotlib.pyplot as plt

# Метод прогонки для тридиагональной системы

def tridiagonal\_solver(a, b, c, d):

    n = len(d)

    alpha = np.zeros(n)

    beta = np.zeros(n)

    # Прямой ход прогонки

    alpha[0] = b[0]

    beta[0] = d[0] / alpha[0]

    for i in range(1, n):

        alpha[i] = b[i] - a[i-1] \* c[i-1] / alpha[i-1]

        beta[i] = (d[i] - a[i-1] \* beta[i-1]) / alpha[i]

    # Обратный ход прогонки

    x = np.zeros(n)

    x[-1] = beta[-1]

    for i in range(n-2, -1, -1):

        x[i] = beta[i] - c[i] \* x[i+1] / alpha[i]

    return x

# Метод простых итераций для решения интегрального уравнения

def simple\_iteration\_solver(K, f, a, b, N, lambda\_val, max\_iter=100, tol=1e-6):

    h = (b - a) / (N - 1)

    x = np.linspace(a, b, N)

    f\_values = f(x)

    u = np.zeros(N)

    for k in range(max\_iter):

        u\_new = np.zeros(N)

        for i in range(N):

            sum\_term = 0

            for j in range(N):

                sum\_term += K(x[i], x[j]) \* u[j] \* h

            u\_new[i] = f\_values[i] + lambda\_val \* sum\_term

        if np.linalg.norm(u\_new - u, np.inf) < tol:

            break

        u = u\_new

    return u, x

# Метод Зейделя для решения интегрального уравнения

def seidel\_solver(K, f, a, b, N, lambda\_val, max\_iter=100, tol=1e-6):

    h = (b - a) / (N - 1)

    x = np.linspace(a, b, N)

    f\_values = f(x)

    u = np.zeros(N)

    for k in range(max\_iter):

        u\_new = u.copy()

        for i in range(N):

            sum\_term = 0

            for j in range(N):

                if j != i:

                    sum\_term += K(x[i], x[j]) \* u[j] \* h

            u\_new[i] = f\_values[i] + lambda\_val \* sum\_term

        if np.linalg.norm(u\_new - u, np.inf) < tol:

            break

        u = u\_new

    return u, x

# Гауссово ядро с масштабом

def K\_gaussian(x, t, sigma=1.0):

    return np.exp(-((x - t) \*\* 2) / (2 \* sigma \*\* 2))  # Гауссово ядро

# Правая часть f(x)

def f(x):

    return np.sin(x)  # Например, правая часть синус

# Метод прогонки для интегрального уравнения

def solve\_integral\_equation(K, f, a, b, N, lambda\_val, sigma):

    h = (b - a) / (N - 1)

    x = np.linspace(a, b, N)

    f\_values = f(x)

    # Формируем коэффициенты для тридиагональной системы

    a\_diag = np.zeros(N-1)

    b\_diag = np.zeros(N)

    c\_diag = np.zeros(N-1)

    # Заполнение диагональных элементов

    for i in range(1, N-1):

        b\_diag[i] = 1 + lambda\_val \* K(x[i], x[i], sigma)

        a\_diag[i-1] = -lambda\_val \* K(x[i], x[i-1], sigma)

        c\_diag[i] = -lambda\_val \* K(x[i], x[i+1], sigma)

    # Граничные значения для основной диагонали

    b\_diag[0] = 1 + lambda\_val \* K(x[0], x[0], sigma)

    b\_diag[-1] = 1 + lambda\_val \* K(x[-1], x[-1], sigma)

    # Преобразуем систему к виду Ax = f

    d = f\_values.copy()

    # Решаем систему с помощью метода прогонки

    u = tridiagonal\_solver(a\_diag, b\_diag, c\_diag, d)

    return u, x

# Параметры задачи

a, b = 0, 10  # границы

N = 100  # количество узлов

lambda\_val = 0.5  # параметр lambda

sigma\_values = [0.1, 0.5, 1.0, 2.0]  # Разные значения масштаба для гауссового ядра

# Получаем решения для разных значений sigma

solutions = {}

for sigma in sigma\_values:

    u\_prog, x = solve\_integral\_equation(K\_gaussian, f, a, b, N, lambda\_val, sigma)

    solutions[sigma] = u\_prog

# Точное решение для сравнения (для примера, можно взять синус)

u\_exact = f(x)

# Оценка точности для разных sigma

errors = {}

for sigma in sigma\_values:

    error = np.linalg.norm(solutions[sigma] - u\_exact, np.inf)

    errors[sigma] = error

# Вывод точности

for sigma in sigma\_values:

    print(f"Точность для sigma = {sigma}: {errors[sigma]:.6e}")

# Отображение решения на графике для разных sigma

plt.figure(figsize=(10, 6))

for sigma in sigma\_values:

    plt.plot(x, solutions[sigma], label=f"Sigma = {sigma}")

plt.plot(x, u\_exact, label="Точное решение (sin(x))", linestyle='-', color='k', linewidth=2)

plt.xlabel("x")

plt.ylabel("u(x)")

plt.title("Решение интегрального уравнения Фредгольма 2-го рода с гауссовым ядром")

plt.legend()

plt.grid(True)

plt.show()