**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Курсовая работа по дисциплине**

**“Численные методы”**

**Вариант 15. Численное решение интегральных уравнений Фредгольма 2-го рода.**

Студент: Блинов М.А.

Группа: 8О-408Б

Преподаватель: Ревизников Д. Л.

Пивоваров Д. Е.

Дата:

Оценка:

**СОДЕРЖАНИЕ:**

[1. ЗАДАНИЕ 3](#_hnw0ysqrl033)

[2. МЕТОД РЕШЕНИЯ 3](#_rafj1onwbpzy)

[3. РЕЗУЛЬТАТЫ 5](#_yl9fsm5tyc0d)

[4. ИСХОДНЫЙ КОД 7](#_e4ppyrsif2fi)

[5. ПРОИЗВОДИТЕЛЬНОСТЬ](#_k2o6548qd2nn) 9

[6. ВЫВОД](#_rg09v4qdm1mf) 10

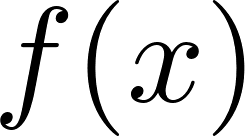
# **1. ЗАДАНИЕ**

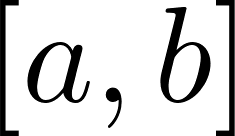
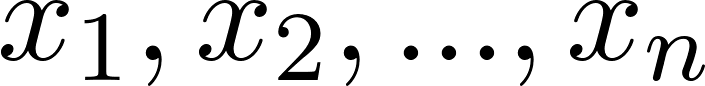
**Постановка задачи:** Написать программу, которая для введенной функции решает интегральное уравнение Фредгольма 2-го рода.

# **2. МЕТОД РЕШЕНИЯ**

Линейное уравнение Фредгольма II рода имеет следующий вид:

[](https://www.codecogs.com/eqnedit.php?latex=y(x)-%5Clambda%20%5Cint_%7Ba%7D%5E%7Bb%7D%20K(x%2C%20s)%20y(s)%20d%20s%3Df(x)%2C%20%5Cquad%20x%20%5Cin%5Ba%2C%20b%5D#0)

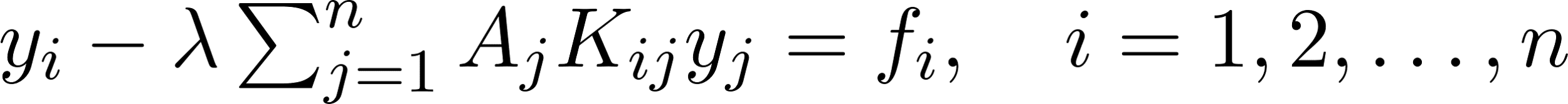
Здесь [](https://www.codecogs.com/eqnedit.php?latex=y(x)#0) — неизвестная функция, [](https://www.codecogs.com/eqnedit.php?latex=K(x%2C%20s)#0) — ядро интегрального уравнения, [](https://www.codecogs.com/eqnedit.php?latex=f(x)#0) — свободный член (правая часть) интегрального уравнения. Для удобства анализа в интегральном уравнении по традиции принято выделять числовой параметр [](https://www.codecogs.com/eqnedit.php?latex=%5Clambda#0), который называют параметром интегрального уравнения.

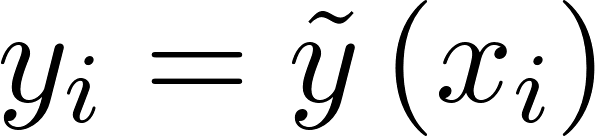
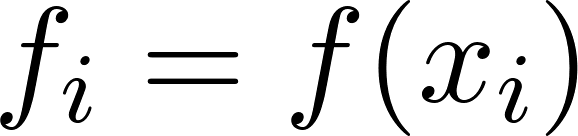
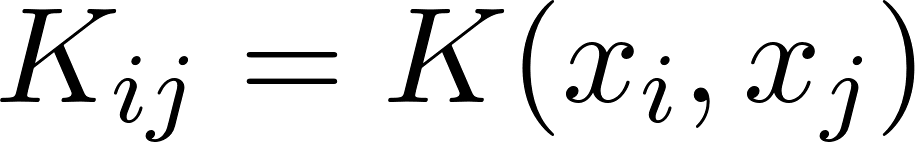
Найдем приближенное решение уравнения методом квадратур. Построим на отрезке [](https://www.codecogs.com/eqnedit.php?latex=%5Ba%2C%20b%5D#0) сетку с узлами [](https://www.codecogs.com/eqnedit.php?latex=x_1%2C%20x_2%2C%20.%20.%20.%20%2C%20x_n#0).

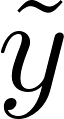
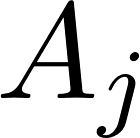
Запишем уравнение (1) в узлах сетки:

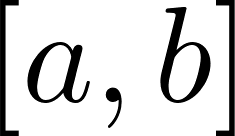
[](https://www.codecogs.com/eqnedit.php?latex=%20y%5Cleft(x_%7Bi%7D%5Cright)-%5Clambda%20%5Cint_%7Ba%7D%5E%7Bb%7D%20K%5Cleft(x_%7Bi%7D%2C%20s%5Cright)%20y(s)%20d%20s%3Df%5Cleft(x_%7Bi%7D%5Cright)%2C%20%5Cquad%20i%3D1%2C2%2C%20%5Cldots%2C%20n%20#0)

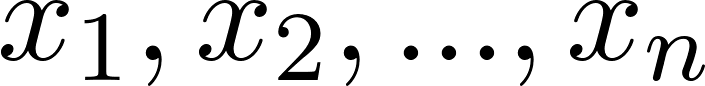
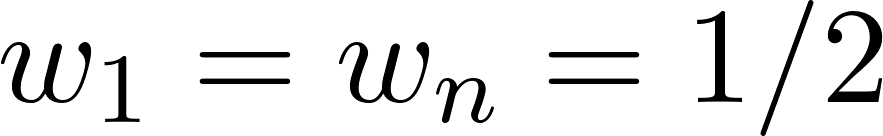
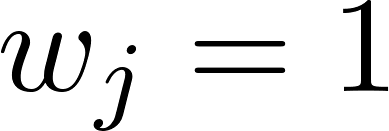
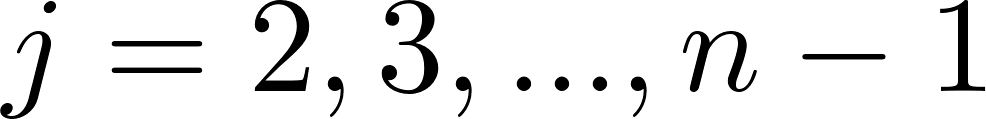
Аппроксимируем интегралы в равенствах (2) конечными суммами с помощью одной из квадратурных формул:

[](https://www.codecogs.com/eqnedit.php?latex=y_%7Bi%7D-%5Clambda%20%5Csum_%7Bj%3D1%7D%5E%7Bn%7D%20A_%7Bj%7D%20K_%7Bi%20j%7D%20y_%7Bj%7D%3Df_%7Bi%7D%2C%20%5Cquad%20i%3D1%2C2%2C%20%5Cldots%2C%20n#0)

Здесь [](https://www.codecogs.com/eqnedit.php?latex=y_%7Bi%7D%3D%5Ctilde%7By%7D%5Cleft(x_%7Bi%7D%5Cright)#0), [](https://www.codecogs.com/eqnedit.php?latex=f_i%20%3D%20f(x_i)#0), [](https://www.codecogs.com/eqnedit.php?latex=K_%7Bij%7D%20%3D%20K(x_i%20%2C%20x_j)#0),

[](https://www.codecogs.com/eqnedit.php?latex=%5Ctilde%7By%7D#0) — приближение к искомой функции [](https://www.codecogs.com/eqnedit.php?latex=y#0), [](https://www.codecogs.com/eqnedit.php?latex=A_j#0) — веса квадратурной формулы.

Решение системы уравнений дает приближенные значения искомой функции в узлах [](https://www.codecogs.com/eqnedit.php?latex=x_i#0) . По ним с помощью интерполяции можно построить приближенное решение интегрального уравнения на всем отрезке [](https://www.codecogs.com/eqnedit.php?latex=%5Ba%2C%20b%5D#0).

Пусть [](https://www.codecogs.com/eqnedit.php?latex=%5Clambda%20%3D%201#0), а сетка [](https://www.codecogs.com/eqnedit.php?latex=x_1%2C%20x_2%2C%20.%20.%20.%20%2C%20x_n#0) — равномерная с шагом [](https://www.codecogs.com/eqnedit.php?latex=h#0). Используем квадратурную формулу трапеций. Тогда система линейных алгебраических уравнений (3) примет следующий вид:где [](https://www.codecogs.com/eqnedit.php?latex=w_1%20%3D%20w_n%20%3D%201%2F2%20#0), [](https://www.codecogs.com/eqnedit.php?latex=w_j%20%3D%201#0) при [](https://www.codecogs.com/eqnedit.php?latex=j%20%3D%202%2C%203%2C%20.%20.%20.%20%2C%20n%20-%201#0).

# 

# **3. РЕЗУЛЬТАТЫ**

# **# Параметры**

# **h = np.pi / 10**

# **a = -np.pi**

# **b = np.pi**

# **\_lambda = 3 / (10 \* np.pi)**

# 

# **# Ядро уравнения**

# **def K(x1, s):**

# **return 1 / (0.64 \* (np.cos((x1 + s) / 2)) \*\* 2 - 1) \* \_lambda**

# 

# **# Правая часть уравнения**

# **def f(x1):**

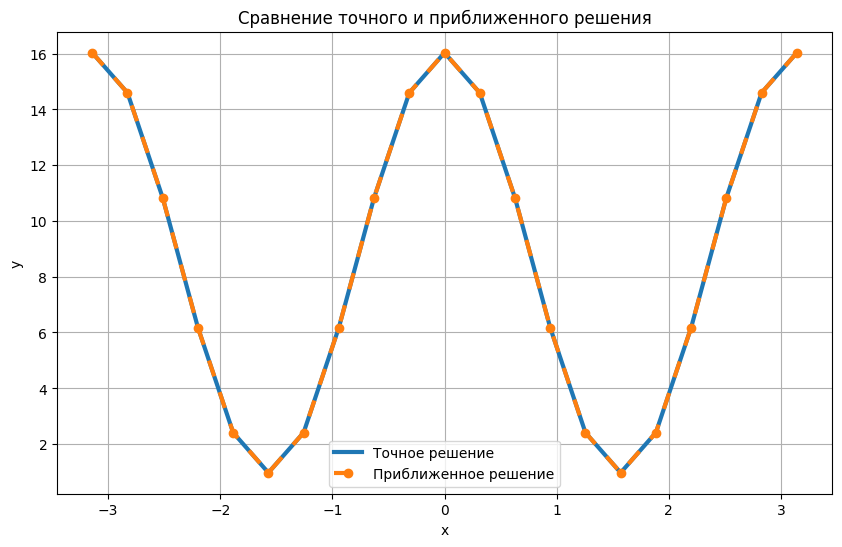
# **return 25 - 16 \* (np.sin(x1)) \*\* 2**

# 

# **# Точное решение**

# **def y\_exact(x1):**

# **return 17 / 2 + 128 / 17 \* np.cos(2 \* x1)**

*Рисунок 1 - График, отображающий точные и численные значения решений на интервалах 𝑥 для примера 1*

# *Рисунок 2 - консольное взаимодействие программы с пользователем*

# 

*Таблица 1 - сравнение точного решения и метода квадратур*

# **4. ИСХОДНЫЙ КОД**

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

**def** fredholm\_method(K, f, a, b, h, iterations**=**1000, tol**=**1e-6):

# Разделяем область интегрировани [a, b] на h частей

x **=** np.arange(a, b **+** h, h)

# Определяем количество отрезков

n **=** len(x)

# Создаем матрицу размера n x n, состоящую из 0

A **=** np.zeros((n, n))

**for** i **in** range(n):

A[i, 0] **=** **-**h **\*** 0.5 **\*** K(x[i], x[0])

**for** j **in** range(1, n**-**1):

A[i, j] **=** **-**h **\*** K(x[i], x[j])

A[i, n**-**1] **=** **-**h **\*** 0.5 **\*** K(x[i], x[n**-**1])

A[i, i] **+=** 1

B **=** np.array([f(xi) **for** xi **in** x]).reshape(**-**1, 1)

# Начальное приближение

x\_0 **=** np.zeros\_like(B)

# Метод простых итераций

x\_new **=** x\_0.copy()

**for** k **in** range(iterations):

**for** i **in** range(n):

# Вычисляем новое значение для x\_i

x\_new[i] **=** (B[i] **-** np.dot(A[i], x\_0) **+** A[i, i] **\*** x\_0[i]) **/** A[i, i]

# Проверяем условие сходимости

**if** np.linalg.norm(x\_new **-** x\_0) **<** tol:

**return** x\_new

# Обновляем предыдущее приближение

x\_0 **=** x\_new.copy()

**return** x\_new

# Параметры

h **=** np.pi **/** 10

a **=** **-**np.pi

b **=** np.pi

\_lambda **=** 3 **/** (10 **\*** np.pi)

# Ядро уравнения

**def** K(x1, s):

**return** 1 **/** (0.64 **\*** (np.cos((x1 **+** s) **/** 2)) **\*\*** 2 **-** 1) **\*** \_lambda

# Правая часть уравнения

**def** f(x1):

**return** 25 **-** 16 **\*** (np.sin(x1)) **\*\*** 2

# Точное решение

**def** y\_exact(x1):

**return** 17 **/** 2 **+** 128 **/** 17 **\*** np.cos(2 **\*** x1)

x\_values **=** np.arange(a, b **+** h, h)

# Решение уравнения

y\_approx **=** fredholm\_method(K, f, a, b, h)

exact **=** y\_exact(x\_values)

print("Точное решение", y\_exact(np.arange(a, b **+** h, h)))

print("Метод квадратур", y\_approx.flatten())

# 

# **5. ПРОИЗВОДИТЕЛЬНОСТЬ** В данном разделе я исследовал время работы программы при изменении точности вычислений (параметр h).

| **Интервалы** | 10 | 100 | 1000 | 10000 |
| --- | --- | --- | --- | --- |
| **Время, ms** | 1.37 | 82.4 | 244.5 | 1532.43 |

# 

# 

# **6. ВЫВОД**

В результате работы над курсовой работой мною были изучены методы реализации интегральных уравнений Фредгольма 2-го рода. Из всех методов мне приглянулся метод квадратур: его простоты и следствию из определения самого интеграла; возможность регулировать временные затраты.