МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Курсовая работа
по курсу «Численные методы»
на тему «Численное решение интегральных уравнений Фредгольма 2-го рода»

Выполнил: А.В. Клитная
Группа: 8О-408Б-20

Москва, 2023

Оглавление

Условие

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

Метод решения

Линейным интегральным уравнением Фредгольма 2-го рода называется уравнение вида

$$\varphi(x) - \lambda^1 \int_a^b [k(x_1 y) \cdot \varphi(y)]\, dy = f(x)$$

Где φ(x) − неизвестная функция, а K(x,y) и f (x) – известные функции, x и t – действительные переменные, изменяющиеся в [a,b], λ-числовой множитель.

Функция K(x,t)называется ядром интегрального уравнения и предполагается, что ядро определено в квадрате a<=x<=b (аналогично для y) на плоскости (x, y) и непрерывно в этом квадрате. Сама же функция f (x) непрерывна или имеет устранимый разрыв(1-го рода).

Я решила в написании программы я решила использовать метод трапеций. Немного о самом методе:

Метод трапеций является одним из численных методов решения интегральных уравнений Фредгольма 2-го рода. Он основан на аппроксимации интеграла с помощью суммы площадей трапеций.

Алгоритм метода трапеций для решения интегрального уравнения Фредгольма 2-го рода выглядит следующим образом:

1. Выберите равномерную сетку узлов на интервале интегрирования от a до b. И вводим саму функцию.
2. Вычислите значения ядра уравнения K(x, y) в каждом узле y и значения правой части f(x) в каждом узле x.
3. Составьте систему линейных уравнений, в которой неизвестными являются значения функции в узлах x. Коэффициенты системы уравнений вычисляются с помощью ядра уравнения K(x, н) и функции правой части f(x).
4. Решите полученную систему линейных уравнений методом.
5. Используя найденные значения функции, вычислите значения интеграла по формуле суммы площадей трапеций:

   F(x) ≈ h/2 * [f(x_0) + 2 * f(x_1) + 2 * f(x_2) + ... + 2 * f(x_n-1) + f(x_n)],

   где h - шаг сетки, x_i - значения узлов.

В результате получаем нужное нам значение.

При этом алгоритм более понятен и раскрыт, благодаря применению библиотек питона, которые позволили реализовать ввод самой функции и ядра с клавиатуры, что делает алгоритм более универсальным для пользователя.

Результаты

Программа работает с функциями, введенными в стиле Питона (те синтаксис подобный x^2 не примется программой). Но синусы и косинусы можно вводить просто названием (см пример ниже картинкой). Для работы с программой необходимо ввести границы, число узлов, ядро и саму функцию:

```
=============== RESTART: C:/labs/4 курс/численные методы/ср/2.py ===============
введите границы интегрирования a и b:
1.0
3.0
Введите количества узлов
100
Введите функцию ядра (от переменных x и y): x+0*y
Введите функцию f (от переменной x): log(x)
Приближенное значение интеграла: 255.68087574334834
Желаете продолжить? (y/n): y
введите границы интегрирования a и b:
0.0
1.0
Введите количества узлов
100
Введите функцию ядра (от переменных x и y): x*y
Введите функцию f (от переменной x): sin(x) * cos(pi*x)
Приближенное значение интеграла: -8.562712605683862
Желаете продолжить? (y/n): y
введите границы интегрирования a и b:
0.0
1.0
Введите количества узлов
1000
Введите функцию ядра (от переменных x и y): x*y
Введите функцию f (от переменной x): asin(x)
Приближенное значение интеграла: 195.76559361727894
```

Выводы

В результате работы над курсовой работой мною были изучены методы реализации интегральных уравнений Фредгольма 2-го рода. Из всех методов мне приглянулся метод трапеций из-за нескольких его свойств: его простоты и следствию из определения самого интеграла; возможность регулировать временные затраты. В результате была проделана довольно интересная работа с ранее неизвестным алгоритмом.

Приложение:

```
from math import sin, cos, exp, pi, log, tan, asin, acos, atan
import math
import numpy as np

# Функция для считывания пользовательской функции
def read_function(prompt):
    while True:
        try:
            # Ввод пользовательской функции
            function_str = input(prompt)
            # Вычисление функции с использованием eval()
            function = eval('lambda x, y: ' + function_str)
            break
        except:
            print("Ошибка! Пожалуйста, введите корректную функцию.")
    return function

def read_functionx(prompt):
    while True:
        try:
            # Ввод пользовательской функции
            function_str = input(prompt)
            # Вычисление функции с использованием eval()
            function = eval('lambda x: ' + function_str)
            break
        except:
            print("Ошибка! Пожалуйста, введите корректную функцию.")
    return function
'''
```

```
# Задание функции ядра
def kernel(x, y):
    # TODO: Вставьте код для вычисления функции ядра
    return x*y

# Задание функции f
def f(x):
    # TODO: Вставьте код для вычисления функции f
    return 0
'''
while True:
    # Задание границы интегрирования
    print("введите границы интегрирования a и b: ")
    a = float(input())#0
    b = float(input())#1

    print("Введите количества узлов")

    # Задание количества узлов
    n = int(input())#100

    # Расчет шага интегрирования
    h = (b - a) / n

    # Считывание функции ядра
    kernel_prompt = "Введите функцию ядра (от переменных x и y): "
    kernel = read_function(kernel_prompt)

    # Считывание функции f
    f_prompt = "Введите функцию f (от переменной x): "
    f = read_functionx(f_prompt)
```

```python
# Вычисление матрицы A
A = [[0] * n for _ in range(n)]
for i in range(n):
    for j in range(n):
        x = a + i * h
        y = a + j * h
        A[i][j] = kernel(x, y)

# Вычисление вектора B
B = [0] * n
for i in range(n):
    x = a + i * h
    B[i] = f(x)

# Решение системы линейных уравнений
x = [0] * n
for i in range(n):
    for j in range(n):
        x[i] += A[i][j] * B[j]

# Вычисление приближенного значения интеграла
integral = 0
for i in range(n):
    integral += x[i] * h

# Вывод результата
print("Приближенное значение интеграла:", integral)
choice = input("Желаете продолжить? (y/n): ")
if choice.lower() != "y":
    break
```