|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**Лабораторная работа №3**

**«Ряды Фурье»**

**ДИСЦИПЛИНА: «Моделирование»**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-62Б | |  |  | ( | Калашников А.С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Никитенко У.В. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

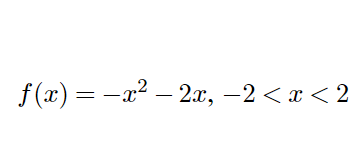
Калуга, 2023

**Цель работы:** сформировать практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений и визуализации результатов численного или приближенно-аналитического решения ДУЧП2 гиперболического типа на основе сравнения результатов.

**Задачи:** решить уравнение, указанное в варианте методом разделения переменных (Фурье), выдвинуть и обосновать гипотезу целесообразности использования того или иного метода в зависимости от предложенной задачи и ее вариаций, точности результата, трудоемкости, сложности алгоритма, сложности обоснования применимости метода, вычислительной эффективности алгоритма. Визуализировать результаты.

**Задача №1**

Разложить функцию



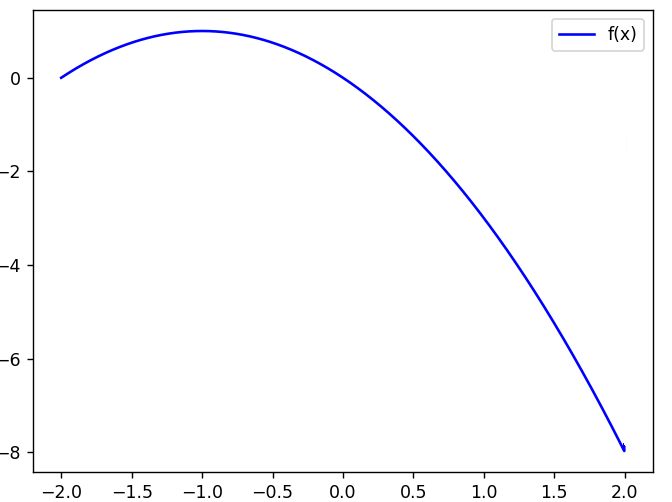
в тригонометрический ряд Фурье. Построить графики функции, суммы ряда, а также частичных сумм S1(x) S2(x) S3(x). Используя данное разложение, аппроксимировать функцию тригонометрическим полиномом третьего порядка и вычислить среднее квадратичное отклонение.

**ПОРЯДОК РЕШЕНИЯ**

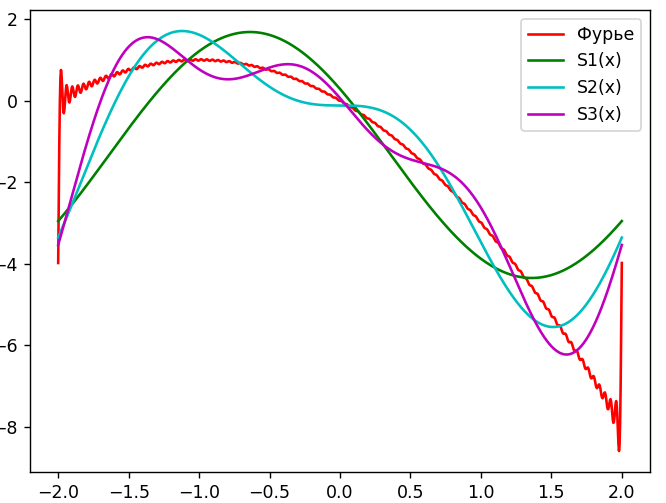
1. Найти коэффициенты
2. Составить разностную схему второго порядка точности.
3. Построить график суммы ряда
4. Вычисляем тригонометрический полином третьего порядка
5. Посчитать среднеквадратичное отклонение.

**Результаты выполнения работы**

Посчитаем коэффициенты a0 an bn:

****

**Рис. 1 График функции f(x)**

****

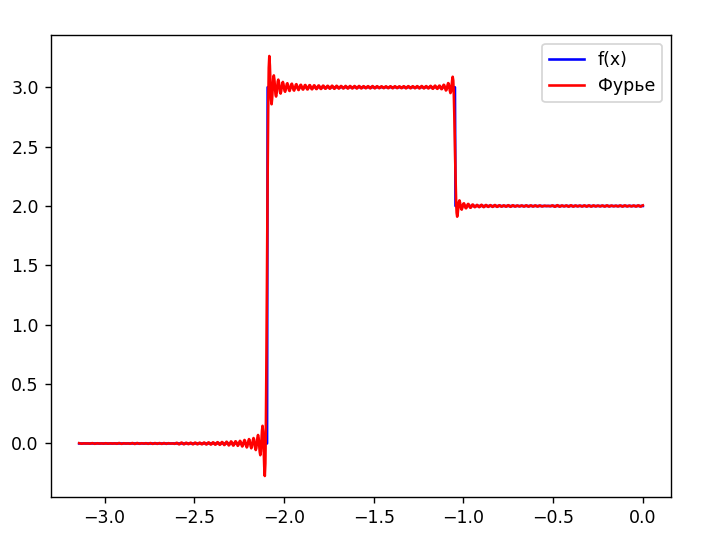
**Рис. 2 Графики функции**

**D:\3-kurs-2-semestr\Моделирование\LB3\Картинки\2.png**

**Рис. 3 Среднее квадратичное отклонение**

**Задача №2**

Функция задана на промежутке (-l,0). Разложить ее по косинусам. В полученный ряд подставить x=l и найти сумму этого числового ряда. Построить графики функции и суммы ряда Фурье



**Рис. 4** График функции и суммы ряда Фурье

**Вывод:** в ходе выполнения работы были сформированы практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений и визуализации результатов численного или приближенно-аналитического решения ДУЧП2 гиперболического типа на основе сравнения результатов.

**ПРИЛОЖЕНИЯ**

**Листинг программы**

**Задание №1**

import numpy as np  
from scipy.integrate import quad  
  
import matplotlib.pyplot as plt  
  
l = 2  
  
def f(x):  
 return np.where((-2 < x) & (x < 2), -x\*\*2 - 2\*x,0)  
  
a0 = 1/l \* (quad(f, -l, l))[0]  
  
  
def an(n): return 1/l \* (quad(lambda x: f(x)  
 \* np.cos(n \* np.pi \* x / l), -l, l))[0]  
  
def bn(n): return 1/l \* (quad(lambda x: f(x)  
 \* np.sin(n \* np.pi \* x / l), -l, l))[0]  
  
N = 100  
a = np.zeros(N+1)  
b = np.zeros(N+1)  
for n in range(1, N+1):  
 a[n] = an(n)  
 b[n] = bn(n)  
  
x = np.linspace(-l, l, 1000)  
  
plt.plot(x, [f(i) for i in x], 'b')  
plt.legend(['f(x)'])  
plt.show()  
  
fourier = a0/2 + sum([a[n] \* np.cos(n \* np.pi \* x / l) +  
 b[n] \* np.sin(n \* np.pi \* x / l) for n in range(1, N+1)])  
plt.plot(x, fourier, 'r')  
  
s1 = a0/2 + sum([a[n] \* np.cos(n \* np.pi \* x / l) + b[n] \*  
 np.sin(n \* np.pi \* x / l) for n in range(1, 2)])  
plt.plot(x, s1, 'g')  
  
s2 = a0/2 + sum([a[n] \* np.cos(n \* np.pi \* x / l) + b[n] \*  
 np.sin(n \* np.pi \* x / l) for n in range(1, 3)])  
plt.plot(x, s2, 'c')  
  
s3 = a0/2 + sum([a[n] \* np.cos(n \* np.pi \* x / l) + b[n] \*  
 np.sin(n \* np.pi \* x / l) for n in range(1, 4)])  
plt.plot(x, s3, 'm')  
  
plt.legend(['Фурье', 'S1(x)', 'S2(x)', 'S3(x)'])  
plt.show()  
  
f\_approx = a0/2 + a[1] \* np.cos(np.pi \* x / l) + b[1] \* np.sin(np.pi \* x / l) + \  
 a[2] \* np.cos(2 \* np.pi \* x / l) + b[2] \* np.sin(2 \* np.pi \* x / l) + \  
 a[3] \* np.cos(3 \* np.pi \* x / l) + b[3] \* np.sin(3 \* np.pi \* x / l)  
  
mse = np.sqrt(np.mean((f(x) - f\_approx)\*\*2))  
  
print("Среднее квадратичное отклонение:", mse)

**Задание №2**

import numpy as np  
from scipy.integrate import quad  
import matplotlib.pyplot as plt  
l = -np.pi  
def f(x):  
 return np.where((-np.pi/3 <= x) & (x <= 0), 2,  
 np.where((-2\*np.pi/3 <= x) & (x < -np.pi/3), 3,  
 np.where((-np.pi <= x) & (x < -2\*np.pi), 1, 0)))  
  
a0 = 2/l \* (quad(f, 0, l))[0]  
def an(n): return 2/l \* (quad(lambda x: f(x)  
 \* np.cos(n \* np.pi \* x / l), 0, l))[0]  
  
N = 100   
a = np.zeros(N+1)  
for n in range(1, N+1):  
 a[n] = an(n)  
x = np.linspace(l, 0, 1000)  
plt.plot(x, [f(i) for i in x], 'b')  
fourier = a0/2 + sum([a[n] \* np.cos(n \* np.pi \* x / l)  
 for n in range(1, N+1)])  
plt.plot(x, fourier, 'r')  
plt.legend(['f(x)', 'Фурье'])  
plt.show()  
res = a0/2 + sum([a[n] \* np.cos(n \* np.pi) for n in range(1, N+1)])  
print(res)