МГТУ им. Н. Э. Баумана

Отчет по лабораторной работе №5 по предмету «Вычислительные алгоритмы»
«Построение и программная реализация алгоритмов численного интегрирования»

Выполнила Параскун София,
ИУ7-44Б
Проверил
Градов В. М.

Москва, 2021 г.

Цель работы: получение навыков построения алгоритма вычисления двукратного интеграла с использованием квадратурных формул Гаусса и Симпсона.

1. Задание

Построить алгоритм и программу для вычисления двукратного интеграла при фиксированном значении параметра τ

$$\varepsilon(\tau) = \frac{4}{\pi}\int_0^{\pi/2} d\varphi \int_0^{\pi/2} [1 - \exp(-\tau \frac{l}{R})] \cos\theta \sin\theta \, d\theta ,$$

$$\text{где} \qquad \frac{l}{R} = \frac{2\cos\theta}{1 - \sin^2\theta\cos^2\varphi},$$

θ, φ - углы сферических координат.

Применить метод последовательного интегрирования. По одному направлению использовать формулу Гаусса, а по другому — формулу Симпсона.

2. Код программы

Программа состоит из файла main.py. Таблица функции генерируется внутри программы.

```
from math import sin, cos, pi, exp, fabs, pow
import numpy as np
import matplotlib.pyplot as plt

Eps = 1e-7

class Integral:
  def __init__(self):
    self.n, self.m = 0, 0
    self.tau = 0
    self.a, self.b, self.c, self.d = 0, pi / 2, 0, pi / 2

  def func(self, tau, teta, phi):
    return (1 - exp(-tau * 2 * cos(teta) / (1 - pow(sin(teta), 2) * pow(cos(phi), 2)))) * c
os(teta) * sin(teta)

  def inputParams(self):
    print("Input N: ", end = '')
    self.n = int(input())
    print("Input M: ", end = '')
    self.m = int(input())
```

```
    print("Input tau: ", end = '')
    self.tau = float(input())

    self.phiInit()

  def phiInit(self):
    self.phi = [0 for i in range(self.n)]
    piece = pi / 2 / self.n
    for i in range(self.n):
      self.phi[i] = piece * i

  def simpsonCalc(self):
    # print("Simpson")
    h = (self.d - self.c) / (self.n - 1)
    res = 0

    for i in range(0, int(self.n / 2 - 1)):
      res += self.integrals[2 * i] + 4 * self.integrals[2 * i + 1] + self.integrals[2 * i +
2]
      # print(res)

    return res * h / 3 * 4 / pi #4/pi - коэф заданной функции

  def gaussCalc(self):
    # print("Gauss")
    x = []
    xLen = 0
    step = 2.0 / self.m
    while (xLen < self.m):
      step /= 2.0
      xLen = 0
      a = -1
      b = a + step
      while (a < 1):
        if self.legendrePolynomCalc(a, self.m) * self.legendrePolynomCalc(b, self.m) < 0:
          xLen += 1
        a = b
        b += step
    a = -1
    b = a + step
    i = 0
    while (a < 1 and i < self.m):
      if self.legendrePolynomCalc(a, self.m) * self.legendrePolynomCalc(b, self.m) < 0:
        x.append(self.bisection(a, b, self.m))
        i += 1
      a = b
      b += step
    rightSLAE = []
    for i in range(0, self.m):
      if (i % 2 == 0):
        rightSLAE.append(2.0 / (i + 1))
      else:
        rightSLAE.append(0)

    helpSLAE = [1 for i in range(self.m)]
    leftSLAE = [[] for i in range(self.m)]
    for i in range(self.m):
```

```
      for j in range(self.m):
        leftSLAE[i].append(helpSLAE[j])
        helpSLAE[j] *= x[j]

    rSLAE = np.asarray(rightSLAE)
    lSLAE = np.asarray(leftSLAE)

    weights = np.linalg.solve(lSLAE, rSLAE)
    # for i in range(self.m):
    #      print("{0} ".format(weights[i]), end = '')
    # print("SLAE ok!")

    for i in range(self.m):
      # M_PI_4 - узнать что за константа!!!!
      x[i] = pi / 4 * (1 + x[i])

    self.integrals = [0 for i in range(self.n)]
    for i in range(self.n):
      for j in range(self.m):
        self.integrals[i] += weights[j] * self.func(self.tau, x[j], self.phi[i])
      self.integrals[i] *= pi / 4
      # M_PI_4 - что за говно вообще??!??!?

  def bisection(self, left, right, n):
    middle = (left + right) / 2
    if fabs(self.legendrePolynomCalc(middle, n) < Eps):
      return middle

    if self.legendrePolynomCalc(left, n) * self.legendrePolynomCalc(middle, n) < 0:
      right = middle
    else:
      left = middle

    while (right - left > Eps):
      if fabs(self.legendrePolynomCalc(middle, n) < Eps):
        return middle

      if self.legendrePolynomCalc(left, n) * self.legendrePolynomCalc(middle, n) < 0:
        right = middle
      else:
        left = middle
      middle = (left + right) / 2
    return middle

  def legendrePolynomCalc(self, x, n):
    if n == 0:
      return 1

    if n == 1:
      return x

    leg0 = 1
    leg1 = x
    leg2 = 0

    for i in range(2, n + 1):
      leg2 = ((2 * i - 1) * x * leg1 - (i - 1) * leg0) / i
```

```
    leg0 = leg1
    leg1 = leg2

  return leg2

def resPlot(self):
  self.equalN()
  self.difN()
  self.difNdifM()

def equalN(self):
  tau = np.linspace(0, 10, 200)
  self.n = 5
  self.phiInit()

  self.m = 2
  res1 = []
  for i in range(len(tau)):
    self.tau = i
    self.gaussCalc()
    res1.append(self.simpsonCalc())
  self.integrals.clear()
  plt.plot(tau, res1, color = "r", label = "N = 5, M = 2")

  self.m = 3
  res2 = []
  for i in range(len(tau)):
    self.tau = i
    self.gaussCalc()
    res2.append(self.simpsonCalc())
  self.integrals.clear()
  plt.plot(tau, res2, color = "g", label = "N = 5, M = 3")

  self.m = 4
  res3 = []
  for i in range(len(tau)):
    self.tau = i
    self.gaussCalc()
    res3.append(self.simpsonCalc())
  self.integrals.clear()
  plt.plot(tau, res3, color = "b", label = "N = 5, M = 4")

  self.m = 5
  res4 = []
  for i in range(len(tau)):
    self.tau = i
    self.gaussCalc()
    res4.append(self.simpsonCalc())
  self.integrals.clear()
  plt.plot(tau, res4, color = "y", label = "N = 5, M = 5")

  plt.xlabel("Tau")
  plt.ylabel("Result")
  plt.grid()
  plt.title("Численное интегрирование")
  plt.legend()
  plt.show()
```

```python
def difN(self):
  tau = np.linspace(0, 10, 200)
  self.n = 3
  self.phiInit()

  self.m = 2
  res1 = []
  for i in range(len(tau)):
    self.tau = i
    self.gaussCalc()
    res1.append(self.simpsonCalc())
  self.integrals.clear()
  plt.plot(tau, res1, color = "r", label = "N = 3, M = 2")

  self.m = 3
  res2 = []
  for i in range(len(tau)):
    self.tau = i
    self.gaussCalc()
    res2.append(self.simpsonCalc())
  self.integrals.clear()
  plt.plot(tau, res2, color = "g", label = "N = 3, M = 3")

  self.n = 5
  self.phiInit()

  self.m = 2
  res3 = []
  for i in range(len(tau)):
    self.tau = i
    self.gaussCalc()
    res3.append(self.simpsonCalc())
  self.integrals.clear()
  plt.plot(tau, res3, color = "b", label = "N = 5, M = 2")

  self.m = 3
  res4 = []
  for i in range(len(tau)):
    self.tau = i
    self.gaussCalc()
    res4.append(self.simpsonCalc())
  self.integrals.clear()
  plt.plot(tau, res4, color = "y", label = "N = 5, M = 3")

  plt.xlabel("Tau")
  plt.ylabel("Result")
  plt.grid()
  plt.title("Численное интегрирование")
  plt.legend()
  plt.show()

def difNdifM(self):
  tau = np.linspace(0, 10, 200)
  self.n, self.m = 3, 3
  self.phiInit()
```

```
    res1 = []
    for i in range(len(tau)):
      self.tau = i
      self.gaussCalc()
      res1.append(self.simpsonCalc())
    self.integrals.clear()
    plt.plot(tau, res1, color = "r", label = "N = 3, M = 3")

    self.n, self.m = 5, 5
    self.phiInit()

    res2 = []
    for i in range(len(tau)):
      self.tau = i
      self.gaussCalc()
      res2.append(self.simpsonCalc())
    self.integrals.clear()
    plt.plot(tau, res2, color = "g", label = "N = 5, M = 5")

    self.n, self.m = 7, 7
    self.phiInit()

    res3 = []
    for i in range(len(tau)):
      self.tau = i
      self.gaussCalc()
      res3.append(self.simpsonCalc())
    self.integrals.clear()
    plt.plot(tau, res3, color = "b", label = "N = 7, M = 7")

    plt.xlabel("Tau")
    plt.ylabel("Result")
    plt.grid()
    plt.title("Численное интегрирование")
    plt.legend()
    plt.show()

if __name__ == "__main__":
  calc = Integral()
  # calc.inputParams()
  # calc.gaussCalc()
  # res = calc.simpsonCalc() * 4 / pi
  # print(res)
  calc.resPlot()
```

3. Результаты работы

1. Описать алгоритм вычисления n корней полинома Лежандра n-ой степени $P_n(x)$ при реализации формулы Гаусса.

Все корни полинома лежат на интервале [-1; 1]. При этом стоит заметить, что интервалы [-1; 0] и [0; 1] – симметричны, так что при поиске корней достаточно рассмотреть интервал [0; 1].

Корни полинома можно вычислить итеративно по методу Ньютона

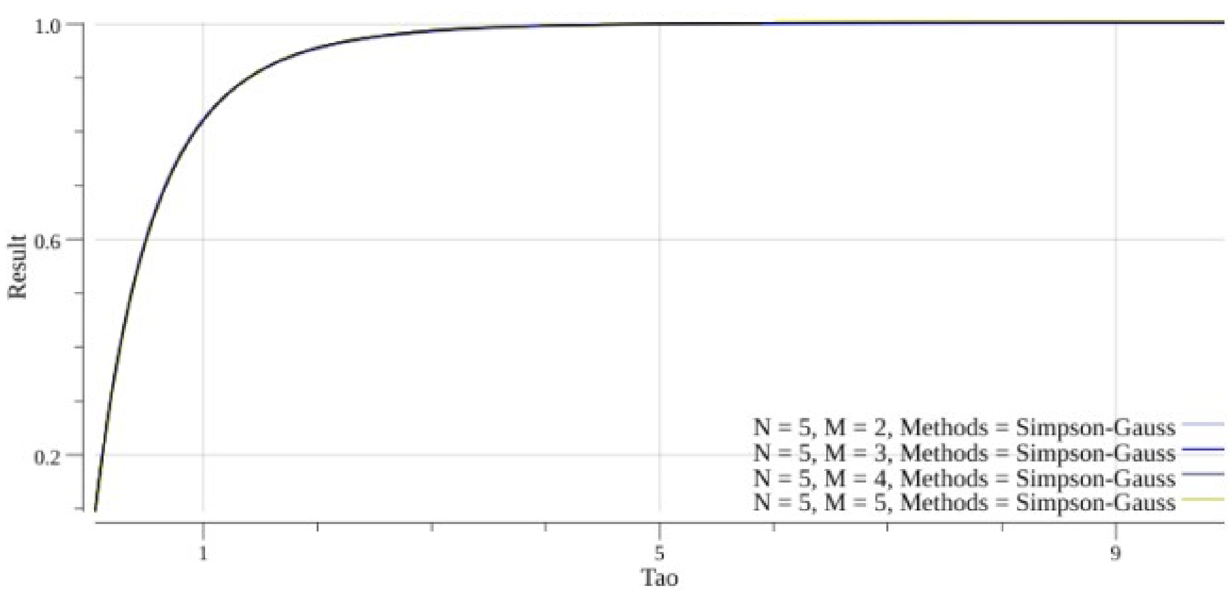
$$x_i^{(k+1)} = x_i^k - \frac{P_n(x_i)^{(k)}}{P_n'(x_i)^{(k)}}$$

Причем начальное приближение для i-ого корня берется по формуле:

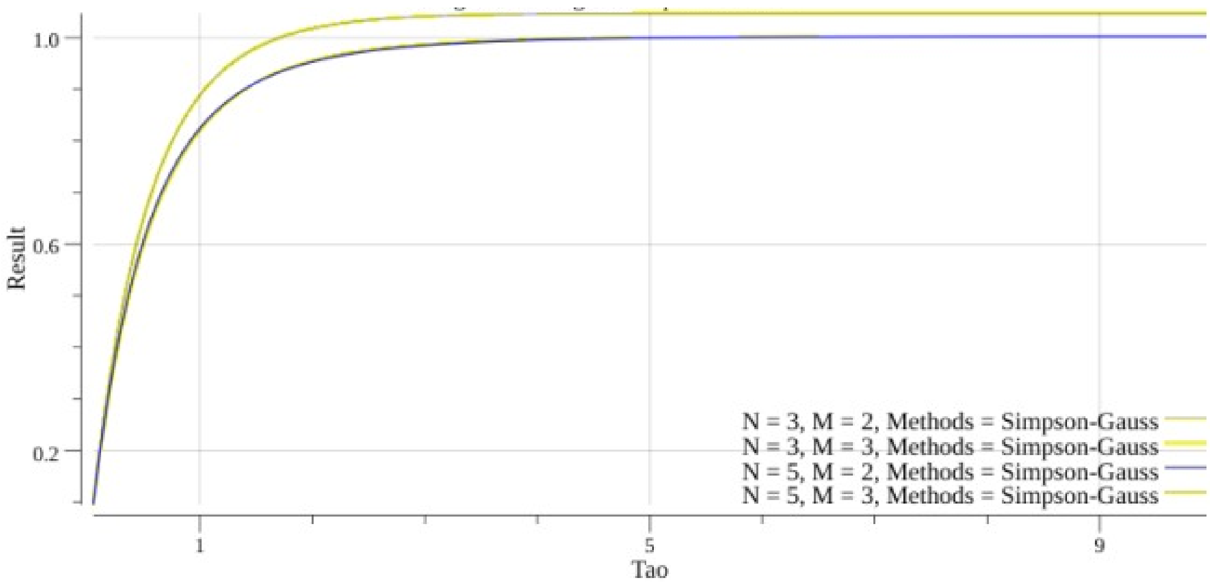
$$x_i^{(0)} = cos[\frac{\pi(4i-1)}{4n+2}]$$

2. Исследовать влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов.

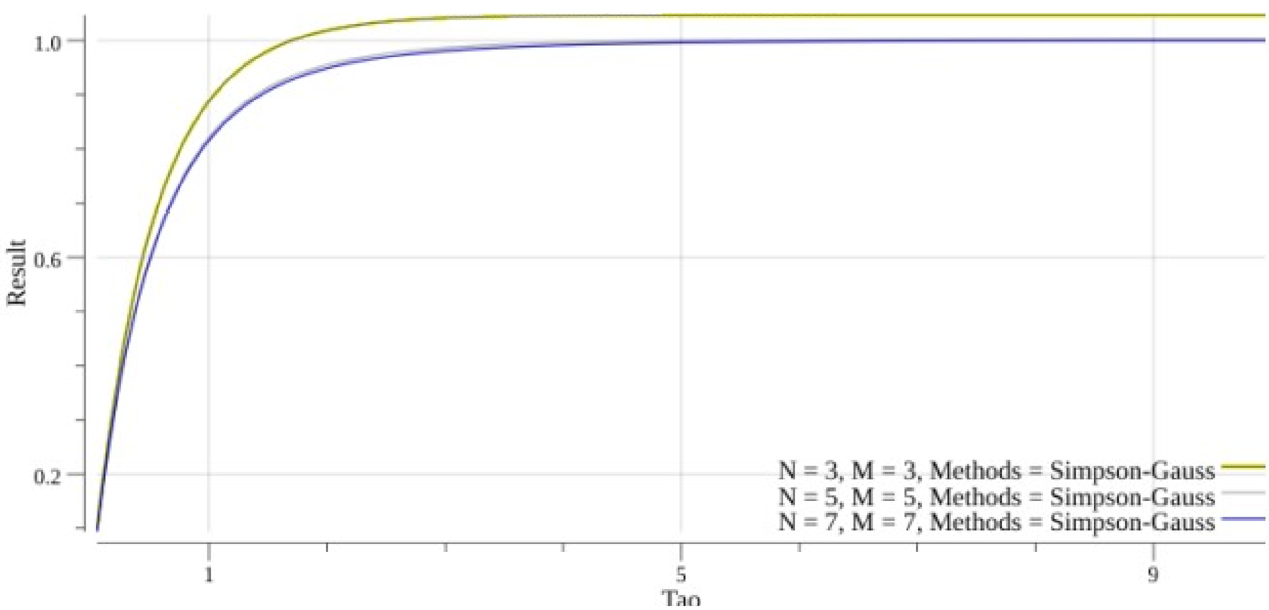
В случае, когда метод Симпсона используется для внешнего интегрирования, при задании для него 5 узлов, метод Гаусса выдает один результат независимо от количества узлов.



При задании меньшего количества узлов для внешнего интегрирования, происходит расхождение с физическим смыслом — больший вклад будет вносить метод, являющийся внешним.



3. Построить график зависимости ε(τ) в диапазоне изменения τ=0.05-10.0. Указать при каком количестве узлов получены результаты.



Исходя из графика, можно сделать вывод, что оптимальной сеткой является 5x5.

4. Вопросы при защите лабораторной работы

1. В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается.

Теоретический порядок квадратурных формул численного интегрирования не достигается в ситуациях, когда подынтегральная функция не имеет соответствующих производных. Порядок точности равен номеру последней существующей производной.

2. Построить формулу Гаусса численного интегрирования при одном узле.

$$\sum_{i=1}^{n} A_i = 2 \quad P_1(x) = x \Rightarrow x = 0$$

Полученная формула:

$$\int_a^b f(x)dx = \frac{b-a}{2} 2f(\frac{b+a}{2} + \frac{b-a}{2} \cdot 0) = (b-a)f(\frac{b+a}{2})$$

3. Построить формулу Гаусса численного интегрирования при двух узлах.

$$P_2(x) = \frac{1}{2}(3x^2 - 1) \ \Rightarrow x = \pm\frac{1}{\sqrt{3}}$$

$$\begin{cases} A_1 + A_2 = 2 \\ -\frac{1}{\sqrt{3}}A_1 + \frac{1}{\sqrt{3}}A_2 = 0 \end{cases} \Rightarrow A_2 = A_1 = 1$$

$$\int_{-1}^{1} f(f)df = f(-\frac{1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}})$$

Полученная формула:

$$\int_a^b f(x)dx = \frac{b-a}{2}(f(\frac{b+a}{2} - \frac{b-a}{2} \cdot \frac{1}{\sqrt{3}}) + f(\frac{b+a}{2} + \frac{b-a}{2} \cdot \frac{1}{\sqrt{3}}))$$

4. Получить обобщенную кубатурную формулу, аналогичную (6.6) из лекции №6, для вычисления двойного интеграла методом последовательного интегрирования на основе формулы трапеций с тремя узлами по каждому направлению.

$$\int_c^d \int_a^b f(x,y)dxdy = \int_a^b dx \int_c^d f(x,y)dy = \int_a^b F(x)dx = h_x(\tfrac{1}{2}F_0 + F_1 + \tfrac{1}{2}F_2) = h_x h_y(\tfrac{1}{2}(\tfrac{1}{2}f(x_0,y_0) + f(x_0,y_1) + \tfrac{1}{2}f(x_0,y_2)) + \tfrac{1}{2}f(x_1,y_0) + f(x_1,y_1) + \tfrac{1}{2}f(x_1,y_2) + \tfrac{1}{2}(\tfrac{1}{2}f(x_2,y_0) + f(x_2,y_1) + \tfrac{1}{2}f(x_2,y_2)))$$

где $h_x = \frac{b-a}{2}$, $h_y = \frac{d-c}{2}$