Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

Лабораторная работа №5

Тема: Численное интегрирование

Студент: Артемьев И.О.

Группа: ИУ7-43Б

Оценка (баллы): _________

Преподаватель: Градов В.М.

Москва.
2021 г.

Задание

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

$$\epsilon(\tau) = \frac{4}{\pi} \int_0^{\frac{\pi}{2}} d\phi \int_0^{\frac{\pi}{2}} [1 - \exp(-\tau \frac{1}{R})] \; cos\theta \; sin\theta \; d\theta \; d\varphi$$

$$\frac{1}{R} = \frac{2cos\theta}{1 - cos^2\theta sin^2\varphi}$$

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

Входные данные

Количество узлов сетки N, M ; значение параметра τ , методы для направлений при последовательном интегрировании.

Выходные данные

Значение интеграла при заданном параметре, график зависимости $\epsilon(\tau)$ в диапазоне $\tau = 0.05 - 10$.

Описание алгоритма

Имеем

$$\int_{-1}^{1} f(t)dt = \sum_{i=1}^{n} A_i f(t_i)$$

Положим

$$\int_{-1}^{1} t^k dt = \sum_{i=1}^{n} A_i f(t_i^k),\ k = 0, 1, 2, ..., 2n-1$$

Тогда, имеем систему

$$\begin{cases} \sum_{i=1}^{n} A_i = 2 \\ \sum_{i=1}^{n} A_i t_i = 0 \\ \sum_{i=1}^{n} A_i t_i^2 = \frac{2}{3} \\ \dots \\ \sum_{i=1}^{n} A_i t_i^{2n-1} = 0 \end{cases}$$

Система нелинейная, найти решение сложно. Для нахождения A_i и t_i можно воспользоваться полиномом Лежандра. Формула полинома:
Узлами формулы Гаусса являются нули полинома Лежандра

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}[(x^2-1)^n],\ n = 0, 1, 2$$

$P_n(t)$, а A_i можно найти из вышеуказанной системы уравнений.

При вычислении интеграла на произвольном интервале $[a, b]$, для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной:

$$x = \frac{b+a}{2} + \frac{b-a}{2}t$$

В таком случае, получаем конечную формулу для произвольного интервала $[a, b]$:

$$\int_a^b f(x)dx = \frac{b-a}{2} \sum_{i=1}^{n} A_i f(x_i)$$

Также, существует квадратнурная формула Симпсона:

$$\int_a^b f(x)dx \approx \frac{h}{3} \sum_{i=0}^{\frac{N}{2}-1} (f_{2i} + 4f_{2i+1} + f_{2i+2})$$

Однако, эти методы можно применять и для приближенной оценки двукратных (и не только) интегралов. Рассмотрим интеграл по прямоугольной области:

$$I = \int_c^d \int_a^b f(x,y)dx\,dy = \int_a^b F(x)dx, \quad \text{где } F(x) = \int_c^d f(x,y)dy$$

По каждой координате введем сетку узлов. Каждый однократный интеграл вычисляют по квадратурным формулам. Для разных направлений можно использовать квадратурные формулы разных порядков точности, в т.ч. и Гаусса.

Конечная формула:

$$I = \int \int_G f(x,y)dx\,dy = \sum_{i=1}^{n} \sum_{j=1}^{m} A_i B_{ij} f(x_i, y_j)$$

где $A_i B_{ij}$ - известные постоянные.

Результаты работы программы

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-го 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.**

Исследование для внешнего направления и метода Симпсона:

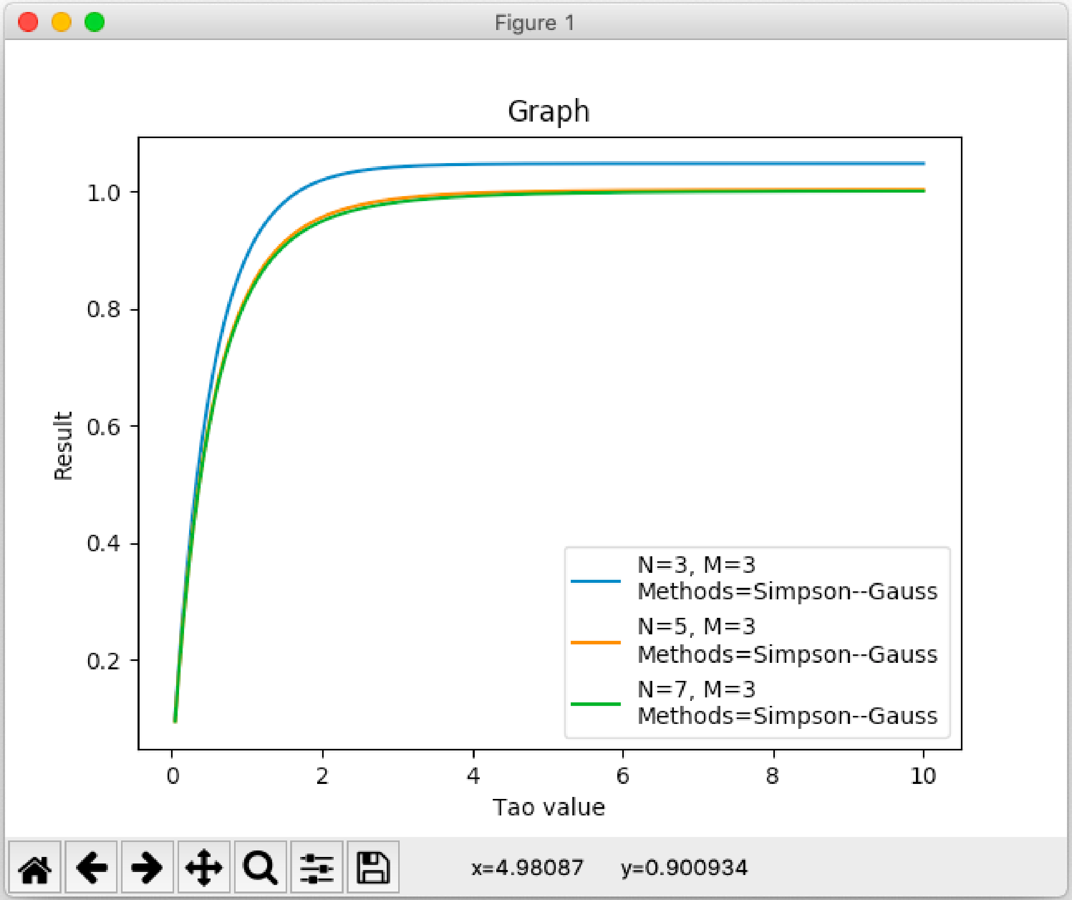


Figure 1
Graph
Result
Tao value
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
10
N=3, M=3
Methods=Simpson--Gauss
N=5, M=3
Methods=Simpson--Gauss
N=7, M=3
Methods=Simpson--Gauss
x=4.98087 y=0.900934

```
 ilyaartemev@Air-Ilya  ~/Desktop/bmstu-ca/lab_05  master ● ?  python3 main.py
N: 3
M: 3
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
1
0
Result: 0.8886948534129957
1 - Continue
0 - End
1
N: 5
M: 3
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
1
0
Result: 0.8208551739111585
1 - Continue
0 - End
1
N: 7
M: 3
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
1
0
Result: 0.8152779841066683
1 - Continue
0 - End
0
```

Видно, что при кол-ве узлов = 3 график отклоняется от графиков с кол-вом узлов 5 и 7. Сравнивая с программой расчета интегралов, можно убедиться, что при большом количестве узлов результат точнее.

Исследование для внутреннего направления и метода Симпсона:

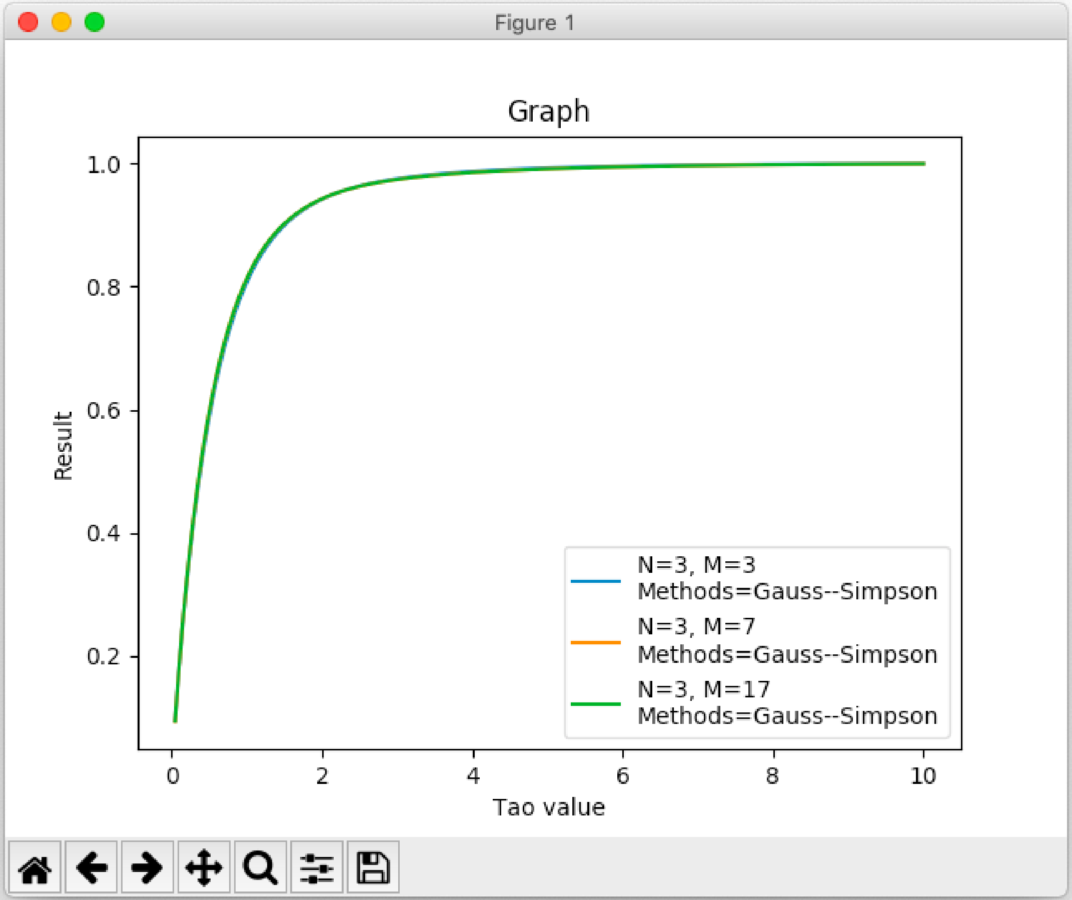
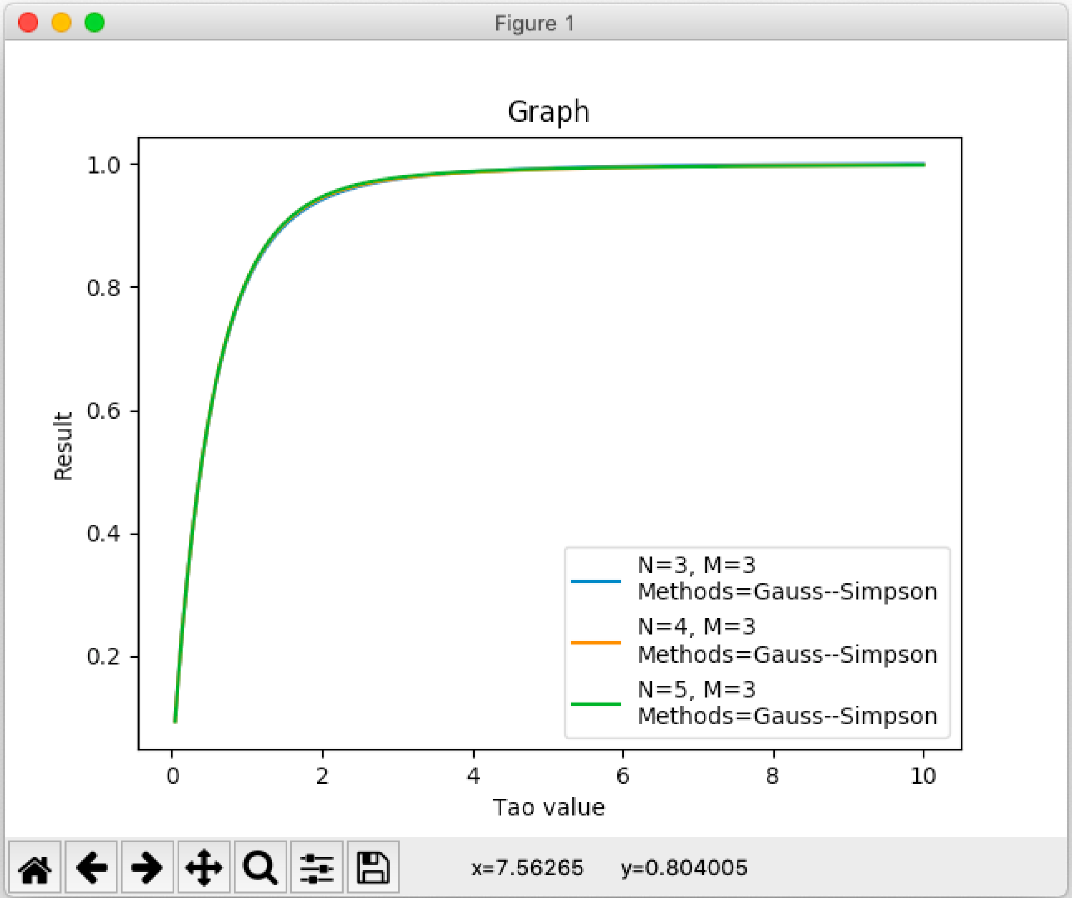


Figure 1
Graph
1.0
0.8
0.6
0.4
0.2
Result
0
2
4
6
8
10
Tao value
N=3, M=3
Methods=Gauss--Simpson
N=3, M=7
Methods=Gauss--Simpson
N=3, M=17
Methods=Gauss--Simpson

```
N: 3
M: 3
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
0
1
Result: 0.8057719561370532
1 - Continue
0 - End
1
N: 3
M: 7
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
0
1
Result: 0.8126077758469114
1 - Continue
0 - End
1
N: 3
M: 17
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
0
1
Result: 0.8122308865577256
1 - Continue
0 - End
0
```

Результаты практически совпадают. Это говорит нам о том, что большее влияние оказывает точность внешнего интегрирования.

Исследования для внешнего направления и метода Гаусса:



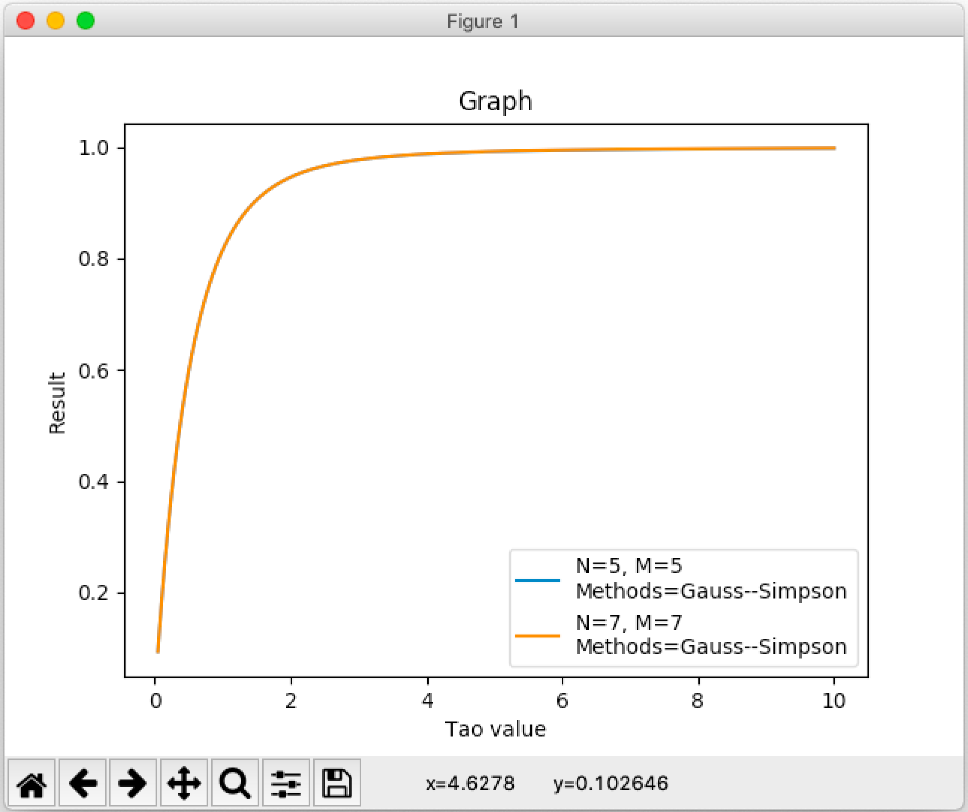
```
N: 3
M: 3
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
0
1
Result: 0.8057719561370532
1 - Continue
0 - End
1
N: 4
M: 3
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
0
1
Result: 0.8091466137324328
1 - Continue
0 - End
1
N: 5
M: 3
Tao: 1
Select a method:
1 - Simpson
0 - Gauss
0
1
Result: 0.8094601626173514
1 - Continue
0 - End
0
```

Результаты близки к действительному даже при малом кол-ве узлов.
Видно, что Гаусс дает более точные результаты, чем Симпсон на внешнем направлении.

Очевидно, что на внутреннем направлении Гаусса большую часть в погрешность будет вносить Симпсон на внешнем направлении.

3. **Построить график зависимости ε(τ) в диапазоне изменения τ =0.05-10. Указать при каком количестве узлов получены результаты.**

Такие графики зависимости есть выше, но здесь можно привести еще один для кол-ва узлов по направлениям (5; 5) и (7; 7).



Код программы

```python
"""
    main.py
"""

from integral_class import Integral


def main():
    exe = Integral()

    exe.fillData()
    exe.drawData()


if __name__ == "__main__":
    main()

"""
    integral_class.py
"""

from numpy import arange
from numpy.polynomial.legendre import leggauss
from math import pi, cos, sin, exp
import matplotlib.pyplot as plt

""" Calculation of the two-fold integral using the quadrature formulas of Gauss and Simpson """

class Integral:

    def __init__(self):
        self.choice = 1

    """ _________________________________METHODS_____________________________________ """


    @staticmethod
    def gauss(func, a, b, numb):
        args, coeffs = leggauss(numb)
        res = 0
```

```
    for i in range(numb):
        res += (b - a) / 2 * coeffs[i] * func((b + a) / 2 + (b - a) * args[i] / 2)

    return res


@staticmethod
def simpson(func, a, b, numb):
    h = (b - a) / (numb - 1)
    x = a
    res = 0

    for _ in range(((numb - 1) // 2):
        res += func(x) + 4 * func(x + h) + func(x + 2 * h)
        x += 2 * h

    return res * (h / 3)


"""__________________________________________________________________"""


@staticmethod
def __getMainFuncLink(param):
    subfunc = lambda x, y: 2 * cos(x) / (1 - (sin(x) ** 2) * (cos(y) ** 2))
    func = lambda x, y: (4 / pi) * (1 - exp(-param * subfunc(x, y))) * cos(x) * sin(x)

    return func


@staticmethod
def __wrapperFunc2(func2, value):
    return lambda x: func2(value, x)


def __integrate(self, func, limits, num_of_nodes, integrators):
    inner = lambda x: integrators[1](self.__wrapperFunc2(func, x), limits[1][0], limits[1][1], num_of_nodes[1])

    return integrators[0](inner, limits[0][0], limits[0][1], num_of_nodes[0])
```

```
@staticmethod
def __taoConstructor(integrate_func, ar_params, label):
    x, y = [], []

    for i in arange(ar_params[0], ar_params[1] + ar_params[2], ar_params[2]):
        x.append(i)
        y.append(integrate_func(i))

    plt.plot(x, y, label=label)


def __getLabel(self, n, m, func1, func2):
    label = "N=" + str(n) + ", M=" + str(m) + "\nMethods="

    if func1 == self.simpson: label += "Simpson"
    else: label += "Gauss"

    if func2 == self.simpson: label += "--Simpson"
    else: label += "--Gauss"

    return label


@staticmethod
def drawData():
    plt.title("Graph")
    plt.legend()
    plt.ylabel("Result")
    plt.xlabel("Tao value")
    plt.show()


def fillData(self):
    self.choice = 1

    while self.choice:
        N = int(input("N: "))
        M = int(input("M: "))
        param = float(input("Tao: "))
```

```
        mode = int(input("Select a method:\n1 - Simpson\n0 - Gauss\n"))
        if mode: func1 = self.simpson
        else: func1 = self.gauss

        mode = int(input())
        if mode: func2 = self.simpson
        else: func2 = self.gauss

        param_integrate = lambda tao: self.__integrate(self.__getMainFuncLink(tao), [[0, pi / 2], [0, pi / 2]], [N, M], [func1, func2])
        print("Result:", param_integrate(param))
        self.__taoConstructor(param_integrate, [0.05, 10, 0.05], self.__getLabel(N, M, func1, func2))

        self.choice = int(input("1 - Continue\n0 - End\n"))
```

Контрольные вопросы

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

   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.

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

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

3. Построить формулу Гаусса численного интегрирования при двух узлах.

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

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

$$\int_{-1}^{1} f(\xi)\, d\xi = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

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

4. Получить обобщенную кубатурную формулу, на основе методе трапеций, с тремя узлами на каждом направлении.

$$\int_c^d \int_a^b f(x,y)\,dx\,dy = \int_a^b dx \int_c^d f(x,y)\,dy = \int_a^b F(x)\,dx =$$

$$= h_x\left(\frac{1}{2}F_0 + F_1 + \frac{1}{2}F_2\right) = h_x h_y \left(\frac{1}{2}\left(\frac{1}{2} f(x_0,y_0) + f(x_0, y_1) + \frac{1}{2} f(x_0,y_2)\right) + \frac{1}{2} f(x_1,y_0) + f(x_1,y_1) + \frac{1}{2} f(x_1, y_2) + \frac{1}{2}\left(\frac{1}{2} f(x_2,y_0) + f(x_2,y_1) + \frac{1}{2} f(x_2,y_2)\right)\right)$$

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