Численное Интегрирование

Фролов А.А, Курылев Г.А., 2к, ИВТ-2

Задача:

Вычислить определенный интеграл, используя различные численные методы и алгоритмы их реализации. Провести сравнительный анализ полученных результатов. Сделать вывод.

Оборудование:

ПК, редактор кода VS Code, язык программирования Python.

Выбранные интегралы:

$$\int_{0,8}^{1,6} \frac{\sqrt{2x+1,6}dx}{1,8+\sqrt{0,3x^2+2,3}}$$

$$\int_{0,8}^{1,6} \frac{x^2+y}{1+x+y^2}$$

Все расчеты производились при n = 10000 (10000 шагов).

Часть 1. Меню.

Меню было реализовано с помощью функций. Файл interface.py содержит функции вывода содержимого каждого уровня меню для каждого выбранного варианта.

interface.py

```python
def tasks():
    print("== Главное меню ==")
    print("1. Численное интегрирование")
    print("2. Дифференциальное уравнение")
    print("3. Элементарная функция")
    print("4. Нелинейное уравнение")
    print("0. Завершить работу")
    print("\n")

def integrationTypes():
    print("== Варианты интеграции ==")
    print("1. Постоянный шаг")
    print("2. Переменный шаг")
    print("3. Кратный интеграл")
    print("0. Назад")
    print("\n")
```

```
17
18
19  def ConstStepMethods():
20      print("== Алгоритмы с постоянным шагом ==")
21      print("1. Метод прямоугольников левых частей")
22      print("2. Метод прямоугольников правых частей")
23      print("3. Метод трапеций")
24      print("4. Метод симпсона")
25      print("5. Все методы вместе (сравнение значений)")
26      print("0. Назад")
27      print("\n")
28
29
30  def VarStepMethods():
31      print("== Алгоритмы с переменным шагом ==")
32      print("1. Алгоритм 1")
33      print("2. Алгоритм 2")
34      print("3. Оба алгоритма (сравнение значений)")
35      print("0. Назад")
36      print("\n")
```

В файле main.py с помощью функций описана сама работа меню. Часть кода можете увидеть ниже:

main.py

```
1   import os
2   from interface import *
3   from Source.integration import *
4
5
6   def clear_console():
7       os.system('cls' if os.name == 'nt' else 'clear')
8
9   def main_menu():
10      while True:
11          clear_console()
12          tasks()
13          choice = input("Выберите пункт меню: ")
14
15          if choice == "1":
16              integration_menu()
17          elif choice == "2":
18              differential_equations_menu()
19          elif choice == "3":
20              elementary_functions_menu()
21          elif choice == "4":
22              nonlinear_equations_menu()
23          elif choice == "0":
```

```
            print("Завершение работы...")
            break
        else:
            print("Неверный ввод. Нажмите Enter чтобы продолжить...")
            input()

def integration_menu():
    while True:
        clear_console()
        integrationTypes()
        choice = input("Выберите тип интегрирования: ")

        if choice == "1":
            const_step_integration_menu()
        elif choice == "2":
            var_step_integration_menu()
        elif choice == "3":
            multiple_integrals_menu()
        elif choice == "0":
            break
        else:
            print("Неверный ввод. Нажмите Enter чтобы продолжить...")
            input()

```

Реализация через функции выбрана потому, что с помощью них можно удобно реализовать переход между уровнями. Меню можно запустить в консоли, это продемонстрировано в скринкасте.

Часть 2. Постоянный шаг.

Для реализации всех методов был создан отдельны файл. Сначала были реализованы функции с постоянным шагом:

integration.py

```
import numpy as np
from Source.functions import f, g

# ПОСТОЯННЫЙ ШАГ

# Метод трапеций
def trapezoid(a, b, n):
    h = (b-a)/n
    r = 0
    for i in np.arange(a + h, b - h + h/2, h):
        r += f(i)
    return h*((f(a)+f(b))/2+r)

```

```python
# Метод прямоугольников левых частей
def left(a, b ,n):
    h = (b-a)/n
    r = 0
    for i in np.arange(a,b-h + h/2,h):
        r += f(i)
    return r*h

# Метод прямоугольников правых частей
def right(a, b, n):
    h = (b-a)/n
    r = 0
    for i in np.arange(a+h,b+h/2,h):
        r += f(i)
    return r*h

# Метод Симпсона (метод парабол)
def simpson(a, b, n):
    s1 = 0
    s2 = 0
    h = (b-a)/n
    for i in np.arange(a+h,b-h+h/2,2*h):
        s1 += f(i)
    for i in np.arange(a+2*h,b-2*h+h/2,2*h):
        s2 += f(i)
    return (h/3)*(f(a)+f(b)+s1*4+s2*2)
```

Результаты вычислений:

Метод	Результат вычислений
Прямоугольник Лев.	0.46176530
Прямоугольник Прав.	0.46177229
Трапеций	0.46176880
Симпсон	0.46176880

Вывод:

На основе представленных результатов вычислений все методы численного интегрирования показали очень близкие значения с расхождением не более 10^{-5}, что подтверждает корректность вычислений. Наиболее точным считается метод Симпсона, который в данном случае дал идентичный результат с методом трапеций - 0.46176880. Методы правых и левых прямоугольников показали минимальное расхождение около 7×10^{-6} между собой. Таким образом, можно заключить, что значение интеграла равно примерно 0.46177 с высокой степенью

точности, а метод Симпсона подтвердил свою эффективность для данного вычисления.

Часть 3. Переменный шаг.

Далее были реализованы функции для вычисления интеграла с переменным шагом:

integration.py

```python
# ПЕРЕМЕННЫЙ ШАГ

#Алгоритм №1
def algorithm1(a, b, n, eps=1e-6):

    """
    a,b — пределы интегрирования
    eps — требуемая точность ε
    n   — начальное число разбиений
    """

    h = (b - a) / n
    In = 0.0
    I2n = 0.0
    R = float('inf')  # начальная разность

    while R > eps:
        # вычисляем интеграл I2n методом левых прямоугольников
        S2 = 0.0
        x = a
        while x ⩽ b - h:
            S2 += f(x)
            x += h
        I2n = h * S2

        # разность между текущим и предыдущим приближением
        R = abs(I2n - In)
        In = I2n  # обновляем

        # уменьшаем шаг вдвое
        h /= 2

    return I2n

#Алгоритм №2
def algorithm2(a, b, n=10, eps=1e-6):

    """
    a,b — пределы интегрирования
```

```
    eps — требуемая точность ε
    n   — начальное число разбиений
    """

    hv = (b - a) / n  # шаг вычисления
    S1 = sum(f(a + i * hv) for i in range(n))
    I1 = S1 * hv

    while True:
        hv /= 2                 # уменьшаем шаг вычисления
        hs = hv                 # шаг смещения
        hd = hv * 2             # шаг движения — как в описании (равен старому hv)

        x = a + hs              # смещение от начала
        S2 = 0.0
        while x < b:
            S2 += f(x)
            x += hd             # двигаемся старым шагом (не hv!)

        # Уточняем интеграл, добавляя новые точки между старыми
        I2 = (I1 / 2) + S2 * hv

        if abs(I2 - I1) < eps:
            return I2

        I1 = I2
```

Результаты вычислений:

Метод	Результат вычислений
Алгоритм 1	0.46176874
Алгоритм 2	0.46176792

Вывод:

Оба алгоритма с переменным шагом показали близкие результаты: 0.46176874 и 0.46176792 соответственно. Разница между значениями составляет 8.2×10^{-7}, что подтверждает корректность реализации методов. Первый алгоритм использует последовательное уменьшение шага с контролем точности через разность приближений, второй применяет схему уточнения интеграла добавлением промежуточных точек. Оба подхода демонстрируют эффективность для численного интегрирования с автоматическим выбором шага.

Часть 4. Кратный интеграл.

Далее была реализована функция для вычисления кратного интеграла:

integration.py

```python
#Кратный интеграл
def double_integral(a1, b1, a2, b2, nx, ny):

    hx = (b1 - a1) / nx
    hy = (b2 - a2) / ny
    s = 0.0

    for i in range(nx):
        x = a1 + i * hx
        for j in range(ny):
            y = a2 + j * hy
            s += g(x, y)

    return s * hx * hy
```

Результаты вычислений:

Метод	Результат вычислений
Кратный интеграл	0.46615090

Вывод:

Реализован алгоритм вычисления кратного интеграла методом прямоугольников. Программа использует двойной цикл для прохода по узлам сетки в областях интегрирования. Полученное значение интеграла составляет 0.46615090. Код корректно выполняет численное интегрирование для функции двух переменных.

Вывод

В ходе выполнения работы были успешно реализованы и протестированы различные методы численного интегрирования. Все алгоритмы показали высокую точность вычислений и корректность реализации.

Для методов с постоянным шагом наивысшую точность продемонстрировали метод Симпсона и метод трапеций, давшие идентичные результаты. Методы прямоугольников показали минимальное расхождение, что подтверждает правильность выбранного подхода.

Алгоритмы с переменным шагом доказали свою эффективность для автоматического достижения заданной точности, при этом оба разработанных алгоритма показали близкие результаты с минимальным расхождением.

Реализация кратного интеграла методом прямоугольников для функции двух переменных также показала работоспособность и может быть использована для решения более сложных многомерных задач.

Структура программы с модульным меню позволяет удобно тестировать различные методы и сравнивать их результаты, что делает программную реализацию удобной для дальнейшего расширения и использования в учебных целях.

Все поставленные задачи выполнены в полном объеме, что подтверждает освоение методов численного интегрирования и их программной реализации на языке Python.