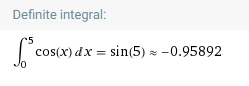
Лабораторная работа №1

**Тема:** Численное интегрирование

**Язык:** Python 3.7.4

**Постановка задачи:** Составить программу, которая реализует методы численного интегрирования с постоянным и переменным шагом для табличных интегралов

**Табличный интеграл:**



**Переменные:**

Модуль main.py

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Значение** | **Тип** |
| main\_menu | Содержит значения пунктов главного меню | dict |
| menu\_constant | Содержит значения пунктов меню для методов решения интегралов с постоянным шагом | dict |
| menu\_var | Содержит значения пунктов меню для методов решения интегралов с переменным шагом | dict |
| cond\_main | Показывает какой пункт в главном меню выбрал пользователь | int |
| cond\_const | Показывает какой пункт в меню для постоянного шага выбрал пользователь | int |
| cond\_var | Показывает какой пункт в меню для переменного шага выбрал пользователь | int |
| arguments | Аргументы, необходимые для вычисления интегралов и составления таблицы | list |
| cond\_checker | Выводит меню и возвращает введенное  пользователем значение. | function |

Модуль integral\_table.py

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Значение** | **Тип** |
| table | Строит таблицу с помощью внешнего модуля prettytable | function |
| left\_parts | Содержит результат вычисления интеграла методом левых частей прямоугольника | list |
| right\_parts | Содержит результат вычисления интеграла методом правых частей прямоугольника | list |
| trap | Содержит результат вычисления интеграла методом трапеции | list |
| par | Содержит результат вычисления интеграла методом Симпсона | list |
| first | Содержит результат вычисления интеграла с помощью первого алгоритма с переменным шагом | list |
| sec | Содержит результат вычисления интеграла с помощью второго алгоритма с переменным шагом | list |
| t | Таблица | Класс PrettyTable |
| column\_names | Содержит названия столбцов | list |

Модуль formulas.py

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Значение** | **Тип** |
| func | Подынтегральная функция | function |
| rectangle\_right | Функция для вычисления интеграла методом правых частей прямоугольника | function |
| rectangle\_left | Функция для вычисления интеграла методом левых частей прямоугольника | function |
| trapeze | Функция для вычисления интеграла методом трапеции | function |
| parabola | Функция для вычисления интеграла методом Симпсона | function |
| n\_from\_h | Вспомогательная функция для получения n из h.  Необходима для более удобного подсчета интеграла методами с переменным шагом. | function |
| double\_recount | Функция для вычисления интеграла по первому алгоритму с переменным шагом | function |
| second\_alogorithm | Функция для вычисления интеграла по первому алгоритму с переменным шагом | function |

Модуль rcounting.py

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Значение** | **Тип** |
| diff | Первая производная от подынтегральной функции | function |
| second\_diff | Вторая производная от подынтегральной функции | function |
| fourth\_diff | Четвертая производная от подынтегральной функции | function |
| max\_diff | Находит максимальное значение производной в пределах интегрирования | function |
| r\_rectangle | Остаточный член для метода правых и левых частей прямоугльника | function |
| r\_trapeze | Остаточный член для метода трапеции | function |
| r\_parabola | Остаточный член для метода Симпсона | function |

Модуль entering.py

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Значение** | **Тип** |
| input\_const | Функция для форматированного ввода параметров,  необходимых при вычислении интеграла методами  с постоянным шагом | function |
| input\_const | Функция для форматированного ввода параметров,  необходимых при вычислении интеграла методами  с переменным шагом | function |
| input\_table | Функция для форматированного ввода параметров,  необходимых для составления таблицы | function |
| input\_double | Функция для форматированного ввода параметров,  необходимых для вычисления двойного интеграла | function |

Модуль double\_integral

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Значение** | **Тип** |
| rectangular\_double | Функция для вычисления двойного интеграла | function |

**Код программы:**

Модуль main.py

from integral\_table import \*  
from entering import \*  
from double\_integral import \*  
  
  
def cond\_checker(menu: dict) -> int:  
 *"""  
 На вход поступает меню в виде словаря.  
 Функция выводит меню и возвращает введенное  
 пользователем значение.  
 """* for i in sorted(menu.keys()):  
 print(i)  
 cond = int(input())  
 return cond  
  
  
def main():  
 cond\_main = 0  
 while cond\_main != main\_menu['5) Выход']:  
 cond\_main = cond\_checker(main\_menu)  
 arguments = list()  
 if cond\_main == main\_menu['1) Методы с постоянным шагом']:  
 cond\_const = 0  
 while cond\_const != menu\_constant['5) Выход']:  
 cond\_const = cond\_checker(menu\_constant)  
 if (cond\_const > 0) & (cond\_const < 5):  
 arguments = input\_const()  
 if cond\_const == 1:  
 print('\nРезультат вычисления: ', rectangle\_left(arguments), ' Остаточный член:',  
 r\_rectangle(arguments), '\n')  
 elif cond\_const == 2:  
 print('\nРезультат вычисления: ', rectangle\_right(arguments), ' Остаточный член:',  
 r\_rectangle(arguments), '\n')  
 elif cond\_const == 3:  
 print('\nРезультат вычисления: ', trapeze(arguments), ' Остаточный член:',  
 r\_trapeze(arguments), '\n')  
 elif cond\_const == 4:  
 print('\nРезультат вычисления: ', parabola(arguments), ' Остаточный член:',  
 r\_parabola(arguments), '\n')  
 elif cond\_const == 5:  
 pass  
 else:  
 print('Ошибка, введите другое значение')  
 elif cond\_main == main\_menu['2) Методы с переменным шагом']:  
 cond\_var = 0  
 while cond\_var != menu\_var['3) Выход']:  
 cond\_var = cond\_checker(menu\_var)  
 if (cond\_var > 0) & (cond\_var < 3):  
 arguments = input\_var()  
 if cond\_var == 1:  
 print('\nРезультат вычисления: ', double\_recount(arguments), '\n')  
 elif cond\_var == 2:  
 print('\nРезультат вычисления: ', second\_algorithm(arguments), '\n')  
 elif cond\_var == 3:  
 pass  
 else:  
 pass  
 elif cond\_main == main\_menu['3) Таблица']:  
 arguments = input\_table()  
 print('\n', table(arguments), '\n')  
 elif cond\_main == main\_menu['4) Двойной интеграл']:  
 args = input\_double()  
 print('\nРезультат вычисления:', rectangular\_double(args), '\n')  
 elif cond\_main == main\_menu['5) Выход']:  
 pass  
 else:  
 print('Ошибка, введите другое значение')  
  
  
main\_menu = {'1) Методы с постоянным шагом': 1,  
 '2) Методы с переменным шагом': 2,  
 '3) Таблица': 3,  
 '4) Двойной интеграл': 4,  
 '5) Выход': 5}  
menu\_constant = {'1) Метод правых частей прямоугольников': 1,  
 '2) Метод левых частей прямоугольников': 2,  
 '3) Метод трапеций': 3,  
 '4) Метод парабол': 4,  
 '5) Выход': 5}  
menu\_var = {'1) Метод по первому алгоритму': 1,  
 '2) Метод по второму алгоритму': 2,  
 '3) Выход': 3}  
  
main()

Модуль integral\_table.py

from prettytable import PrettyTable  
from formulas import \*  
from rcounting import \*  
  
  
def table(args):  
 *"""  
 Выводит таблицу с результатом подсчета интеграла всеми методами.  
 """* left\_parts = [str(round(rectangle\_left([args[0], args[1], args[2]]), 5)) + ', R = ' + str(round(r\_rectangle(  
 [args[0], args[1], args[2]]), 5))]  
 right\_parts = [str(round(rectangle\_right([args[0], args[1], args[2]]), 5)) + ', R = ' + str(round(r\_rectangle(  
 [args[0], args[1], args[2]]), 5))]  
 trap = [str(round(trapeze([args[0], args[1], args[2]]), 5)) +  
 ', R = ' + str(round(r\_trapeze([args[0], args[1], args[2]]), 8))]  
 par = [str(round(parabola([args[0], args[1], args[2]]), 5)) +  
 ', R = ' + str(round(r\_parabola([args[0], args[1], args[2]]), 8))]  
 first = [str(round(double\_recount([args[0], args[1], args[3]]), 5))]  
 sec = [str(round(second\_algorithm([args[0], args[1], args[3]]), 5))]  
 t = PrettyTable()  
 column\_names = ['Прямоугольник левых частей ', 'Прямоугольник правых частей', 'Трапеция', 'Парабола',  
 'Первый алгоритм', 'Второй алгоритм']  
 t.add\_column(column\_names[0], left\_parts)  
 t.add\_column(column\_names[1], right\_parts)  
 t.add\_column(column\_names[2], trap)  
 t.add\_column(column\_names[3], par)  
 t.add\_column(column\_names[4], first)  
 t.add\_column(column\_names[5], sec)  
 return t

Модуль formulas.py

*"""  
В модуле содержатся функции для подсчета интегралов  
"""*import math  
  
  
def func(x):  
 return math.cos(x)  
  
  
def rectangle\_right(args: list) -> float:  
 *"""Интегралы методом правых частей прямоугольника.  
 Args - это a, b и количество разбиений"""* h = (args[1]-args[0])/args[2]  
 i = args[0]+h  
 res = 0  
 while i <= args[1]:  
 res += h \* func(i)  
 i += h  
 return res  
  
  
def rectangle\_left(args: list) -> float:  
 *"""args - это a, b и количество разбиений"""* h = (args[1]-args[0])/args[2]  
 i = args[0]  
 res = 0  
 while i <= args[1]-h:  
 res += h \* func(i)  
 i += h  
 return res  
  
  
def trapeze(args: list) -> float:  
 *"""args - это a, b и количество разбиений"""* h = (args[1]-args[0])/args[2]  
 i = args[0] + h  
 res = 0  
 while i <= args[1]-h:  
 res += func(i)  
 i += h  
 res += (func(args[0]) + func(args[1]))/2  
 res \*= h  
 return res  
  
  
def parabola(args: list) -> float:  
 *"""args - это a, b и количество разбиений"""* h = (args[1] - args[0]) / args[2]  
 i = args[0] + h  
 odd = 0  
 even = 0  
 while i <= args[1] - h:  
 odd += func(i)  
 i += 2 \* h  
 i = args[0] + 2 \* h  
 while i <= args[1] - 2 \* h:  
 even += func(i)  
 i += 2 \* h  
 res = h / 3 \* (func(args[0]) + func(args[1]) + 4 \* odd + 2 \* even)  
 return res  
  
  
def n\_from\_h(args):  
 *"""Вспомогательная функция для получения n из h.  
 Необходима для более удобного подсчета интеграла.  
 Аргументы: a, b, h"""* return (args[1] - args[0])/args[2]  
  
  
def double\_recount(args\_var):  
 *"""args\_var - это a, b и точность"""* h = args\_var[2]  
 a = args\_var[0]  
 b = args\_var[1]  
 res1 = trapeze([a, b, n\_from\_h([a, b, h])])  
 res2 = trapeze([a, b, n\_from\_h([a, b, h/2])])  
 while abs(res2-res1) > args\_var[2]:  
 h /= 2  
 res1 = trapeze([a, b, n\_from\_h([a, b, h])])  
 res2 = trapeze([a, b, n\_from\_h([a, b, h])])  
 return res2  
  
  
def second\_algorithm(args\_var):  
 h\_v = args\_var[2]  
 h\_s = h\_v/2  
 a = args\_var[0]  
 b = args\_var[1]  
 res1 = rectangle\_right([a, b, n\_from\_h([a, b, h\_v])])  
 res2 = 0  
 while abs(res2 - res1) > args\_var[2]:  
 res1 = rectangle\_right([a, b, n\_from\_h([a, b, h\_v])])  
 h\_v += h\_s  
 h\_d = h\_v/2  
 res2 = rectangle\_right([a, b, n\_from\_h([a, b, h\_d])])  
 h\_v /= 2  
 h\_s /= 2  
 return res2

Модуль rcounting.py

*"""  
В модуле содержатся функции для подсчета остаточных членов  
"""*import math  
  
  
def diff(x):  
 return -1 \* math.sin(x)  
  
  
def second\_diff(x):  
 return -1 \* math.cos(x)  
  
  
def fourth\_diff(x):  
 return math.cos(x)  
  
  
def max\_diff(args, func):  
 i = args[0]  
 res = []  
 while args[1] >= i:  
 res.append(func(i))  
 i += 0.001  
 return abs(max(res))  
  
  
def r\_rectangle(args):  
 return max\_diff(args, diff)\*(args[1] - args[0])\*\*2 / (2 \* args[2])  
  
  
def r\_trapeze(args):  
 return max\_diff(args, second\_diff)\*(args[1] - args[0])\*\*3 / (12 \* args[2]\*\*2)  
  
  
def r\_parabola(args):  
 return max\_diff(args, fourth\_diff)\*(args[1] - args[0])\*\*5 / (180 \* (2 \* args[2])\*\*2)

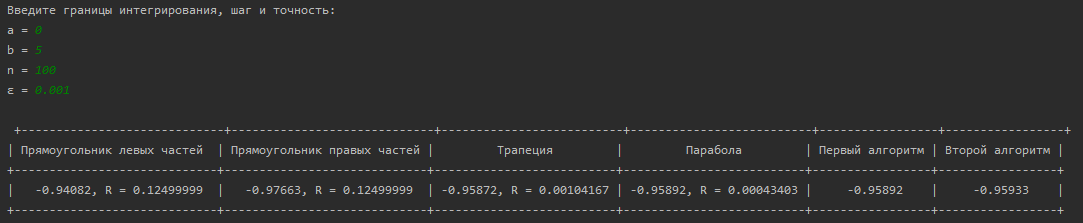
Модуль entering.py

def input\_const():  
 *"""  
 Функция для форматированного ввода параметров,  
 необходимых при вычислении интеграла методами  
 с постоянным шагом  
 """* print('Введите границы интегрирования и шаг:\na = ', end='')  
 a = float(input())  
 print('b = ', end='')  
 b = float(input())  
 print('n = ', end='')  
 n = float(input())  
 res = [a, b, n]  
 return res  
  
  
def input\_var():  
 *"""  
 Функция для форматированного ввода параметров,  
 необходимых при вычислении интеграла методами  
 с переменным шагом  
 """* print('Введите границы интегрирования и точность:\na = ', end='')  
 a = float(input())  
 print('b = ', end='')  
 b = float(input())  
 print('ε = ', end='')  
 e = float(input())  
 res = [a, b, e]  
 return res  
  
  
def input\_table():  
 *"""  
 Функция для форматированного ввода параметров,  
 необходимых для составления таблицы  
 """* print('Введите границы интегрирования, шаг и точность:\na = ', end='')  
 a = float(input())  
 print('b = ', end='')  
 b = float(input())  
 print('n = ', end='')  
 n = float(input())  
 print('ε = ', end='')  
 e = float(input())  
 res = [a, b, n, e]  
 return res  
  
  
def input\_double():  
 *"""  
 Функция для форматированного ввода параметров,  
 необходимых для вычисления двойного интеграла  
 """* res = [i\*0 for i in range(6)]  
 print('Введите границы интегрирования и шаг для первого интеграла:\na = ', end='')  
 res[0] = float(input())  
 print('b = ', end='')  
 res[1] = float(input())  
 print('n1 = ', end='')  
 res[4] = int(input())  
 print('Введите границы интегрирования и шаг для второго интеграла:\nc = ', end='')  
 res[2] = float(input())  
 print('d = ', end='')  
 res[3] = float(input())  
 print('n2 = ', end='')  
 res[5] = int(input())  
 return res

Модуль double\_integral.py

*"""  
Модуль необходимый для вычисления двойного интеграла.  
"""*def rectangular\_double(args):  
 f = lambda x, y: x \* 2 + y  
 a = args[0]  
 b = args[1]  
 c = args[2]  
 d = args[3]  
 nx = args[4]  
 ny = args[5]  
 hx = (b - a) / float(nx)  
 hy = (d - c) / float(ny)  
 res = 0  
 for i in range(nx):  
 for j in range(ny):  
 xi = a + hx/2 + i\*hx  
 yj = c + hy/2 + j\*hy  
 res += hx\*hy\*f(xi, yj)  
 return res

**Полученные значения:**



**Не табличный интеграл:**

