МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

# ОТЧЁТ

по лабораторной работе №3

" ИНТЕРПОЛИРОВАНИЕ ФУНКЦИИ

МНОГОЧЛЕНОМ НЬЮТОНА И МНОГОЧЛЕНОМ ЛАГРАНЖА"

по дисциплине

*Вычислительная математика*

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Панкратова А.З.\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Халеев А.А. \_

(подпись) (фамилия, и.,о.)

\_\_\_\_\_\_\_21-ВМз-4\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2023

**Тема работы:**

Интерполирование функции многочленом Ньютона и многочленом Лагранжа

**Цель работы**:

Изучить численные методы и алгоритмы решения задачи интерполяции функций

**Постановка задачи:**

Реализовать изученные алгоритмы интерполяции и провести сравнение методов

**Вариант №7:**

1. Найти приближенное значение функции при данных значениях аргумента с помощью интерполяционного многочлена Лагранжа, если функция задана в неравноотстоящих узлах таблицы, оценить погрешность

|  |  |  |  |
| --- | --- | --- | --- |
| x | y |  |  |
| 0.43 | 1.63597 | 0.702 | 0.503 |
| 0.48 | 1.73234 | 0.512 | 0.441 |
| 0.55 | 1.87686 | 0.645 | 0.602 |
| 0.62 | 2.03345 | 0.736 | 0.732 |
| 0.70 | 2.22846 |  |  |
| 0.75 | 2.35973 |  |  |

1. Используя первую или вторую интерполяционную формулу Ньютона, вычислить значения функции при данных значениях аргумента

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | Значение аргумента | | | |
| 1.340 | 4.25562 | x1 | x2 | x3 | x4 |
| 1.345 | 4.35325 | 1.3617 | 1.3921 | 1.3359 | 1.400 |
| 1.350 | 4.45522 | 1.3463 | 1.3868 | 1.335 | 1.3990 |
| 1.355 | 4.56184 | 1.3432 | 1.3936 | 1.3365 | 1.3975 |
| 1.360 | 4.67344 |
| 1.365 | 4.79038 |
| 1.370 | 4.91306 |
| 1.375 | 5.04192 |
| 1.380 | 5.17744 |
| 1.385 | 5.32016 |
| 1.390 | 5.47069 |
| 1.395 | 5.62968 |

***Теория:***

***Метод центральных прямоугольников***

Метод непосредственно использует замену определенного интеграла интегральной суммой. В качестве точек может выбираться любая точка в промежутке . В зависимости от выбора этой точки различают методы левых, правых и центральных прямоугольников.

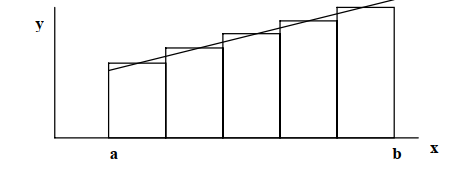
Середина интервала, метод центральных прямоугольников:

Обычно, когда рассматривают метод прямоугольников, разбивают на n равных отрезков:

Заметим, что в пределах одного шага подынтегральная функция заменяется  
(аппроксимация) отрезком горизонтальной прямой т.е. первым членом полинома

Коэффициент ищется из условия прохождения кривой через точки .  
Широко распространенным и более точным является вид формулы  
прямоугольников, использующий значения функции в средних точках элементарных  
отрезков, т.е.

В этом случае формула прямоугольника имеет вид:



***Исходные данные:***

***n1 = 8; n2 = 20***

***Решение в Excel:***

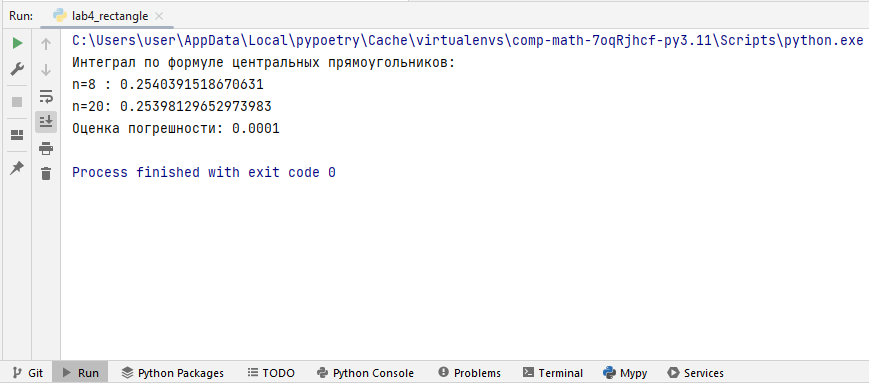
|  |  |  |  |
| --- | --- | --- | --- |
| Вычисление интеграла методом центральных прямоугольников | | | |
| Нижний предел | Верхний предел | n | h |
| 0,80 | 1,60 | 8,00 | 0,10 |
| k | x | Xi-1/2 | Fk(x) |
| 1 | 0,90 | 0,85 | 0,027783439 |
| 2 | 1,00 | 0,95 | 0,029402596 |
| 3 | 1,10 | 1,05 | 0,030736762 |
| 4 | 1,20 | 1,15 | 0,031822237 |
| 5 | 1,30 | 1,25 | 0,03269311 |
| 6 | 1,40 | 1,35 | 0,033380293 |
| 7 | 1,50 | 1,45 | 0,033911158 |
| 8 | 1,60 | 1,55 | 0,034309557 |
|  |  | |= | 0,254039152 |

|  |  |  |  |
| --- | --- | --- | --- |
| Вычисление интеграла методом центральных прямоугольников | | | |
| Нижний предел | Верхний предел | n | h |
| 0,80 | 1,60 | 20,00 | 0,04 |
| k | x | Xi-1/2 | Fk(x) |
| 1 | 0,84 | 0,82 | 0,010894642 |
| 2 | 0,88 | 0,86 | 0,011183693 |
| 3 | 0,92 | 0,9 | 0,011452381 |
| 4 | 0,96 | 0,94 | 0,011701646 |
| 5 | 1,00 | 0,98 | 0,011932436 |
| 6 | 1,04 | 1,02 | 0,012145699 |
| 7 | 1,08 | 1,06 | 0,012342367 |
| 8 | 1,12 | 1,1 | 0,012523355 |
| 9 | 1,16 | 1,14 | 0,012689554 |
| 10 | 1,20 | 1,18 | 0,012841824 |
| 11 | 1,24 | 1,22 | 0,01298099 |
| 12 | 1,28 | 1,26 | 0,013107846 |
| 13 | 1,32 | 1,3 | 0,013223147 |
| 14 | 1,36 | 1,34 | 0,013327613 |
| 15 | 1,40 | 1,38 | 0,013421925 |
| 16 | 1,44 | 1,42 | 0,01350673 |
| 17 | 1,48 | 1,46 | 0,013582638 |
| 18 | 1,52 | 1,5 | 0,013650223 |
| 19 | 1,56 | 1,54 | 0,013710027 |
| 20 | 1,60 | 1,58 | 0,01376256 |
|  |  | |= | 0,253981297 |

***Программа на Python:***

import math  
  
  
def func(x: float) -> float:  
 return math.log10(x \*\* 2 + 1) / x *# подынтегральная функция*def rectangular\_rule(a: float, b: float, n: int) -> float:  
 *"""  
 Вычисляет определенный интеграл функции с использованием метода центральных прямоугольников.  
  
 Аргументы:  
 a (float): Нижний предел интегрирования.  
 b (float): Верхний предел интегрирования.  
 n (int): Количество интервалов, на которые разбивается область интегрирования.  
  
 Возвращает:  
 float: Приближенное значение определенного интеграла.  
 """* h = (b - a) / n *# шаг* integral = 0 *# аккумулятор значения интеграла* for i in range(n):  
 x = a + h \* (i + 0.5) *# вычисление значения Xi-1/2 с учетом 0-индексированного цикла* integral += h \* func(x) *# аккумуляция значения интеграла* return integral  
  
  
def main() -> None:  
 *# Вычисление интегралов* integral\_rectangular\_8 = rectangular\_rule(a=0.8, b=1.6, n=8)  
 integral\_rectangular\_20 = rectangular\_rule(a=0.8, b=1.6, n=20)  
 error\_rectangular = abs(integral\_rectangular\_20 - integral\_rectangular\_8)  
 print("Интеграл по формуле центральных прямоугольников:")  
 print(f"n=8 : {integral\_rectangular\_8}\nn=20: {integral\_rectangular\_20}")  
 print(f"Оценка погрешности: {round(error\_rectangular, 4)}")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Вывод программы:

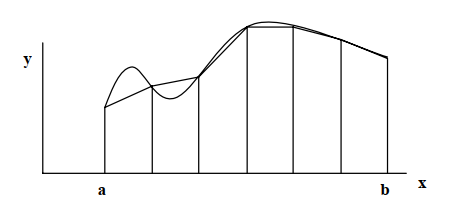


***Метод трапеций***

Отличается от метода прямоугольников способом аппроксимации отрезка Δ*xi* . В методе прямоугольников аппроксимация осуществлялась отрезком горизонтальной прямой, а в методе трапеций – прямой общего вида, т.е. полиномом 1-й степени:

Коэффициенты вычисляются из условия прохождения этой прямой через две точки (значения подынтегральной функции на краях интервала Δ*xi* ).

График функции *y* = *f* (*x*) представляется в виде ломаной, соединяющей точки(*xi* , *yi* ):



Площадь всей фигуры складывается из элементарных прямоугольных трапеций.  
Площадь каждой элементарной трапеции может быть записана в виде:

*Площадь всей фигуры:*

Если (интегрирование с постоянным шагом), то формула трапеций принимает вид:

*Исходные данные:*

***n1 = 8; n2 = 20***

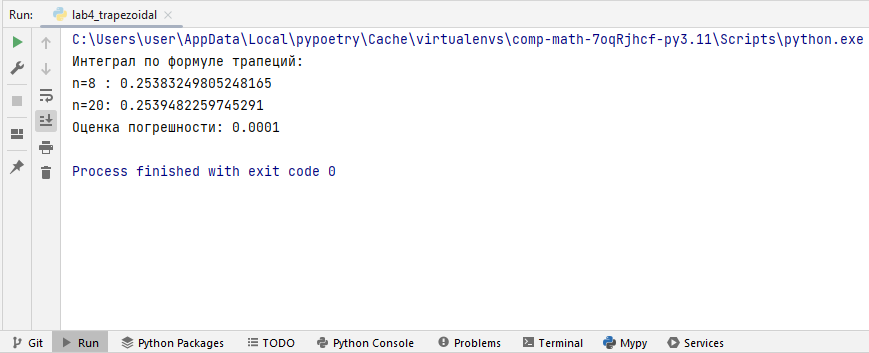
***Решение в Excel:***

|  |  |  |  |
| --- | --- | --- | --- |
| Вычисление интеграла методом трапеций | | | |
| Нижний предел | Верхний предел | n | h |
| 0,80 | 1,60 | 8,00 | 0,10 |
| k | x | Yk(x) |  |
| 0 | 0,80 | 0,26855481 |  |
| 1 | 0,90 | 0,286309528 |  |
| 2 | 1,00 | 0,301029996 |  |
| 3 | 1,10 | 0,313083885 |  |
| 4 | 1,20 | 0,322824855 |  |
| 5 | 1,30 | 0,330578677 |  |
| 6 | 1,40 | 0,336636936 |  |
| 7 | 1,50 | 0,341255574 |  |
| 8 | 1,60 | 0,344656249 |  |
|  |  | |= | 0,253832498 |
|  |  |  |  |
|  |  |  |  |
| Вычисление интеграла методом трапеций | | | |
| Нижний предел | Верхний предел | n | h |
| 0,80 | 1,60 | 20,00 | 0,04 |
| k | x | Yk(x) |  |
| 0 | 0,80 | 0,26855481 |  |
| 1 | 0,84 | 0,276044271 |  |
| 2 | 0,88 | 0,283013101 |  |
| 3 | 0,92 | 0,289484556 |  |
| 4 | 0,96 | 0,295482281 |  |
| 5 | 1,00 | 0,301029996 |  |
| 6 | 1,04 | 0,30615123 |  |
| 7 | 1,08 | 0,310869118 |  |
| 8 | 1,12 | 0,315206228 |  |
| 9 | 1,16 | 0,319184442 |  |
| 10 | 1,20 | 0,322824855 |  |
| 11 | 1,24 | 0,326147714 |  |
| 12 | 1,28 | 0,329172374 |  |
| 13 | 1,32 | 0,331917273 |  |
| 14 | 1,36 | 0,334399928 |  |
| 15 | 1,40 | 0,336636936 |  |
| 16 | 1,44 | 0,338643991 |  |
| 17 | 1,48 | 0,340435903 |  |
| 18 | 1,52 | 0,342026627 |  |
| 19 | 1,56 | 0,343429295 |  |
| 20 | 1,60 | 0,344656249 |  |
|  |  | |= | 0,253948226 |

***Программа на Python:***

import math  
  
  
def func(x: float) -> float:  
 return math.log10(x \*\* 2 + 1) / x *# подынтегральная функция*def trapezoidal\_rule(a: float, b: float, n: int) -> float:  
 *"""  
 Вычисляет определенный интеграл функции с использованием метода трапеций.  
  
 Аргументы:  
 a (float): Нижний предел интегрирования.  
 b (float): Верхний предел интегрирования.  
 n (int): Количество интервалов, на которые разбивается область интегрирования.  
  
 Возвращает:  
 float: Приближенное значение определенного интеграла.  
 """* h = (b - a) / n *# шаг* integral = 0 *# аккумулятор значения интеграла* for i in range(n + 1):  
 x = a + i \* h *# вычисление значения x на основании шага и индекса текущей итерации* if i == 0 or i == n:  
 integral += func(x) / 2 *# первое и последнее значение делятся на 2* else:  
 integral += func(x) *# остальные значения прибавляются без изменений* integral \*= h *# умножение на шаг* return integral  
  
  
def main() -> None:  
 *# Вычисление интегралов* integral\_trapezoidal\_8 = trapezoidal\_rule(a=0.8, b=1.6, n=8)  
 integral\_trapezoidal\_20 = trapezoidal\_rule(a=0.8, b=1.6, n=20)  
 error\_rectangular = abs(integral\_trapezoidal\_20 - integral\_trapezoidal\_8)  
 print("Интеграл по формуле трапеций:")  
 print(f"n=8 : {integral\_trapezoidal\_8}\nn=20: {integral\_trapezoidal\_20}")  
 print(f"Оценка погрешности: {round(error\_rectangular, 4)}")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Вывод программы:



***Метод Симпсона***

Разобьем отрезок [*a*, *b*]на четное число *n* равных частей с шагом *h* на каждом  
отрезке.

Возьмем отрезки, равные двум шагам: [*x*0, *x*2], [*x*2 ,*x*4 ], ..., [*xi*-1, *xi*+1], ... ,[*xn*-2 , *xn*] . На  
каждом из них подынтегральную функцию заменим интерполяционным полиномом  
второй степени:

Коэффициенты, , могут быть найдены из условий равенства многочлена в  
точках соответствующим табличным значениям В качестве можно взять интерполяционный многочлен Лагранжа, проходящий через точки

Элементарная функция - площадь криволинейной трапеции (рис. 8) вычисляется с помощью определенного интеграла, подынтегральной функцией которого является многочлен Лагранжа:

Данное выражение принимается в качестве значения определенного интеграла:

***Это формула Симпсона***

*Исходные данные:*

***n1 = 8; n2 = 20***

***Решение в Excel:***

*При n=8:*

|  |  |  |  |
| --- | --- | --- | --- |
| Вычисление интеграла методом Симпсона | | | |
| Нижний предел | Верхний предел | n | h |
| 0,80 | 1,60 | 8,00 | 0,10 |
| k | x | Yk(x) |  |
| 0 | 0,80 | 0,26855481 |  |
| 1 | 0,90 | 0,286309528 |  |
| 2 | 1,00 | 0,301029996 |  |
| 3 | 1,10 | 0,313083885 |  |
| 4 | 1,20 | 0,322824855 |  |
| 5 | 1,30 | 0,330578677 |  |
| 6 | 1,40 | 0,336636936 |  |
| 7 | 1,50 | 0,341255574 |  |
| 8 | 1,60 | 0,344656249 |  |
|  |  | |= | 0,253970176 |

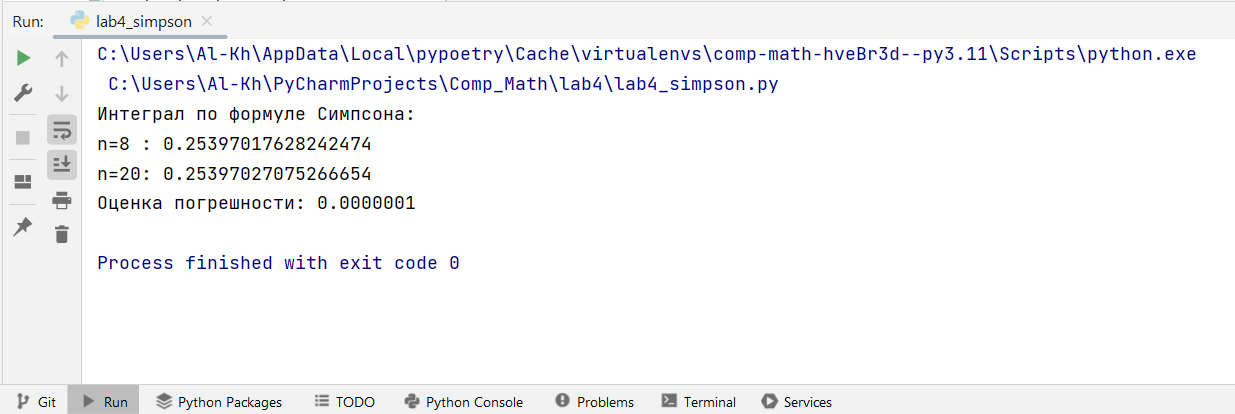
*При n=20:*

|  |  |  |  |
| --- | --- | --- | --- |
| Вычисление интеграла методом Симпсона | | | |
| Нижний предел | Верхний предел | n | h |
| 0,80 | 1,60 | 20,00 | 0,04 |
| k | x | Yk(x) |  |
| 0 | 0,80 | 0,26855481 |  |
| 1 | 0,84 | 0,276044271 |  |
| 2 | 0,88 | 0,283013101 |  |
| 3 | 0,92 | 0,289484556 |  |
| 4 | 0,96 | 0,295482281 |  |
| 5 | 1,00 | 0,301029996 |  |
| 6 | 1,04 | 0,30615123 |  |
| 7 | 1,08 | 0,310869118 |  |
| 8 | 1,12 | 0,315206228 |  |
| 9 | 1,16 | 0,319184442 |  |
| 10 | 1,20 | 0,322824855 |  |
| 11 | 1,24 | 0,326147714 |  |
| 12 | 1,28 | 0,329172374 |  |
| 13 | 1,32 | 0,331917273 |  |
| 14 | 1,36 | 0,334399928 |  |
| 15 | 1,40 | 0,336636936 |  |
| 16 | 1,44 | 0,338643991 |  |
| 17 | 1,48 | 0,340435903 |  |
| 18 | 1,52 | 0,342026627 |  |
| 19 | 1,56 | 0,343429295 |  |
| 20 | 1,60 | 0,344656249 |  |
|  |  | |= | 0,253970271 |

***Программа на Python:***

import math  
  
  
def func(x: float) -> float:  
 return math.log10(x \*\* 2 + 1) / x *# подынтегральная функция*def simpson\_rule(a: float, b: float, n: int) -> float:  
 *"""  
 Вычисляет определенный интеграл функции с использованием правила Симпсона.  
  
 Параметры:  
 a (float): Нижний предел интегрирования.  
 b (float): Верхний предел интегрирования.  
 n (int): Количество подинтервалов. Должно быть четным числом.  
  
 Возвращает:  
 float: Приближенное значение определенного интеграла.  
 """* if n % 2 != 0:  
 raise ValueError("число n должно быть четным")  
  
 h = (b - a) / n *# шаг* integral = 0 *# аккумулятор значения интеграла* for i in range(n + 1):  
 x = a + i \* h *# вычисление значения x на основании шага и индекса текущей итерации* if i == 0 or i == n:  
 integral += func(x) *# первый и последний индекс без множителя* elif i % 2 == 0:  
 integral += 2 \* func(x) *# для четных индексов множитель = 2* else:  
 integral += 4 \* func(x) *# для нечетных индексов множитель = 4* integral \*= h / 3 *# умножаем полученную сумму на коэффициент (шаг деленный на 3)* return integral  
  
  
def main() -> None:  
 *# Вычисление интегралов* integral\_simpson\_8 = simpson\_rule(a=0.8, b=1.6, n=8)  
 integral\_simpson\_20 = simpson\_rule(a=0.8, b=1.6, n=20)  
 error\_rectangular = abs(integral\_simpson\_20 - integral\_simpson\_8)  
 print("Интеграл по формуле Симпсона:")  
 print(f"n=8 : {integral\_simpson\_8}\nn=20: {integral\_simpson\_20}")  
 print(f"Оценка погрешности: {error\_rectangular:.7f}")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Вывод программы:



**Вывод:**

В данной лабораторной работе были изучены следующие методы численного решения определенных интегралов:

- метод прямоугольников

- метод трапеций

- метод Симпсона

Были получены следующие значения точности вычислений при разбиении области интегрирования на 8 и 20 равных частей:

Метод прямоугольников : 0.0001

Метод трапеций : 0.0001

Метод Симпсона : 0.0000001

Таким образом, метод Симпсона является наиболее точным, поэтому его чаще  
всего используют при работе на ЭВМ. На ЭВМ есть стандартные программы,  
вычисляющие значения определенного интеграла методом Симпсона