

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

|  |
| --- |
| **ШКОЛА ЕСТЕСТВЕННЫХ НАУК**  **кафедра информатики, математического и компьютерного моделирования** |

**О Т Ч Е Т**

о выполнении лабораторной работы №2 по предмету «Методы сплайн-функций»

Направление подготовки 01.03.02 «Прикладная математика и информатика»

|  |  |  |
| --- | --- | --- |
|  |  | Выполнил студент гр. Б8116-01.03.02-ПМИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Глушков В.К. |
|  |  | (*подпись*) *(Ф.И.О.)* |
|  |  |  |

г. Владивосток

2020

## Цель работы

Найти интеграл функции через интегрирование дискретного интерполяционного сплайна и оценить погрешность интегрирования с разными значениями производных на концах.

## Задание

Интегрировать следующие функции:

|  |  |
| --- | --- |
| , где | (1) |
| , где | (2) |

путем интегрирования дискретного сплайна с первым типом краевых условий

|  |  |
| --- | --- |
|  | (3) |

## Выполнение задание

Пусть на отрезке [] в узлах сетки заданы значения некоторой функции

Дискретным интерполяционным кубическим сплайном называют функцию которая:

1. на каждом из интервалов является кубическим многочленом

|  |  |
| --- | --- |
|  | (4) |

1. при заданных

|  |  |
| --- | --- |
|  | (5) |
|  | (6) |

Введем обозначение

|  |  |
| --- | --- |
|  | (7) |

В таком случае сплайн при записывается в виде

|  |  |
| --- | --- |
|  | (8) |

где

Для нахождения моментов при первом граничном условии будем использовать следующие формулы

|  |  |
| --- | --- |
|  | (9) |

***Теорема 1***. *Если интерполирует функцию и удовлетворяет краевым условиям одного из четырех типов, то*

|  |  |
| --- | --- |
|  | (10) |

*если то*

*если то*

Наиболее простым способом получения формул численного интегрирования для интеграла

|  |  |
| --- | --- |
|  | (11) |

является замена на интерполяционный сплайн и расчет приближенного значения интеграла

|  |  |
| --- | --- |
|  | (12) |

Так как искомый сплайн ограничен первыми краевыми условиями и будет построен на равномерной сетке, то формула для вычисления интеграла непериодической функции примет вид

|  |  |
| --- | --- |
|  | (13) |

а формула периодической функции с периодом упрощается до

|  |  |
| --- | --- |
|  | (14) |

Погрешность вычисления интеграла можно оценить следующим образом

|  |  |
| --- | --- |
|  | (15) |

Следовательно, достаточно иметь оценку погрешности приближения функции сплайном , которая указана в теореме 1.

## Код программы

|  |
| --- |
| main.py |
| import numpy as np import math import matplotlib.pyplot as plt  from lab1.configurations import configurations from scipy import integrate   class Lab2:  def \_\_init\_\_(self, points, diff2):  self.points = points  self.n = len(points) - 1  self.diff2 = diff2  self.m\_res = []  self.\_\_calc\_matrix\_\_()   def \_\_get\_x\_\_(self, i):  \_, x\_, \_ = self.points[i]  return x\_   def \_\_get\_y\_\_(self, i):  \_, \_, y\_ = self.points[i]  return y\_   def \_\_get\_eps\_\_(self, i):  eps\_, \_, \_ = self.points[i]  return eps\_   def calc\_i(self, x):  for i in range(len(self.points) - 1):  x1 = self.\_\_get\_x\_\_(i)  x2 = self.\_\_get\_x\_\_(i + 1)  if x1 <= x <= x2:  return i  assert False, "ERROR: 'i' not found!"   def \_\_h\_\_(self, i):  if i + 1 < len(self.points):  return self.\_\_get\_x\_\_(i + 1) - self.\_\_get\_x\_\_(i)  else:  return self.\_\_get\_x\_\_(i) - self.\_\_get\_x\_\_(i - 1)   def \_\_calc\_matrix\_\_(self):  m\_a = np.zeros((self.n + 1) \*\* 2).reshape(self.n + 1, self.n + 1)  m\_a[0][0] = 2  m\_a[0][1] = 1  for i in range(1, self.n):  m\_a[i][i - 1] = self.\_\_mu\_\_(i) \* (1 - math.pow(self.\_\_get\_eps\_\_(i), 2) / math.pow(self.\_\_h\_\_(i - 1), 2))  m\_a[i][i] = 2 + math.pow(self.\_\_get\_eps\_\_(i), 2) / (self.\_\_h\_\_(i - 1) \* self.\_\_h\_\_(i))  m\_a[i][i + 1] = self.\_\_lmbd\_\_(i) \* (1 - math.pow(self.\_\_get\_eps\_\_(i), 2) / math.pow(self.\_\_h\_\_(i), 2))  m\_a[self.n][self.n - 1] = 1  m\_a[self.n][self.n] = 2   m\_b = np.zeros(self.n + 1)  term1 = ((self.\_\_get\_y\_\_(1) - self.\_\_get\_y\_\_(0)) / self.\_\_h\_\_(0) - self.diff2(self.\_\_get\_x\_\_(0)))  m\_b[0] = 6 / self.\_\_h\_\_(0) \* term1  for i in range(1, self.n):  term1 = 6 / (self.\_\_h\_\_(i - 1) + self.\_\_h\_\_(i))  term2 = (self.\_\_get\_y\_\_(i + 1) - self.\_\_get\_y\_\_(i)) / self.\_\_h\_\_(i)  term3 = (self.\_\_get\_y\_\_(i) - self.\_\_get\_y\_\_(i - 1)) / self.\_\_h\_\_(i - 1)  m\_b[i] = term1 \* (term2 - term3)   term1 = 3 \* ((self.\_\_get\_y\_\_(self.n) - self.\_\_get\_y\_\_(self.n - 1)) / self.\_\_h\_\_(self.n))  term2 = self.\_\_h\_\_(self.n) \* self.diff2(self.\_\_get\_x\_\_(self.n)) / 2  m\_b[self.n] = term1 + term2   self.m\_res = np.linalg.solve(m\_a, m\_b)   def sdx(self):  term1 = 0.5 \* sum([self.\_\_h\_\_(j) \* (self.\_\_get\_y\_\_(j) + self.\_\_get\_y\_\_(j + 1)) for j in range(0, self.n - 1)])  ss = [math.pow(self.\_\_h\_\_(j), 3) \* (self.m\_res[j] + self.m\_res[j + 1]) for j in range(0, self.n - 1)]  term2 = 1 / 24 \* sum(ss)  return term1 - term2   def \_\_mu\_\_(self, i):  return self.\_\_h\_\_(i - 1) / (self.\_\_h\_\_(i - 1) + self.\_\_h\_\_(i))   def \_\_lmbd\_\_(self, i):  return 1 - self.\_\_mu\_\_(i)   def main(configuration):  func = configuration["function"]  diff = configuration["diff"]  diff2 = configuration["diff2"]  a = configuration["a"]  n = configuration["n"]  x = np.linspace(configuration["x\_start"], configuration["x\_end"], num=n + 1)  y = np.array([func(a, x\_) for x\_ in x])  e = np.linspace(1e-5, 1, num=100)  d = []  for e\_ in e:  eps = [e\_ for i in range(n + 1)]  solver = Lab2(list(zip(eps, x, y)), lambda x\_: diff2(a, x\_))   y\_res = integrate.quad(lambda x\_: func(a, x\_), configuration["x\_start"], configuration["x\_end"])[0]  s\_res = solver.sdx()   d\_ = abs(y\_res - s\_res)  # print("eps {} : error {}".format(e\_, d\_))  d.append(d\_)  fig, ax = plt.subplots()  ax.grid()  ax.plot(e, d, label="error")  ax.legend()  ax.set\_xlabel('eps')  ax.set\_ylabel('error')  plt.show()  return 0   if \_\_name\_\_ == "\_\_main\_\_":  for i in [0, 2, 5, 9, 10, 12, 15, 19]:  print("a = {}".format(i % 10 + 1))  main(configurations[i]) |

## Результат

|  |  |  |
| --- | --- | --- |
| Периодическая функция | | |
| Параметр | 1 | 3 |
| График |  |  |

|  |  |  |
| --- | --- | --- |
| Периодическая функция | | |
| Параметр | 6 | 10 |
| График |  |  |

|  |  |  |
| --- | --- | --- |
| Непериодическая функция | | |
| Параметр | 1 | 3 |
| График |  |  |

|  |  |  |
| --- | --- | --- |
| Непериодическая функция | | |
| Параметр | 6 | 10 |
| График |  |  |

## Вывод

Были успешно вычислены приближенные интегралы функций с помощью формулы интегрирования дискретного интерполяционного сплайна, ограниченного краевыми условиями первого типа. Погрешность тем ниже, чем меньше параметр дискретного сплайна . Также было установлено, что оценка погрешности интегрирования зависит от оценки погрешности интерполирования.