**Потоцкая Анастасия, Б8116-01.03.02**

**Отчёт по лабораторной работе № 2**

1. **Задание**

Найти интеграл функции через интерполяционный сплайн, используя первый тип краевых условий.

2. Алгоритм выполнения задания

Пусть на сетке заданы значения

Интеграл можно найти по формуле

Погрешность вычисленного интеграла можно оценить следующим образом

Следовательно, достаточно иметь оценку погрешности приближения функции сплайном .

4. Результатa

В качестве параметры эксперимента рассмотрим

1. Размер равномерной выборки
2. Значения функции: точные и с помехами
3. Значения моментов функции: точные и приближенные

Рассмотрим функцию на сетке .

|  |  |
| --- | --- |
| Параметры эксперимента | Ошибка |
|  | 1.17e-06 |
|  | 6.64e-02 |
|  | 3.06e-05 |
|  | 1.38e-01 |
|  | 7.31e-08 |
|  | 2.04e-02 |
|  | 7.44e-06 |
|  | 7.18e-02 |
|  |  |
|  | 5.42e-06 |
|  | 4.88e-02 |
|  | 1.41e-04 |
|  | 4.74e-02 |
|  | 3.38e-07 |
|  | 1.21e-02 |
|  | 3.42e-05 |
|  | 1.06e-01 |
|  |  |
|  | 1.49e-05 |
|  | 2.06e-01 |
|  | 3.82e-04 |
|  | 5.88e-02 |
|  | 9.29e-07 |
|  | 5.51e-02 |
|  | 9.27e-05 |
|  | 1.63e-02 |

Рассмотрим функцию на равномерной сетке .

|  |  |
| --- | --- |
| Вычисленное значение | Ошибка |
|  | 3.32e-08 |
|  | 1.83e-02 |
|  | 2.13e-06 |
|  | 1.95e-02 |
|  | 2.07e-09 |
|  | 1.97e-02 |
|  | 5.25e-07 |
|  | 1.30e-02 |
|  |  |
|  | 8.84e-08 |
|  | 2.83e-02 |
|  | 5.66e-06 |
|  | 1.70e-02 |
|  | 5.52e-09 |
|  | 3.99e-03 |
|  | 1.40e-06 |
|  | 1.85e-02 |

6.Выводы

При точных значениях функции, увеличение выборки увеличивает точность на порядок. При наличии помех в значениях функции или в моментах, увеличение выборки незначительно влияет на точность.

Больше всего на точность интеграла влияет помехи в значениях функции, помехи уменьшают точность на 2-3 порядка. Помехи в моментах уменьшают точность на порядок.

5. Код программы

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import random  def h(x, i):  return x[i + 1] - x[i]  def getInte(configuration):  x = np.linspace(configuration["x\_start"],  configuration["x\_end"],  num=configuration["n"] +1)  y = configuration["y"]  diff = configuration["diff"]  a, b = 0, 0  print(len(x), len(y))  for i in range(len(y) - 1):  a += h(x, i) \* (y[i] + y[i + 1])  b += pow(h(x, i), 2) \* (diff[i] - diff[i + 1])  return 0.5 \* a + (1/12) \* b  def main(conf):  func = conf["func"]  fdiff = conf["diff"]  conf["x"] = np.linspace(conf["x\_start"], conf["x\_end"], num=conf["n"] + 1)  conf["y"] = np.array([func(conf["a"], x\_) for x\_ in conf["x"]])  conf["diff"] = np.array([fdiff(conf["a"], x\_) for x\_ in conf["x"]])  res = getInte(conf)  true\_res = conf["funcint"](conf["a"], conf["x\_end"]) - conf["funcint"](conf["a"], conf["x\_start"])  print(res)  print(true\_res)  print(abs(true\_res - res))  def func1(a, x):  return np.sin(a \* x)  def func1dx(a, x):  return a \* np.cos(a \* x)  def func1int(a, x):  return -(1/a) \* np.cos(a\*x)  configurations = [  {"func": func1, "funcint": func1int, "diff": func1dx, "a": 3, "n": 20, "x\_start": 0, "x\_end": np.pi},  {"func": func1, "funcint": func1int, "diff": func1dx, "a": 3, "n": 50, "x\_start": 0, "x\_end": np.pi},  {"func": func1, "funcint": func1int, "diff": func1dx, "a": 3, "n": 100, "x\_start": 0, "x\_end": np.pi},  {"func": func1, "funcint": func1int, "diff": func1dx, "a": 3, "n": 200, "x\_start": 0, "x\_end": np.pi}  ]  if \_\_name\_\_ == "\_\_main\_\_":  for conf in configurations:  main(conf) |