**Потоцкая Анастасия, Б8116-01.03.02**

**Отчёт по лабораторной работе № 1**

1. **Задание**

Построение кубического сплайна через наклоны, используя первый тип краевых условий

2. Алгоритм выполнения задания

Путсть на сетке заданы значения

Кубический сплайн на отрезке можно записать

Моментами называют

Сплайн на отрезке

, где

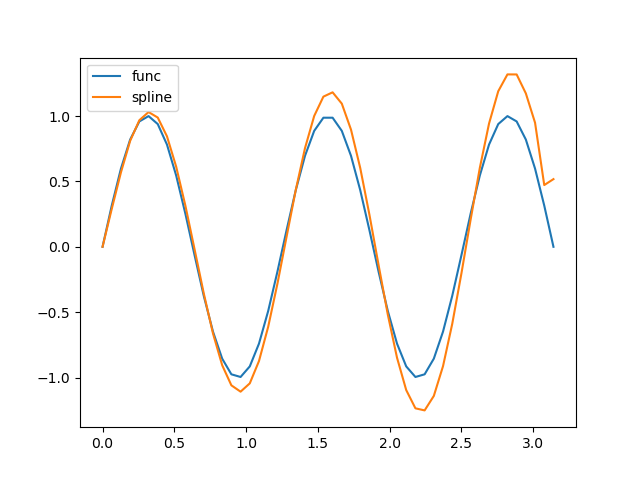
Наклоны находятся из решения системы уравнений с неизвестными:

Для замыкания системы необходимо добавить первое краевое условия:

Откуда получаем

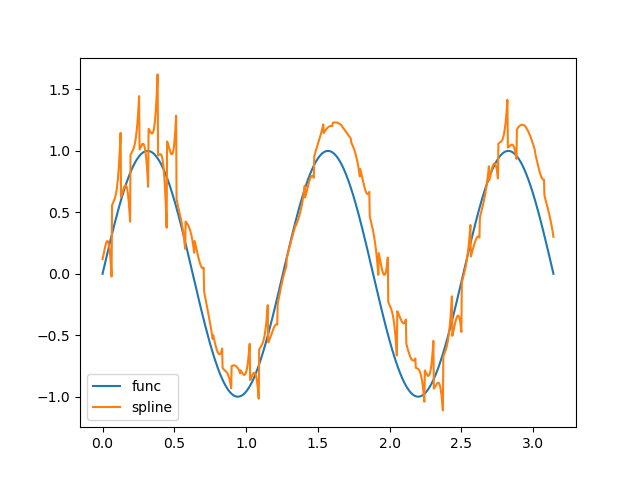
4. Результат

Была построенная интерполяция функции по сетки размеров 100. Точность .

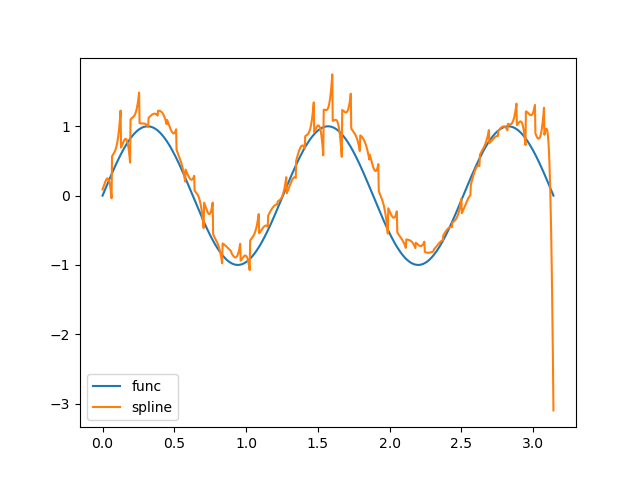


|  |  |  |
| --- | --- | --- |
| x | Точное значение | Приблизительное значение |
| 0 | 0 | 0 |
| 0.06411414 | 0.315108218 | 0.28990533 |
| 0.12822827 | 0.598110530 | 0.56940555 |
| 0.19234241 | 0.820172255 | 0.8101898 |
| 0.25645654 | 0.958667853 | 0.96795101 |
| 0.32057068 | 0.999486216 | 1.03041175 |
| 0.38468481 | 0.938468422 | 0.98837613 |
| 0.44879895 | 0.781831482 | 0.84493611 |
| 0.51291309 | 0.545534901 | 0.61313161 |
| 0.57702722 | 0.253654584 | 0.31534985 |

Результат с плохими граничным условиями



Точность .



Точность .

5. Код программы

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import random  def h(x, i):  return x[i] - x[i - 1]  def s(x\_, x, y, dy):  for i in range(0, len(x)):  if x[i] <= x\_ <= x[i + 1]:  a = 6 / h(x, i + 1)  a \*= (y[i + 1] - y[i]) / h(x, i + 1) - (dy[i + 1] + 2 \* dy[i]) / 3  b = 12 / np.power(h(x, i + 1), 2)  b \*= (dy[i + 1] + dy[i]) / 2 - (y[i + 1] - y[i]) / h(x, i + 1)  с = dy[i]    result = y[i] + с \* (x\_ - x[i])  result += (a / 2) \* np.power(x\_ - x[i], 2)  result += (b / 2) \* np.power(x\_ - x[i], 3)  return result  def mu(x, i):  return h(x, i + 1) / (h(x, i) + h(x, i + 1))  def lambda\_(x, i):  return h(x, i) / (h(x, i) + h(x, i + 1))  def getM(configuration):  x = np.linspace(configuration["x\_start"],  configuration["x\_end"],  num=configuration["n"])  y = configuration["y"]  diff = configuration["diff"]  n = len(x)  m\_a = np.zeros(n \*\* 2).reshape(n, n)  m\_a[0][0] = 1  for i in range(1, n - 1):  m\_a[i][i - 1] = mu(x, i)  m\_a[i][i] = 2  m\_a[i][i + 1] = lambda\_(x, i)  m\_a[-1][-2] = 1  m\_b = np.zeros(n)  m\_b[0] = diff[0] # (y[1] - y[0]) / h(x, 0)  for i in range(1, n - 1):  m\_b[i] = lambda\_(x, i) \* (y[i + 1] - y[i]) / h(x, i + 1)  m\_b[i] += mu(x, i) \* (y[i] - y[i - 1]) / h(x, i)  m\_b[i] \*= 3  m\_b[-1] = diff[-1] # (y[-1] - y[-2]) / h(x, len(x) - 1)  return np.linalg.solve(m\_a, m\_b)  def setUp(conf):  func = conf["func"]  fdiff = conf["diff"]  tr = conf["tr\_func"]  conf["func"] = lambda x: func(conf["a"], x)  conf["diff"] = lambda x: fdiff(conf["a"], x)  conf["tr\_func"] = lambda x: tr(conf["a"], x)  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"]])  return getM(conf)  def plot(n, configuration):  true\_func = configuration["tr\_func"]  x\_dots = np.linspace(configuration["x\_start"], configuration["x\_end"], num=n)  y\_dots = np.array([true\_func(x\_) for x\_ in x\_dots])  s\_dots = np.array([s(x\_, x\_dots, y\_dots, configuration["m\_res"]) for x\_ in x\_dots])  plt.plot(x\_dots, y\_dots, label="func")  plt.plot(x\_dots, s\_dots, label="spline")  plt.legend()  plt.show()  d = max([abs(y\_dots[i] - s\_dots[i]) for i in range(0, len(y\_dots))])  print(d)  return  def main(conf):  conf["m\_res"] = setUp(conf)  plot(conf["n"], conf)  def func1(a, x):  return np.sin(a \* x)  def func1dx(a, x):  return a \* np.cos(a \* x)  def func1app(a, x):  offset = random.uniform(0.5, 1)  return func1(a, x) + offset  def func1dxapp(a, x):  dx = 0.1  return (func1(a, x) - func1(a, x - dx)) / (x - x + dx)  def func1dxfalse(a, x):  return a + x  configurations = [  {"func": func1, "tr\_func": func1, "diff": func1dx, "a": 5, "n": 50, "x\_start": 0, "x\_end": np.pi},  {"func": func1app, "tr\_func": func1, "diff": func1dxapp, "a": 5, "n": 50, "x\_start": 0, "x\_end": np.pi},  {"func": func1app, "tr\_func": func1, "diff": func1dxfalse, "a": 5, "n": 50, "x\_start": 0, "x\_end": np.pi}  ]  if \_\_name\_\_ == "\_\_main\_\_":  main(configurations[0])  main(configurations[1])  main(configurations[2]) |