

**НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ УКРАИНЫ
“КИЕВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ ИМЕНИ ИГОРЯ
СИКОРСКОГО”
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ №5

Дисциплина: «Методы вычислений»

Тема: «Интерполяция»

Выполнила

студентка 3 курса группы ФИ-41

Лавягина Ольга Алексеевна

Проверила

Стёпочкина Ирина Валерьевна

ОГЛАВЛЕНИЕ

1	Задание	2
2	Интерполяционный полином с обозначением узлов интерполяции	3
3	Интерполирующая сплайн-функция.	5
4	Значение погрешности	7
5	Листинг программы.	9
	Выводы	12

1 ЗАДАНИЕ

Задана функция

$$f(x) = \frac{1}{\sin x}$$

на отрезке

$$\left[\frac{\pi}{6}, \frac{\pi}{2} \right].$$

Нужно построить таблицу значений функции в узлах на заданном отрезке.

По таблично заданной:

- построить интерполяционный полином в форме Ньютона или Лагранжа;
- совершить интерполяцию спланами (второго или третьего порядка);
- построить график погрешности интерполяции $\varepsilon = |P_n(x) - f(x)|$. Нужно вывести на график с шагом, меньше в 5-6 раз, чем шаг интерполяции, соответственные значения полинома, сплайн-функции и точной функции.

Если погрешность очень мала, применить масштабирование.

Сравнить полученный результат с теоретической оценкой погрешности.

2 ИНТЕРПОЛЯЦИОННЫЙ ПОЛИНОМ С ОБОЗНАЧЕНИЕМ УЗЛОВ ИНТЕРПОЛЯЦИИ

Пусть на

$$\left[\frac{\pi}{6}, \frac{\pi}{2}\right]$$

задана $n + 1$ точка — узлов интерполяции x_0, x_1, \dots, x_n и значений функции в этих узлах $f(x_0), f(x_1), \dots, f(x_n)$.

x:

[0.5235987755982988, 0.6283185307179586, 0.7330382858376184, 0.8377580409572782, 0.942477796076938, 1.0471975511965979, 1.1519173063162575, 1.2566370614359172, 1.361356816555577, 1.4660765716752366, 1.5707963267948963]

f:

[2.0000000000000004, 1.7013016167040798, 1.4944765498646086, 1.3456327296063761, 1.2360679774997896, 1.1547005383792515, 1.0946362785060468, 1.0514622242382672, 1.0223405948650293, 1.0055082795635164, 1.0]

Нужно построить интерполяционный полином по этой таблице значений.

Необходимо, чтобы $L_n(x_i) = y_i$. Отсюда

$$L_n(x) = \sum_{i=0}^n y_i \cdot \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}.$$

Для заданной таблицы

	10		9		8		7		6		5		4
1.105 x	-	12.73 x	+	66 x	-	203.1 x	+	412.4 x	-	579.7 x	+	575.5 x	
		3		2									
- 403.3 x	+	195.5 x	-	62.21 x	+	11.79							

Полученный полином изображён на рисунке 2.1. Красным изображены узлы интерполяции.

Для данных узлов получен интерполяционный полином Ньютона (рис. 2.2).

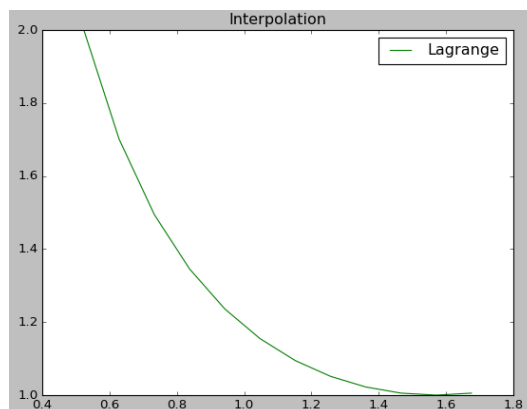


Рисунок 2.1 — Полином Лагранжа

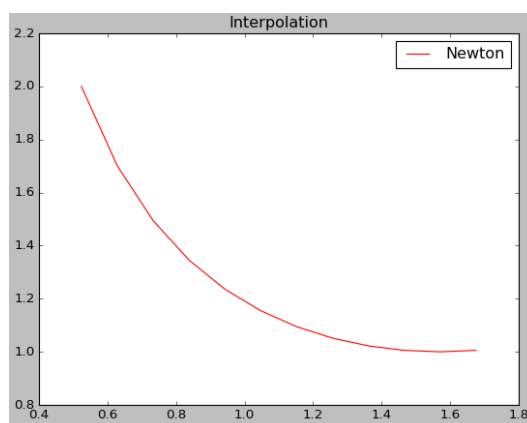


Рисунок 2.2 — Полином Ньютона

3 ИНТЕРПОЛИРУЮЩАЯ СПЛАЙН-ФУНКЦИЯ

Составлена ЛСУ для нахождения коэффициентов при сплайн-функции. Использовано то, что все

$$h = \frac{\pi}{30}.$$

Система имеет вид $Am = b$, где матрица A имеет вид

[illegible]

вектор-столбец b имеет вид

$$b = \begin{bmatrix} 0.87732554713739574 \\ 0.55368012000205091 \\ 0.37508747138267329 \\ 0.26926450461833418 \\ 0.20343037684714838 \\ 0.16128958271650951 \\ 0.1341907730636353 \\ 0.11735430490342991 \\ 0.10813657580708488 \\ 0.10520039045588443 \end{bmatrix}$$

Проведена интерполяция функции кубическим сплайном кубическим сплайном (рис. 3.1).

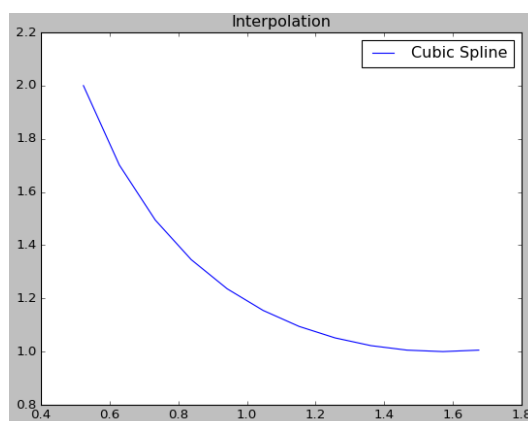


Рисунок 3.1 — Кубический сплайн

4 ЗНАЧЕНИЕ ПОГРЕШНОСТИ

Погрешности были найдены в узлах, которые отстают друг от друга на

$$\frac{\pi}{150}$$

на отрезке

$$\left[\frac{\pi}{6}, \frac{\pi}{2} \right]$$

для всех использованных типов интерполяции. Была использована формула

$$|f(x) - \text{polynom}(x)|.$$

На рисунке 4.1 изображена погрешность для интерполяции полиномом Лагранжа, на рисунке 4.2 — для полинома Ньютона, на рисунке 4.3 — для интерполяции кубическими сплайнами.

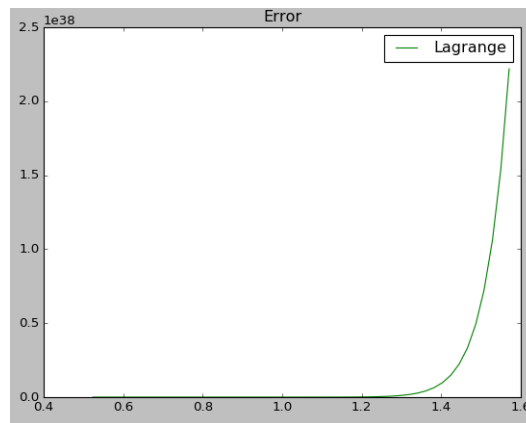


Рисунок 4.1 — Погрешность для полинома Лагранжа

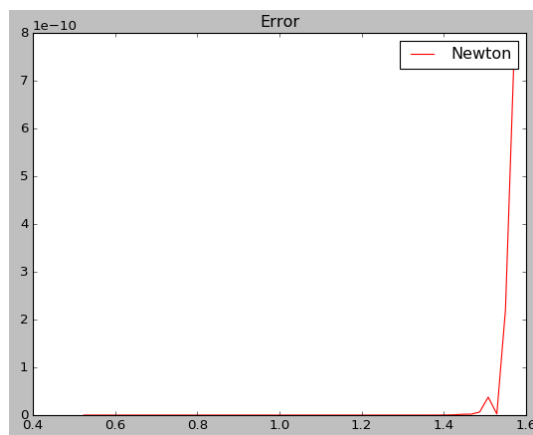


Рисунок 4.2 — Погрешность для полинома Ньютона

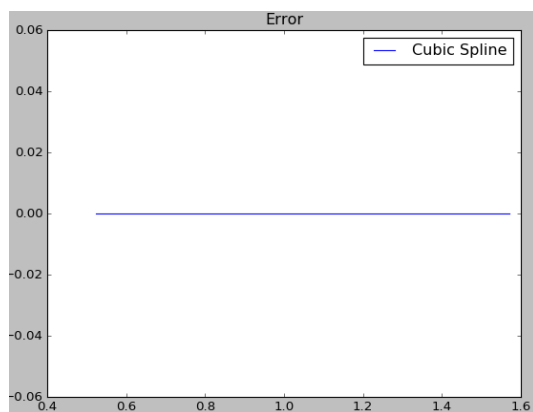


Рисунок 4.3 — Погрешность для сплайнов

5 ЛИСТИНГ ПРОГРАММЫ

Листинг файла `__main__.py`

```

import math, scipy, numpy
import matplotlib.pyplot as plt
from scipy import interpolate
from scipy.interpolate import CubicSpline

from lagrange import f, lagrange
from newton import newton_interpolation
from spline import spline_system

if __name__ == '__main__':
    x = []
    x.append(math.pi/6)
    while x[-1] <= math.pi/2:
        x.append(x[-1] + math.pi/30)
    x = scipy.array(x)
    y = [];
    for element in x:
        y.append(f(element))
    y = scipy.array(y)

    differences = spline_system(y)

    lagrange_polynom = lagrange(x, y)

    newton = []
    for element in x:
        newton.append(newton_interpolation(x, y, element))

    cs = CubicSpline(x, y)

    plt.plot(x, lagrange_polynom(x), 'g', x, newton, 'r', x, cs(x), 'b')
    plt.legend(['Lagrange', 'Newton', 'Cubic Spline'])
    plt.title('Interpolation')
    plt.show()

    xnew = []
    xnew.append(math.pi/6)
    while xnew[-1] <= math.pi/2:
        xnew.append(xnew[-1] + math.pi/150)
    xnew = scipy.array(xnew)
    ynew = [];
    for element in xnew:
        ynew.append(f(element))

```

```

ynew = scipy.array(ynew)

difference_lagrange = []
new_lagrange_polynom = lagrange(xnew, ynew)

difference_newton = []

difference_spline = []
new_cs = CubicSpline(xnew, ynew)

for element in xnew:
    difference_lagrange.append(abs(f(element) - new_lagrange_polynom(element)))
    difference_newton.append(abs(f(element) - newton_interpolation(xnew, ynew, element)))
    difference_spline.append(abs(f(element) - new_cs(element)))

difference_lagrange = scipy.array(difference_lagrange)
difference_newton = scipy.array(difference_newton)
difference_spline = scipy.array(difference_spline)

plt.plot(xnew, difference_lagrange, 'g', xnew, difference_newton, 'r', xnew, difference_spline, 'b')
plt.legend(['Lagrange', 'Newton', 'Cubic Spline'])
plt.title('Error')
plt.show()

```

Листинг файла lagrange.py с вычислением интерполяционного полинома Лагранжа

```

import scipy, numpy, math
import matplotlib.pyplot as plt

def f(x):
    return 1 / math.sin(x)

def lagrange(x, y):
    result = scipy.poly1d([0.0]) #setting result = 0

    for i in range(0, len(x)): #number of polynomials  $L_k(x)$ .
        temp_numerator = scipy.poly1d([1.0]) # resets temp_numerator such that a new numerator can be created
        denominator = 1.0 #resets denominator such that a new denominator can be created for each i.
        for j in range(0, len(x)):
            if i != j:
                temp_numerator *= scipy.poly1d([1.0, -x[j]]) #finds numerator for  $L_i$ 
                denominator *= x[i] - x[j] #finds denominator for  $L_i$ 
        result += (temp_numerator/denominator) * y[i] #linear combination
    return result

```

Листинг файла newton.py с вычислением интерполяционного полинома

Ньютона

```
def newton_interpolation(x, y, u):
    '''
    Parameters
    -----
    x : list of floats
    y : list of floats
    u : float

    Returns
    -----
    float
        an estimate at the point u
    '''
    g = y[:]
    s = g[0]
    for i in range(len(y)-1):
        g = [(g[j+1]-g[j])/(x[j+1]-x[j]) for j in range(len(g)-1)]
        s += g[0] * product(u-x[j] for j in range(i+1))
    return s

def product(a):
    p = 1
    for i in a: p *= i
    return p
```

Листинг файла spline.py с вычислением вектора-столбца b для системы $Am = b$, с помощью которой находятся коэффициенты m сплайн-функции

```
import math

def spline_system(y):
    h = math.pi/30
    differences = []
    for y1, y0 in zip(y[1:], y[:-1]):
        differences.append((y1 - y0) / h)
    b = []
    for d1, d0 in zip(differences[1:], differences[:-1]):
        b.append(d1 - d0)
    print('Vector b:')
    print(b)
    return b
```

ВЫВОДЫ

К функции была применена интерполяция методами Лагранжа, Ньютона, сплайнов. Для этого были составлены узлы интерполяции на отрезке и найдены значения функции в этих узлах.

Построены графики погрешностей для данных методов, которые получились незначительными. Интерполяция сплайнами дала наилучший результат.