Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №5**

по «Вычислительной математике»

Интерполяция фунции

Выполнил:

Студент группы P32151

Кортыш А.О.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2023

**Цель работы**

Решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.

**Вычислительная реализация**

Таблица конечных разностей.

| i |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2,10 | 3,7587 | 0,4274 | 0,3083 | -0,6171 | 1,0778 | -1,7774 | 2,9757 |
| 1 | 2,15 | 4,1861 | 0,7357 | -0,3088 | 0,4607 | -0,6996 | 1,1983 |  |
| 2 | 2,20 | 4,9218 | 0,4269 | 0,1519 | -0,2389 | 0,4987 |  |  |
| 3 | 2,25 | 5,3487 | 0,5788 | -0,087 | 0,2598 |  |  |  |
| 4 | 2,30 | 5,9275 | 0,4918 | 0,1728 |  |  |  |  |
| 5 | 2,35 | 6,4193 | 0,6646 |  |  |  |  |  |
| 6 | 2,40 | 7,0839 |  |  |  |  |  |  |

X1 = 2.112

Используем первую формулу Ньютона:

X2 = 2.205

Используем вторую формулу Гаусса:

**Листинг программы**

from typing import List, Tuple

from sympy import Add, Symbol, lambdify, sin

var = Symbol("x")

def get\_finite\_differences(y: List[float]) -> List[List[float]]:

differences = [[y[i]] for i in range(len(y))]

for col in range(1, len(y)):

for row in range(len(y)):

if col + row == len(y):

break

differences[row].append(differences[row+1][col-1] - differences[row][col-1])

return differences

def get\_lagrange\_polynom(x: List[float], y: List[float], x\_ans: float) -> Tuple[Add, float]:

lagrange\_polynom: Add = Add(0)

for i in range(len(x)):

cur\_term: Add = Add(y[i])

for j in range(len(x)):

if i != j:

cur\_term \*= (var - x[j]) / (x[i] - x[j])

lagrange\_polynom += cur\_term

callable\_lagrange\_polynom = lambdify(var, lagrange\_polynom, "numpy")

return lagrange\_polynom, callable\_lagrange\_polynom(x\_ans)

def check\_algebraic\_progression(x: List[float], possible\_delta=1e6):

diff = x[1] - x[0]

for index, el in enumerate(x):

if abs(x[0] + index \* diff - el) > possible\_delta:

raise ValueError

def get\_gauss\_polynom(x: List[float], y: List[float], x\_ans: float) -> Tuple[Add, float]:

# Check that there is h

check\_algebraic\_progression(x)

# make sure that number of points is odd

cur\_x = x[::]

cur\_y = y[::]

if len(cur\_x) % 2 == 0:

cur\_x = cur\_x[:-1]

cur\_y = cur\_y[:-1]

first\_formula\_fl = x\_ans > cur\_x[len(cur\_x) // 2]

gauss\_polynom = get\_gauss(cur\_x, cur\_y, first\_formula\_fl)

callable\_gauss\_polynom = lambdify(var, gauss\_polynom, "numpy")

return gauss\_polynom, callable\_gauss\_polynom(x\_ans)

def get\_gauss(x: List[float], y: List[float], greater: bool) -> Add:

differences = get\_finite\_differences(y)

middle = len(x) // 2 # cur\_x[middle] = x\_0 in terms of lecture

h = x[1] - x[0]

t = (var - x[middle]) / h

gauss\_polynom = Add(y[middle])

addition = 0

cur\_term = 1

for i in range(1, len(x)):

if i % 2 == 0:

addition += 1

if not greater:

if i % 2 == 0:

cur\_term \*= (t + addition)

else:

cur\_term \*= (t - addition)

else:

if i % 2 == 0:

cur\_term \*= (t - addition)

else:

cur\_term \*= (t + addition)

cur\_term /= i

cur\_delta = differences[middle - (i + 1) // 2][i] if not greater else differences[middle - i // 2][i]

gauss\_polynom += cur\_delta \* cur\_term

return gauss\_polynom

def get\_stirling\_polynom(x: List[float], y: List[float], x\_ans: float) -> Tuple[Add, float]:

# Check that there is h

check\_algebraic\_progression(x)

# make sure that number of points is odd

cur\_x = x[::]

cur\_y = y[::]

if len(cur\_x) % 2 == 0:

cur\_x = cur\_x[:-1]

cur\_y = cur\_y[:-1]

stirling\_polynom = (get\_gauss(cur\_x, cur\_y, True) + get\_gauss(cur\_x, cur\_y, False)) / 2

callable\_stirling\_polynom = lambdify(var, stirling\_polynom, "numpy")

return stirling\_polynom, callable\_stirling\_polynom(x\_ans)

def get\_bessel\_polynom(x: List[float], y: List[float], x\_ans: float) -> Tuple[Add, float]:

# Check that there is h

check\_algebraic\_progression(x)

# make sure that number of points is even

cur\_x = x[::]

cur\_y = y[::]

if len(cur\_x) % 2 == 1:

cur\_x = cur\_x[:-1]

cur\_y = cur\_y[:-1]

cur\_term = Add(1)

h = cur\_x[1] - cur\_x[0]

middle = len(cur\_x) // 2 - 1

t = (var - cur\_x[middle]) / h

diff = get\_finite\_differences(cur\_y)

bessel\_polynom = Add(0)

for i in range(len(cur\_x)):

if i != 0:

cur\_term /= i

if i % 2 == 0:

bessel\_polynom += cur\_term\*(diff[middle - i//2][i] + diff[middle - i//2+1][i]) / 2

else:

bessel\_polynom += cur\_term\*(t - 1/2)\*diff[middle - i//2][i]

cur\_term \*= (t - (i + 1) // 2)\*(t + (i + 1) // 2 - 1)

callable\_bessel\_polynom = lambdify(var, bessel\_polynom, "numpy")

return bessel\_polynom, callable\_bessel\_polynom(x\_ans)

functions = [

sin(var),

1 / (var \*\* 2 + 1)

]

**Примеры и результат работы программы**

**Пример 1**

Введите 0 для ввода из консоли, 1 для ввода из файла, 2 для выбора функции:

1

Введите имя файла:

1.txt

Введите интересующее вас значение аргумента:

0.32

y\_i Δy\_i Δ^2y\_i Δ^3y\_i Δ^4y\_i

1.25 1.13 0.28 -0.04 -0.15

2.38 1.41 0.24 -0.19

3.79 1.65 0.05

5.44 1.7

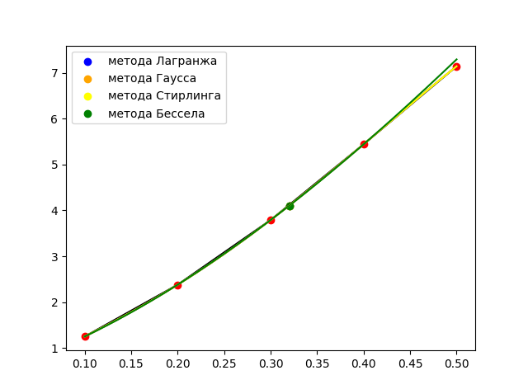
7.14

Результат метода Лагранжа: 4.1047

Результат метода Гаусса: 4.1047

Результат метода Стирлинга: 4.1047

Результат метода Бессела: 4.1021



**Пример 2**

Введите 0 для ввода из консоли, 1 для ввода из файла, 2 для выбора функции:

1

Введите имя файла:

2.txt

Введите интересующее вас значение аргумента:

0.28

Некорректные данные

**Пример 3**

Введите 0 для ввода из консоли, 1 для ввода из файла, 2 для выбора функции:

2

0. sin(x)

1. 1/(x\*\*2 + 1)

Введите индекс функции:

0

Введите концы интервала через пробел:

2 8

Введите количество промежутков:

10

Введите интересующее вас значение аргумента:

5.78

y\_i Δy\_i Δ^2y\_i Δ^3y\_i Δ^4y\_i Δ^5y\_i Δ^6y\_i Δ^7y\_i Δ^8y\_i Δ^9y\_i

0.9093 -0.452 -0.1958 0.2774 -0.0349 -0.1038 0.0594 0.019 -0.0336 0.0062

0.4573 -0.6478 0.0816 0.2425 -0.1388 -0.0444 0.0784 -0.0146 -0.0273

-0.1906 -0.5662 0.3241 0.1037 -0.1832 0.034 0.0639 -0.0419

-0.7568 -0.2422 0.4278 -0.0795 -0.1491 0.0979 0.0219

-0.999 0.1856 0.3483 -0.2286 -0.0512 0.1198

-0.8133 0.5339 0.1197 -0.2799 0.0686

-0.2794 0.6536 -0.1602 -0.2113

0.3742 0.4933 -0.3715

0.8675 0.1219

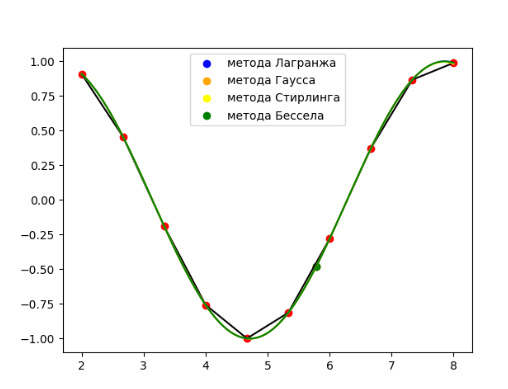
0.9894

Результат метода Лагранжа: -0.4822

Результат метода Гаусса: -0.4822

Результат метода Стирлинга: -0.4822

Результат метода Бесселя: -0.4822



**Выводы**

Интерполяция функций достаточно разительно отличается по сложности от аппроксимации, что лично для меня удивительно. И выражения для получаемых полиномов довольно сложно читать и воспринимать на глаз. Также не совсем понятно, чем объясняется такое количество способов построения полинома для интерполяции, ведь получаемый полином единственен.