«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия  
Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №5

Вариант 4

Студент:

*Ильин Н. С.*

*Р3210*

Преподаватель:

*Наумова Н. А.*

Санкт-Петербург, 2025 г.

**Оглавление**

[**Цель работы**: 3](#_Toc196202348)

[**1 Вычислительная реализация задачи:** 3](#_Toc196202349)

[**2. Программная реализация** 5](#_Toc196202350)

[**Блок схемы** 5](#_Toc196202351)

[**Листинг программы** 8](#_Toc196202352)

[**Примеры и результаты работы программы** 10](#_Toc196202353)

[**Выводы**: 12](#_Toc196202354)

# **Цель работы**:

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

# **1 Вычислительная реализация задачи:**

1. Выбрать таблицу 𝑦 = 𝑓(𝑥):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 1.4 | x | y | вариант | X1 | X2 |
| 1,05 | 0,1213 | 4 | 1,051 | 1,277 |
| 1,15 | 1,1316 |
| 1,25 | 2,1459 |
| 1,35 | 3,1565 |
| 1,45 | 4,1571 |
| 1,55 | 5,1819 |
| 1,65 | 6,1969 |

2. Построить таблицу конечных разностей:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | x­i | yi | Δyi | Δ2yi | Δ3yi | Δ4yi | Δ5yi | Δ6yi |
| 0 | 1,05 | 0,1213 | 1,0103 | 0,004 | -0,0077 | 0,0014 | 0,0391 | -0,1478 |
| 1 | 1,15 | 1,1316 | 1,0143 | -0,0037 | -0,0063 | 0,0405 | -0,1087 |  |
| 2 | 1,25 | 2,1459 | 1,0106 | -0,01 | 0,0342 | -0,0682 |  |  |
| 3 | 1,35 | 3,1565 | 1,0006 | 0,0242 | -0,034 |  |  |  |
| 4 | 1,45 | 4,1571 | 1,0248 | -0,0098 |  |  |  |  |
| 5 | 1,55 | 5,1819 | 1,015 |  |  |  |  |  |
| 6 | 1,65 | 6,1969 |  |  |  |  |  |  |

3. Вычислить значения функции для аргумента 𝑋1 используя интерполяционную формулу Ньютона

Воспользуемся формулой Ньютона для интерполирования вперед(первая формула), так как X1 = 0.502 лежит в левой половине отрезка

Для X1 = 0.502:

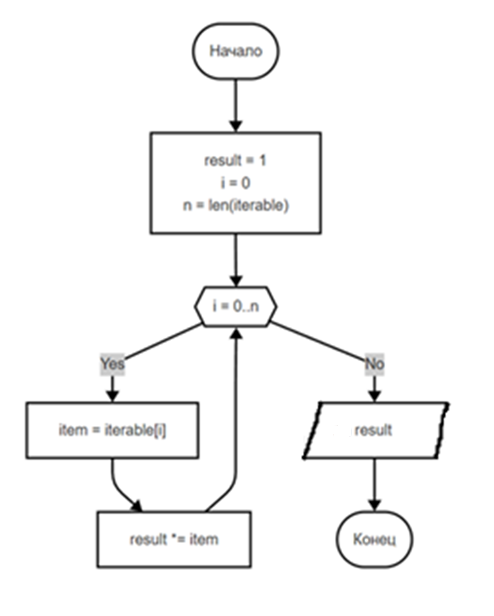
4. **Вычислить значения функции для аргумента 𝑋2**, используя интерполяционную формулу **Гаусса**:

Воспользуемся формулой Ньютона для интерполирования вперед(вторая формула), так как X2 = 1.277 лежит в левой половине отрезка

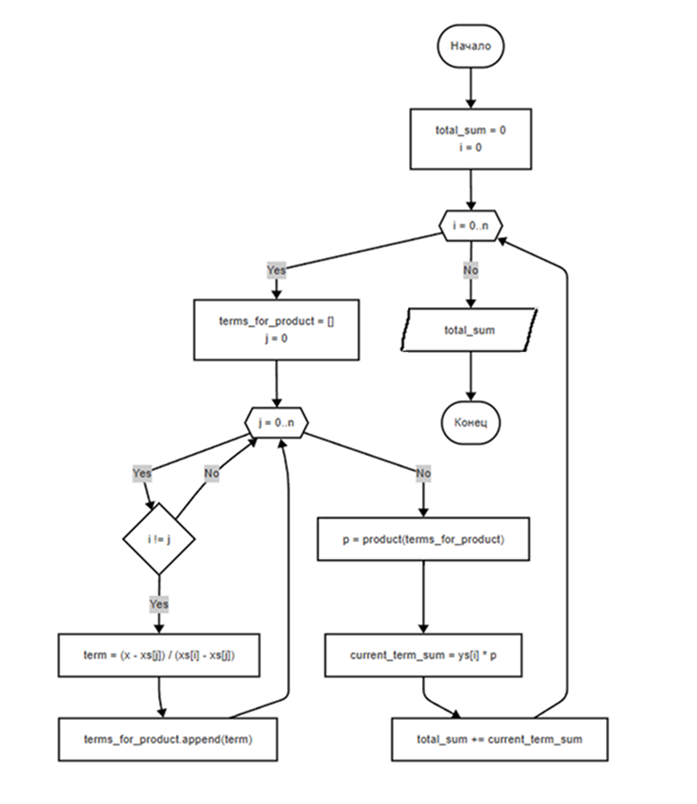
# **2. Программная реализация**

# **Блок схемы**

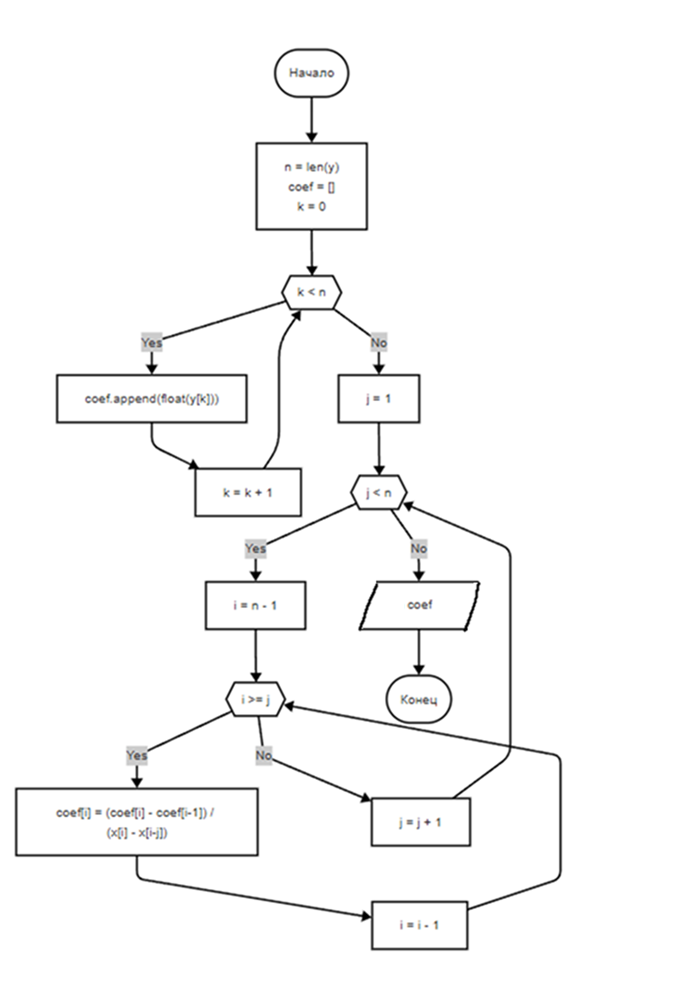
product



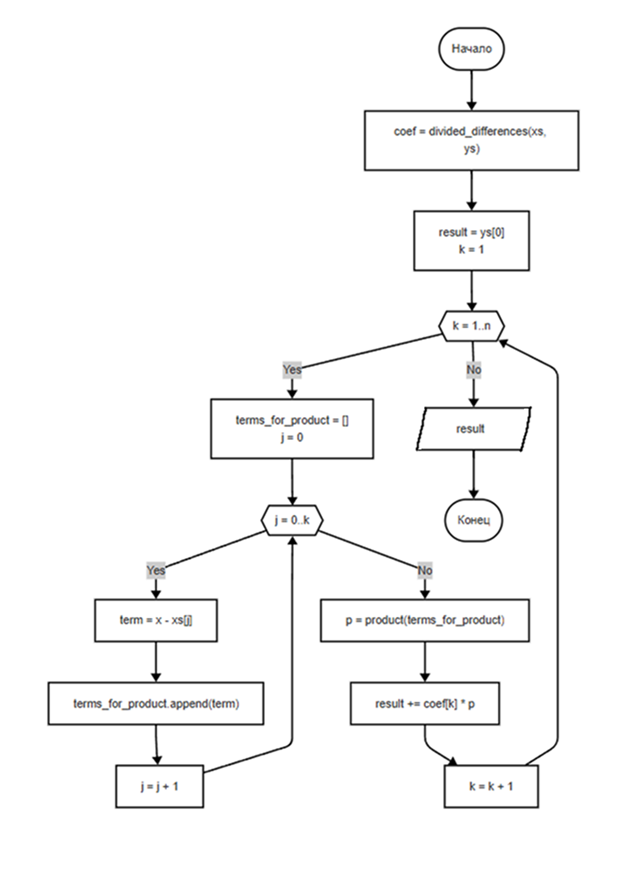
lagrange\_polynomial



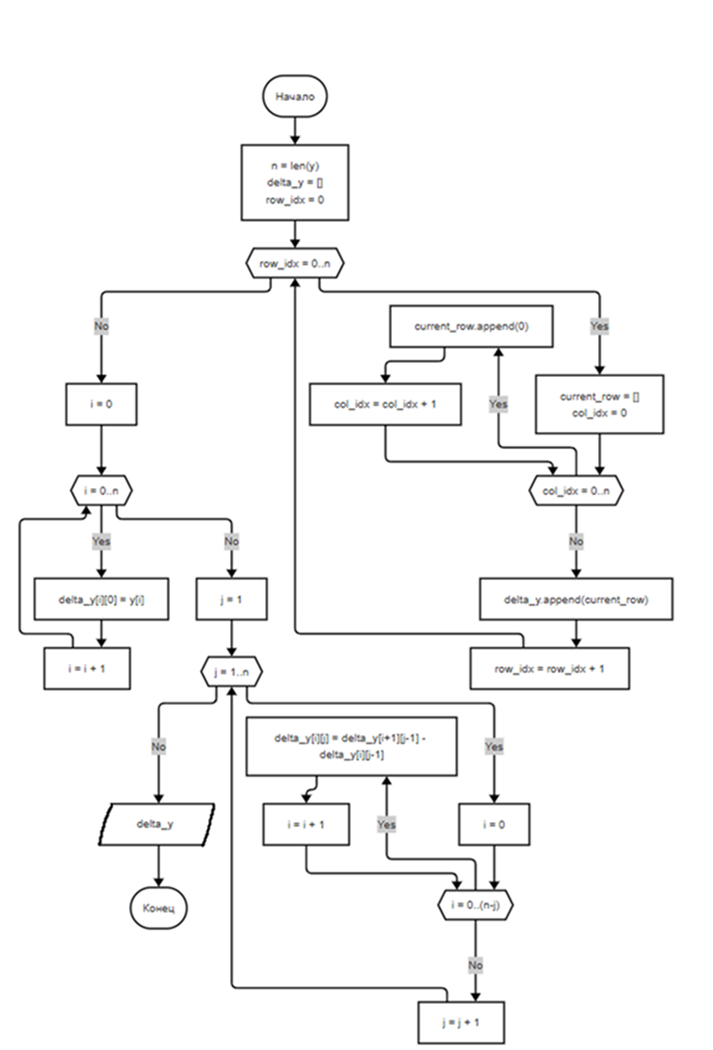
divided\_differences



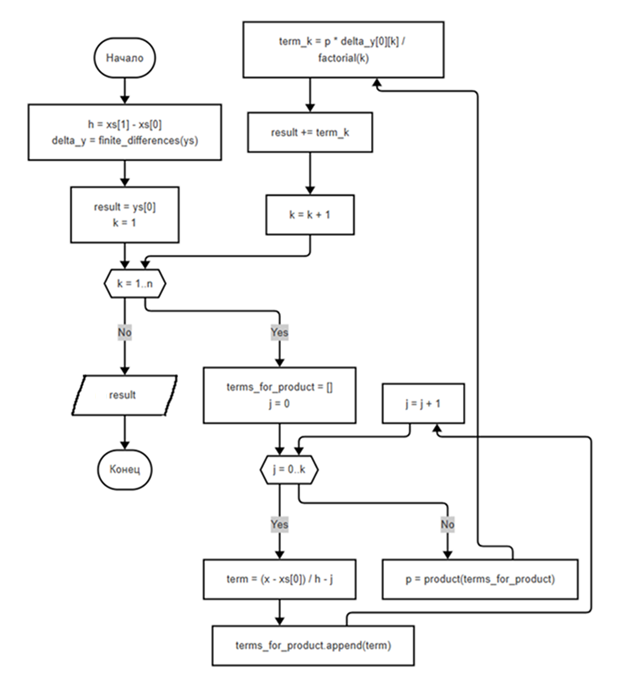
newton\_divided\_difference\_polynomial



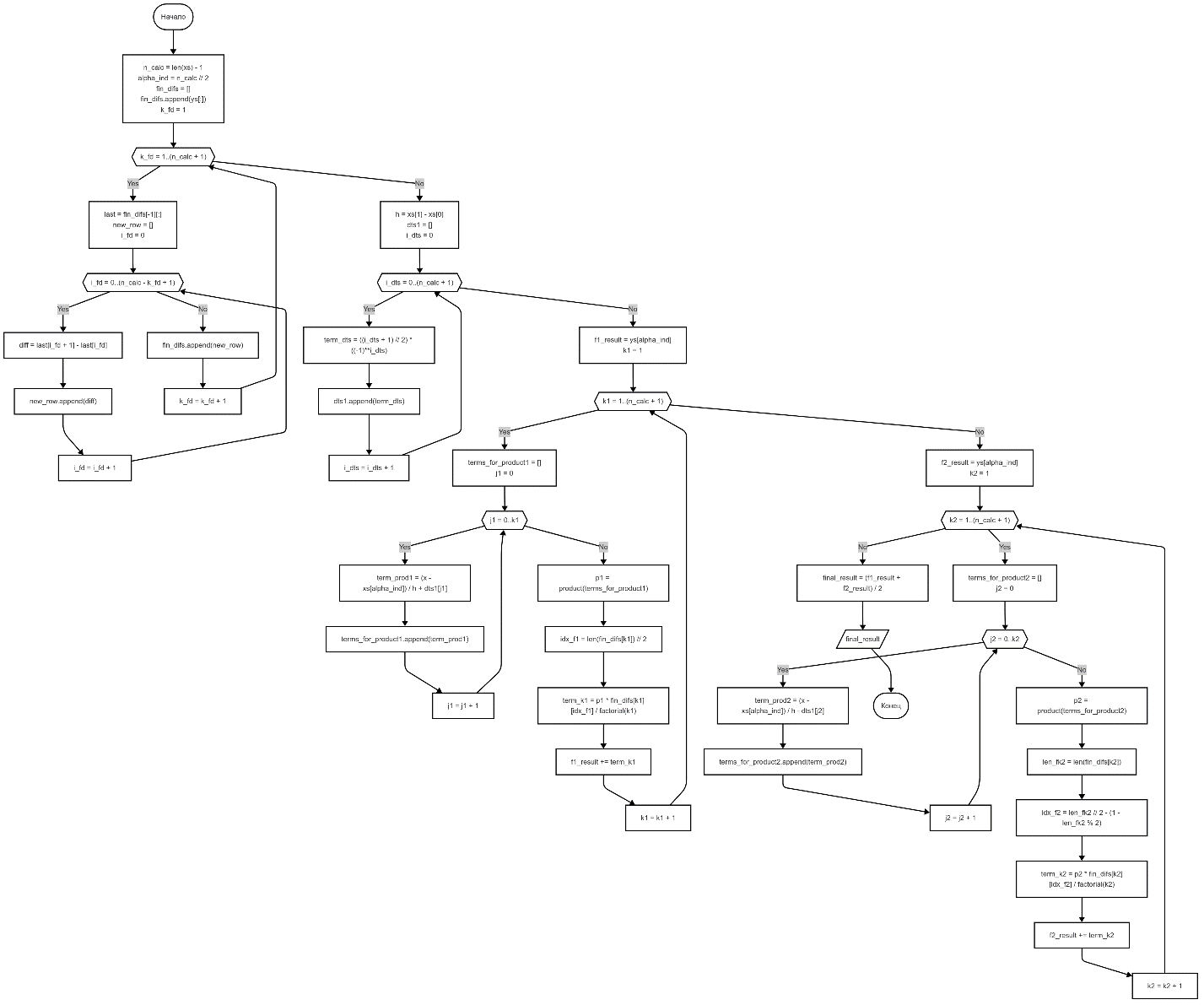
finite\_differences



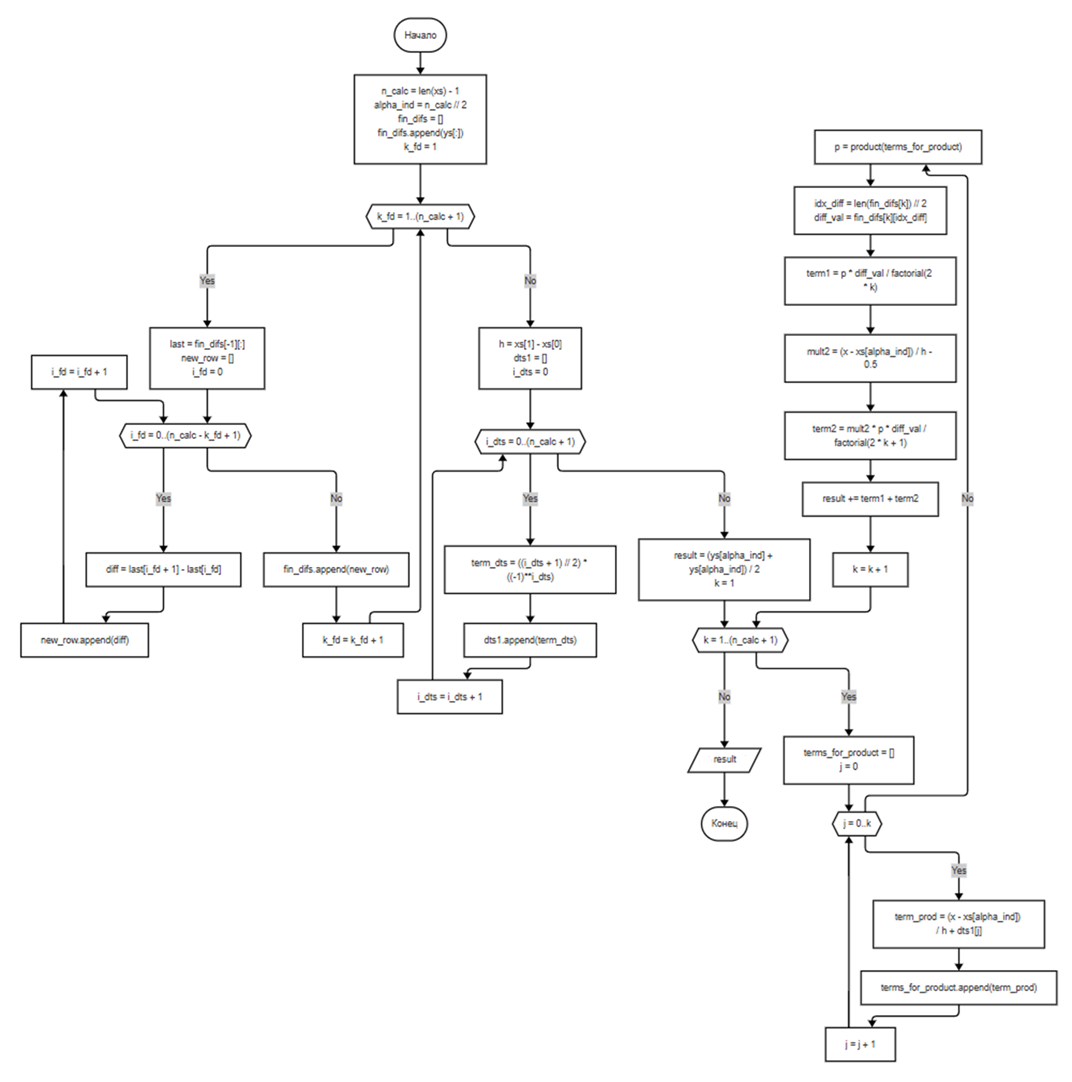
newton\_finite\_difference\_polynomial



stirling\_polynomial



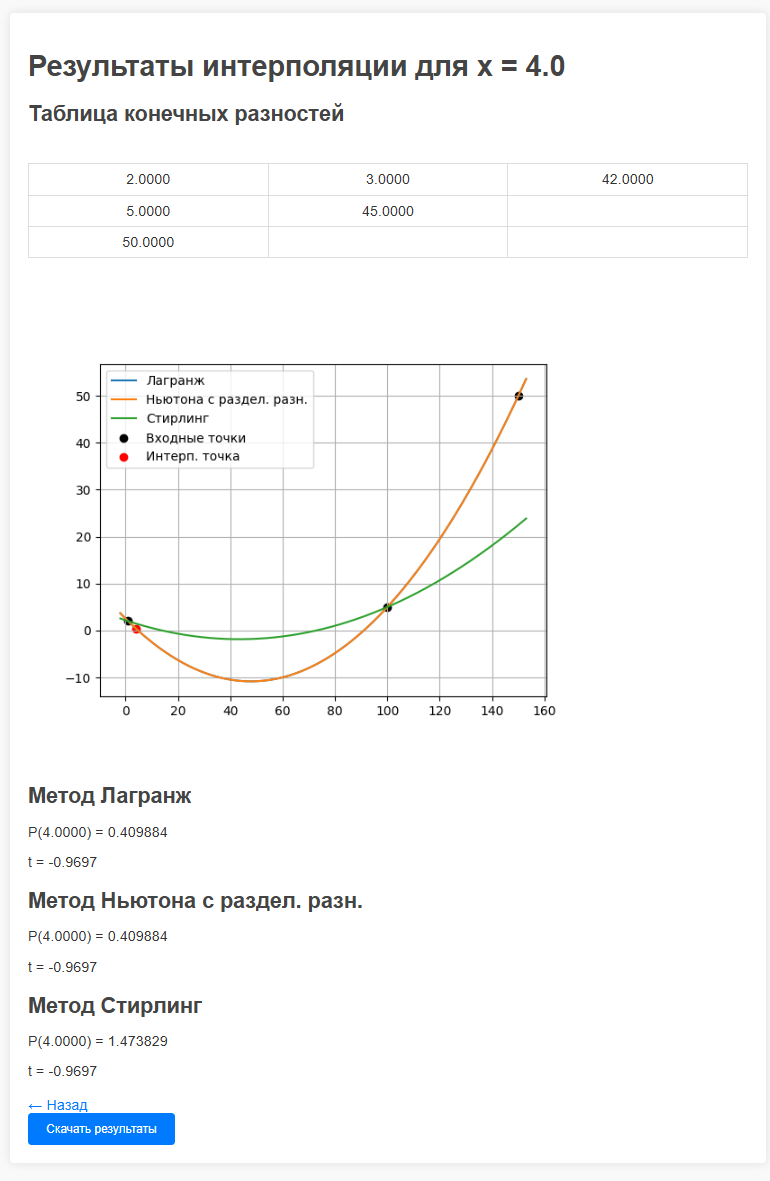
bessel\_polynomial



# **Листинг программы**

|  |
| --- |
| from copy import copy  from math import factorial  from matplotlib import pyplot as plt  def product(iterable):  result = 1  for item in iterable:  result \*= item  return result  def lagrange\_polynomial(xs, ys, n):  return lambda x: sum([  ys[i] \* product(  [(x - xs[j]) / (xs[i] - xs[j]) for j in range(n) if i != j]  )  for i in range(n)  ])  def divided\_differences(x, y):  n = len(y)  coef = [float(val) for val in y]  for j in range(1, n):  for i in range(n-1, j-1, -1):  coef[i] = (coef[i] - coef[i-1]) / (x[i] - x[i-j])  return coef  def newton\_divided\_difference\_polynomial(xs, ys, n):  coef = divided\_differences(xs, ys)  return lambda x: ys[0] + sum([  coef[k] \* product([x - xs[j] for j in range(k)]) for k in range(1, n)  ])  def finite\_differences(y):  n = len(y)  delta\_y = [[0 for \_ in range(n)] for \_ in range(n)]  for i in range(n):  delta\_y[i][0] = y[i]  for j in range(1, n):  for i in range(n-j):  delta\_y[i][j] = delta\_y[i+1][j-1] - delta\_y[i][j-1]  return delta\_y  def newton\_finite\_difference\_polynomial(xs, ys, n):  h = xs[1] - xs[0]  delta\_y = finite\_differences(ys)  return lambda x: ys[0] + sum([  product([(x - xs[0]) / h - j for j in range(k)]) \* delta\_y[k][0] / factorial(k)  for k in range(1, n)  ])  def gauss\_polynomial(xs, ys, n):  n = len(xs) - 1  alpha\_ind = n // 2  fin\_difs = [ys[:]]  for k in range(1, n + 1):  last = fin\_difs[-1][:]  fin\_difs.append([last[i + 1] - last[i] for i in range(n - k + 1)])  h = xs[1] - xs[0]  # generate step offsets sequence: 0, -1, 1, -2, 2, ... up to needed length  dts1 = [((i+1)//2) \* ((-1)\*\*i) for i in range(n+1)]  f1 = lambda x: ys[alpha\_ind] + sum([  product(  [(x - xs[alpha\_ind]) / h + dts1[j] for j in range(k)])  \* fin\_difs[k][len(fin\_difs[k]) // 2] / factorial(k)  for k in range(1, n + 1)])  f2 = lambda x: ys[alpha\_ind] + sum([  product(  [(x - xs[alpha\_ind]) / h - dts1[j] for j in range(k)])  \* fin\_difs[k][len(fin\_difs[k]) // 2 - (1 - len(fin\_difs[k]) % 2)] / factorial(k)  for k in range(1, n + 1)])  return lambda x: f1(x) if x > xs[alpha\_ind] else f2(x)  def stirling\_polynomial(xs, ys, n):  n = len(xs) - 1  alpha\_ind = n // 2  fin\_difs = [ys[:]]  for k in range(1, n + 1):  last = fin\_difs[-1][:]  fin\_difs.append([last[i + 1] - last[i] for i in range(n - k + 1)])  h = xs[1] - xs[0]  # dynamic step offsets for Stirling  dts1 = [((i+1)//2) \* ((-1)\*\*i) for i in range(n+1)]  f1 = lambda x: ys[alpha\_ind] + sum([  product(  [(x - xs[alpha\_ind]) / h + dts1[j] for j in range(k)])  \* fin\_difs[k][len(fin\_difs[k]) // 2] / factorial(k)  for k in range(1, n + 1)])  f2 = lambda x: ys[alpha\_ind] + sum([  product(  [(x - xs[alpha\_ind]) / h - dts1[j] for j in range(k)])  \* fin\_difs[k][len(fin\_difs[k]) // 2 - (1 - len(fin\_difs[k]) % 2)] / factorial(k)  for k in range(1, n + 1)])  return lambda x: (f1(x) + f2(x)) / 2  def bessel\_polynomial(xs, ys, n):  n = len(xs) - 1  alpha\_ind = n // 2  fin\_difs = [ys[:]]  for k in range(1, n + 1):  last = fin\_difs[-1][:]  fin\_difs.append([last[i + 1] - last[i] for i in range(n - k + 1)])  h = xs[1] - xs[0]  # dynamic step offsets for Bessel  dts1 = [((i+1)//2) \* ((-1)\*\*i) for i in range(n+1)]  return lambda x: (ys[alpha\_ind] + ys[alpha\_ind]) / 2 + sum([  product(  [(x - xs[alpha\_ind]) / h + dts1[j] for j in range(k)])  \* fin\_difs[k][len(fin\_difs[k]) // 2] / factorial(2 \* k) +  ((x - xs[alpha\_ind]) / h - 1 / 2) \*  product(  [(x - xs[alpha\_ind]) / h + dts1[j] for j in range(k)])  \* fin\_difs[k][len(fin\_difs[k]) // 2] / factorial(2 \* k + 1)  for k in range(1, n + 1)  ]) |

## **Примеры и результаты работы программы**



Github: [https://github.com/MrTheFall/computational\_math/tree/main/lab](https://github.com/MrTheFall/computational_math/tree/main/lab5)5

# **Выводы**:

В рамках лабораторной работы мною были изучены и применены на практике интерполяционные методы Ньютона и Гаусса для работы с предоставленными табличными данными. Эти подходы продемонстрировали возможность определения значений функции в точках, не указанных в исходном наборе данных, через аппроксимацию.

Разработанное ПО позволило получить оценки функции для целевых аргументов обоими методами. Сравнительный анализ выявил идентичность результатов при использовании разных алгоритмов, что подтверждает корректность их реализации.