«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 1

Студент:

Алхимовици А.

Р3210

Преподаватель:

Наумова Н. А.

Санкт-Петербург, 2025 г.

**Цель лабораторной 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 1.1 | X | y | вариант | X1 | X2 |
| 0.25 | 1.2557 | 1 | 0.251 | 0.402 |
| 0.30 | 2.1764 |
| 0.35 | 3.1218 |
| 0.40 | 4.0482 |
| 0.45 | 5.9875 |
| 0.50 | 6.9195 |
| 0.55 | 7.8359 |

2. Построить таблицу конечных разностей:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | **x** | **Δ0** | **Δ1** | **Δ2** | **Δ3** | **Δ4** | **Δ5** | **Δ6** |
| 0 | 0.25 | 1.2557 | 2.1764 | 3.1218 | 4.0482 | 5.9875 | 6.9195 | 7.8359 |
| 1 | 0.30 | 0.9207 | 0.9454 | 0.9264 | 1.9393 | 0.9320 | 0.9164 |  |
| 2 | 0.35 | 0.0247 | -0.0190 | 1.0129 | -1.0073 | -0.0156 |  |  |
| 3 | 0.40 | -0.0437 | 1.0319 | -2.0202 | 0.9917 |  |  |  |
| 4 | 0.45 | 1.0756 | -3.0521 | 3.0119 |  |  |  |  |
| 5 | 0.50 | -4.1277 | 6.0640 |  |  |  |  |  |
| 6 | 0.55 | 10.1917 |  |  |  |  |  |  |

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

Воспользуемся формулой Ньютона для интерполирования вперед(первая формула), так как X1 = 0.251 лежит в левой половине отрезка

Для X1 = 0.251:

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

Воспользуемся формулой Гауса для интерполирования вперед(вторая формула), так как X2 = 0.402 лежит в правой половине отрезка

*P(0.402)​=ymid​+T1​+T2​+T3​=4.111195+0.041276−0.008523−0.002966≈4.140982​*

# **2. Программная реализация**

# **Блок схемы**

Многочлен Ньютона с разделенными разностями

A diagram of a graph

AI-generated content may be incorrect.

Формула прямых конечных разностей

A diagram of a graph

AI-generated content may be incorrect.

Формула обратных конечных разностей

A diagram of a computer program

AI-generated content may be incorrect.

Интерполяция Лагранжа

A diagram of a graph

AI-generated content may be incorrect.

Схема Стирлинга

A diagram of a algorithm

AI-generated content may be incorrect.

Схема Бесселя

A diagram of a diagram

AI-generated content may be incorrect.

# **Листинг программы**

<https://github.com/senya-2011/Vu4Math/tree/main/lab5>

|  |
| --- |
| def stirling(pol\_xs, pol\_ys, x):  *"""Схема Стирлинга для равноотстоящих узлов"""* n = len(pol\_xs) - 1  mid = n // 2  diffs = finite\_diff\_table(pol\_ys)  h = pol\_xs[1] - pol\_xs[0]  t = (x - pol\_xs[mid]) / h  # dts: последовательность шагов для центральной схемы  dts = [(j // 2) \* ((-1) \*\* (j)) for j in range(n + 1)]  result = pol\_ys[mid]  for k in range(1, n + 1):  # выбор центрального элемента конечных разностей  idx = len(diffs[k]) // 2  term = 1.0  for j in range(k):  term \*= (t - dts[j])  result += term \* diffs[k][idx] / factorial(k)  return result   def bessel(pol\_xs, pol\_ys, x):  *"""Схема Бесселя для равноотстоящих узлов"""* n = len(pol\_xs) - 1  mid = n // 2  diffs = finite\_diff\_table(pol\_ys)  h = pol\_xs[1] - pol\_xs[0]  t = (x - pol\_xs[mid]) / h  # dts: последовательность шагов для центральной схемы  dts = [(j // 2) \* ((-1) \*\* (j)) for j in range(n + 1)]  result = pol\_ys[mid]  for k in range(1, n + 1):  idx = len(diffs[k]) // 2  # первая часть  term1 = 1.0  for j in range(k):  term1 \*= (t - dts[j])  # вторая часть (сдвиг на 0.5)  term2 = 1.0  for j in range(k):  term2 \*= (t + dts[j])  result += 0.5 \* (term1 + term2) \* diffs[k][idx] / factorial(k)  return result  def gauss\_forward(xs, ys, x):  h = xs[1] - xs[0]  n = len(xs)  mid = n // 2  table = finite\_diff\_table(ys)  t = (x - xs[mid]) / h  result = ys[mid]  p = 1.0   for k in range(1, mid + 1):  if k % 2 == 1:  idx = mid - (k // 2)  else:  idx = mid - (k // 2) - 1   if not (0 <= idx < len(table[k])):  raise IndexError(f"gauss\_forward: k={k}, idx={idx}, len(table[{k}])={len(table[k])}")   p \*= (t - (k - 1) / 2)  result += p \* table[k][idx] / factorial(k)   return result   def gauss\_backward(xs, ys, x):  h = xs[1] - xs[0]  n = len(xs)  mid = n // 2  table = finite\_diff\_table(ys)  t = (x - xs[mid]) / h  result = ys[mid]  p = 1.0   for k in range(1, mid + 1):  if k % 2 == 1:  idx = mid - (k // 2)  else:  idx = mid - (k // 2) - 1   if not (0 <= idx < len(table[k])):  raise IndexError(f"gauss\_backward: k={k}, idx={idx}, len(table[{k}])={len(table[k])}")   p \*= (t + (k - 1) / 2)  result += p \* table[k][idx] / factorial(k)  return result  def lagrange(xs, ys, x):  *"""  Интерполяция Лагранжа.  """* n = len(xs)  total = 0.0  for i in range(n):  term = ys[i]  for j in range(n):  if j != i:  term \*= (x - xs[j]) / (xs[i] - xs[j])  total += term  return total  # Многочлен Ньютона с разделёнными разностями def newton\_divided(xs, ys, x):  *"""  Интерполяция Ньютона (разделённые разности).  """* n = len(xs)  coeff = list(ys)  for j in range(1, n):  for i in range(n - 1, j - 1, -1):  coeff[i] = (coeff[i] - coeff[i - 1]) / (xs[i] - xs[i - j])  result = coeff[0]  prod = 1.0  for i in range(1, n):  prod \*= (x - xs[i - 1])  result += coeff[i] \* prod  return result  # Многочлен Ньютона с конечными разностями (прямая формула) def newton\_forward(xs, ys, x):  *"""Формула прямых конечных разностей. Только для равноотстоящих узлов."""* h = xs[1] - xs[0]  table = finite\_diff\_table(ys)  t = (x - xs[0]) / h  result = ys[0]  for k in range(1, len(xs)):  coeff = 1.0  for j in range(k):  coeff \*= (t - j)  result += coeff \* table[k][0] / factorial(k)  return result  # Многочлен Ньютона с конечными разностями (обратная формула) def newton\_backward(xs, ys, x):  *"""Формула обратных конечных разностей. Только для равноотстоящих узлов."""* h = xs[1] - xs[0]  table = finite\_diff\_table(ys)  n = len(xs)  t = (x - xs[-1]) / h  result = ys[-1]  for k in range(1, n):  coeff = 1.0  for j in range(k):  coeff \*= (t + j)  result += coeff \* table[k][n - k - 1] / factorial(k)  return result |

## 

## **Примеры и результаты работы программы**

A screenshot of a graph

AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect.

# **Вывод**:

В ходе выполнения лабораторной работы я изучил и практично применил интерполяционные методы Ньютона и Гаусса для обработки табличных данных. Эти методы позволили аппроксимировать значения функции в точках, отсутствующих в исходном наборе.

Разработанная программа обеспечила вычисление значений функции в заданных точках с использованием обоих методов. Проведённый сравнительный анализ показал совпадение результатов, что свидетельствует о правильной реализации и работоспособности выбранных алгоритмов.