МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Отчёт о выполнении лабораторной работы №2

Вычислительная математика

Выполнил:

Манулин Павел

Б03-107

Долгопрудный
2023

1 Постановка задачи

При исследовании некоторой химической реакции через каждые 2 секунды измерялась температура смеси. Результаты измерений представлены в таблице.

t,с	5	7	9	11	13	15	17	19	21
T, К	296	520	744	982	1248	1570	2256	2256	2256

Таблица 1: Результаты измерений

С помощью интерполяции найти t^*, при котором производная $\frac{dT}{dt}$ максимальна.
Использовать интерполяционные полиномы в форме Ньютона и Лагранжа.

2 Теоретические сведения

2.1 Алгебраическая интерполяция

Если в качестве линейно независимых функций выбраны степенные функции $\varphi_k(x) = x^k, k = \overrightarrow{1,n}$, то интерполяция называется алгебраической. Система $\{x^k\}_{k=0}^n$ является системой Чебышева, т.к. согласно «основной теореме алгебры» алгебраический многочлен с действительными коэффициентами степени не выше n не может иметь более чем n нулей. Будем представлять алгебраический интерполяционный многочлен в виде $P_n(x) = a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n = 0$, $a_0 \neq 0$.

2.2 Задача алгебраической интерполяции

Задача алгебраической интерполяции – это задача построения алгебраического многочлена степени не выше n , который в заданных попарно различных точках $x_j \neq x_k$, если $j \neq k, \{x_j\}_{j=0}^n$ принимал бы значения $\{f_j\}_{j=0}^n$. В дальнейшем вместо термина алгебраический интерполяционный многочлен будем использовать термин интерполяционный многочлен. Приведём несколько форм записи этого многочлена.

2.3 Интерполяционный многочлен в форме Ньютона

Эта форма записи интерполяционного многочлена во многом аналогичена той, что уже была приведена, если считать, что $\{x_j\}_{j=i}^{i+n}$ - узлы интерполяции, а $\{f_j\}_{j=i}^{i+n}$ - значения функции в узлах интерполяции:

$$P_n(x) = \sum_{k=0}^{n} b_k \prod_{j=i}^{i+k-1} (x - x_j) = b_0 + b_1(x - x_i) + \cdots + b_n(x - x_i)(x - x_{i+1}) \cdots (x - x_{i+n-1})$$

Коэффициенты $\{b_i\}_{i=0}^n$ являются разделёнными разностями i-го порядка и определяются как

$$\begin{aligned}
&b_0 = f_1,\\
&b_1 = f(x_i, x_{i+1}) = (f_{i+1} - f_i)/(x_{i+1} - x_i),\\
&b_2 = f(x_i, x_{i+1}, x_{i+2}) = (f(x_{i+1}, x_{i+2}) - f(x_i, x_{i+1}))/(x_{i+2} - x_i), \cdots,\\
&\quad \cdots\\
&b_n = f(x_i, \cdots, x_{i+n}) = (f(x_{i+1}, \cdots, x_{i+n}) - f(x_i, \cdots, x_{i+n-1}))/(x_{i+n} - x_i)
\end{aligned}$$

2.4 Остаточный многочлен. Погрешность интерполяции

Пусть приближаемая функция $f(x)$ достаточно гладкая и имеет $n+1$ непрерывную производную на отрезке $x \in [x_o, x_n]$. Тогда для любого значения $x \in [x_o, x_n]$ справедливо равенство $f(x) = P_n(x) + R_n(x)$, где $R_n(x)$ остаточный многочлен, который можно представить в виде $R_n(x) = \frac{f_x^{(n+1)}(\xi)}{(n+1)!}(x - x_0)\cdots(x - x_n)$, а ξ принадлежит интервалу (x_0, x_n). Справедлива оценка

$$|R_n(x) <= |\frac{M_{n+1}}{(n+1)!}(x - x_0)...(x - x_n)|, M_{n+1} = max|f_x^{n+1}(x)|$$

2.5 Интерполяционный многочлен в форме Лагранжа

Для узлов $\{x_j\}_{j=0}^n$ и значений функции $\{f_j\}_{j=0}^n$ интерполяционный многочлен Лагранжа имеет вид $P_n(x) = \sum_{i=0}^n f_i \prod_{j=0, j\neq i}^{n} (x - x_j)/(x_i - x_j)$ или более подробно

$$P_n(x) = f_o\frac{(x - x_1)\cdots(x - x_n)}{(x_0 - x_1)\cdots(x_0 - x_n)} + f_1\frac{(x - x_0)(x - x_2)\cdots(x - x_n)}{(x_1 - x_0)(x_1 - x_2)\cdots(x_1 - x_n)} + \cdots$$
$$\cdots + f_n\frac{(x - x_0)(x - x_1)\cdots(x - x_{n-1})}{(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1})}.$$

3 Ход работы

3.1 Метод Ньютона

Вычислим разделённые разности

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

time = np.array([5, 7, 9, 11, 13, 15, 17, 19, 21])
T = np.array([296, 520, 744, 982, 1248, 1570, 2256, 2256, 2256])

'''Вычислим первые разделённые разности, чтобы понять,
между какими точками следует искать t* '''

b1 = np.array([(T[i + 1] - T[i]) / (time[i + 1] - time[i]) for i in range(
    len(time) - 1)])
b1
```

```
array([112., 112., 119., 133., 161., 343.,   0.,   0.])
```

```
'''Видим, что максимальная скачок наблюдается между 15 и 17 секундами
значит между этими точками будет лежать t* в которой производная максимальна.
Построим интерполяционные многочлен в форме Ньютона
по точками 11, 13, 15, 17, 19, 21'''

b1 = b1[3:]
print(f'Первые разделёные разности: {b1}')
b2 = np.array([(b1[i + 1] - b1[i]) / (time[6] - time[4]) for i in range(4)])
print(f'Вторые разделёные разности: {b2}')
```

```
Первые разделёные разности: [133. 161. 343.   0.   0.]
Вторые разделёные разности: [  7.    45.5  -85.75   0.  ]
```

```
b1
```

```
array([112., 112., 119., 133., 161., 343.,   0.,   0.])
```

```
b3 = np.array([(b2[i + 1] - b2[i]) / (time[6] - time[3]) for i in range(3)])
print(f'Третьи разделёные разности: {b3}')
b4 = np.array([(b3[i + 1] - b3[i]) / (time[6] - time[2]) for i in range(2)])
print(f'Четвёртые разделённые разности: {b4}')
b5 = (b4[1] - b4[0]) / (time[6] - time[1])
print(f'Пятая   разделённая разность: {b5}')

def polinom4(x):
    # Многочлен 4-й степени по точкам 13, 15, 17, 19
    return 1248 + 161 * (x - 13) + 45.5 * (x - 13) * (x - 15) - \
           21.875 * (x - 13) * (x - 15) * (x - 17) + \
           4.52 * (x - 13) * (x - 15) * (x - 17) * (x - 19)

def derivative(x):
    # Производная многочлена
    return 91 * x - 1113 - \
    21.875 * (2 * x * x - 34 * x - 28 * x + 476 + x * x - 28 * x + 195) + \
    4.52 * (2 * x * x * x - 72 * x * x + 646 * x - 28 * x * x + 1008 * x - \
    9044 + 2 * x * x * x - 56 * x * x + 390 * x - 36 * x * x + 1008 * x - 7020)
```

```
Третьи разделёные разности: [  6.41666667 -21.875       14.29166667]
Четвёртые разделённые разности: [-3.53645833  4.52083333]
Пятая   разделённая разность: 0.8057291666666666
```

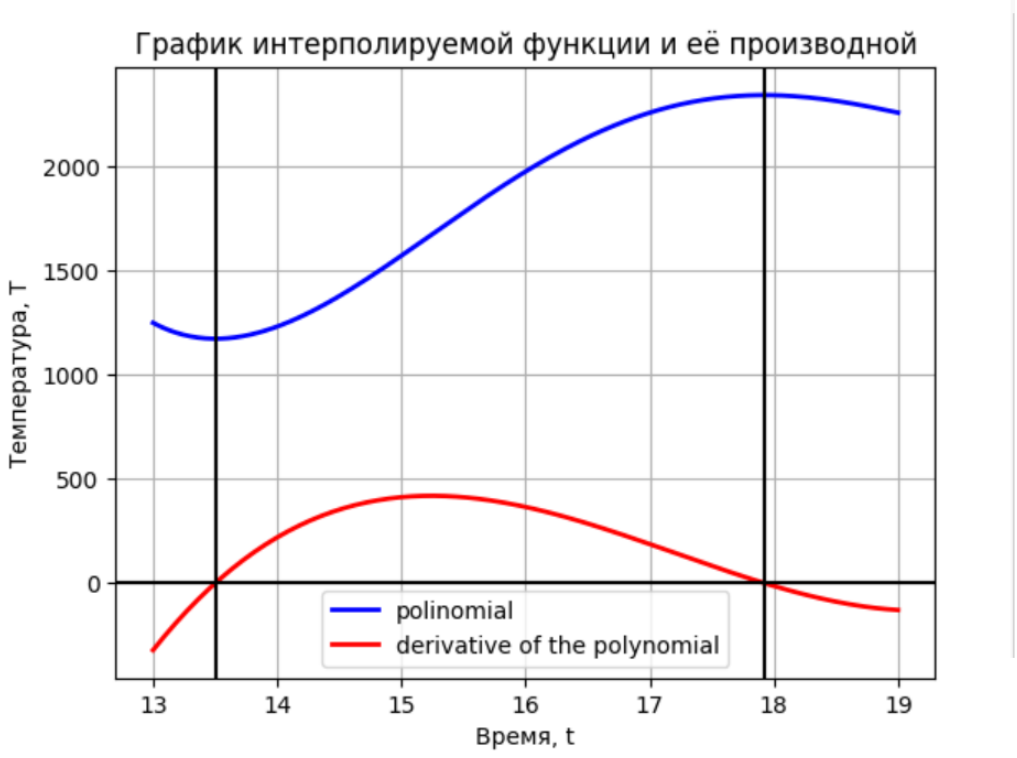
x_i	$f(x_i, x_{i+1})$	$f(x_i, x_{i+1}, x_{i+2})$	$f(x_i, ..., x_{i+3})$	$f(x_i, ..., x_{i+4})$	$f(x_i, ..., x_{i+5})$
11	982.000				
		7.000			
13	1248.000		6.417		
		45.500		-3.536	
15	1570.000		-21.875		0.805
		-85.750		4.521	
17	2256.000		14.291		
		0.000			
19	2256.000				

Построим график нашей приближённой методом Ньютона функции и её производной.

```
first_nulls = np.array([])
second_nulls = np.array([])
for i in np.linspace(13, 14, num=2000):
    if np.absolute(derivative(i)) <= 2e-1:
        first_nulls = np.append(first_nulls, i)
for i in np.linspace(17, 19, num=2000):
    if np.absolute(derivative(i)) <= 2e-2:
        second_nulls = np.append(second_nulls, i)

plt.plot(np.linspace(13, 19, num=200), polinom4(np.linspace(13, 19, num=200)),
         linewidth=2, color='b', label='polinomial')
plt.plot(np.linspace(13, 19, num=200), derivative(np.linspace(13, 19, num=200)),
         linewidth=2, color='r', label='derivative of the polynomial')

plt.axvline(x = first_nulls[0], color = 'black')
plt.axvline(x = second_nulls[0], color = 'black')
plt.axhline(y=0, color='black')
plt.title('График интерполируемой функции и её производной')
plt.xlabel('Время, t')
plt.ylabel('Температура, T')
plt.grid(True)
plt.legend()
plt.show()
```



Видим, что максимум производной достигается между 15-й и 16-й секундой. Вычислим значение аргумента, при котором достигается максимум. Затем вычислим погрешность метода

```
# Найдем точку, в которой производная максимальна
for i in np.linspace(13, 19, num=200):
    if derivative(i) == derivative(np.linspace(13, 19, num=200)).max():
        founded_time = i
founded_time
```

```
15.231155778894472
```

```
#Погрешность метода
sigma = (b5 / (5 * 4 * 3 * 2)) * (founded_time - time[4]) * \
 (founded_time - time[5]) * (founded_time - time[6]) * \
  (founded_time - time[7]) * (founded_time - time[8])
print(sigma)
```

```
-0.1331769527933134
```

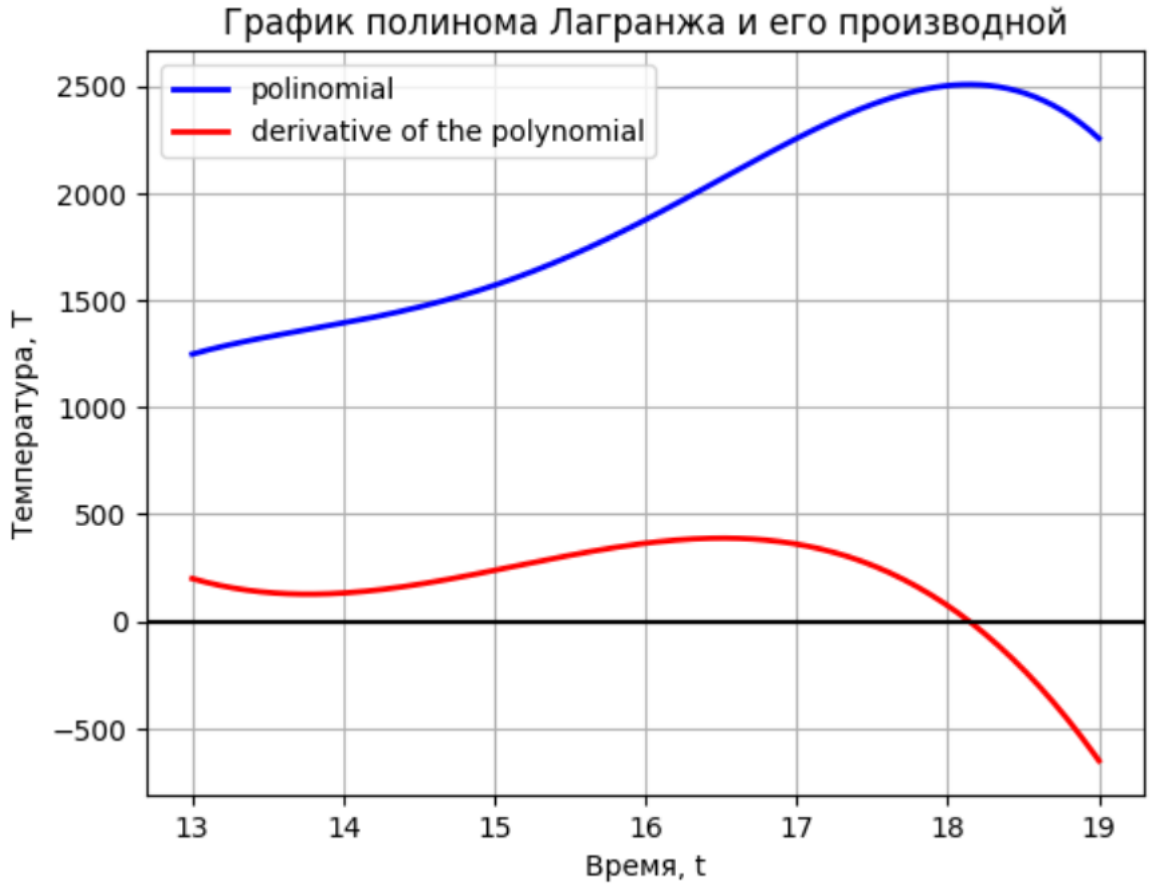
3.2 Метод Лагранжа

Построим полином Лагранжа

```
def lagrange_polynomial(x):
    return 982 * ((x - 13) * (x - 15) * (x - 17) * (x - 19)) / \
     ((11 - 13) * (11 - 15) * (11 - 17) * (11 - 19) * (11 - 21)) + \
    1248 * ((x - 11) * (x - 15) * (x - 17) * (x - 19)) / \
     ((13 - 11) * (13 - 15) * (13 - 17) * (13 - 19)) + \
    1570 * ((x - 11) * (x - 13) * (x - 17) * (x - 19)) / \
     ((15 - 11) * (15 - 13) * (15 - 17) * (15 - 19)) + \
    2256 * ((x - 11) * (x - 13) * (x - 15) * (x - 19)) / \
     ((17 - 11) * (17 - 13) * (17 - 15) * (17 - 19)) + \
    2256 * ((x - 11) * (x - 13) * (x - 15) * (x - 17)) / \
     ((19 - 11) * (19 - 13) * (19 - 15) * (19 - 17))

def lagrange_derivative(x):
    return (-1/1920)*(491 * (x - 15) * (x - 17) * (x - 19) + \
    491 * (x - 13) * (x - 17) * (x - 19) + 491 * (x - 13) * (x - 15) * \
     (x - 19) + 491 * (x - 13) * (x - 15) * (x - 17)) - 13 * (x - 15) * \
      (x - 17) * (x - 19) - 13 * (x - 11) * (x - 17) * (x - 19) - 13 * \
       (x - 11) * (x - 15) * (x - 19) - 13 * (x - 11) * (x - 15) * (x - 17) + \
       (1 / 32) * (785 * (x - 13) * (x - 17) * (x - 19) + 785 * (x - 11) * \
        (x - 17) * (x - 19) + 785 * (x - 11) * (x - 13) * (x - 19) + 785 * \
         (x - 11) * (x - 13) * (x - 17)) - (1 / 2) * (47 * (x - 13) * \
          (x - 15) * (x - 19) + 47 * (x - 11) * (x - 15) * (x - 19) + 47 * \
           (x - 11) * (x - 13) * (x - 19) + 47 * (x - 11) * (x - 13) * \
            (x - 15)) + (1 / 8) * (47 * (x - 13) * (x - 15) * (x - 17) + 47 * \
             (x - 11) * (x - 15) * (x - 17) + 47 * (x - 11) * (x - 13) * \
              (x - 17) + 47 * (x - 11) * (x - 13) * (x - 15))
```

```
plt.plot(np.linspace(13, 19, num=200), lagrange_polynomial(
    np.linspace(13, 19, num=200)),
         linewidth=2, color='b', label='polinomial')
plt.plot(np.linspace(13, 19, num=200), lagrange_derivative(
    np.linspace(13, 19, num=200)),
         linewidth=2, color='r', label='derivative of the polynomial')
plt.axhline(y=0, color='black')
plt.title('График полинома Лагранжа и его производной')
plt.xlabel('Время, t')
plt.ylabel('Температура, T')
plt.grid(True)
plt.legend()
plt.show()
```



Видим, что максимум производной достигается между 16-й и 17-й секундой. Теперь уточним эту точку

```
for i in np.linspace(13, 19, num=200):
    if lagrange_derivative(i) == lagrange_derivative(
                                np.linspace(13, 19, num=200)).max():
        print(i)
```

```
16.49748743718593
```

4 Обсуждение результатов и выводы

Итак, как мы выяснили в начале, максимум производной dT/dt лежит в интервале между 15-й и 17-й секундой. Для уточнения t_*, в которой достигается максимум производной, мы воспользовались двумя методами интерполяции: методом Ньютона и методом Лагранжа. Проведя построение интерполяционного полинома каждым из методов, а затем вычислив первую производную, мы нашли точку, в которой достигается максимум производной, то есть искомое t_*. Однако для каждого метода мы получили различные значения t_*:

$t_{Ньютон} = 15.231$, $t_{Лагранж} = 16.497$

Из чего следует вывод, что для этой задачи и имеющихся у нас данных эти методы не очень хороши для уточнения точки, в которой достигается максимум производной ис-

ходной функции. Возможно, нужны дополнительные данные о производной, к примеру, значения производной в некоторых узлах интерполяции, или более плотное распределение узлов. Всё это может помочь более точно и однозначно провести построение интерполяционного полинома, и уточнение точки, в которой его производная максимальна.