Інтерполяційний многочлен Лагранжа

Підготувала:
Студентка ПМІ-23
Шувар Софія

Мета: побудувати інтерполяційний многочлен Лагранжа та за його допомогою знайти наближене значення функції у вказаній точці. Вивести його графік. Продемонструвати роботу програми на конкретному прикладі.

Завдання:

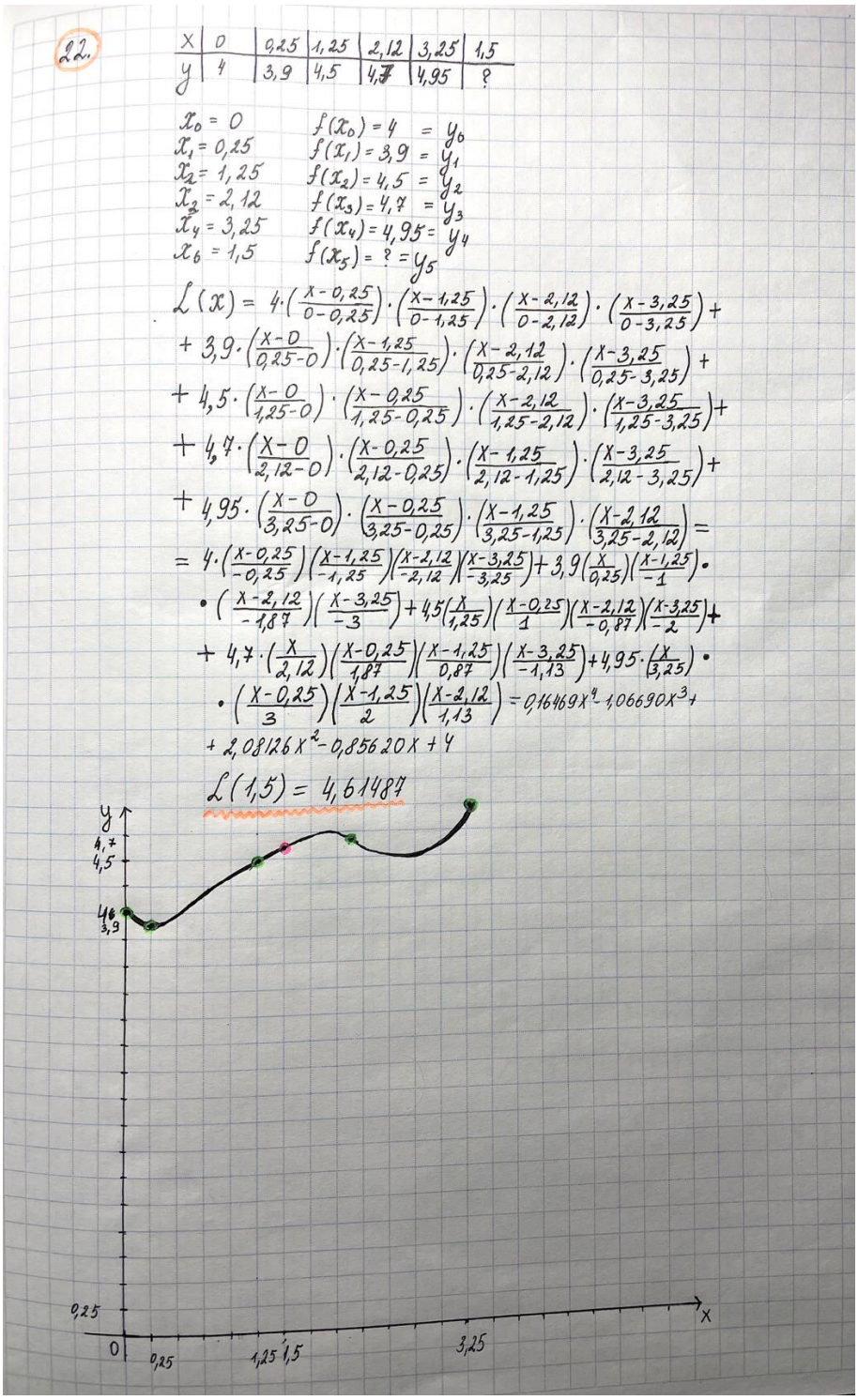
22.

x	0	0.25	1.25	2.12	3.25	1.5
y	4	3.9	4.5	4.7	4.95	?

Хід роботи:

1. Побудувати інтерполяційний многочлен Лагранжа та за його допомогою знайти наближене значення функції у вказаній точці.
2. Реалізувans алгоритм побудови інтерполяційного многочлена Лагранжа на довільний мові програмування.
3. Продемонструвати результати роботи алгоритму на конкретному прикладі.

Розв’язок:



Реалізація алгоритму:

Для реалізації алгоритму я обрала мову програмування Python. Алгоритм написаний з використанням бібліотеки numpy та matplotlib.

Функція user_input :

Аргументи: -

Функція повертає масив points точок (x_i, y_i) введені користувачем та точку, у якій повинне бути знайдене наближене значення функції теж введене користувачем.

```
5   def user_input():
6       n = int(input('Enter number of data points: '))
7       points = np.zeros((n, 2))
8       print('Enter data for x and y: ')
9       for i in range(n):
10          points[i][0] = float(input('x['+str(i)+'] = '))
11          points[i][1] = float(input('y['+str(i)+'] = '))
12      xx = float(input('Enter interpolation point: '))
13      return points, xx
14
```

Функція lagrange_interpolation (реалізація алгоритму):

Аргументи: points – масив точок (x_i, y_i), xx – точка, у якій повинне бути знайдене наближене значення функції.

Алгоритм побудови інтерполяційного многочлена Лагранжа складається з наступних частин:

Рядок 17 – присвоюємо змінній sum значення 0;

Рядок 18 – розпочинаємо цикл довжини кількості елементів масиву x;

Рядок 19, 20 – присвоюємо змінним cur_x, cur_y – поточні значення (x_i, y_i) відповідно;

Рядок 21 – розпочинаємо цикл довжини кількості елементів масиву x;

Рядок 22 – обчислюємо нове значення cur_y використовуючи формулу:

$$P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}.$$

Рядок 23 – знаходимо значення полінома в точці xx, за формулою:

$$P(x) = \sum_{j=1}^{n} P_j(x),$$

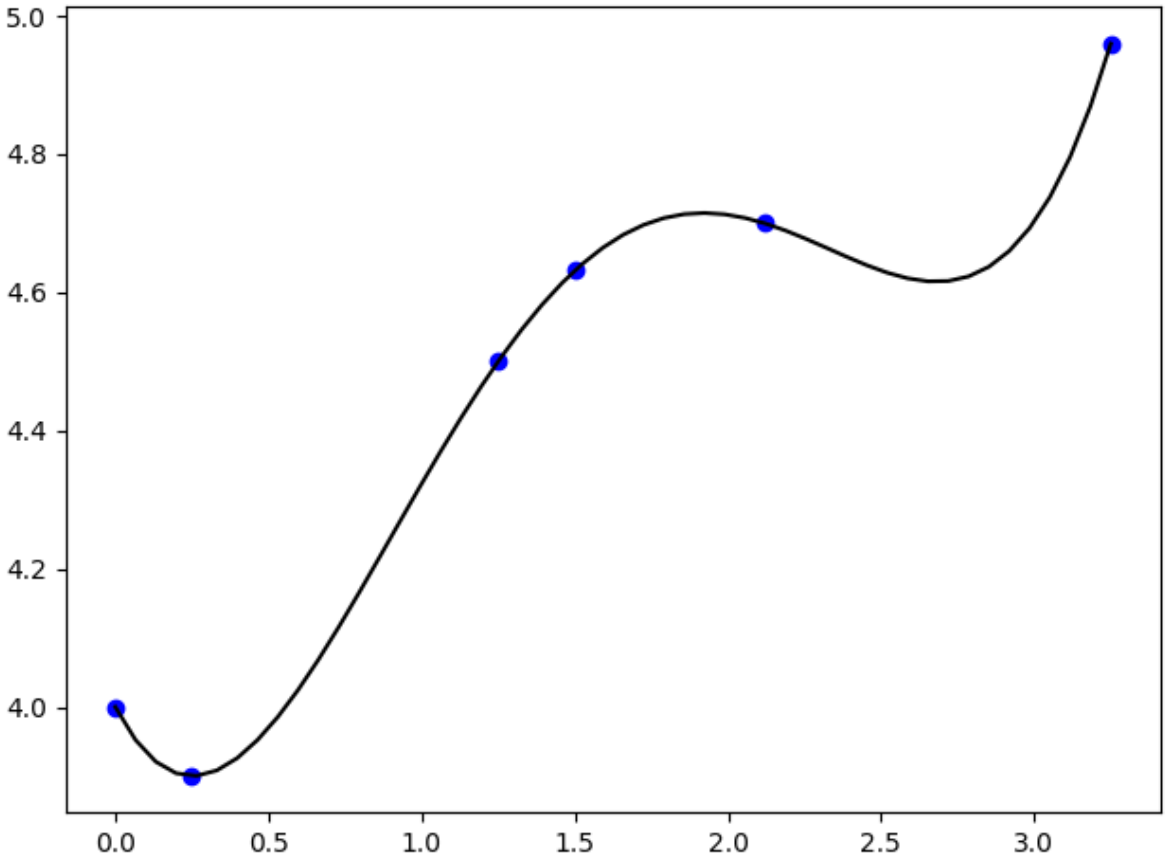
```
15
16  def lagrange_interpolation(points, xx):
17      sum = 0
18      for i in range(len(points)):
19          cur_y = points[i][1]
20          cur_x = points[i][0]
21          for j in range(len(points)):
22              cur_y = cur_y*(xx - points[j][0])/(cur_x-points[j][0]) if i != j else cur_y
23          sum += cur_y
24      return sum
```

Функція demonstration(вивід усіх отриманих результатів):
Рядки[33, 34] - демонстрація отриманих результатів на графіку.

```
27  def demonstration():
28      points, xx = user_input()
29      yy = lagrange_interpolation(points, xx)
30      print('x['+str(len(points))+'] = ' + str(yy))
31      points = np.vstack((points, [xx, yy]))
32      X = np.linspace(np.min(points[:, 0]), np.max(points[:, 0]))
33      plt.plot(points[:, 0], points[:, 1], 'bo', X, [lagrange_interpolation(points, i) for i in X], 'k')
34      plt.show()
35
36
37  if __name__ == "__main__":
38      demonstration()
39
```

Результати роботи програми на конкретному прикладі:

```
Enter number of data points: 5
Enter data for x and y:
x[0] = 0
y[0] = 4
x[1] = 0.25
y[1] = 3.9
x[2] = 1.25
y[2] = 4.5
x[3] = 2.12
y[3] = 4.7
x[4] = 3.25
y[4] = 4.95
Enter interpolation point: 1.5
x[5] = 4.6314870642745225
```



Висновок

Виконуючи дану практичну роботу, я навчилась будувати інтерполяційні многочлени Лагранжа та реалізувала алгоритм знаходження такого многочлена на мові програмування Python з використанням можливостей бібліотек numpy та matplotlib.