6. Aproksimacija funkcija

1. Date su tačke:

```
x = np.array([0.0000,    1.2500,    2.5000,    3.7500,    5.0000])
fX = np.array([1.7499,    0.9830,    1.2554,    3.0802,    2.3664])
```

2. Nacrtati poznate tačke:

```
plt.scatter(x, fX, c='black')
```

3. Naći polinom 3. stepena koji aproksimira funkciju kroz poznate tačke

```
p = np.polyfit(x, fX, 3)
```

Rezultat:

```
p =
   -0.1527    1.2208   -2.1641    1.8157
```

4. Naći ukupnu kvadratnu grešku po svim tačkama:

```
pX = np.polyval(p, x)
sumSquaredErr = np.sum((fX - pX) ** 2)
```

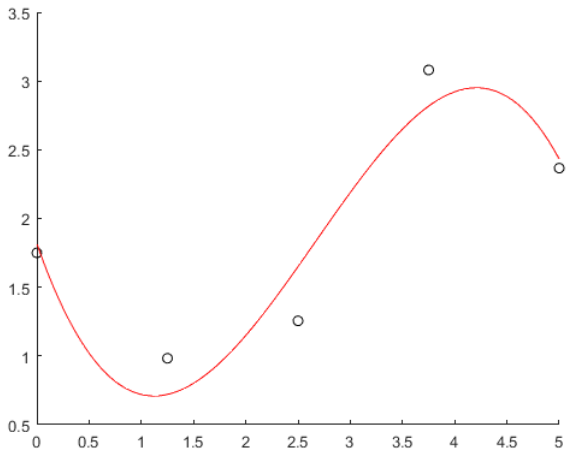
Rezultat:

```
sumSquaredErr =
    0.3028
```

5. Nacrtati pronađeni polinom:

```
x = np.linspace(np.min(x), np.max(x), 100)
pX = np.polyval(p, x)
plt.plot(x, pX, 'red')
```

Rezultat:



Slika 1. Polinom

Metoda najmanjih kvadrata

Zadatak 1. Napisati metodu najmanjih kvadrata za aproksimaciju funkcije kroz proizvoljan broj poznatih tačaka polinomom proizvoljnog stepena (bar za 1 manjeg od broja tačaka).

Potrebno je naći polinom $(m-1)$-tog stepena ($m \leq n$), gde je n broj poznatih tačaka:

$$p(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_1 x_1 + a_0$$

Da bi ukupna kvadratna greška po svim tačkama bila što manja, potrebno je rešiti sledeći problem:

$$\frac{\partial \sum_{i=1}^{n} \sum_{j=0}^{m} \left(a_j x_i^{\,j} - f(x_i)\right)^2}{\partial a_0} = 0$$

$$\frac{\partial \sum_{i=1}^{n} \sum_{j=0}^{m} \left(a_j x_i^{\,j} - f(x_i)\right)^2}{\partial a_1} = 0$$

$$\vdots$$

$$\frac{\partial \sum_{i=1}^{n} \sum_{j=0}^{m} \left(a_j x_i^{\,j} - f(x_i)\right)^2}{\partial a_m} = 0$$

Rešenje problema je:

$$\sum_{i=1}^{n} \sum_{j=0}^{m} a_j x_i^{\,j} x_i^{\,0} - \sum_{i=1}^{n} f(x_i) x_i^{\,0} = 0$$

$$\sum_{i=1}^{n} \sum_{j=0}^{m} a_j x_i^{\,j} x_i^{\,1} - \sum_{i=1}^{n} f(x_i) x_i^{\,1} = 0$$

$$\vdots$$

$$\sum_{i=1}^{n} \sum_{j=0}^{m} a_j x_i^{\,j} x_i^{\,m} - \sum_{i=1}^{n} f(x_i) x_i^{\,m} = 0$$

Rešenje problema se može zapisati i na sledeći način:

$$\sum_{j=0}^{m} \sum_{i=1}^{n} x_i^{\,j} x_i^{\,0} a_j = \sum_{i=1}^{n} f(x_i) x_i^{\,0}$$

$$\sum_{j=0}^{m} \sum_{i=1}^{n} x_i^{\,j} x_i^{\,1} a_j = \sum_{i=1}^{n} f(x_i) x_i^{\,1}$$

$$\vdots$$

$$\sum_{j=0}^{m} \sum_{i=1}^{n} x_i^{\,j} x_i^{\,m} a_j = \sum_{i=1}^{n} f(x_i) x_i^{\,m}$$

Matrični zapis rešenja je:

$$(A^T A) a = A^T f_x$$

$$A = \begin{bmatrix} x_1^{\,0} & x_1^{\,1} & \cdots & x_1^{\,m} \\ x_2^{\,0} & x_2^{\,1} & & x_2^{\,m} \\ \vdots & & \ddots & \vdots \\ x_n^{\,0} & x_n^{\,1} & \cdots & x_n^{\,m} \end{bmatrix} \qquad a = ¿ \qquad f_x = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}$$

Vektor a je rešenje sistema jednačina:

$$a = \left(A^T A\right) ¿ \left(A^T f_x ¿\right)$$

Pokušati prvo ručno nalaženje polinoma za skup tačaka:

```
x = np.array([0.0000,    1.2500,    2.5000,    3.7500,    5.0000])
fX = np.array([1.7499,    0.9830,    1.2554,    3.0802,    2.3664])

order = 3
```

1. Naći broj poznatih tačaka n i broj množioca polinoma m:

```
n = x.size
m = order + 1
```

2. Vektore `x` i `fX` je potrebno transponovati da bi dimenziono odgovarali izrazima u matričnoj jednačini (jer su dati kao vektori vrsta):

```
x = x.T
fX = fX.T
```

3. Formirati matricu A:

```
A = np.zeros((n, m))
for it in range(m):
    A[:, it] = x ** it
```

4. Rešiti sistem jednačina:

```
a = np.linalg.solve(np.matmul(A.T * A), np.matmul(A.T * fX))
```

Rešenje će nastati u sledećem obliku:

$$a = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}$$

5. Potrebno je dovesti rešenje u sledeći oblik:

$$a = \begin{bmatrix} a_m & a_{m-1} & \dots & a_0 \end{bmatrix}$$

```
p = a[::-1]
```

Rezultat:

```
p =
    -0.1527    1.2207   -2.1640    1.8157
```

6. Sada je moguće definisati funkciju koja sadrži prethodni postupak:

```
def least_squares_regression(x, fX, order):
    .
    .
```

7. Testirati funkciju na primeru:

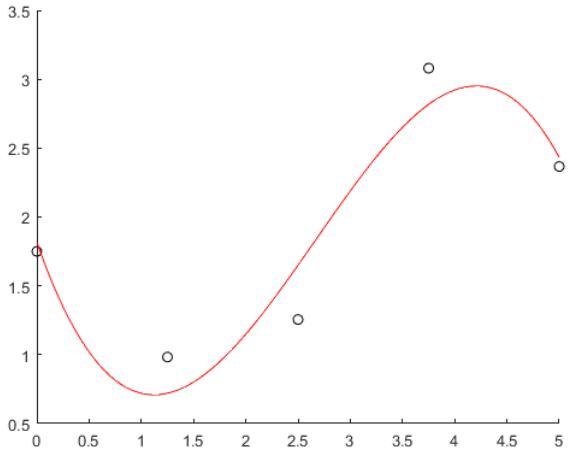
```
x = np.array([0.0000,    1.2500,    2.5000,    3.7500,    5.0000])
fX = np.array([1.7499,    0.9830,    1.2554,    3.0802,    2.3664])
plt.scatter(x, fX, c='black')

p = least_squares_regression(x, fX, 3)

x = np.linspace(np.min(x), np.max(x), 100)
pX = np.polyval(p, x)
plt.plot(x, pX, 'red')
```

Rezultat:

```
p =
    -0.1527    1.2207   -2.1640    1.8157
```



Slika 2. Niži stepen polinoma

8. Povećati stepen polinoma:

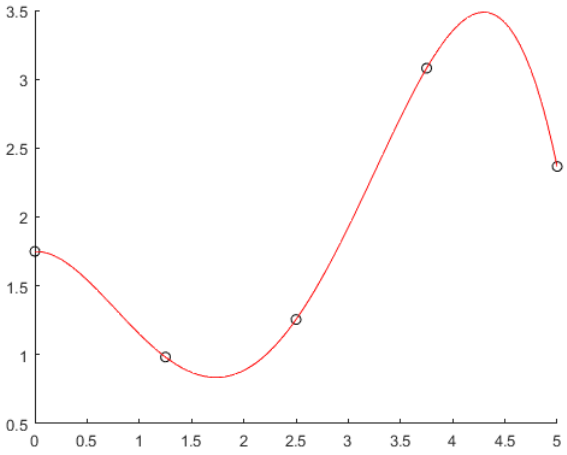
```
x = np.array([0.0000,    1.2500,    2.5000,    3.7500,    5.0000])
fX = np.array([1.7499,    0.9830,    1.2554,    3.0802,    2.3664])
plt.scatter(x, fX, c='black')

p = least_squares_regression(x, fX, 4)

x = np.linspace(np.min(x), np.max(x), 100)
pX = np.polyval(p, x)
plt.plot(x, pX, 'red')
```

Rezultat:

```
p =
   -0.0786    0.6331   -1.1822    0.0284    1.7499
```



Slika 3. Maksimalni stepen polinoma