

# Laboratorium 11

## Optymalizacja

Iga Antonik, Helena Szczepanowska

### Zadanie 1

Wyznacz punkty krytyczne każdej z poniższych funkcji. Scharakteryzuj każdy znaleziony punkt jako minimum, maksimum lub punkt siodłowy. Dla każdej funkcji zbadaj, czy posiada minimum globalne lub maksimum globalne na zbiorze  $R^2$ .

$$f_1(x, y) = x^2 - 4xy + y^2$$

$$f_2(x, y) = x^4 - 4xy + y^4$$

$$f_3(x, y) = 2x^3 - 3x^2 - 6xy(x - y - 1)$$

$$f_4(x, y) = (x - y)^4 + x^2 - y^2 - 2x + 2y + 1$$

### Rozwiązanie

$$\frac{\partial f_1}{\partial x} = 2x - 4y = 0$$

$$\frac{\partial f_1}{\partial y} = 2y - 4x = 0$$

Z powyższych równań wynika, że punkt krytyczny funkcji  $f_1$  to  $(0, 0)$

$$\frac{\partial^2 f_1}{\partial x^2} = 2$$

$$\frac{\partial^2 f_1}{\partial y^2} = 2$$

$$\frac{\partial^2 f_1}{\partial x \partial y} = -4$$

$$\frac{\partial^2 f_1}{\partial y \partial x} = -4$$

$$\begin{bmatrix} 2 & -4 \\ -4 & 2 \end{bmatrix} < 0$$

Więc punkt krytyczny  $(0, 0)$  jest punktem siodłowym.

$$\frac{\partial f_2}{\partial x} = 4x^3 - 4y = 0$$

$$\frac{\partial f_2}{\partial y} = 4y^3 - 4x = 0$$

Z powyższych równań wynika, że punkty krytyczne funkcji  $f_2$  to  $(0, 0); (-1, -1); (1, 1)$

$$\frac{\partial^2 f_2}{\partial x^2} = 12x^2$$

$$\frac{\partial^2 f_2}{\partial y^2} = 12y^2$$

$$\frac{\partial^2 f_2}{\partial x \partial y} = -4$$

$$\frac{\partial^2 f_2}{\partial y \partial x} = -4$$

$$\begin{bmatrix} 12x^2 & -4 \\ -4 & 12y^2 \end{bmatrix}$$

Dla  $(0, 0)$

$$\begin{bmatrix} 0 & -4 \\ -4 & 0 \end{bmatrix} < 0$$

Więc punkt krytyczny  $(0, 0)$  jest punktem siodłowym.

Dla  $(1, 1)$

$$\begin{bmatrix} 12 & -4 \\ -4 & 12 \end{bmatrix} > 0$$

$$\frac{\partial^2 f_2}{\partial x^2} = 12 > 0$$

Więc punkt krytyczny  $(1, 1)$  jest minimum.

Dla  $(-1, -1)$

$$\begin{bmatrix} 12 & -4 \\ -4 & 12 \end{bmatrix} > 0$$

$$\frac{\partial^2 f_1}{\partial x^2} = 12 > 0$$

Więc punkt krytyczny  $(-1, -1)$  jest minimum.

$$\frac{\partial f_3}{\partial x} = 6x^2 - 6x - 12xy + 6y^2 + 6y = 0$$

$$\frac{\partial f_3}{\partial y} = -6x^2 + 12xy + 6x = 0$$

Z powyższych równań wynika, że punkty krytyczne funkcji  $f_3$  to  $(0, 0)$ ;  $(1, 0)$ ;  $(0, -1)$ ;  $(-1, -1)$

$$\frac{\partial^2 f_3}{\partial x^2} = 12x - 6 - 12y$$

$$\frac{\partial^2 f_3}{\partial y^2} = 12x$$

$$\frac{\partial^2 f_3}{\partial x \partial y} = -12x + 12y + 6$$

$$\frac{\partial^2 f_3}{\partial y \partial x} = -12x + 12y + 6$$

$$\begin{bmatrix} 12x - 6 - 12y & -12x + 12y + 6 \\ -12x + 12y + 6 & 12x \end{bmatrix}$$

Dla  $(0, 0)$

$$\begin{bmatrix} -6 & 6 \\ 6 & 0 \end{bmatrix} < 0$$

Więc punkt krytyczny  $(0, 0)$  jest punktem siodłowym.

Dla  $(1, 0)$

$$\begin{bmatrix} 6 & -6 \\ -6 & 12 \end{bmatrix} > 0$$

$$\frac{\partial^2 f_3}{\partial x^2} = 6 > 0$$

Więc punkt krytyczny  $(1, 0)$  jest minimum.

Dla  $(0, -1)$

$$\begin{bmatrix} 6 & -6 \\ -6 & 0 \end{bmatrix} < 0$$

Więc punkt krytyczny  $(0, -1)$  jest punktem siodłowym.

Dla  $(-1, -1)$

$$\begin{bmatrix} -6 & 6 \\ 6 & -12 \end{bmatrix} > 0$$

$$\frac{\partial^2 f_3}{\partial x^2} = -6 < 0$$

Więc punkt krytyczny  $(-1, -1)$  jest maksimum.

$$\frac{\partial f_4}{\partial x} = 4x^3 - 12x^2y + 12xy^2 - 4y^3 + 2x - 2 = 0$$

$$\frac{\partial f_4}{\partial y} = -4x^3 + 12x^2y - 12xy^2 + y^3 - 2y + 2 = 0$$

$$2x - 2y = 0$$

$$x = y = 0$$

Z powyższych równań wynika, że punkty krytyczne funkcji  $f_4$  to  $(0, 0)$

$$\frac{\partial^2 f_4}{\partial x^2} = 12x^2 - 24xy + 12y^2 + 2$$

$$\frac{\partial^2 f_4}{\partial y^2} = 12x^2 - 24xy + 12y^2 - 2$$

$$\frac{\partial^2 f_4}{\partial x \partial y} = -12x^2 + 24xy - 12y^2$$

$$\frac{\partial^2 f_4}{\partial y \partial x} = -12x^2 + 24xy - 12y^2$$

Dla  $(0, 0)$

$$\begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix} < 0$$

Więc punkt krytyczny  $(0, 0)$  jest punktem siodłowym.

## Zadanie 2

Należy wyznaczyć najkrótszą ścieżkę robota pomiędzy dwoma punktami  $x^{(0)}$  i  $x^{(n)}$ . Problemem są przeszkody usytuowane na trasie robota, których należy unikać. Zadanie polega na minimalizacji funkcja kosztu, która sprowadza problem nieliniowej optymalizacji z ograniczeniami do problemu nieograniczonej optymalizacji. Macierz  $X \in R^{(n+1) \times 2}$  opisuje ścieżkę złożoną z  $n + 1$  punktów  $x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(n)}$ . Każdy punkt posiada 2 współrzędne,  $x^{(i)} \in R^2$ . Punkty początkowy i końcowy ścieżki,  $x^{(0)}$  i  $x^{(n)}$ , są ustalone. Punkty z przeszkodami (punkty o 2 współrzędnych),  $r^{(i)}$  dane są w macierzy przeszkód  $R \in R^{k \times 2}$ . W celu optymalizacji ścieżki robota

należy użyć metody największego spadku. Funkcja celu użyta do optymalizacji  $F(x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(n)})$  zdefiniowana jest jako

$$F(x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(n)}) = \lambda_1 \sum_{i=1}^n \sum_{j=1}^k \frac{1}{\varepsilon + \|x^{(i)} - r^{(j)}\|_2^2} + \lambda_2 \sum_{i=0}^{n-1} \|x^{(i+1)} - x^{(i)}\|_2^2$$

Symbole użyte we wzorze mają następujące znaczenie:

- Stałe  $\lambda_1$  i  $\lambda_2$  określają wpływ każdego członu wyrażenia na wartość  $F(X)$ .
- $\lambda_1$  określa wagę składnika zapobiegającego zbytniemu zbliżaniu się do przeszkody
- $\lambda_2$  określa wagę składnika zapobiegającego tworzeniu bardzo długich ścieżek
- $n$  jest liczbą odcinków, a  $n + 1$  liczbą punktów na trasie robota.
- $k$  jest liczbą przeszkód, których robot musi unikać.
- Dodanie  $\varepsilon$  w mianowniku zapobiega dzieleniu przez zero.

1. Wyprowadź wyrażenie na gradient  $\nabla F$  funkcji celu  $F$  względem  $x^{(i)}$ :  $\nabla F = [\frac{\partial F}{\partial x^{(0)}}, \dots, \frac{\partial F}{\partial x^{(n)}}]$ . Wzór wyraż przez wektory  $x^{(i)}$  i ich składowe, wektory  $r^{(j)}$  i ich składowe,  $\varepsilon$ ,  $\lambda_1$ ,  $\lambda_2$ ,  $n$  i  $k$  (niekoniecznie wszystkie).

Wskazówka.  $\frac{\partial F \|x\|^2}{\partial x} = 2x$ .

2. Opisz matematycznie i zaimplementuj kroki algorytmu największego spadku z przeszukiwaniem liniowym, który służy do minimalizacji funkcji celu  $F$ . Do przeszukiwania liniowego (ang. line search) użyj metody złotego podziału (ang. golden section search). W tym celu załóż, że  $F$  jest unimodalna (w rzeczywistości tak nie jest) i że można ustalić początkowy przedział, w którym znajduje się minimum.

3. Znajdź najkrótszą ścieżkę robota przy użyciu algorytmu zaimplementowanego w w poprzednim punkcie. Przyjmij następujące wartości parametrów:

- $n = 20, k = 50$
- $x^{(0)} = [0, 0], x^{(n)} = [20, 20]$
- $r^{(i)} \sim U(0, 20) \times U(0, 20)$
- $\lambda_1 = \lambda_2 = 1$
- $\varepsilon = 10^{-13}$
- liczba iteracji = 400

Ponieważ nie chcemy zmieniać położenia punktu początkowego i końcowego,  $x^{(0)}, x^{(n)}$ , wyzeruj gradient funkcji  $F$  względem tym punktów. Obliczenia przeprowadź dla 5 różnych losowych inicjalizacji punktów wewnątrz ścieżki  $x^{(0)}, \dots, x^{(n-1)}$ . Narysuj przykładowy wykres wartości funkcji  $F$  w zależności od iteracji.

## Rozwiązanie

### 1. Wyprowadzenie gradientu funkcji celu $F$

$$F(x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(n)}) = \lambda_1 \sum_{i=1}^n \sum_{j=1}^k \frac{1}{\varepsilon + \|x^{(i)} - r^{(j)}\|_2^2} + \lambda_2 \sum_{i=0}^{n-1} \|x^{(i+1)} - x^{(i)}\|_2^2$$

Gradient składnika unikania przeszkód:

$$\frac{\partial}{\partial x^{(i)}} \frac{1}{\varepsilon + \|x^{(i)} - r^{(j)}\|_2^2} = -\frac{2(x^{(i)} - r^{(j)})}{\varepsilon + \|x^{(i)} - r^{(j)}\|_2^2}$$

Gradient składnika długości ścieżki:

$$\frac{\partial}{\partial x^{(i)}} \|x^{(i+1)} - x^{(i)}\|_2^2 = 2(x^{(i)} - x^{(i+1)})$$

$$\nabla F_{x^{(i)}} = -\lambda_1 \frac{2(x^{(i)} - r^{(j)})}{\varepsilon + \|x^{(i)} - r^{(j)}\|_2^2} + 2(x^{(i)} - x^{(i+1)})$$

### 2. Algorytm największego spadku z przeszukiwaniem liniowym

- $f: \mathbb{R}^n \rightarrow \mathbb{R}$
- $x_k \in \mathbb{R}^n$
- $\alpha_k \in \mathbb{R}$

## Steepest descent

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

## Line search

$$\min_{\alpha_k} f(x_k - \alpha_k \nabla f(x_k))$$

## Biblioteki

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt
```

## Podane parametry

```
In [ ]: n = 20
k = 30
x0 = np.array([0, 0])
xn = np.array([20, 20])
r = np.random.uniform(0, 20, (k, 2))
lambda1 = 1
lambda2 = 1
epsilon = 1e-3
num_iterations = 400
x = np.linspace(x0, xn, n+1)
```

## Funkcja kosztu

- term1: Kara za bliskość do przeszkód.
- term2: Kara za długość ścieżki.
- lambda1 i lambda2 równoważą te dwie składowe.

```
In [ ]: def compute_F(x, r, lambda1, lambda2):
    term1 = sum(1 / (np.linalg.norm(x[i] - r[j])**2 + epsilon) for i in range(1, n) for j in range(k))
    term2 = sum(np.linalg.norm(x[i+1] - x[i])**2 for i in range(n))
    return lambda1 * term1 + lambda2 * term2
```

## Gradient

Obliczamy gradient funkcji  $F$  względem każdego punktu  $x^{(i)}$  z wyjątkiem punktów końcowych.

```
In [ ]: def compute_gradient(x, r, lambda1, lambda2):
    grad = np.zeros_like(x)
    for i in range(1, n):
        term1_grad = np.zeros(2)
        for j in range(k):
            term1_grad += (x[i] - r[j]) / (np.linalg.norm(x[i] - r[j])**2 + epsilon)**2
        term1_grad *= -2 * lambda1
```

```

term2_grad = np.zeros(2)
if i > 0:
    term2_grad += 2 * lambda2 * (x[i] - x[i-1])
if i < n:
    term2_grad += 2 * lambda2 * (x[i] - x[i+1])

grad[i] = term1_grad + term2_grad
return grad

```

## Metoda Złotego Podziału do Przeszukiwania Liniowego

Znajdujemy optymalny krok  $\alpha$  dla największego spadku, minimalizując funkcję kosztu wzdłuż kierunku przeszukiwania.

```

In [ ]: def golden_section_search(func, a, b, tol=1e-5):
    phi = (1 + np.sqrt(5)) / 2
    while (b - a) > tol:
        c = b - (b - a) / phi
        d = a + (b - a) / phi
        if func(c) < func(d):
            b = d
        else:
            a = c
    return (b + a) / 2

```

## Gradientowy Spadek z Przeszukiwaniem Liniowym

- Iteracyjnie aktualizuje ścieżkę, poruszając się w kierunku przeciwnym do gradientu.
- Używa przeszukiwania metodą złotego podziału do określenia optymalnego kroku  $\alpha$  przy każdej iteracji.

```

In [ ]: def gradient_descent_with_line_search(x, r, lambda1, lambda2, num_iterations):
    history = []
    for _ in range(num_iterations):
        grad = compute_gradient(x, r, lambda1, lambda2)

        # Funkcja do minimalizacji wzdłuż kierunku przeciwnym do gradientu
        def line_search_func(alpha):
            new_x = x.copy()
            new_x[1:n] = new_x[1:n] - alpha * grad[1:n]
            return compute_F(new_x, r, lambda1, lambda2)

        alpha = golden_section_search(line_search_func, 0, 1)

        x[1:n] = x[1:n] - alpha * grad[1:n]

        history.append(compute_F(x, r, lambda1, lambda2))
    return x, history

```

## 3. Nakrótka ścieżka robota

```

In [ ]: optimal_path, history = gradient_descent_with_line_search(x, r, lambda1, lambda2, num_iteration

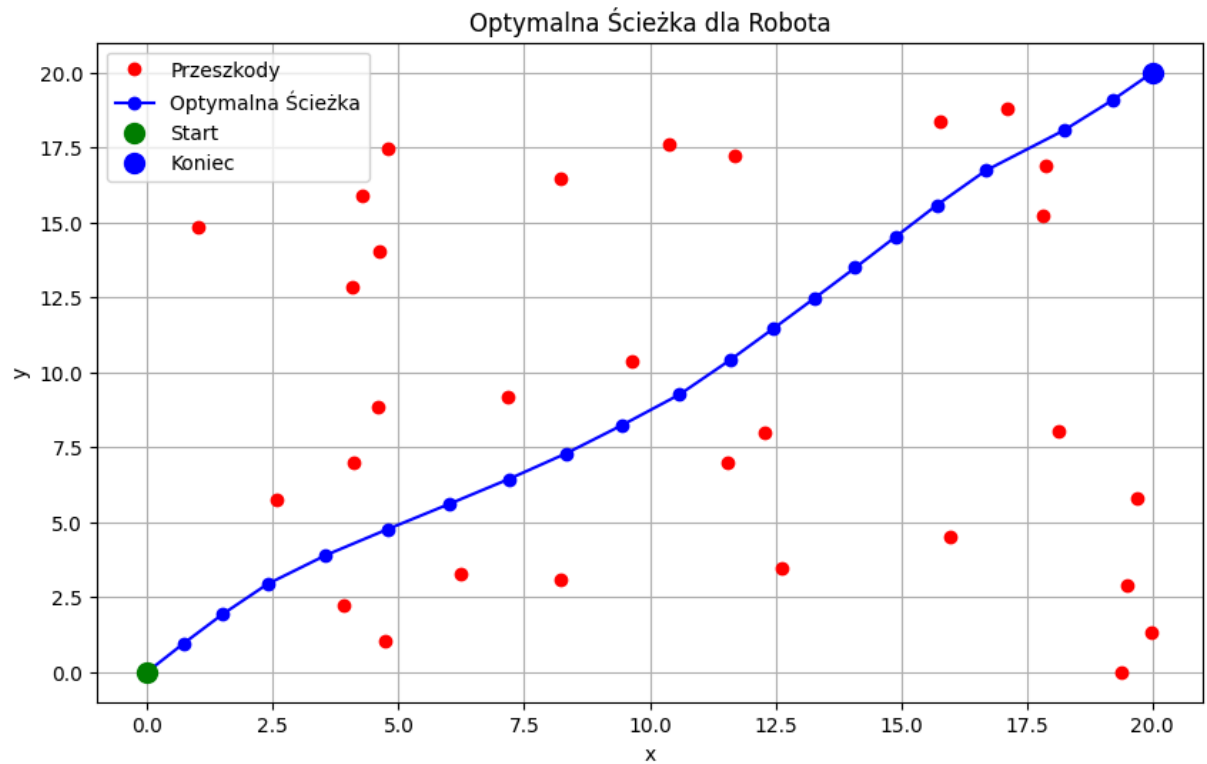
```

## Wykres 1

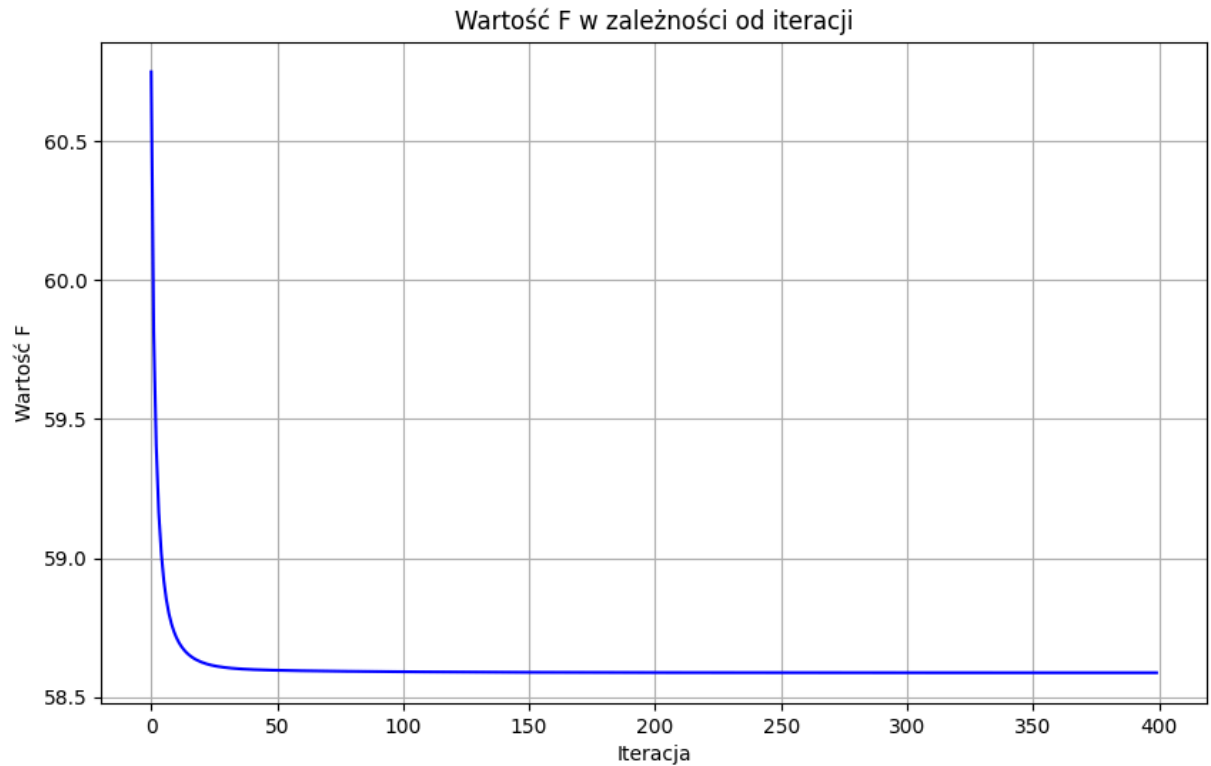
```

In [ ]: plt.figure(figsize=(10, 6))
plt.plot(r[:, 0], r[:, 1], 'ro', label='Przeszkody')
plt.plot(optimal_path[:, 0], optimal_path[:, 1], 'b-', marker='o', label='Optymalna Ścieżka')
plt.plot(x0[0], x0[1], 'go', markersize=10, label='Start')
plt.plot(xn[0], xn[1], 'bo', markersize=10, label='Koniec')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Optymalna Ścieżka dla Robota')
plt.legend()
plt.grid(True)
plt.show()

```



```
In [ ]: plt.figure(figsize=(10, 6))
plt.plot(history, 'b-')
plt.xlabel('Iteracja')
plt.ylabel('Wartość F')
plt.title('Wartość F w zależności od iteracji')
plt.grid(True)
plt.show()
```

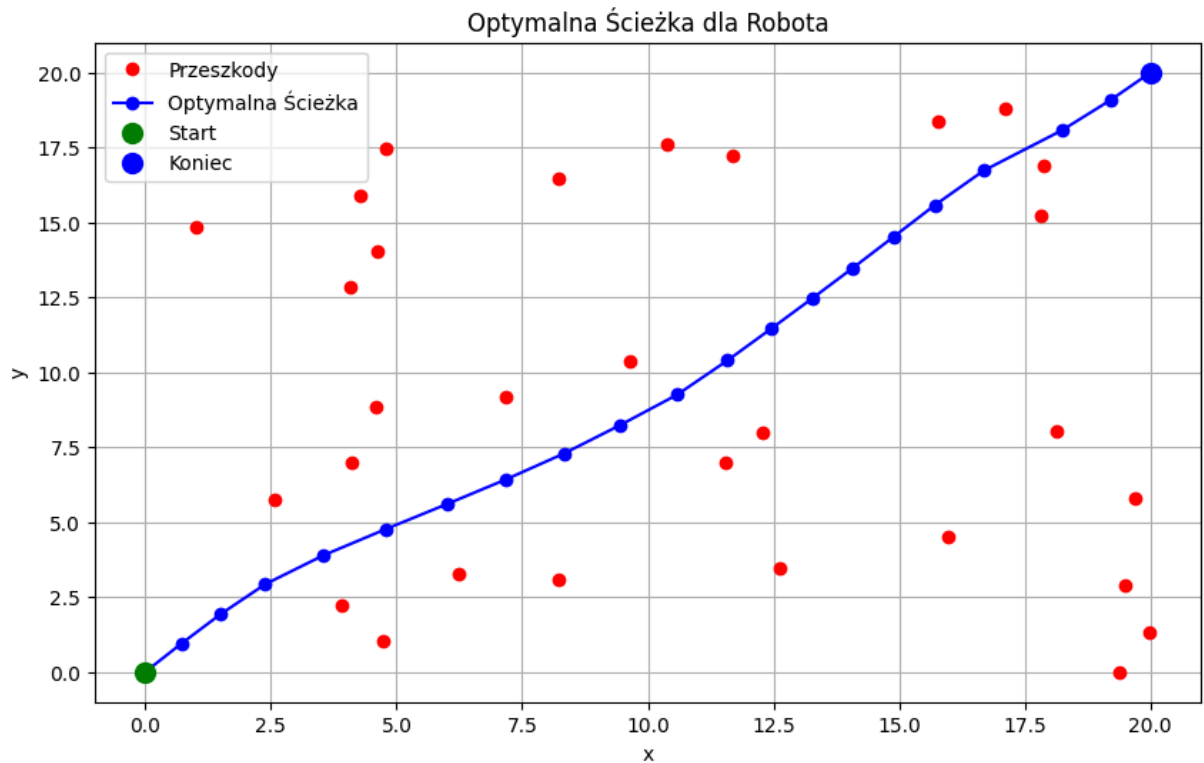


## Wykres 2

```
In [ ]: optimal_path, history = gradient_descent_with_line_search(x, r, lambda1, lambda2, num_iteration)
```

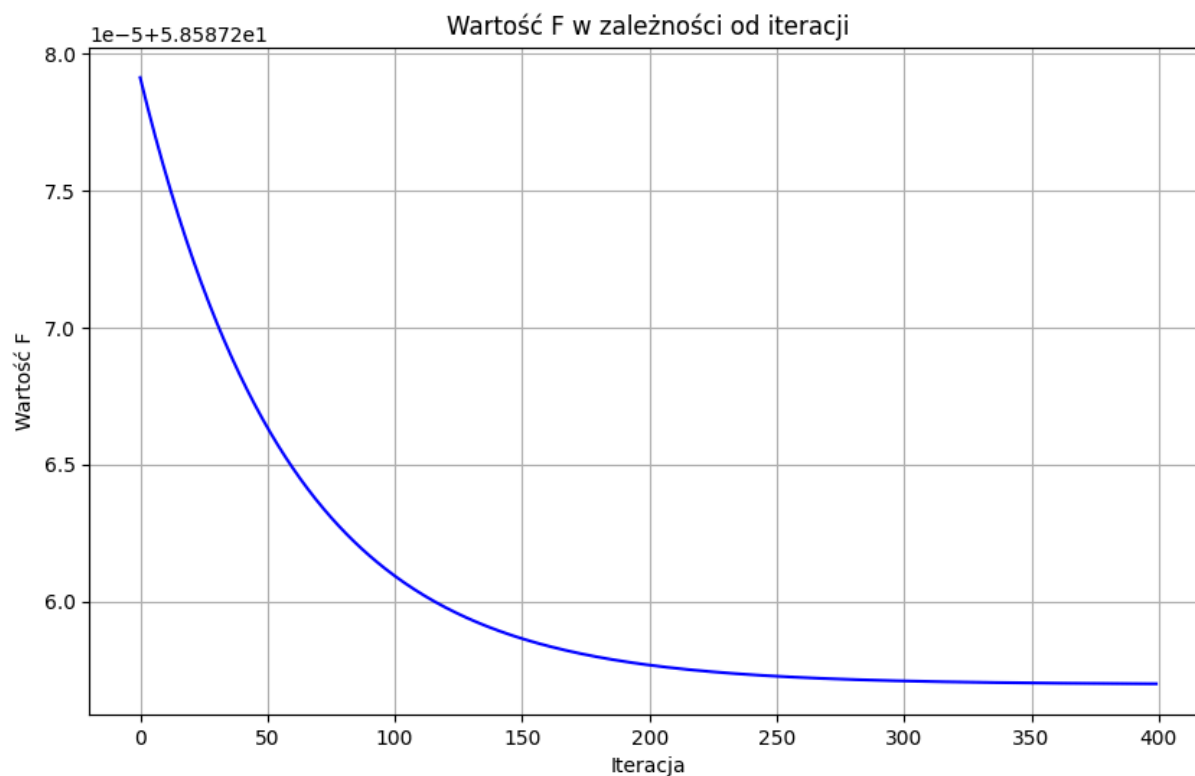
```
In [ ]: plt.figure(figsize=(10, 6))
plt.plot(r[:, 0], r[:, 1], 'ro', label='Przeszkody')
```

```
plt.plot(optimal_path[:, 0], optimal_path[:, 1], 'b-', marker='o', label='Optymalna Ścieżka')
plt.plot(x0[0], x0[1], 'go', markersize=10, label='Start')
plt.plot(xn[0], xn[1], 'bo', markersize=10, label='Koniec')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Optymalna Ścieżka dla Robota')
plt.legend()
plt.grid(True)
plt.show()
```



```
In [ ]: plt.figure(figsize=(10, 6))
plt.plot(history, 'b-')
plt.xlabel('Iteracja')
plt.ylabel('Wartość F')
plt.title('Wartość F w zależności od iteracji')
plt.grid(True)
plt.show()
```





## Wnioski

Liczba iteracji algorytmu gradientowego spadku wpływa na dokładność znalezionej ścieżki. Większa liczba iteracji pozwala na bardziej precyzyjne dostrojenie ścieżki, choć kosztem czasu obliczeń.

Optymalizacja ścieżki robota przy użyciu algorytmu gradientowego spadku z przeszukiwaniem liniowym okazała się skuteczna. Algorytm ten poprawnie minimalizuje funkcję kosztu, znajdując optymalną ścieżkę, która omija przeszkody i utrzymuje rozsądną długość. Wartości parametrów  $\lambda_1$ ,  $\lambda_2$  oraz  $\epsilon$  mają kluczowy wpływ na wyniki optymalizacji. Jak widać na wykresach powyżej, każde uruchomienie algorytmu może dać inny wynik/ inną ścieżkę robota.

## Bibliografia

prezentacja Optimization - Marcin Kuta

Michael T. Heath, [https://heath.cs.illinois.edu/scicomp/notes/cs450\\_chapt06.pdf](https://heath.cs.illinois.edu/scicomp/notes/cs450_chapt06.pdf)