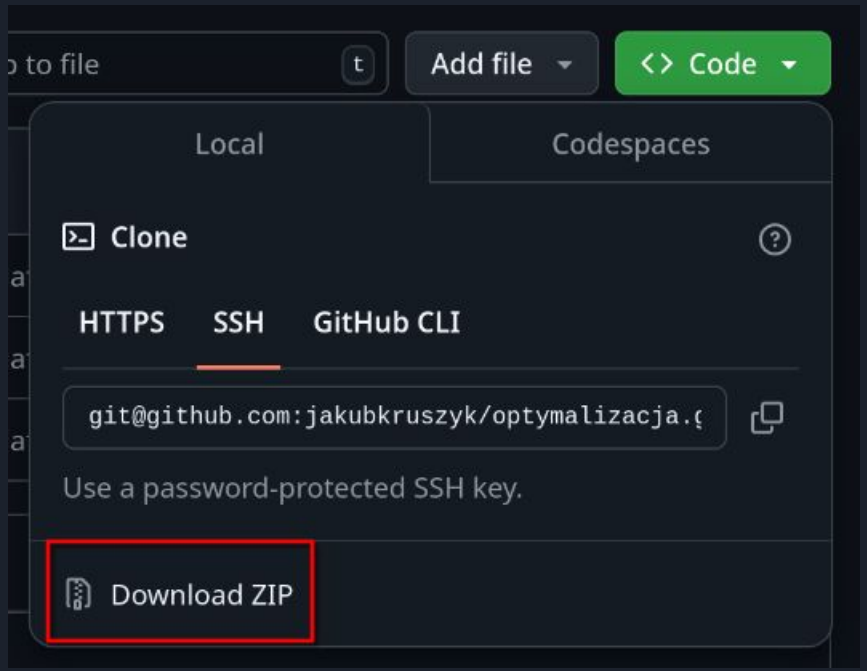
Programowanie nieliniowe

Jakub Kruszyk
Maciej Duszczak
Sławomir Forszpaniak
Tomasz Sobecki

https://github.com/jakubkruszyk/optymalizacja



Programowanie nieliniowe bez ograniczeń

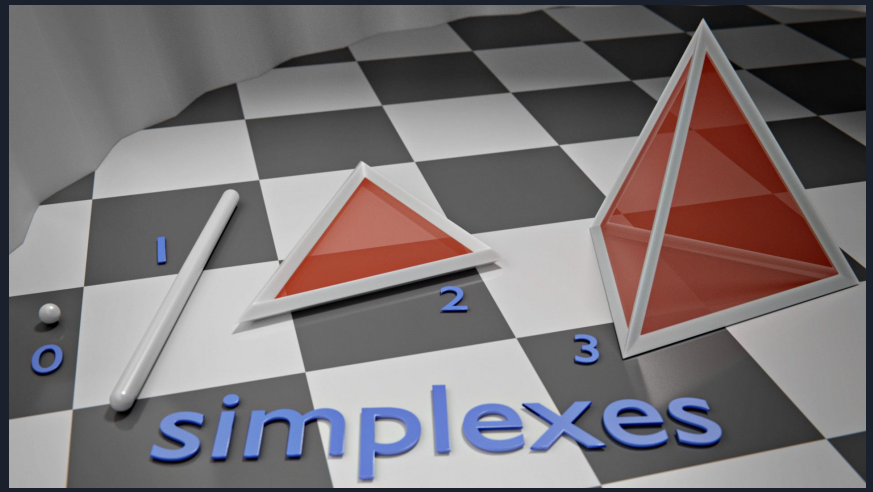
- Szukają minimum lokalnego dowolnej funkcji wielu zmiennych
- Wymagają podania punktu startowego
- Metody iteracyjne - powtarzane dopóki obliczone rozwiązanie nie będzie wystarczająco blisko optimum

Metoda Neldera-Meada

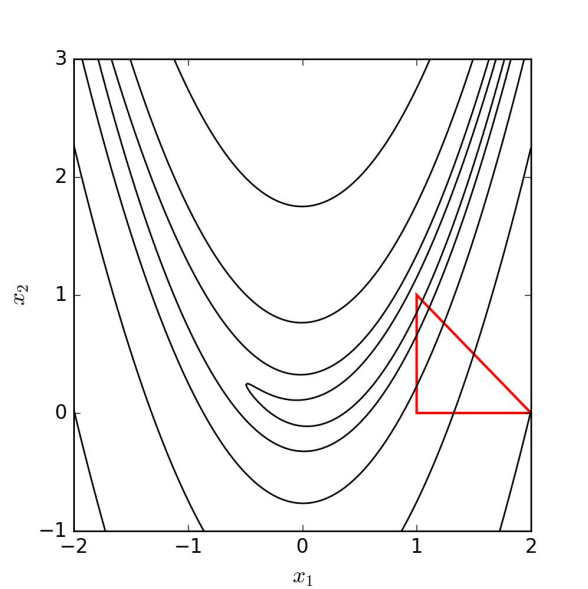
Polega na przekształcaniu w-wymiarowego simpleksu, czyli figury składającej się z n+1 punktów, gdzie n to liczba zmiennych optymalizowanej funkcji.

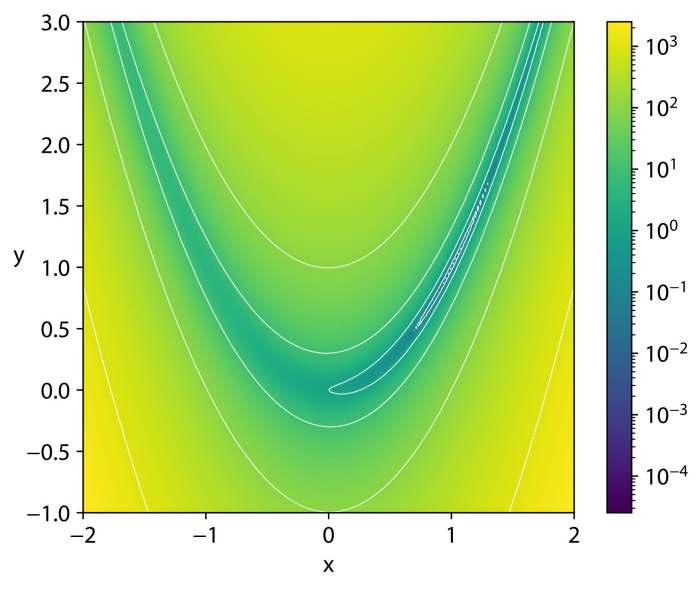
W każdym kroku iteracji obliczana jest wartość funkcji w punktach simpleksu. Następnie punkt, w którym wartość funkcji jest największa(najgorsza) zostaje wymieniony.

Algorytm kończy pracę gdy odległość pomiędzy środkiem symetrii simpleksu a jego wierzchołkami przekroczy zadaną wartość.



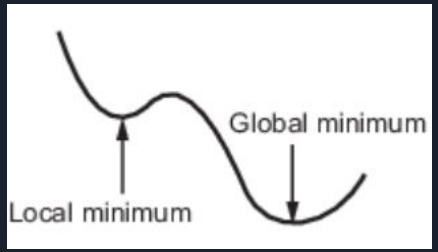
Metoda Neldera-Meada





Metoda Neldera-Meada

Funkcja fminsearch znajduje minimum lokalne nieograniczonej funkcji wielu zmiennych metodą Neldera-Meada.



[x, fval] = fminsearch(fun, x0, options)

gdzie:

fun - funkcja poddawana minimalizacji

x0 - punkt początkowy w postaci jednowymiarowego wektora

options - opcjonalny argument pozwalający na zmianę parametrów optymalizacji

Metoda Neldera-Meada

Zadanie 1

Znajdź minimum poniższej funkcji:

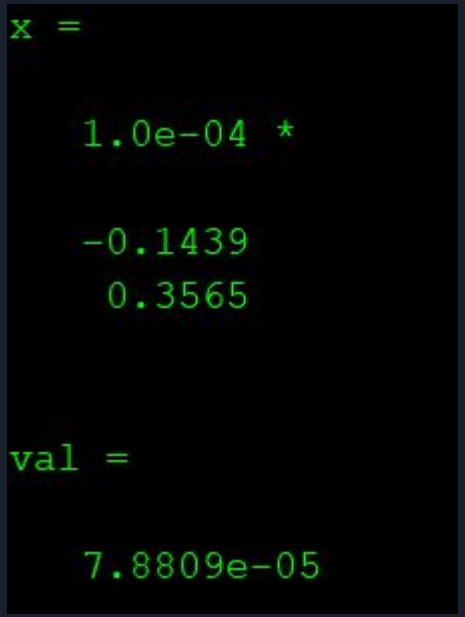
$$y = fun(x) = 3|x_1| + |x_2|$$

Zapis zmiennych funkcji anonimowych w języku Matlab:

$$f@(x)... -> x_n = x(n)$$

Metoda Neldera-Meada

Rozwiązanie:



Metoda Neldera-Meada

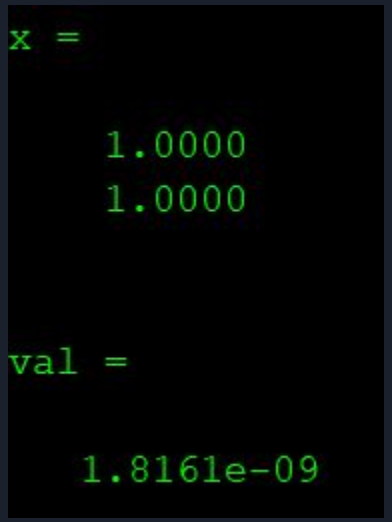
Zadanie 2

Znajdź minimum podanej funkcji:

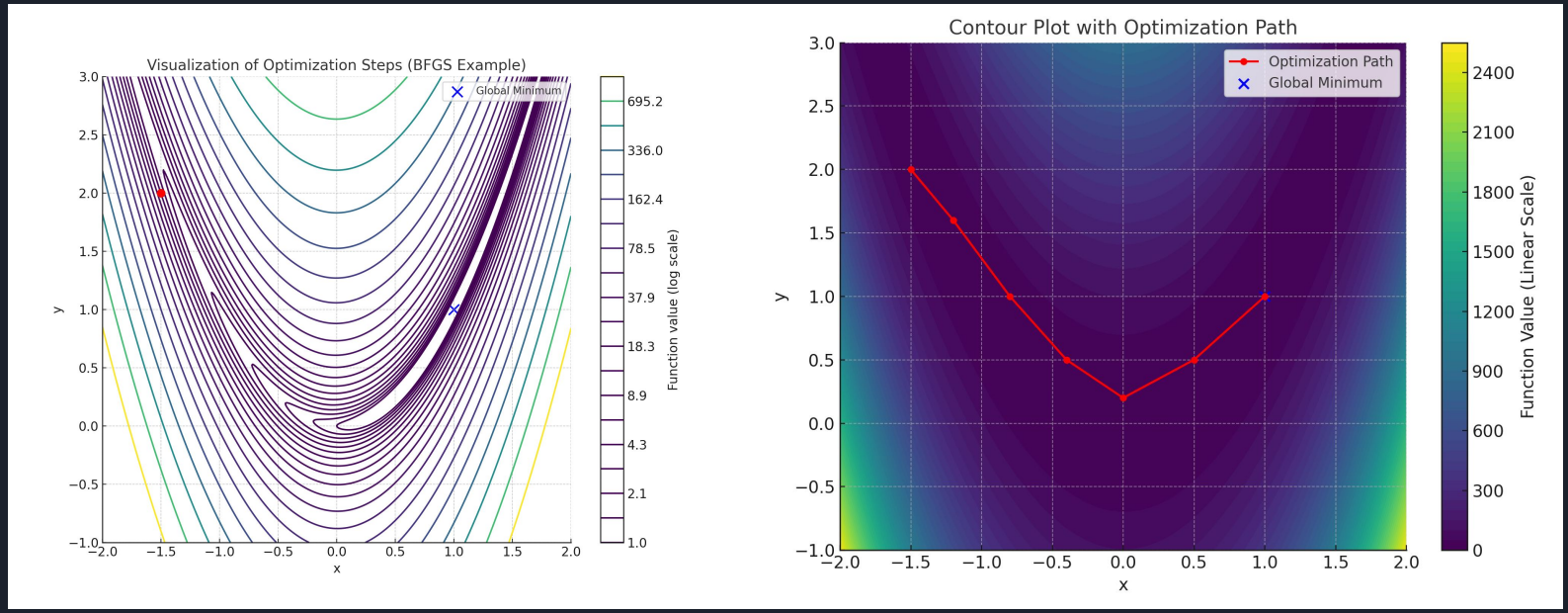
$$z = 5(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Metoda Neldera-Meada

Rozwiązanie:



Metoda Broydena-Fletchera-Goldfarba-Shanno



Metoda Broydena-Fletchera-Goldfarba-Shanno

Metoda Broydena-Fletchera-Goldfarba-Shanno (BFGS)

Polega na iteracyjnym znajdowaniu minimum funkcji celu poprzez aktualizację macierzy aproksymującej Hesjan, co pozwala uniknąć bezpośredniego obliczania odwrotności macierzy Hesjana.

W każdym kroku iteracji:

1. Wyznaczany jest gradient funkcji w bieżącym punkcie.
2. Aproksymowana jest odwrotność Hesjana na podstawie gradientów z kolejnych iteracji.
3. Wybierany jest kierunek poprawy, a następnie wykonywany jest krok minimalizacyjny wzdłuż tego kierunku.

Algorytm kończy pracę, gdy norma gradientu funkcji celu jest mniejsza od zadanej wartości lub długość kroku minimalizacyjnego osiągnie zadaną precyzję.

Metoda Broydena-Fletchera-Goldfarba-Shanno

Funkcja fminunc znajduje minimum nieograniczonej funkcji wielu zmiennych metodą Broydena-Fletchera-Goldfarba-Shanno.

[x, fval] = fminunc(fun, x0, options)

gdzie:

fun - funkcja poddawana minimalizacji

x0 - punkt początkowy w postaci jednowymiarowego wektora

options - opcjonalny argument pozwalający na zmianę parametrów optymalizacji

Metoda Broydena-Fletchera-Goldfarba-Shanno

Zadanie Optymalizacji Nieliniowej (Bez gradientu)

Znajdź minimum funkcji celu:

$$f(x) = 3x_1^2 + 2x_1x_2 + x_2^2$$

przy warunkach poczatkowych:

$$x_0 = [1, 1].$$

```
f = @(x)3*x(1)^2+2*x(1)*x(2)+x(2)^2;
x0 = [1, 1];
options = optimset('Display','iter');
[x, fval, exitflag, output] = fminunc(f, x0, options)
```

```
x =

   1.0e-06 *

    0.2541   -0.2029


fval =

   1.3173e-13
```

Metoda Broydena-Fletchera-Goldfarba-Shanno

Zadanie Optymalizacji Nieliniowej (Z gradientem)

Znajdź minimum funkcji celu:

$$f(x) = 3x_1^2 + 2x_1x_2 + x_2^2$$

przy warunkach poczatkowych:

$$x_0 = [1, 1].$$

Dodatkowo wykorzystaj informacje o gradientzie funkcji celu:

$$\frac{\partial f}{\partial x_1} = 6x_1 + 2x_2, \quad \frac{\partial f}{\partial x_2} = 2x_1 + 2x_2.$$

Plik: zad2_fun.m

```
function [f, g] = zad2_fun(x)
    % Funkcja celu
    f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;

    % Gradient funkcji celu
    if nargout > 1 % Sprawdzenie, czy gradient jest wymagany
        g(1) = 6*x(1)+2*x(2);
        g(2) = 2*x(1)+2*x(2);
    end
end
```

Plik: zad2.m

```
x0 = [1, 1];

options = optimoptions('fminunc', ...
 'SpecifyObjectiveGradient', true, ...
 'Algorithm', 'quasi-newton', ...
 'TolX', 1e-12, ...
 'TolFun', 1e-12, ...
 'Display', 'iter');

[x, val, exitflag, output] = fminunc(@zad2_fun, x0, options)
```

```
x =

   1.0e-13 *

   -0.1585    0.4090


val =

   1.1298e-27
```

Programowanie nieliniowe z ograniczeniami

- Podstawowe podejście polega na zamianie zadania z ograniczeniami na zadanie bez ograniczeń, najprostszym rozwiązaniem jest modyfikacja funkcji celu
- Metoda funkcji kar
  - Wielokrotna modyfikacja funkcji celu, do której wprowadzony został czynnik *"kary"* sztucznie zawyżający wartość minimalizacji, który w wyniku iteracji jest modyfikowany i stopniowo zanika gdy spełniane są ograniczenia
- Metoda kierunków dopuszczalnych
  - Metoda polega na przeniesieniu punktu początkowego do obszaru dopuszczalnego poprzez minimalizację odległości między punktem a ograniczeniami. Gwarantuje pozostanie w obszarze dopuszczalnym, ale może być kosztowna obliczeniowo przy dużej liczbie ograniczeń.
- Metoda rzutu ortogonalnego
  - Znajduje kierunek poprawy funkcji celu, który jednocześnie spełnia ograniczenia. W każdym kroku optymalizuje funkcję wzdłuż dopuszczalnego kierunku, zapewniając efektywne poruszanie się w obszarze dopuszczalnym.

Programowanie nieliniowe z ograniczeniami

- Metoda SQP (Sequential Quadratic Programming)
  - Należy do metod modyfikacji kierunku poszukiwań jak i poprawy
  - Iteracyjnie rozwiązuje zadanie programowania kwadratowego, aby kierunek był możliwie bliski kierunkowi metody Newtona, po czym dokonywana jest minimalizacja w kierunku
  - Hesjan uaktualniany jest z zasadami metod zmiennej metryki
- Metoda równań Kuhna-Tuckera
  - Najefektywniejsza z metod
  - Ustala warunki optymalności z użyciem mnożników Lagrange'a

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{x}^*) = 0,$$

$$\nabla g_i(\mathbf{x}^*) = 0, \quad i = \overline{1, m_e},$$

$$\lambda_i^* = 0, \quad i = \overline{m_e + 1, m},$$

Programowanie nieliniowe z ograniczeniami

Funkcja fmincon znajduje minimum problemu opisanego jako:

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub, \end{cases}$$

[x, fval] = fmincon (fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options)

gdzie:

fun - minimalizowana funkcja

x0 - punkt początkowy w postaci jednowymiarowego wektora

A, b - ograniczenia liniowe (nierówności)

Aeq, beq - ograniczenia liniowe (równości)

lb, ub - ograniczenie dolne i górne

nonlcon - ograniczenia nieliniowe reprezentowane jako c(x) oraz ceq(x)

options - opcjonalny parametr pozwalający m.in. na wybór algorytmu optymalizacji (domyślny algorytm: interior-point)

Programowanie nieliniowe z ograniczeniami

Przykład 1

Znajdź minimum funkcji celu:

$$f(x) = -x_1 * x_2 * x_3$$

przy warunkach początkowych:

$$x_0 = [10; 10; 10]$$

oraz przy obecności ograniczeń liniowych:

$$-x_1 - 2x_2 - 2x_3 \leq 0,$$
$$x_1 + 2x_2 + 2x_3 \leq 72$$

Ograniczenia te można zapisać w postaci macierzowej:

$$A * x \leq b,$$

gdzie:

$$A = \begin{bmatrix} -1 & -2 & -2 \\ 1 & 2 & 2 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 72 \end{bmatrix}$$

Programowanie nieliniowe z ograniczeniami

Przykład 1

```
% Funkcja celu
f = @(x) -x(1) * x(2) * x(3); % f(x) = -x1*x2*x3

% Ograniczenia liniowe w postaci macierzowej
A = [-1, -2, -2;   % -x1 - 2x2 - 2x3 <= 0
      1,  2,  2];  % x1 + 2x2 + 2x3 <= 72
b = [0; 72];

% Warunki początkowe
x0 = [10; 10; 10]; % Startowa wartość zmiennych

% Rozwiązanie problemu optymalizacji
[x, fval] = fmincon(f, x0, A, b, [], [], [], []);

% Wyświetlenie wyników
disp('Optymalne rozwiązanie:');
disp(x);
disp('Wartość funkcji celu:');
disp(fval);
```

```
Optymalne rozwiązanie:
   24.0000
   12.0000
   12.0000

Wartość funkcji celu:
  -3.4560e+03
```

Programowanie nieliniowe z ograniczeniami

Zadanie 1

Znajdź minimum poniższej funkcji przy zadanych ograniczeniach

$$f = 5(x_2 - x^2_1)^2 + (1 - x_1)^2$$

$$x_0 = (-3, 4)$$

Ograniczenia: $x_1 - x_2 + 4 \leq 0$

Programowanie nieliniowe z ograniczeniami

Zadanie 1 - rozwiązanie

```
f = @(x)(1-x(1))^2+5*(x(2)-x(1)^2)^2;
A = [1, -1];
b = -4;

x0 = [-3.0; 4.0];
options = optimset('Display','iter');
[x, fval] = fmincon(f, x0, A, b, [], [], [], [], [], options)
```

```
x =

   -1.5311
    2.4689


fval =

    6.4841
```

Programowanie nieliniowe z ograniczeniami

Zadanie 2

Funcja

$$f(x) = 0.5x_1^2 + 0.5x_2^2 - x_1 - 2x_2 + 5$$

przy $x_0 = (1.5, 0.5)$ i ograniczeniach

$$\begin{aligned} 2x_1 + 3x_2 &\leq 6, \\ x_1 + 4x_2 &\leq 5, \\ x_1, x_2 &\geq 0 \end{aligned}$$

Programowanie nieliniowe z ograniczeniami

Zadanie 2 - rozwiązanie

```
% Funkcja celu
f = @(x) 0.5 * x(1)^2 + 0.5 * x(2)^2 - x(1) - 2*x(2) + 5;

% Ograniczenia liniowe
A = [2, 3;      % 2x1 + 3x2 <= 6
     1, 4];     % x1 + 4x2 <= 5
b = [6; 5];

% Dolne ograniczenia (x1, x2 >= 0)
lb = [0, 0];
ub = []; % Brak górnych ograniczeń

% Warunki początkowe
x0 = [1.5, 0.5]; % Punkt startowy

% Rozwiązanie problemu optymalizacji
[x, fval] = fmincon(f, x0, A, b, [], [], lb, ub);

% Wyświetlenie wyników
disp('Optymalne rozwiązanie:');
disp(x);
disp('Wartość funkcji celu:');
disp(fval);
```

```
Optymalne rozwiązanie:
    0.7647    1.0588

Wartość funkcji celu:
    2.9706
```