Przegląd metod optymalizacji numerycznej

Krzysztof Malczewski

Numeryczne metody optymalizacji

Deterministyczne (klasyczne)
* bez ograniczeń:
 - bezgradientowe:
  + **simpleks Neldera-Meada,**
  + spadku względem współrzędnych (Gaussa-Seidela),
  + Powella, Rosenbrocka, Hooke'a-Jeevesa,...
 - gradientowe pierwszego rzędu:
  + największego spadku,
  + gradientów sprzężonych.
 - gradientowe drugiego rzędu i „superliniowe":
  + Newtona, BFGS,
  + „***Trust region***".

* z ograniczeniami:
  + eliminacji zmiennych,
  + Lagrange'a,
  + z funkcją kary,
  + SQP.

Niedeterministyczne - Monte Carlo, symulowane wyżarzanie, algorytmy genetyczne i ewolucyjne, algorytmy rojowe itp.

Cel

Znaleźć $\min_x f(x),$

$x \in \Re^n$, gdy zadanie bez ograniczeń,

$x \in D \subset \Re^n$, gdy zadanie z ograniczeniami.

Szukanie postępuje krok po kroku:

$x_{k+1} = x_k + \alpha_k p_k, \quad k = 0,1,2,\ldots$

p_k - wektor wyznaczający kierunek kroku,

α_k - liczba określająca długość kroku.

Oznaczenia

Funkcja celu $f : \Re^n \to \Re, \quad x \mapsto f(x), \quad x \notin \Re^n.$

Gradient

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Hesjan

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Metoda simpleksów Neldera-Meada

1. W przestrzeni *n* wymiarowej tworzymy wokół punktu x0 simpleks n+1 wymiarowy.
2. Porządkujemy punkty simpleksu tak, aby $f(x_i)<f(x_{i+1})$, i=1,...,n.
3. Generujemy punkt $r=2m-x_{n+1}$, gdzie $m=(x_1+...+x_n)/n$.
4. Jeżeli $f(x_1)<=f(r)<f(x_n)$, to akceptujemy r, **reflect.**
5. Jeżeli $f(r)<=f(x_1)$, to obliczamy s=m+2 i
   a. jeżeli f(s)<f(r), to akceptujemy s, **expand.**
   b. jeżeli nie, to akceptujemy r, **reflect.**
6. Jeżeli $f(r)>=f(x_n)$, to kontrakcja między m a lepszym(r,x_{n+1})
   a. jeżeli $f(r)<f(x_{n+1})$, to obliczamy c=m+(r-m)/2,
      jeżeli f(c)<f(r), to akceptujemy c, **contract outside,**
      jeżeli nie, to do punktu 7,
   b. jeżeli $f(r)>=f(x_{n+1})$, to obliczamy cc=m+(x_{n+1}-m)/2,
      jeżeli $f(cc)<f(x_{n+1})$, to akceptujemy cc, **contract inside,**
      jeżeli nie, to do punktu 7.
7. Obliczamy n punktów $v_i=x_1+(x_i-x_1)/2$, i=2,...,n+1, **shrink.**

Ew. unikanie degradacji.

Opis szczegółowy

1. Zacznij w dowolnym punkcie i zbuduj sympleks

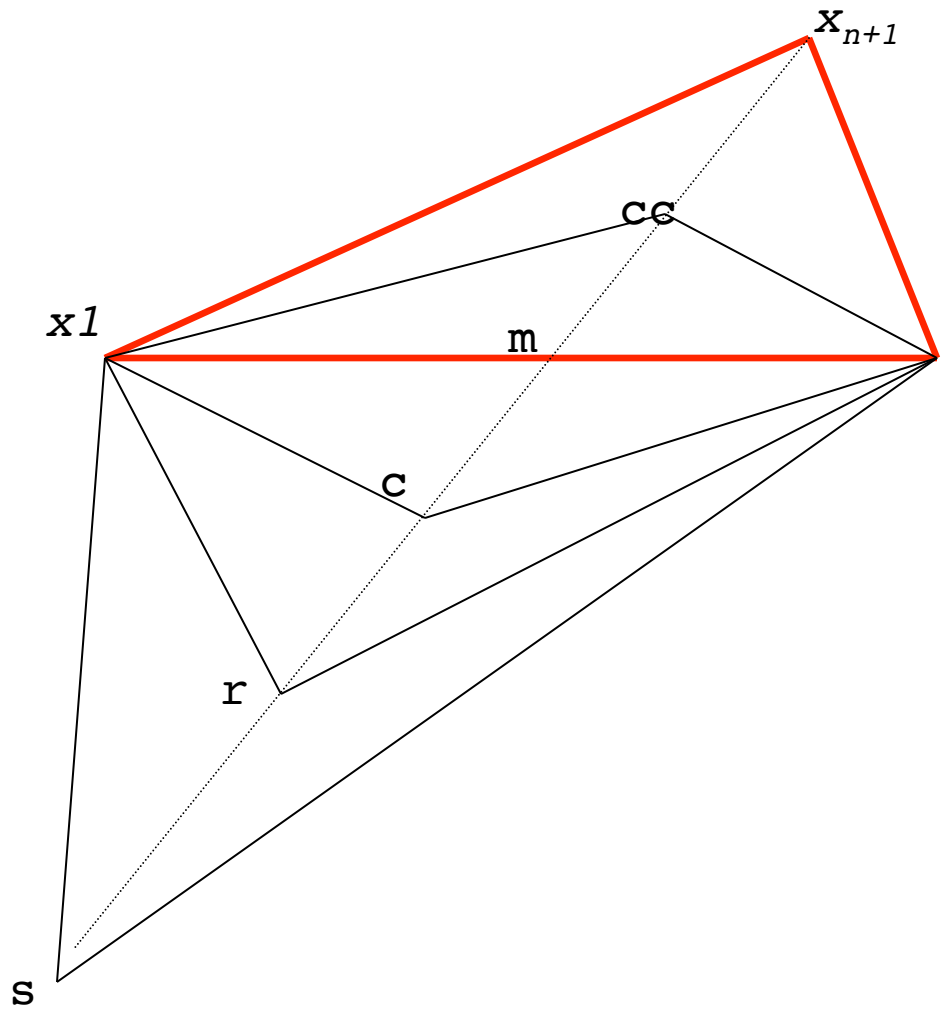
2. Zbadaj wartości funkcji w wierzchołkach sympleksu. Znajdź największą i najmniejszą.

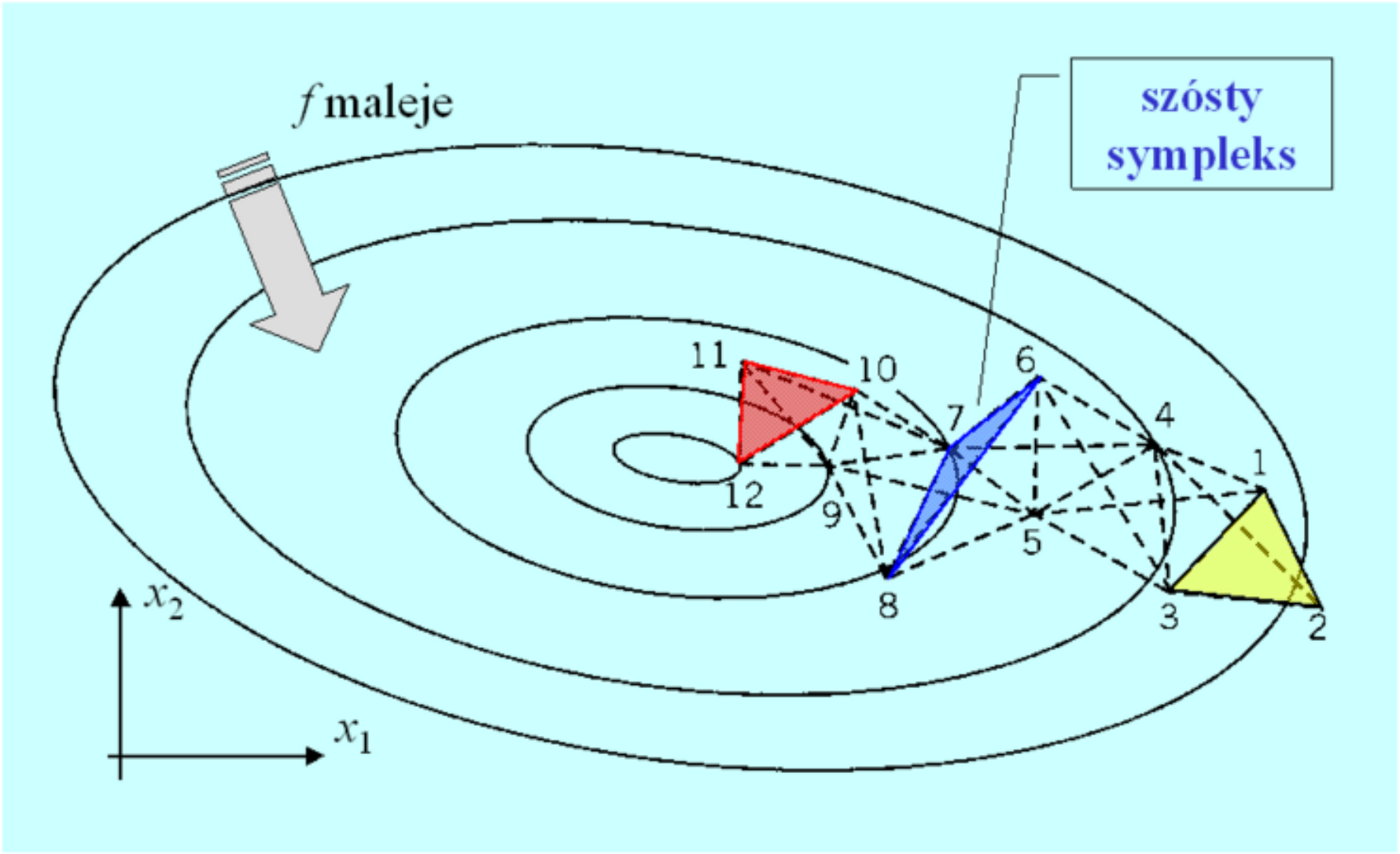
3. Punkt w którym jest największa wartość, zastąp nowym, tworząc przy okazji nowy sympleks. Jeśli ”zbliżasz się” do minimum, przy okazji zmniejsz sympleks, jeśli nie — zwiększ w kierunku, w którym spadają wartości.

4. Kontynuuj poszukiwania, aż sympleks będzie dostatecznie mały, by jego środek dobrze przybliżał minimum.

Algorytm Neldera i Meada jest też zwany algorytmem pełzającego sympleksu. W algorytmie tym otoczeniem punktu bieżącego nie jest kula ale **sympleks** - zbiór punktów o liczności o jeden większej niż wymiar przestrzeni poszukiwań.

Metody rozwiązywania układów równań nieliniowych
Metoda simpleks (Nelder-Mead)




f maleje
szósty
sympleks
11
10
6
7
4
1
12
9
5
8
3
2
x_2
x_1

Podstawowe schematy jednego kroku metod optymalizacji

A:

1. Wyznaczenie kierunku poszukiwania minimum p_k.
2. Wykonanie kroków (w tym kierunku) do minimum.

B: (*Trust Region*)

1. Określenie maksymalnej długości kroku.
2. Wyznaczenie kierunku, w którym należy wykonać krok.

Przejdźmy teraz do omówienia podejścia opartego na założeniu, że Algorytm **dysponuje *a priori* pewną wiedzą o lokalnym zachowaniu funkcji wyboru** (drugi sposób) i przyjmiemy dodatkowo, że jest na tyle odważny, że chce (i potrafi) tę wiedzę „rozciągnąć" na otoczenie dużo większe.

Metody oparte na takim rozumowaniu od połowy lat dziewięćdziesiątych XX w nazywa się:

Metodami obszaru zaufania

(Trust region methods)

Poszukiwanie na kierunku (*line search methods*)

Szukanie minimum na kierunku:

- Obliczanie wartości funkcji w kolejnych punktach, aż do napotkania wzrostu wartości funkcji.
- Interpolacja wielomianem Lagrange'a 3 ostatnich punktów.
- Przybliżone minimum w minimum paraboli.

Kryterium wyboru długości "kroku" α

- Warunek Wolfe'a:
  Redukcja wartości funkcji powinna być proporcjonalna do długości kroku oraz do pochodnej kierunkowej:

$$f(x_k + \alpha p_k) \le f(x_k) + c\alpha \nabla f_k^T p_k, \qquad \text{(nierówność Armijo)}$$

$$c \approx 10^{-4}$$

Wybór metody zależy od kosztu zmiany kierunku (zerowy albo obliczanie gradientu lub hesjanu).

Metoda spadku względem współrzędnych (Gauss-Seidel, *coordinate descent method*)

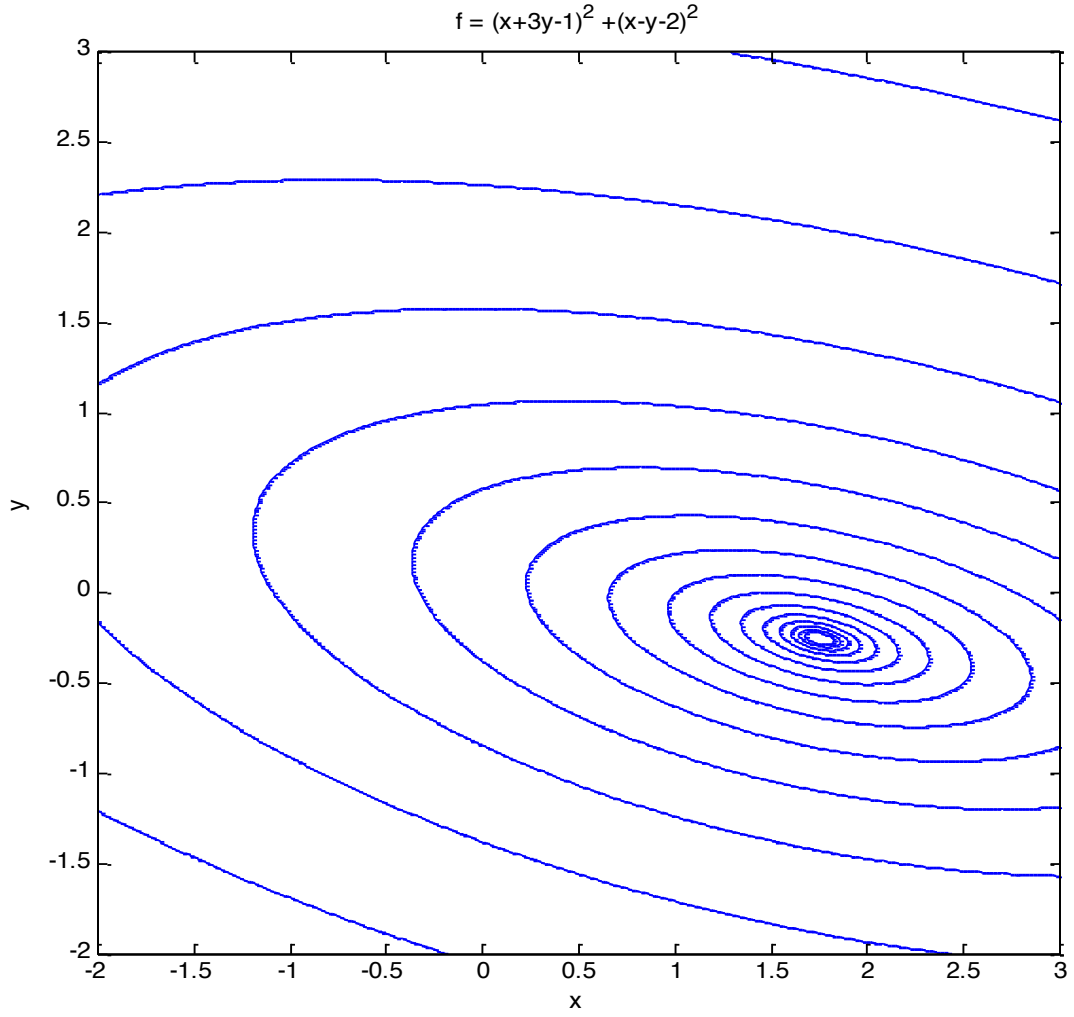
Grupa A,
Bez ograniczeń,
bezgradientowa:

Idea:

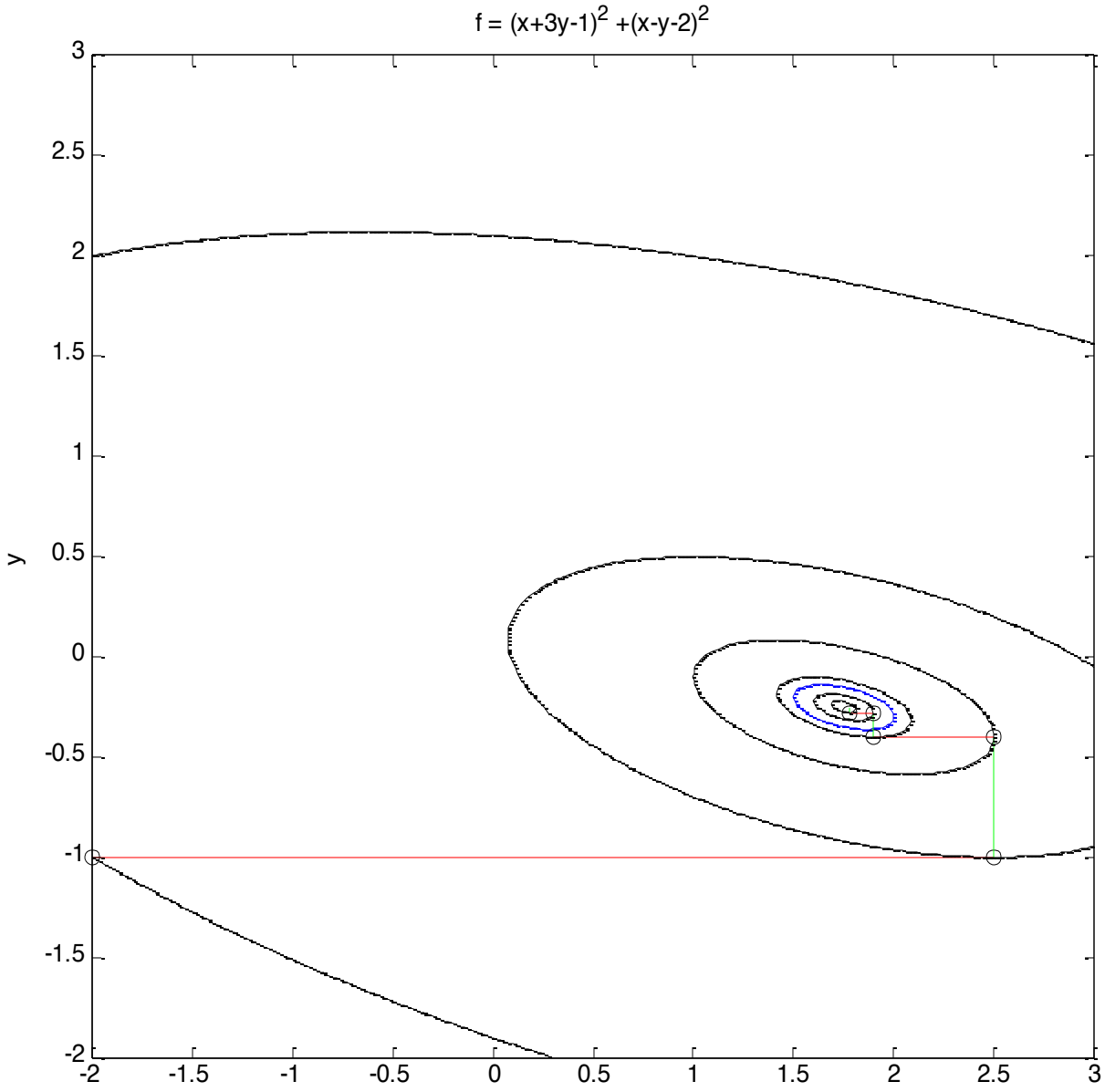
Szukanie minimum na kierunkach kolejnych współrzędnych.

Przykład: $f(x) = (x_1 + 3x_2 - 1)^2 + (x_1 - x_2 - 2)^2$

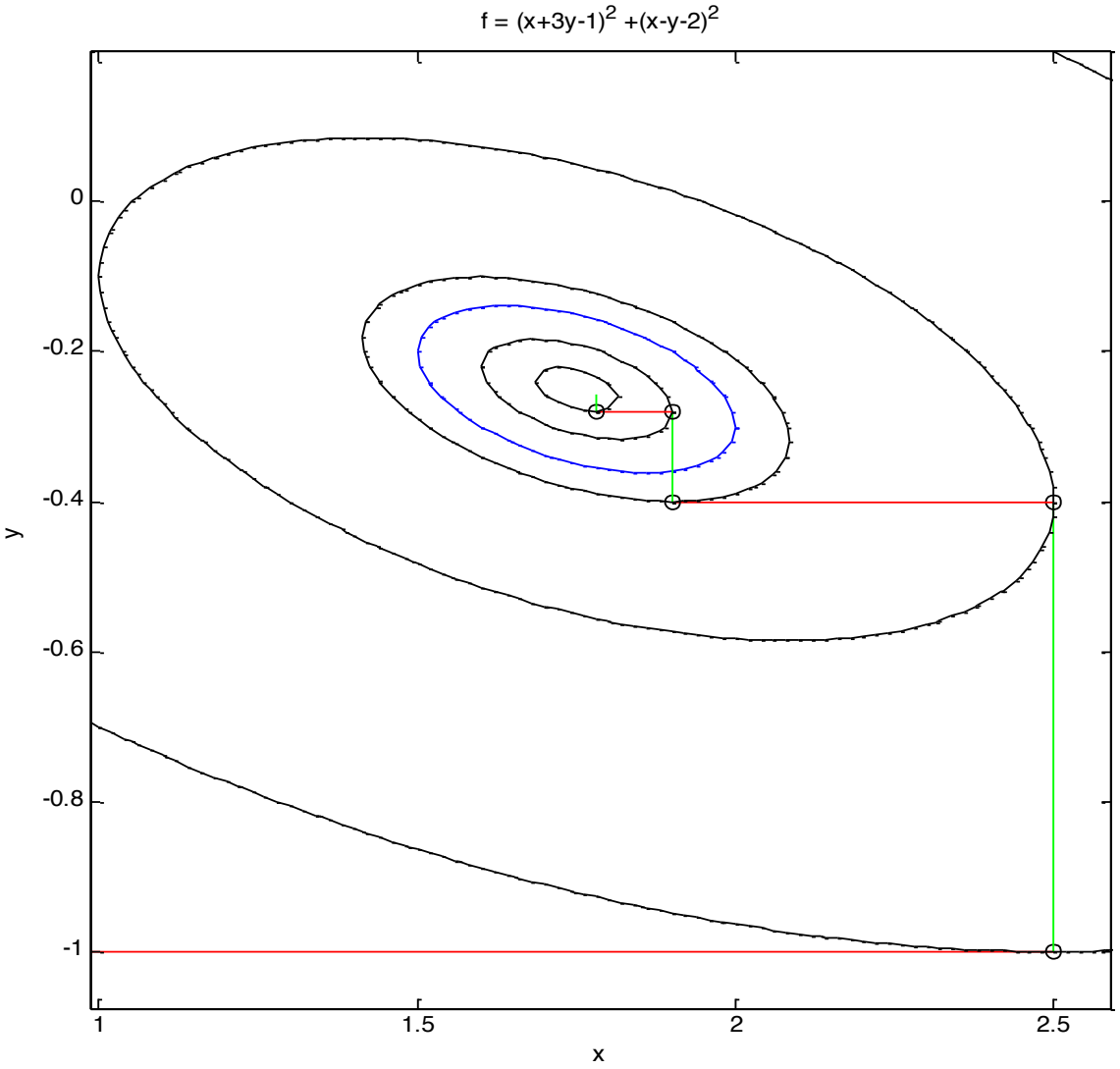
$$x_{\min} = \begin{bmatrix} \frac{7}{4} \\ -\frac{1}{4} \end{bmatrix}$$



Metoda spadku względem współrzędnych



Metoda spadku względem współrzędnych



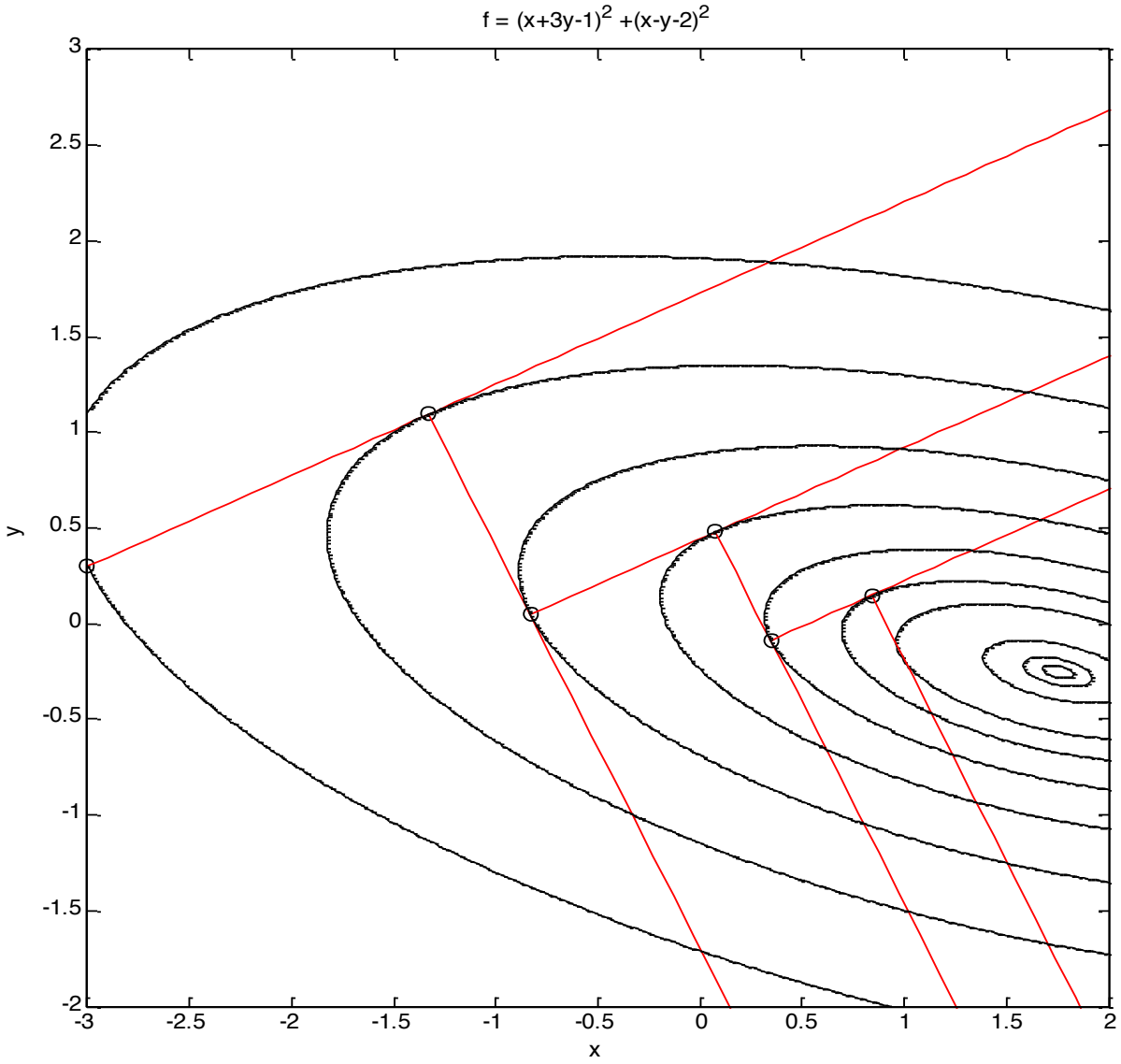
Metoda najszybszego spadku (*steepest descent*)

Grupa A
Bez ograniczeń
gradientowa (pierwszego rzędu)

Idea:

$$p_k = -\nabla f\left(x_k\right).$$

Metoda najszybszego spadku



Metoda gradientów sprzężonych
Conjugate gradient (CG) method

Grupa A
Bez ograniczeń
Gradientowa (pierwszego rzędu)

Idea:

$$p_k = -\nabla f\left(x_k\right) + \beta_k p_{k-1}.$$

Numeryczne metody optymalizacji

Metoda Newtona

- Szybka -metoda gradientowa drugiego rzędu.

- Skuteczna
  - w pobliżu minimum (ogólnie - gdy model 2. rzędu dobrze przybliża funkcję),
  - gdy hesjan jest macierzą dodatnio określoną.

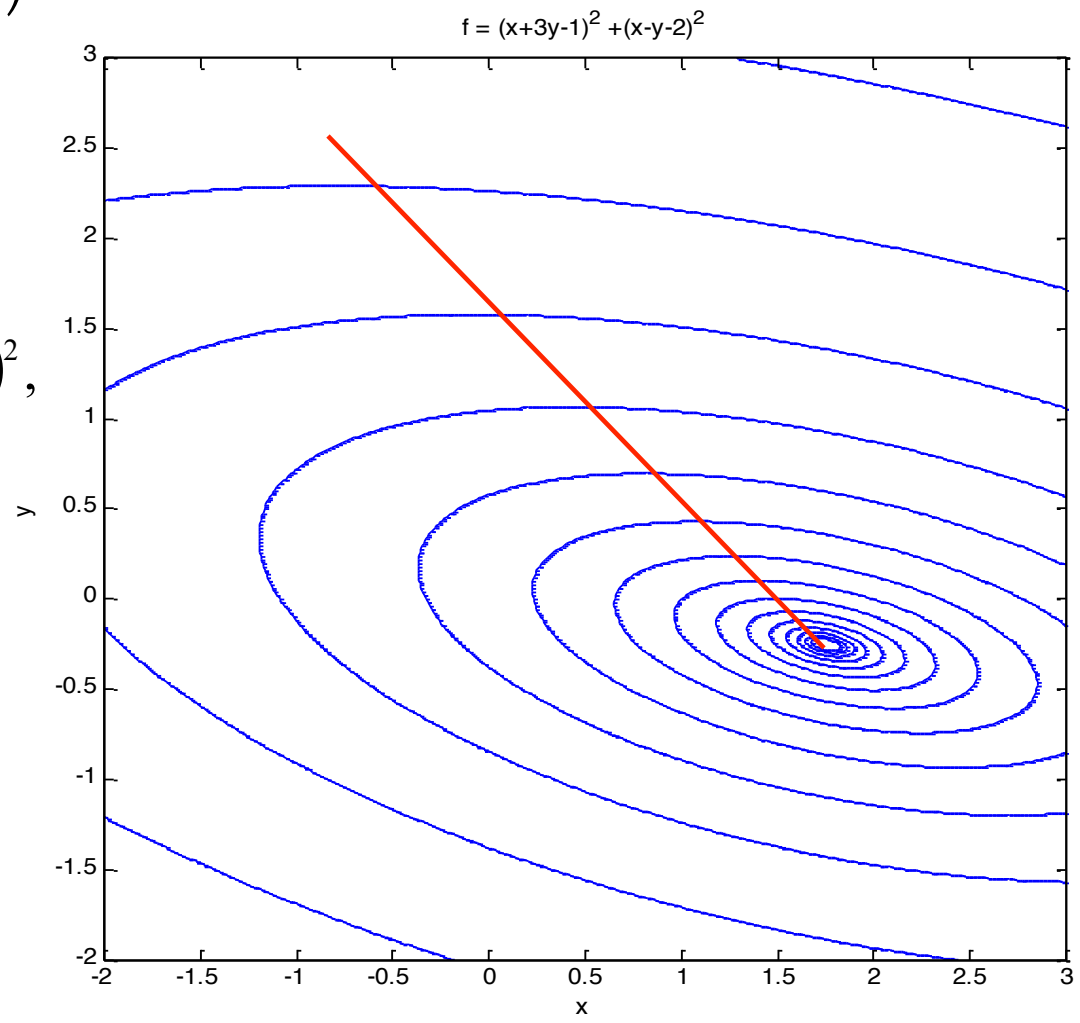
- Kosztowna - wymaga częstego obliczania hesjanu.

Metoda Newtona (Grupa A)

$$p^{(k)} = -\nabla^2 f^{-1}\left(x^{(k)}\right)\cdot \nabla f\left(x^{(k)}\right)$$
$$x^{(k+1)} = x^{(k)} + p^{(k)}$$

Przykład:

$$f(x) = \left(x_1 + 3x_2 - 1\right)^2 + \left(x_1 - x_2 - 2\right)^2,$$
$$\nabla f = \begin{bmatrix} 4x_1 + 4x_2 - 6 \\ 4x_1 + 20x_2 - 2 \end{bmatrix},$$
$$\nabla^2 f = \begin{bmatrix} 4 & 4 \\ 4 & 20 \end{bmatrix},$$
$$\nabla^2 f^{-1} = \tfrac{1}{16}\begin{bmatrix} 5 & -1 \\ -1 & 1 \end{bmatrix},$$
$$p = -\begin{bmatrix} x_1 - \frac{7}{4} \\ x_2 + \frac{1}{4} \end{bmatrix}.$$



Numeryczne metody optymalizacji (Grupa A)

Quasi-Newton

- Zbieżność superliniowa (rząd > 1).
- Mniejszy koszt - nie wymagają obliczania hesjanu.
- Hesjan zastąpiony macierzą $\boldsymbol{B_k}$ - uaktualnianą informacją o funkcji uzyskaną w kolejnych krokach - zmiana gradientu wzdłuż kierunku poszukiwań dostarcza przybliżoną informację o drugiej pochodnej.

Najbardziej popularna **BFGS**
(Broyden, Fletcher, Goldfarb,Shanno):

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \quad s_k = x_{k+1} - x_k, \quad y_k = B_{k+1} s_k.$$

Trust-Region Methods

(Grupa B)

Idea:

- Budujemy model otoczenia punktu x_k

$$m_k(p) = f_k + \nabla f_k^T p + \tfrac{1}{2} p^T B_k p$$

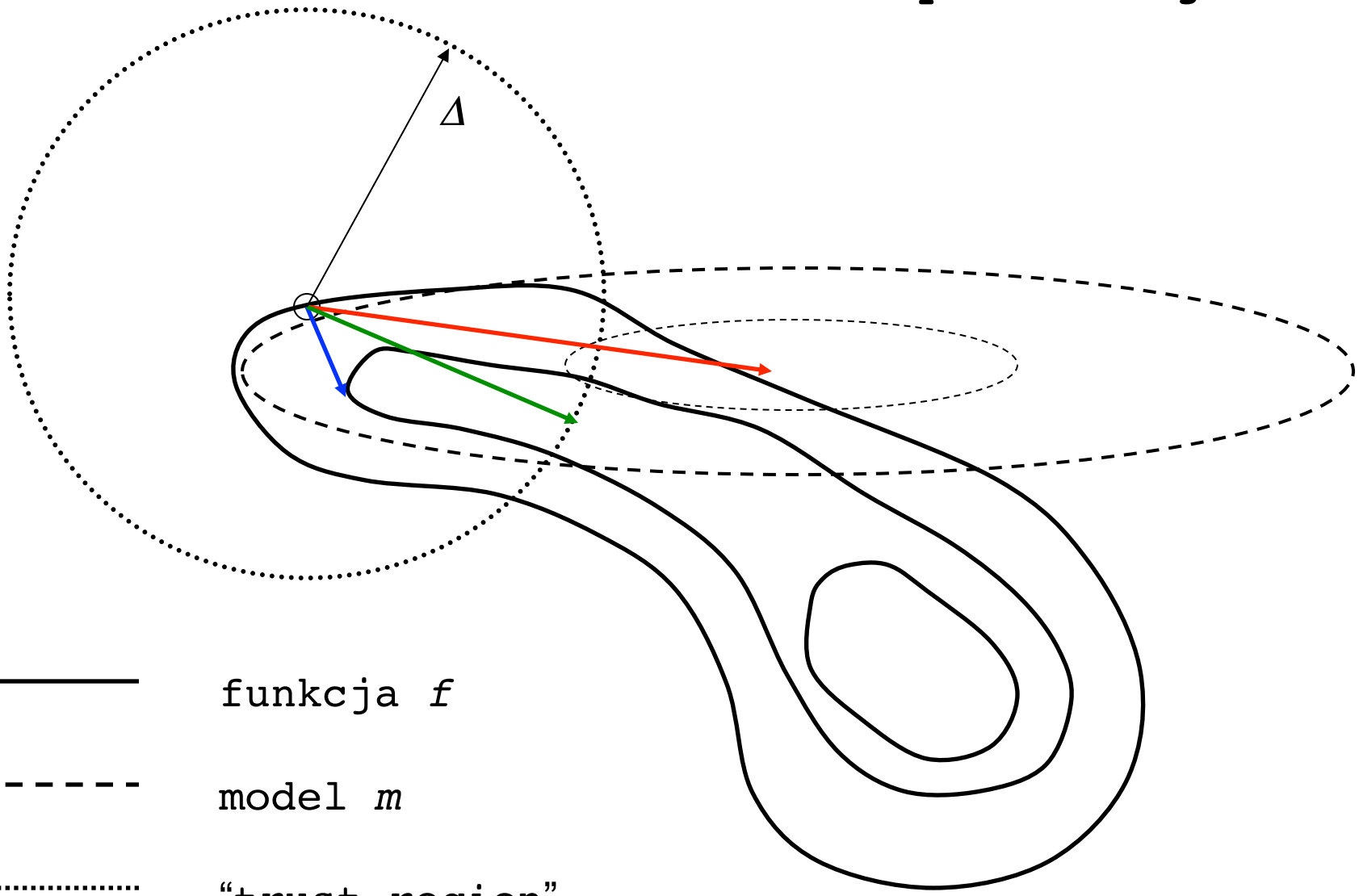
B_k- *wg Newtona, CG, BFGS itp..*

- Szukamy minimum w tym otoczeniu

$$\min_{p \in \Re^n} m_k(p), \quad \|p\| \le \Delta_k .$$

- Promień Δ ustalany na podstawie zgodności modelu i funkcji celu w poprzednim kroku.

Idea metody Trust-Region



Trust-Region Methods

Metoda *Dogleg*

Schemat algorytmu:

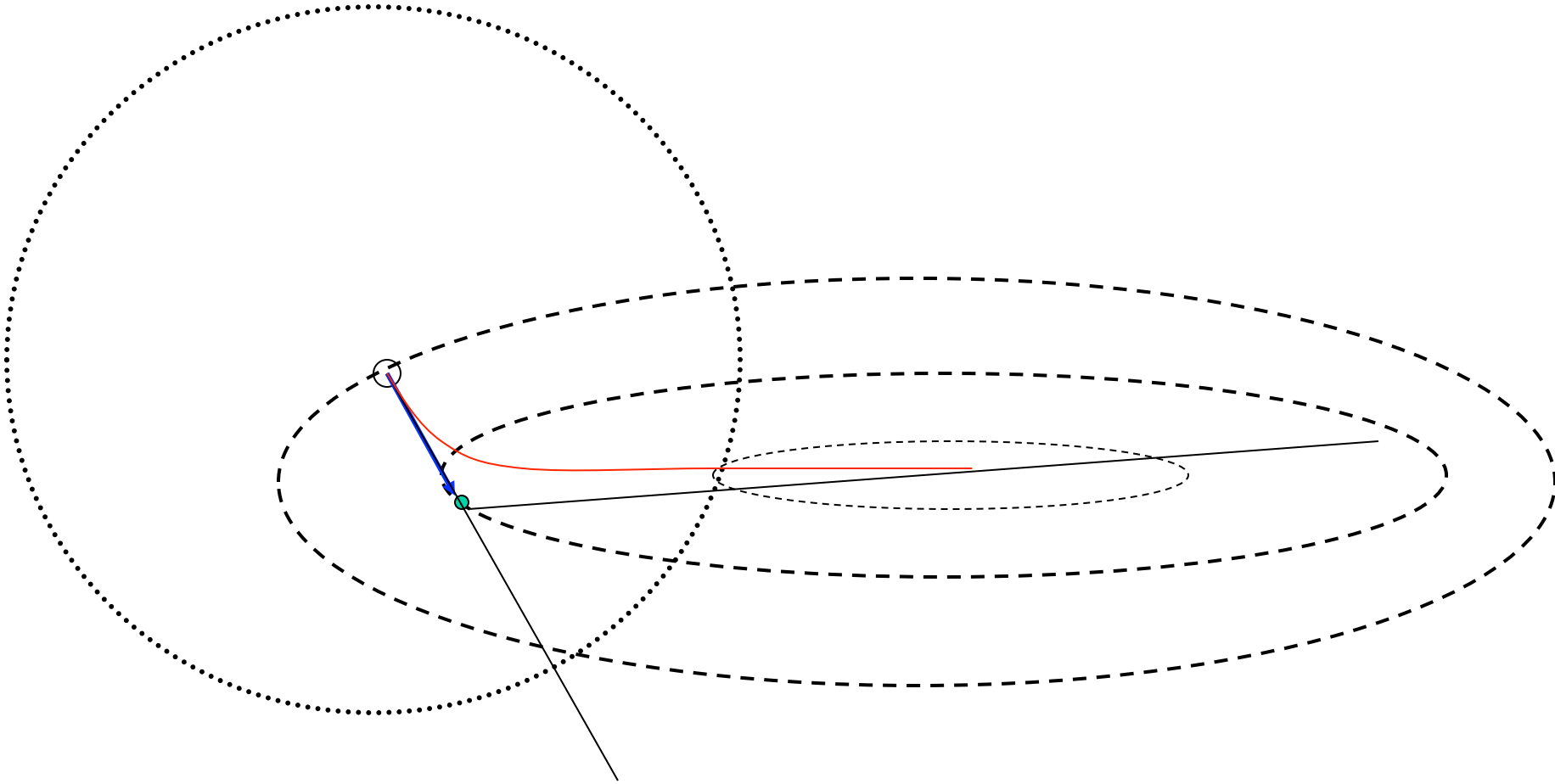
- Wyznaczamy punkt Cauchy'ego (punkt na kierunku antygradientu, w którym model ma minimum).

- Drogę do punktu x_{k+1} składamy z odcinka do punktu Cauchy'ego i odcinka od punktu Cauchy'ego do minimum modelu.

- Jeżeli wiarygodność modelu jest mała, to dominuje najszybszy spadek.
- Jeżeli wiarygodność jest duża, to dominuje model 2. rzędu.

- Są metody bardziej zaawansowane np. **Levenberga - Marquardta**

Ilustracja kroku metody metody *Dogleg*



Numeryczne metody optymalizacji z ograniczeniami

- ograniczenia równościowe i nierównościowe,

- Metody:
  * eliminacja zmiennych,
  * metody z funkcją kary,
  * metoda Lagrange'a

- Warunek konieczny 1. rzędu
- Warunek 2. rzędu

- Quadratic Programming
- SQP - Sequential Quadratic Programming

Kryterium stopu

Minimum lokalne a globalne.

W zastosowaniu do rozwiązywania układów równań nieliniowych - istotna jest znajomość funkcji celu punkcie minimalnym.

“Numeryczna niejednoznaczność” zera
- błąd numeryczny obliczania wartości funkcji.

Numeryczne metody optymalizacji

Metody niedeterministyczne:

- Monte Carlo,
- symulowane wyżarzanie,
- algorytmy genetyczne i ewolucyjne,
- algorytmy rojowe,
- wykorzystujące sztuczne sieci neuronowe.

O czym należy pamiętać:

- Algorytm poszukiwania minimum w kierunku.
- Podstawowe schematy jednego kroku metod optymalizacji.
- W których metodach jest obliczany gradient funkcji.
- W których metodach jest obliczany hesjan.
- Ogólny schemat algorytmu:
  - Neldera-Meadʹa,
  - spadku wzdłuż współrzędnych,
  - najszybszego spadku,
  - gradientu sprzężonego,
  - Newtona,
  - quasi-newtonowskiego (np.BFGS),
  - trust-region,
  - Monte-Carlo.
- Zalety i wady poszczególnych metod.

OPTYMALIZACJA W ŚRODOWISKU MATLAB

1. Cel ćwiczeń

Celem ćwiczeń jest zaznajomienie studentów z podstawową obsługą środowiska obliczeń inżynierskich Matlab oraz zapoznanie się z możliwościami przeprowadzenia procesu optymalizacji w tym środowisku. Po ukończeniu tych zajęć student powinien umieć samodzielnie utworzyć plik funkcyjny zawierający funkcję celu, zdefiniować ograniczenia oraz uruchomić procedurę optymalizacji.

2. MATLAB – podstawowe operacje

a) Definiowanie macierzy i wektorów

X=[1 3 2 8] – przez podanie kolejnych elementów,
X=[1:7] – przez podanie zakresu,
X=[1:2:13] – przez podanie zakresu oraz skoku,
X=[1 3 2 8; 2 4 5 6; 3 2 1 8] – podawanie elementów,
X=[-5:1; 1:7; 1:2:13] – operowanie zakresem i skokiem

b) Odwołania do elementów macierzy i wektorów

X(1,2) – odwołanie do jednego elementu,
X(1:2,4:7) – odwołanie zakresowe,
X(1:2,1:2:7) – odwołanie zakresowe ze skokiem

c) Składanie macierzy i wektorów

A=[1:4, 4:7],
B=[2:2:8, 10:10:40],
C=[A;B] – składanie w pionie,
C=[A,B] – składanie w poziomie,
C=[A,B;B,A] – składanie mieszane

d) Definiowanie macierzy z wykorzystaniem funkcji systemu Matlak

X=eye(5) – macierz jednostkowa 5x5,
X=ones(5) – macierz jedynkowa 5x5 ,
X=zeros(5) - macierz zerowa 5x5,
X=rand(5) - macierz losowa 5x5,
X=randn(5) - macierz losowa o rozkładzie normalnym 5x5,
X=linspace(x1,x2,N) – wektor liniowy,
X=logspace(x1,x2,N) – wektor logarytmiczny

e) Manipulacje macierzami

A=rot90(B) – obrót,
A=flipr(B) – odbicie w pionie,

A=flipuprot90(B) – odbicie w poziomie,
A=reshape(B,2,5) – połamanie wektora,
A=diag(B) – macierz diagonalna

f) Działania na macierzach i wektorach

C=A+/-B – dodawanie i odejmowanie macierzy,
C=A*B – mnożenie macierzy,
C=A.*B – mnożenie tablicowe macierzy,
C=A' – transponowanie macierzy,
C=A^(-1) – odwrotność macierzy,
C=A.^2 – podniesienie do potęgi elementów macierzy,
C=A./B – dzielenie tablicowe

g) Inne funkcje macierzowe

Rank(A) – rząd macierzy
[m,n]=size(A) – wymiar macierzy,
N=length(X) – długość wektora,
Inv(A) – macierz odwrotna,

h) Liczby zespolone

5+6i, 5+6*sort(-1) – deklaracja liczby zespolonej,
abs(z) – moduł liczby zespolonej,
angle(z) – argument liczby zespolonej,
real(z) – część rzeczywista liczby zespolonej,
imag(z) – część urojona liczby zespolonej,
conj(z) – sprzężenie liczby zespolonej,

Dokładniejszy opis poszczególnych funkcji oferowanych przez system Matlab znaleźć można w rozległej literaturze [1,2,3,4] oraz systemie pomocy Matlaba.

3. OPTYMALIZACJA bez ograniczeń - fminunc

Zadanie:
Znajdź minimum nieograniczonej funkcji wielu zmiennych $\min_x f(x)$, gdzie x jest wektorem zmiennych decyzyjnych a f(x) funkcją celu.

Składnia funkcji fminunc:

x = fminunc(fun,x0)
x = fminunc(fun,x0,options)
x = fminunc(fun,x0,options,P1,P2,...)
[x,fval] = fminunc(...)
[x,fval,exitflag] = fminunc(...)
[x,fval,exitflag,output] = fminunc(...)
[x,fval,exitflag,output,grad] = fminunc(...)
[x,fval,exitflag,output,grad,hessian] = fminunc(...)

Wyjaśnienie występujących oznaczeń:

- fun – funkcja celu, musi być zdefiniowana w pliku funkcyjnym matlaba,
- x – wektor zawierający optimum,
- x0 – punkt startowy optymalizacji,
- fval – wartość funkcji celu odpowiadająca w optimum,
- exitflag – znacznik wyjścia,
- output – zawiera ogólne informacje dotyczące procesu optymalizacji,
- grad – gradient,
- hessian – wartość hessiana funkcji celu
- options – opcje optymalizacji, ustawiane poleceniem optimset

Opcje optymalizacji – funkcja optimset:

- LargeScale – On/Off,
- Display – On/Off
- TolX – tolerancja na wektor X,
- TolFun – tolerancja funkcji celu

4. OPTYMALIZACJA z definicją gradientu

Ten typ optymalizacji pozwala zdefiniować dodatkowo gradient funkcji celu.

Składnia:
[x,fval,exitflag,output]=fminunc({@fcelu,@fgradientu},x0,options)

Jedyną różnicą pomiędzy tym rodzajem optymalizacji, a zwykłą optymalizacją bez ograniczeń jest konieczność zdefiniowania funkcji gradientu. Jest to osobny m-plik funkcyjny zawierający wyrażenia na pochodne cząstkowe funkcji celu po poszczególnych zmiennych decyzyjnych.

5. OPTYMALIZACJA z ograniczeniami - fmincon

Zadanie:
Znajdź minimum ograniczonej funkcji wielu zmiennych $\min_x f(x)$, gdzie x jest wektorem zmiennych decyzyjnych a f(x) funkcją celu oraz:

$$c(x) \le 0$$
$$ceq(x) = 0$$
$$A \cdot x \le b$$
$$Aeq \cdot x \le beq$$
$$lb \le b \le ub$$

gdzie x, b, beq, lb oraz ub są wektorami, A i Aeq są macierzami, c(x) i ceq(x) są funkcjami zwracającymi wektory oraz f(x) jest funkcją zwracającą skalar. Funkcje f(x), c(x) i ceq(x) mogą być funkcjami nieliniowymi.

Składnia funkcji:

x = fmincon(fun,x0,A,b,Aeq,beq)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2, ...)
[x,fval] = fmincon(...)
[x,fval,exitflag] = fmincon(...)
[x,fval,exitflag,output] = fmincon(...)
[x,fval,exitflag,output,lambda] = fmincon(...)
[x,fval,exitflag,output,lambda,grad] = fmincon(...)
[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)

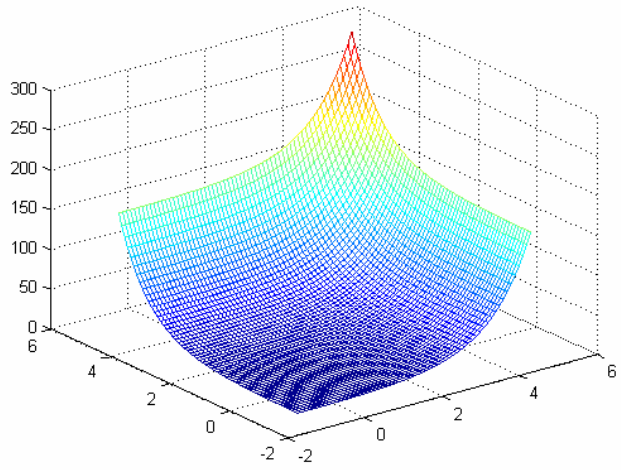
Funkcja nonlcon może być użyta do wygodnego definiowania wszystkich ograniczeń nieliniowych w osobnym pliku funkcyjnym, zamiast ich definiowania podczas wywoływania procedury optymalizacji.

6. Przykłady

Przykład 1

Znajdź minimum funkcji $y = e^{x_1} + e^{x_2}$, dla następujących ograniczeń równościowych: x(1)^2+x(2)^2-9=0 oraz nierównościowych $-x_1-x_2+1<0$, $-x_1<0$; $x_2>0$

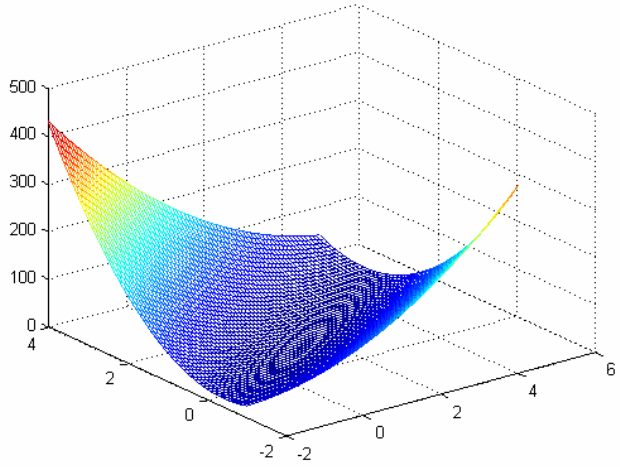
Rozwiązanie: x=[2.12 2.12], wartość funkcji celu = 16.6843



Przykład 2

Znajdź minimum funkcji
$y = (x_1 - 2)^2 + (x_2 - 1)^2 +$
$+ 0.04 \cdot \left(\frac{x_1^2}{4} - x_2^2 + 1 \right) + 5 \cdot (x_1 - 2x_2 + 1)^2$
w granicach x_1=[-2:0.1:5] i x_2=[-1:0.1:4];

Rozwiązanie: x=[1.8442, 1.4047], wartość funkcji celu = 0.1212



Przykład 3

Znajdź minimum funkcji danej wzorem:

$f(x, y) = 100\left(y - x^2\right)^2 + (1 - x)^2$

Jest to funkcja, którą trudno jest zoptymalizować. Dlatego należy zastosować optymalizację z gradientem.

Punkt startowy: x0=[-1.9, 2];

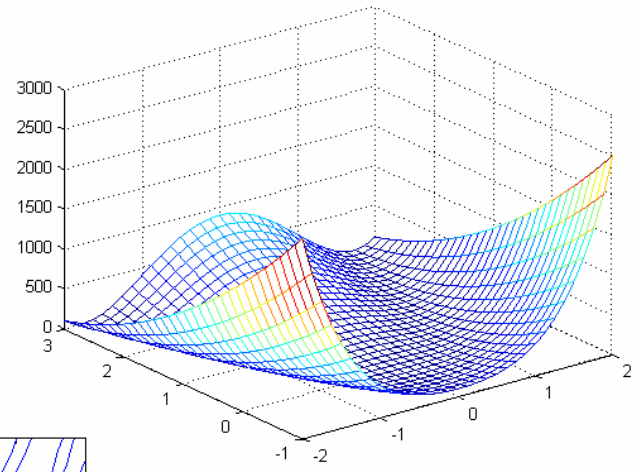
Minimum należy poszukiwać w granicach $x \in < -2,2 >$, $y \in < -1,3 >$

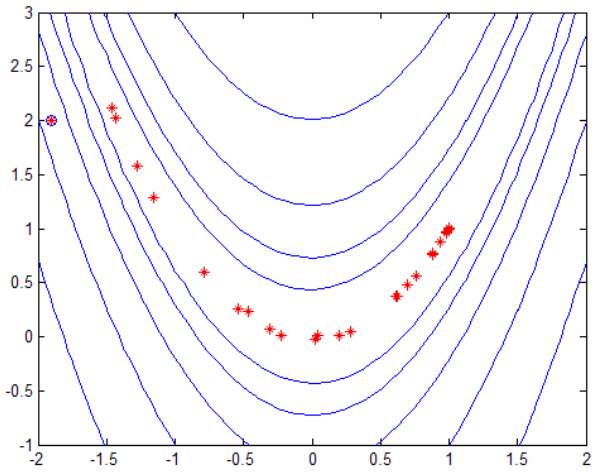
W celu rozwiązania zadania należy wyznaczyć pochodne cząstkowe funkcji celu:

$$\frac{\partial f}{\partial x} = -400\left(y - x^2\right)x - 2(1 - x)$$

$$\frac{\partial f}{\partial y} = 200\left(y - x^2\right)$$

Przed wywołaniem funkcji optymalizującej należy poprawnie ustawić opcje optymalizacji:
Options=optimset('largescale','off','linesearchtype','cubicpoly','gradobj','on')





Wyniki optymalizacji: wartość funkcji celu w optimum 8.9856e-009, dla x = 1.0001 oraz y = 1.0002

7. Zadania do samodzielnego wykonania.

Zadanie 1

Znajdź optimum funkcji $f(x, y) = e^{x^2 + 2y^2} - 10xy$ w granicach x = [-1.1:1.1], y = [-1:1].

Zadanie 2

Znajdź optimum funkcji $f(x, y) = \dfrac{\left(y^2 - 0.5\right)^2}{\left(0.132 - e^{-(x-1)^2}\right)}$ w granicach x = [0:2], y = [-1.1:1.1].

Zadanie 3

Znajdź optimum funkcji $f(x, y) = 2x^3 + y^2 + x^2 y^2 + 4y + 3$ w granicach x = [4:13], y = [0:10].

8. Literatura

[1] A.Zalewski, R.Cegieła: Matlab – obliczenia numeryczne i ich zastosowania. Wydawnictwo Nakom, Poznań 1996

[2] W. Regel: Obliczenia symboliczne i numeryczne w programie Matlab. Wydawnictwo Mikom, Warszawa 2004

[3] M.Czajka: Matlab – ćwiczenia. Wydawnictwo Helion, Gliwice 2005

[4] B.Mrozek, Z.Mrozek: Matlab – uniwersalne środowisko do obliczeń naukowo technicznych. Wyd.3, Warszawa 1996