Teoria i metody optymalizacji

Oleksandr Sokolov

Wydział Fizyki, Astronomii i Informatyki Stosowanej
UMK
http://fizyka.umk.pl/~osokolov/TMO/

Optymalizacja

Optymalizacja – metoda wyznaczania najlepszego (optymalnego) rozwiązania (**poszukiwanie ekstremum funkcji/funkcjonału**) z punktu widzenia określonego kryterium (wskaźnika) jakości (np. kosztu, drogi, wydajności).

Stosuje się **optymalizacje jedno i wielokryterialne**.
Optymalizacja wielokryterialna występuje w wielu różnych dziedzinach: w projektowaniu produktu i procesu produkcji, finansów, projektowaniu samolotów, w przemyśle chemicznym, projektowaniu samochodów, wszędzie tam gdzie optymalne decyzje muszą być podjęte w obecności ***kompromisów pomiędzy dwoma lub więcej sprzecznymi celami***. Przykładem wielokryterialnej optymalizacji jest **maksymalizacja zysków i minimalizacji kosztów** produktu, maksymalizacja wydajności przy ograniczaniu zużycia paliwa pojazdu, czy też obniżenie masy urządzenia przy jednoczesnej maksymalizacji wytrzymałości poszczególnych jego komponentów.

Optymalizacja. Ogólna postać

Optymalizacja – problem polegający na znalezieniu **ekstremum zadanej funkcji celu**.

Zadania optymalizacji dzielimy na dwie podstawowe klasy:

optymalizację statyczną (sprowadzającą się do poszukiwania ekstremum funkcji)

$$y(x) = f(x)$$

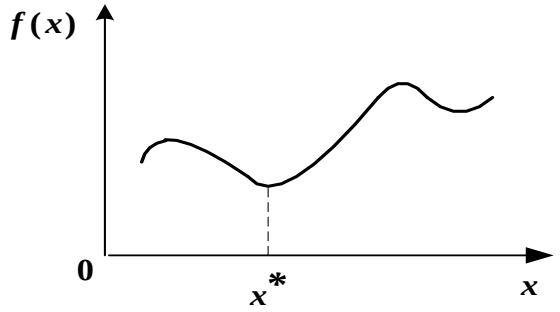
(niewiadomy jest **wektor x**)
oraz
optymalizację dynamiczną, sprowadzającą się do poszukiwania ekstremum funkcjonału

$$y\left(g\left(x\right)\right) = f\left(g\left(x\right)\right)$$

(niewiadoma jest **funkcja g(x)**)

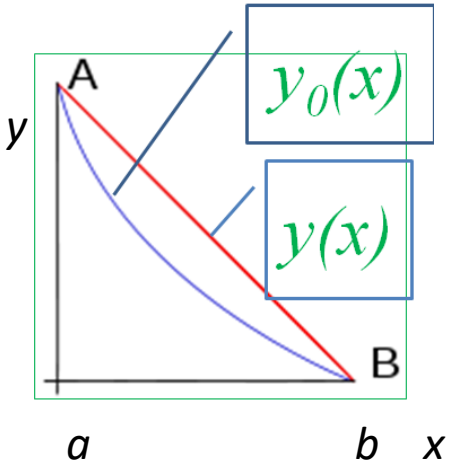
Optymalizacja statyczna vs dynamiczna

$$\min_{x} f(x) = x^*$$



$$J\left(y(x)\right) = \int_a^b F(x, y(x), y'(x))dx$$

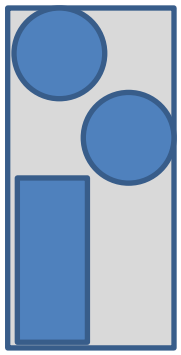
$$\min_{y(x)} J\left(y(x)\right) = y_0(x)$$



Maksymalizacja objętości

Producent cylindrycznych pojemników chce **zmaksymalizować objętość** produkcji V dla danej powierzchni zastosowanego materiału S_0.

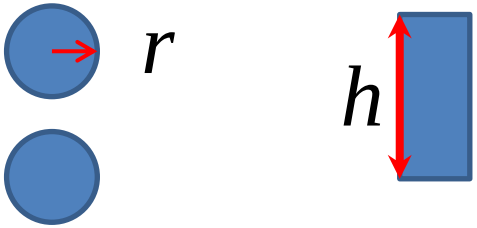
S_0



V

$$V(r,h) = \pi r^2 h \to \max$$

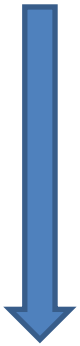
$$S(r,h) = 2\pi r^2 + 2\pi rh = S_0,$$

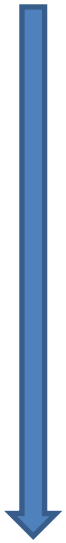


Funkcja optymalizacji

$$2\pi r^2 + 2\pi rh = S_0,$$

$$V(r,h) = \pi r^2 h$$





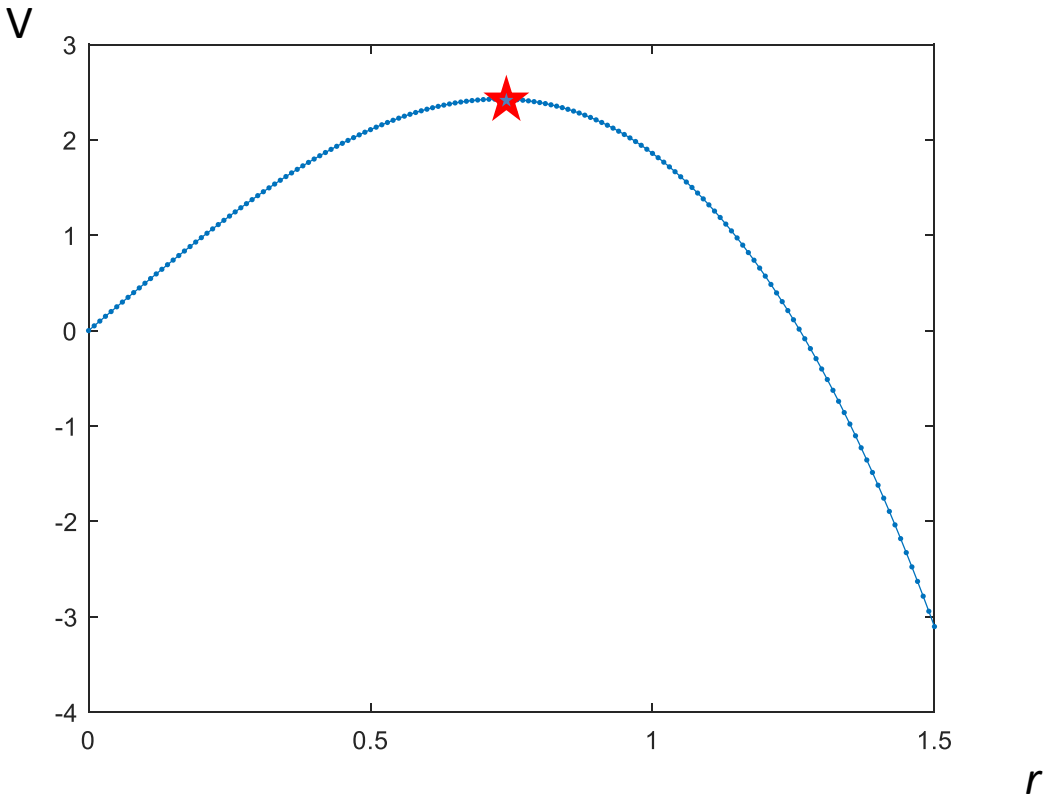
$$h = (S_0 - 2\pi r^2)/2\pi r$$

$$V(r) = \pi r^2[(S_0 - 2\pi r^2)/2\pi r] = \frac{r}{2}S_0 - \pi r^3$$

Zagadnienie optymalizacji

$$V(r^*) = \frac{r^*}{2} S_0 - \pi r^{*3}$$

$$\max_r V(r)$$



Rozwiązanie analityczne

$$V(r^*) = \frac{r^*}{2} S_0 - \pi r^{*3}$$

$$\frac{dV(r)}{dr} = \frac{S_0}{2} - 3\pi r^2 = 0,$$

$$r^* = \sqrt{S_0 / 6\pi}$$

```
>> r=sqrt(S0/(6*pi))
r=    0.7284
```

```
>> V=S0/2-3*pi*r^3
V =    1.3582
```

$$h^* = (S_0 - 2\pi r^{*2}) / 2\pi r^* \Rightarrow h^* = \sqrt{\frac{2S_0}{3\pi}}$$

```
>> h=sqrt(2*S0/(3*pi))
h =    1.4567
```

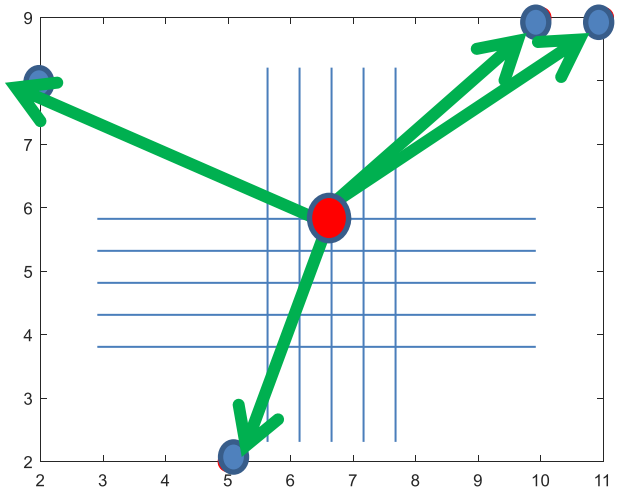
$$q = \frac{h^*}{r^*} = \sqrt{\frac{2S_0}{3\pi}} \Big/ \sqrt{S_0 / 6\pi} = \sqrt{4} = 2$$

Ciekawy praktyczny wynik

Lokalizacja centrum handlowego

Aby zapewnić wygodę wszystkim mieszkańcom dzielnicy, konieczne jest takie ustawienie centrum handlowego, aby **całkowita odległość między nim a obszarami mieszkalnymi była minimalna**.

Osiedle	Współrzędne	
	Oś x_1: a_k	Oś x_2: b_k
Osiedle1	2	8
Osiedle2	10	9
Osiedle3	5	2
Osiedle4	11	9

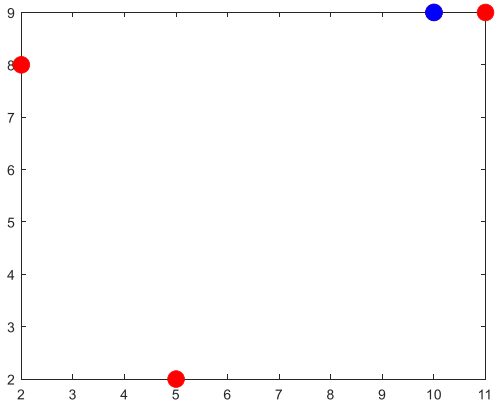


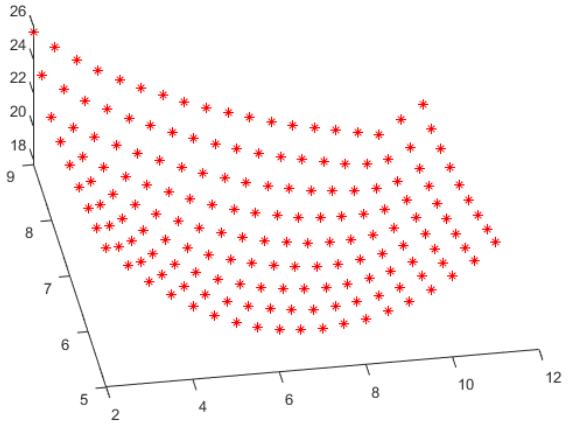
$$f(x) = \sum_{k=1}^{n} \sqrt{(x_1 - a_k)^2 + (x_2 - b_k)^2} \rightarrow \min$$

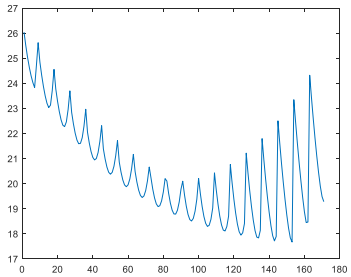
Osiedle1.m

Wyszukiwanie wyczerpujące

```
a=[2 10 5 11];
b=[8 9 2 9];

Out=[];
for x1=2:0.5:11
   for x2=5:0.5:9
      %sum(sqrt((x1-a).^2+(x2-b).^2))
      f=0;
      for i=1:4
         f=f+sqrt((x1-a(i))^2+(x2-b(i))^2);
      end;

      Out=[Out;x1 x2 f];
   end;
end;
[v i]=min(Out(:,3));
Out(i,:)
plot(a,b,'.r','MarkerSize',40);
hold on;
plot(Out(i,1),Out(i,2),'.b','MarkerSize',40);
```







plot(Out(:,3))

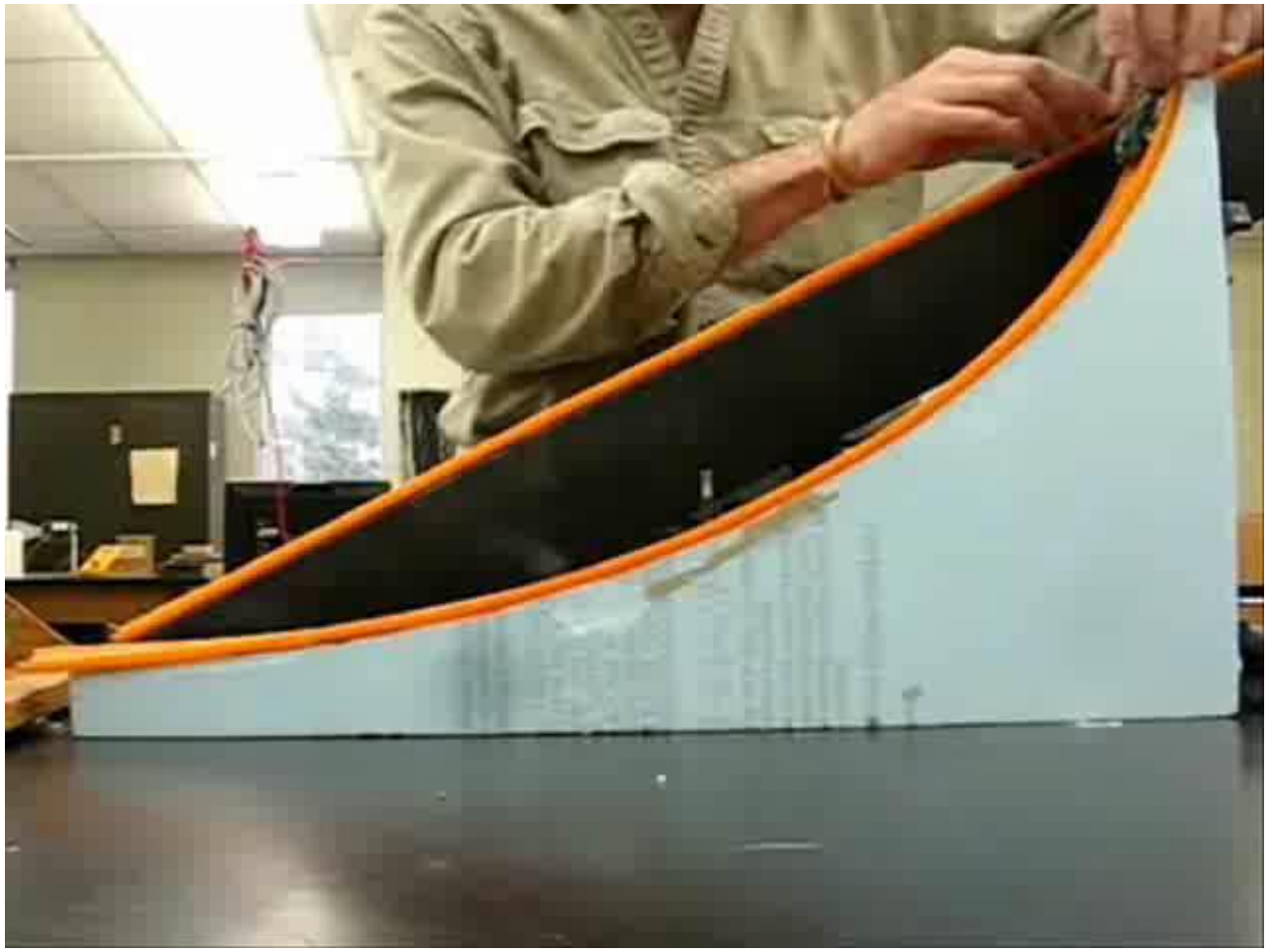
plot3(Out(:,1),Out(:,2),Out(:,3),'r*')

Osiedle1.m

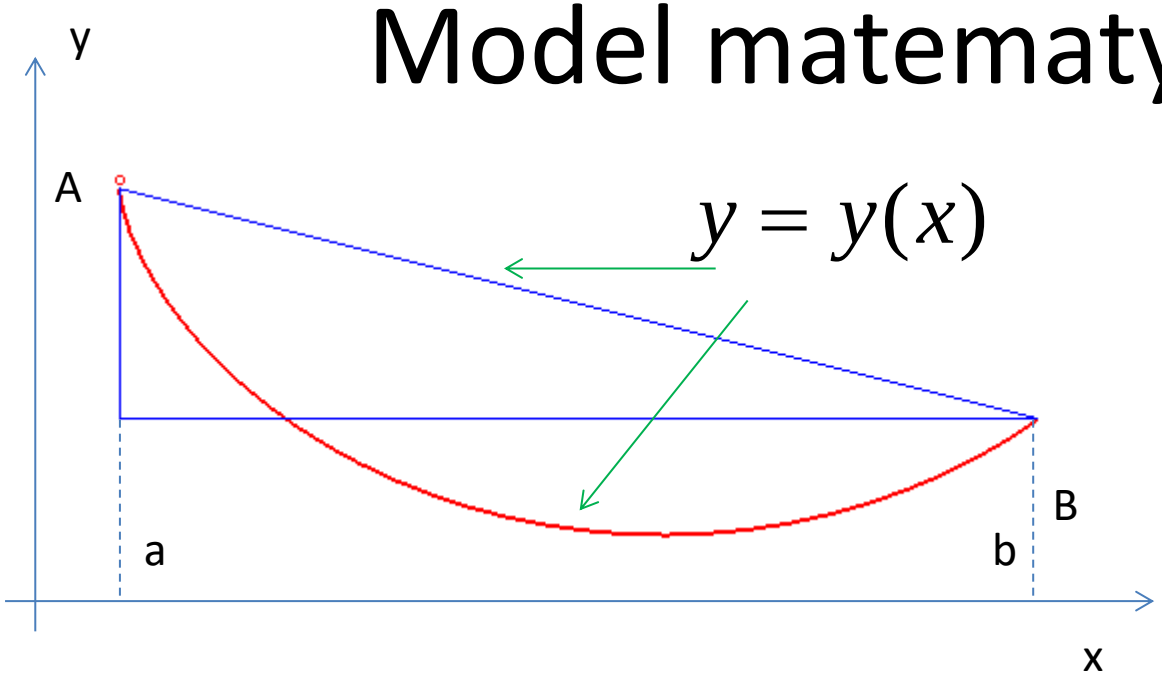
Brachistochrona

Zagadnienie brachistochrony było jednym z pierwszych, do rozwiązania którego wykorzystano rachunek wariacyjny. Postawiony w 1696 przez Jakuba Bernoulliego problem znalezienia krzywej najszybszego spadku został rozwiązany niezależnie przez Leibniza, Newtona, Jana Bernoulliego oraz de L'Hospitala.

Eksperyment



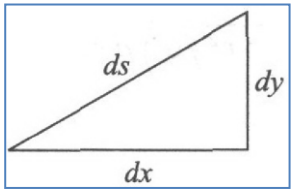
Model matematyczny



Trzeba znaleźć

$$y = y(x)$$

długość krzywej



$$ds^2 = dx^2 + dy^2$$

$$dy = y'(x) \cdot dx$$

Kryterium optymalności

Problem sterowania ujmuje funkcjonał kosztów

W każdym punkcie prędkość ma być maksymalną

$$ds = \sqrt{1 + y'(x)^2}\, dx$$

$$J = \int_a^b \frac{ds}{v} \rightarrow \min$$

$$v = y'(x)$$

Brachistochrona

$$\frac{dy}{dx}=\sqrt{\frac{C-y}{y}},\ C>0$$

parametr

$$y(t)=\frac{C}{2}-\frac{C}{2}\cos t=C\sin^2\frac{t}{2}$$

$$\frac{y'(t)}{x'(t)}=\sqrt{\frac{C\cos^2\frac{t}{2}}{C\sin^2\frac{t}{2}}}\Rightarrow\frac{C\cos\frac{t}{2}\sin\frac{t}{2}}{x'(t)}=\frac{\cos\frac{t}{2}}{\sin\frac{t}{2}}\Rightarrow x'(t)=C\sin^2\frac{t}{2}$$

$$x'(t)=\frac{C}{2}(1-\cos t)\Leftrightarrow x(t)=\frac{C}{2}(t-\sin t)+C_1,$$

$$C_1=0,\ x(0)=0$$

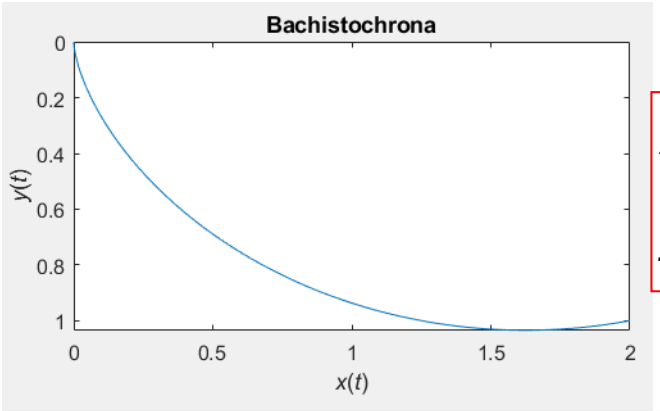
$$y(a)=A,\quad y(b)=B$$

$$x(t)=\frac{C}{2}(t-\sin t)$$

$$y(t)=\frac{C}{2}(1-\cos t)$$

Bach2.m

```
clear all
disp('Brachistochrona')
x2=2;
y2=1;
fprintf('Punkt prawy: y(%d)=%d\n',x2,y2)
eq1=['C1*(t2-sin(t2))=' num2str(x2)];
eq2=['C1*(1-cos(t2))=' num2str(y2)];
fprintf('System rownian:\n%s\n%s\n',eq1,eq2)
syms C1 t2
eq1=C1*(t2-sin(t2))==2;
eq2=C1*(1-cos(t2))==1;

%Sol=solve(eq1,eq2,C1,t2);
Sol=vpasolve(eq1,eq2,C1,t2);

C1=eval(Sol.C1);
t2=eval(Sol.t2);
disp('Rozwiazanie:')
fprintf('Stala C1=%10.5f\n',C1)
fprintf('Parametr t2=%10.5f\n',t2)
disp('Rownianie brachistochrony:')
fprintf('x(t)=%10.5f(t-sin(t))\n',C1)
fprintf('y(t)=%10.5f(1-cos(t))\n',C1)
```



$$x(t)=\frac{C}{2}(t-\sin t)$$

$$y(t)=\frac{C}{2}(1-\cos t)$$

```
t=linspace(0,t2);
x=C1*(t-sin(t));
y=C1*(1-cos(t));
figure %
plot(x,y) %
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',12)
axis ij %
xlim([0 x2])
da=daspect;
da(1:2)=min(da(1:2));
daspect(da);
title ('\bfBrachistochrona')
xlabel ('\itx\rm(\itt\rm)')
ylabel ('\ity\rm(\itt\rm)')
```

Problem miękkiego lądowania

$$F = P - G; \quad m\ddot{h} = -\mu\dot{m} - mg$$

Warunki brzegowe:

$$t_0 = 0: \quad h(0) = h_0; \quad v(0) = dh(t)/dt\big|_{t=t_0} = v_0.$$

$$t = t_1: \quad h(t_1) = 0, \quad v(t_1) = dh(t)/dt\big|_{t=t_1} = 0.$$

Kryterium optymalności:

$$J = \int_{t_0}^{t_1} \dot{m}\,dt = \min.$$

zużycie paliwa

Równania stanu:

$$\begin{cases} \dot{x}_0 = -u, & f_0 = -u, \\ \dot{x}_1 = x_2, & f_1 = x_2, \\ \dot{x}_2 = \mu\dfrac{u}{x_0} - g, & f_2 = \mu\dfrac{u}{x_0} - g \end{cases}$$

$$x_1 = h; \quad x_2 = v; \quad x_0 = J; \quad u = -\dot{m}.$$

Wyszukiwanie wyczerpujące?

Optymalne sterowanie

Pasywna sekcja zrzutu:

$$0 \le t < \xi_0,\ u = 0$$

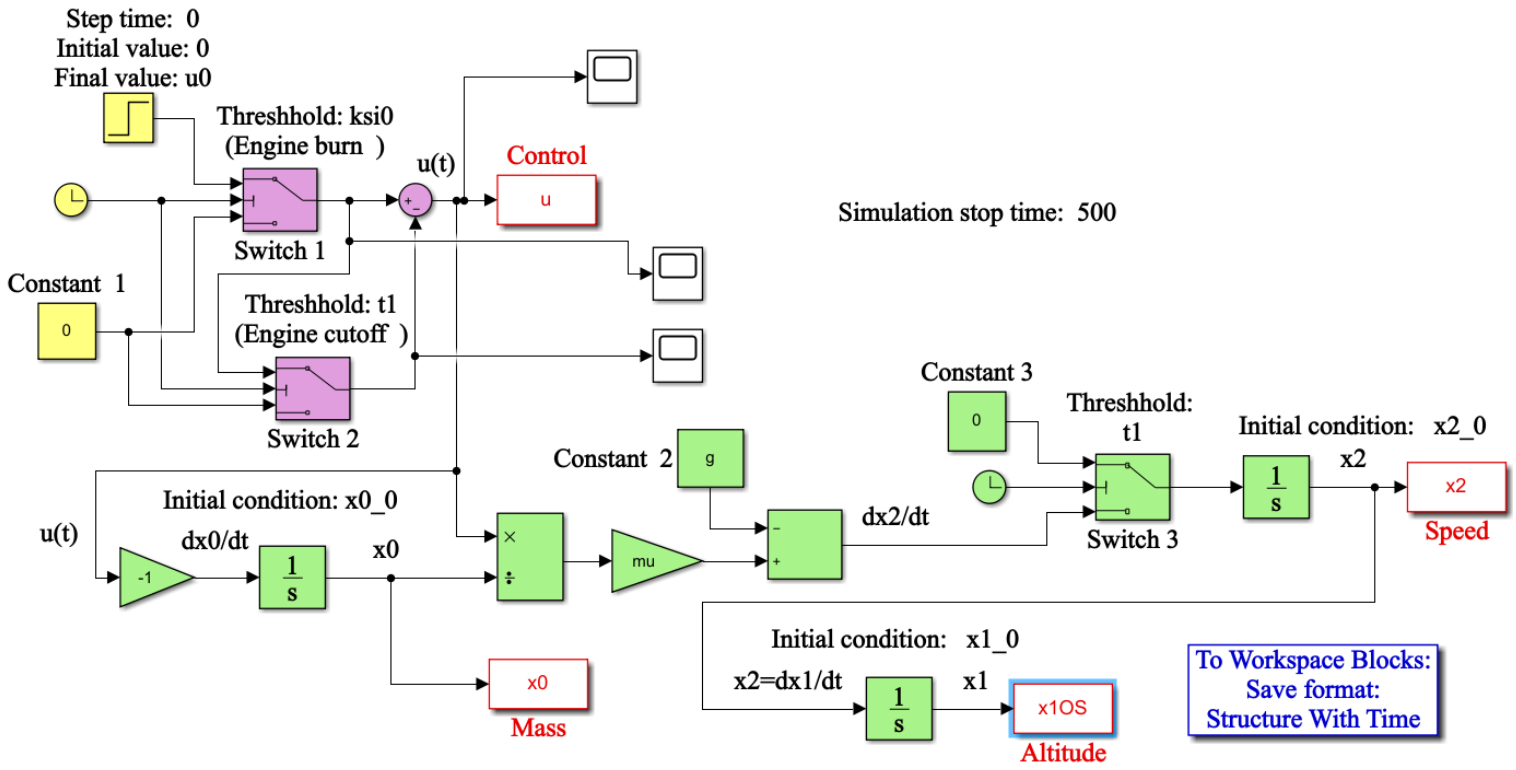
$$\begin{cases} x_0(t) = m_0, \\ x_2(t) = v_0 - gt, \\ x_1(t) = h_0 + v_0 t - gt^2 / 2. \end{cases}$$

Aktywna sekcja zrzutu:

$$\xi_0 \le t < t_1,\ u = u_0$$

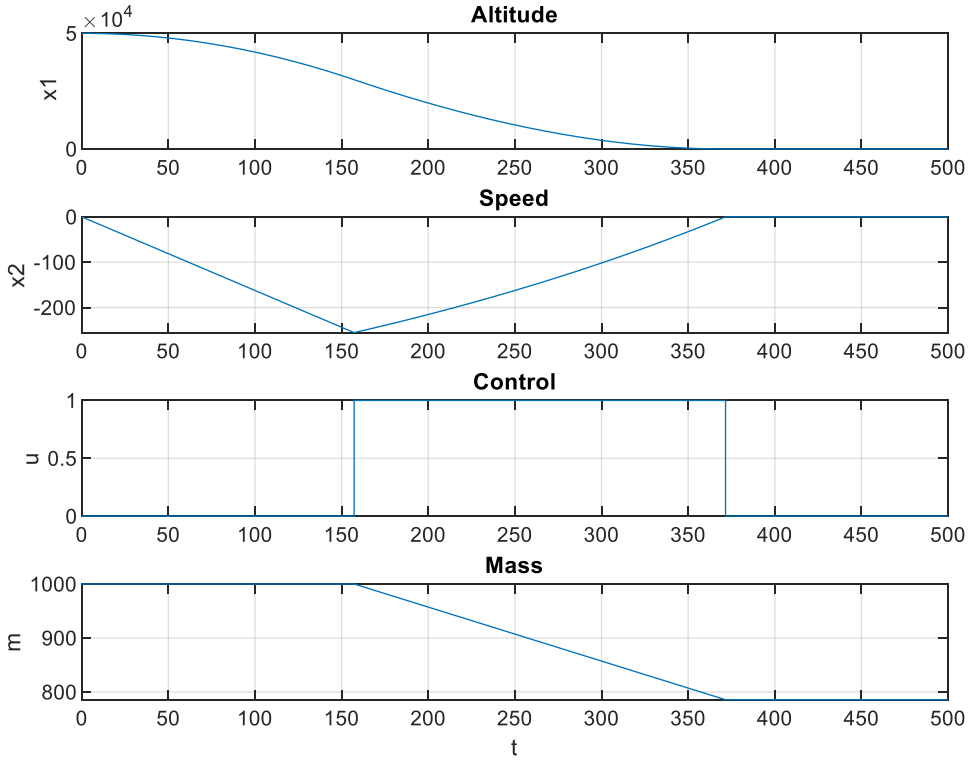
$$\begin{cases} x_0(t) = -u_0(t - \xi_0) + m_0, \\ x_1(t) = x_1(\xi_0) + x_2(\xi_0)(t - \xi_0) - \dfrac{g}{2}(t - \xi_0)^2 + \\ +\mu \dfrac{m_0}{u_0}\left[\left(1 - \dfrac{u_0(t - \xi_0)}{m_0}\right)\ln\left(1 - \dfrac{u_0(t - \xi_0)}{m_0}\right) + \dfrac{u_0}{m_0}(t - \xi_0)\right], \\ x_2(t) = x_2(\xi_0) - g(t - \xi_0) - \mu \ln\left(1 - \dfrac{u_0(t - \xi_0)}{m_0}\right). \end{cases}$$

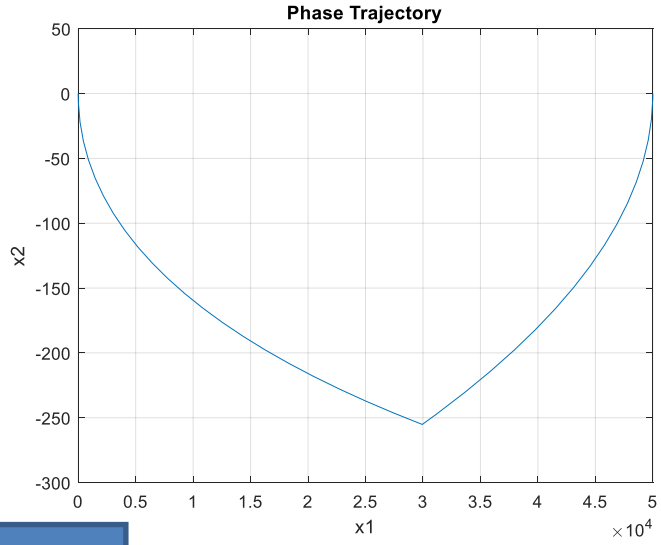
Model w Simulinku



moonshipOS

Wyniki symulacji

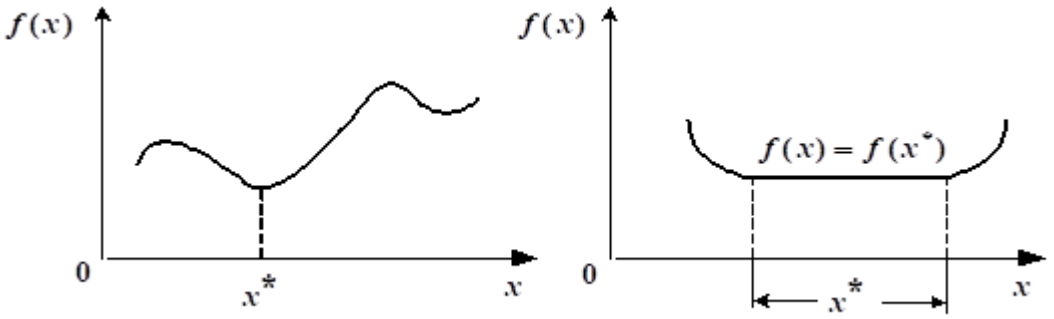




Optymalizacja bezwarunkowa

$$\min_{x} f(x)$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \mathrm{L} \\ x_n \end{bmatrix} \in R^n$$



$$f(x^*) < f(x)$$

$$x^* \in X$$

$$f(x^*) \le f(x)$$

$$x^* = [x^* \in X : f(x) = f(x^*)]$$

właściwe minimum globalne

Optymalizacja warunkowa

$$\min_{x} f(x)$$

$$g_i(x) = 0, \quad i = 1, 2, \text{K}, n$$
$$h_j(x) \le 0, \quad j = 1, 2, \text{K}, m$$
$$a \le x \le b,$$

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \text{L} \\ a_n \end{bmatrix}; \; b = \begin{bmatrix} b_1 \\ b_2 \\ \text{L} \\ b_n \end{bmatrix}$$

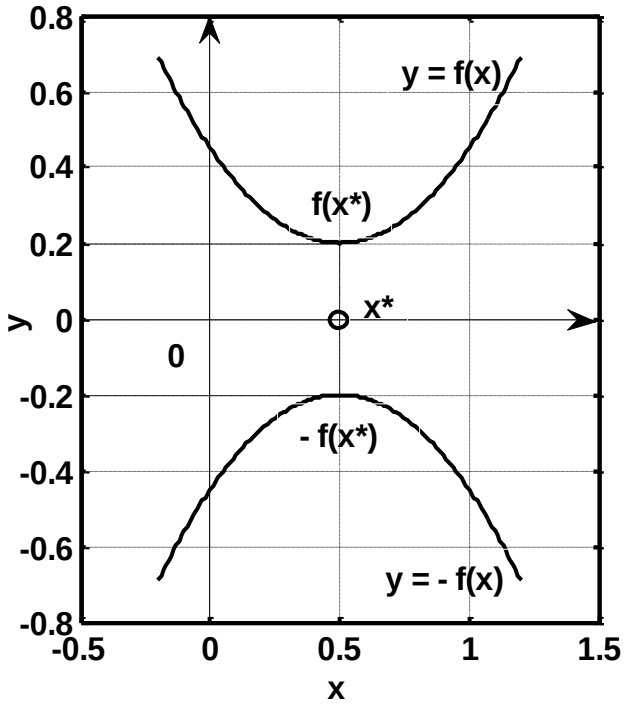
Maksymalizacja funkcji vs minimalizacja funkcji

$$\min_x f(x) = -\max_x f(x)$$

$$\min_x f(x_1, x_2, \ldots, x_n) = -\max_x f(x_1, x_2, \ldots, x_n)$$

$$y(x) = f(x) = (x - 0{,}5)^2 + 0{,}2$$

```
>> x = -0.2 : 0.01 : 1.2;
y = (x-0.5).^2+0.2;
plot (x,y)
hold on
y = -((x-0.5).^2+0.2);
plot (x,y)
grid on
xlabel('x'), ylabel('y')
```



Metody bezpośrednich poszukiwań

$$\min_{x} f(x)$$

Minimum funkcji znajduje się w dwóch fazach:

- Metody **ustalania przedziału**, w którym znajduje się minimum (ang. bracketing methods)
- Metody znajdowania minimum **z zadana dokładnością**:
  - Metody eliminowania obszarów (ang. region elimination methods)
  - Metoda estymacji punktowej (ang. *point estimation method*)
  - Metody oparte na gradientach (ang. *gradient based methods*)

Źródło: https://web.sgh.waw.pl/~mlewan1/Site/MO_files/mo_skrypt_21_12.pdf

Metoda **wyczerpującego** poszukiwania (ang. exhaustive search method)

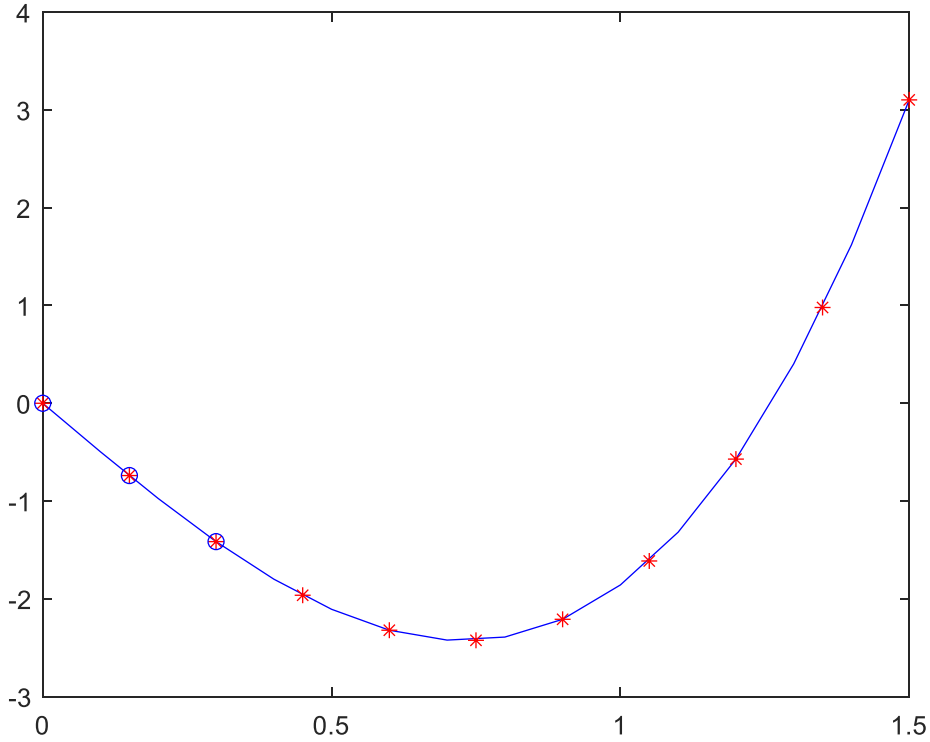
Metoda ta polega na porównywaniu wartości funkcji celu **dla punktów jednakowo od siebie odległych**. Zazwyczaj poszukiwania zaczyna się od dolnego ograniczenia zmiennej i w pojedynczej iteracji porównuje się wartości trzech kolejnych punktów wykorzystując założenie **unimodalności**.

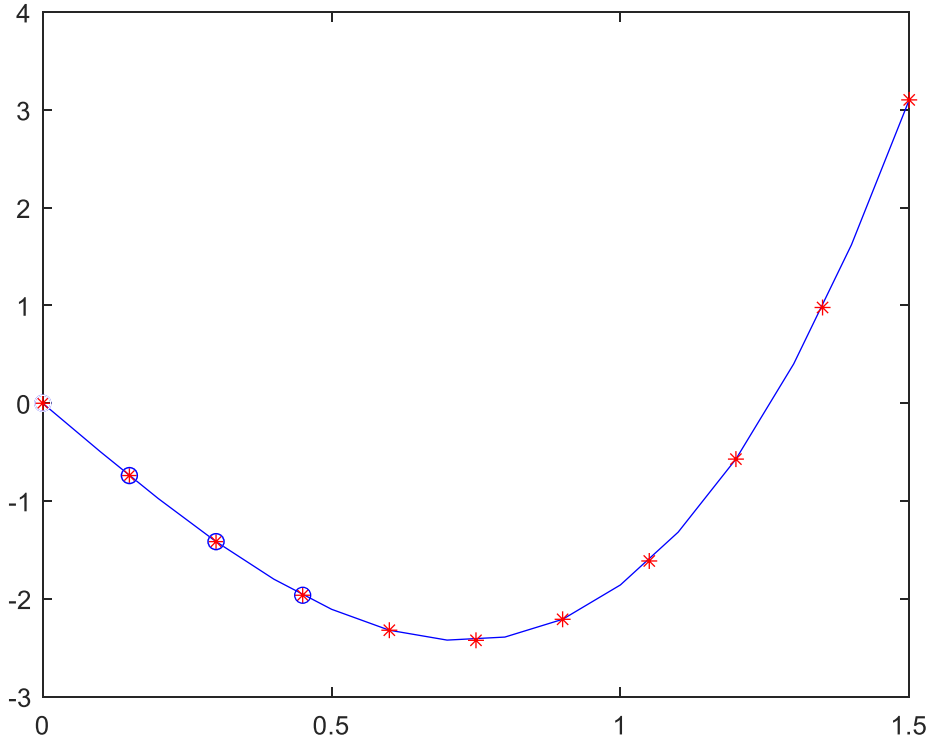
1) Ustal $x_1 = a$, $\Delta x = (b - a)/n$ (n jest liczbą punktów pośrednich), $x_2 = x_1 + \Delta x$, i $x_3 = x_2 + \Delta x$.

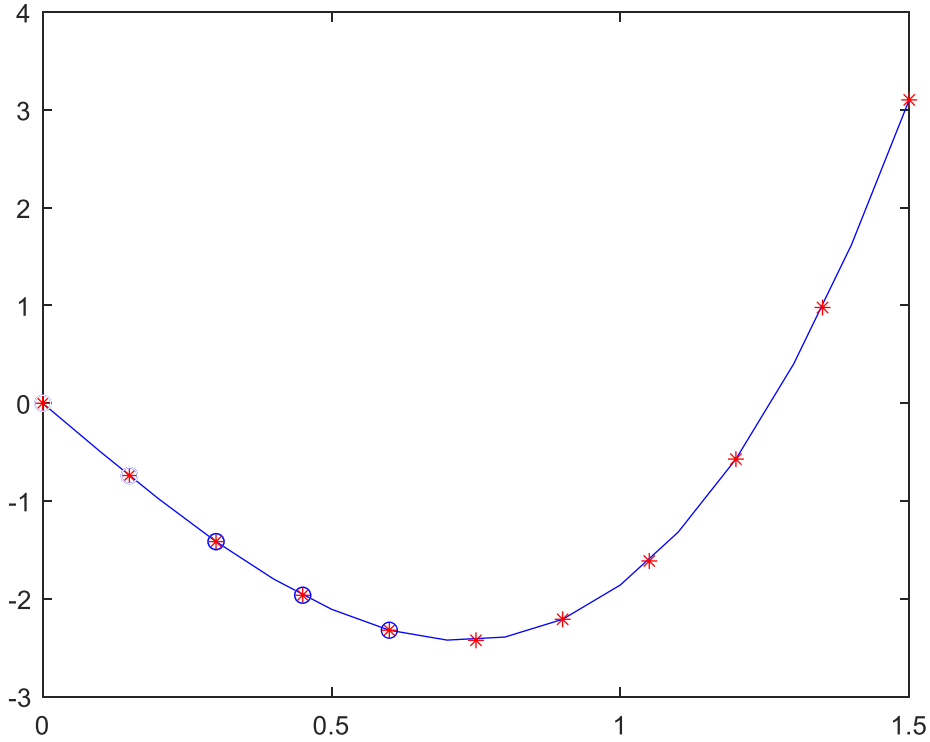
2) Jeśli $f(x_1) \geqslant f(x_2) \leqslant f(x_3)$, minimum znajduje się w (x_1, x_3), **Zakończ**; W przeciwnym przypadku $x_1 = x_2$, $x_2 = x_3$, $x_3 = x_2 + \Delta x$ i przejdź do kroku 3).

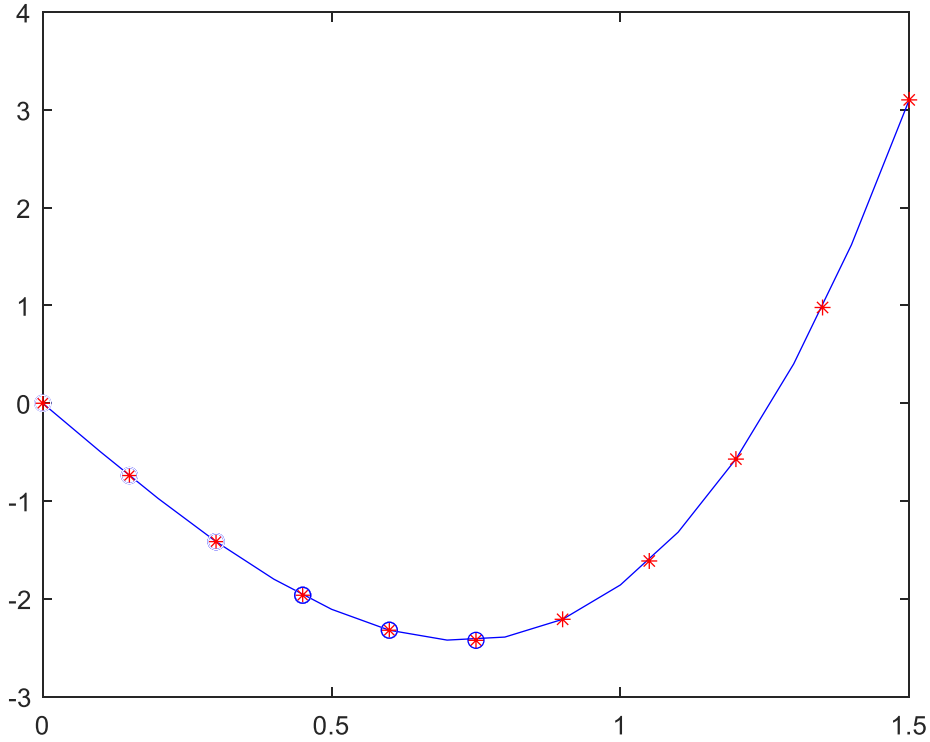
3) Czy $x_3 \leqslant b$? Jeśli tak, to idź do kroku 2);
Jeśli nie, to nie istnieje minimum w przedziale (a, b) lub punkt brzegowy (a lub b) jest puntem minimalnym.

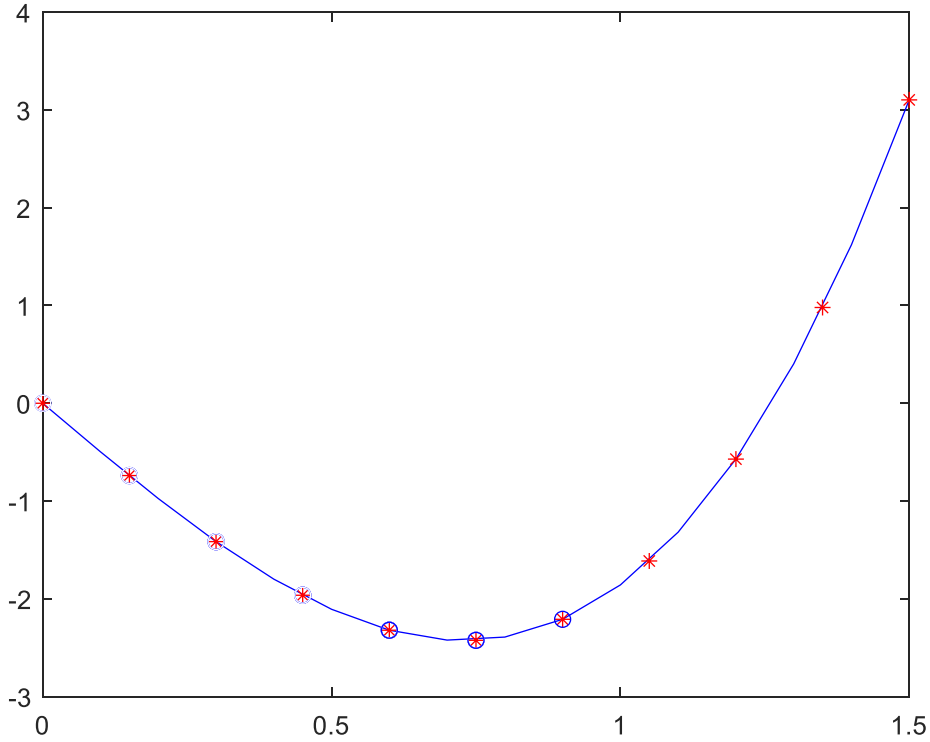
\ProgramOS\MetWyczPoszukiwaniaOS.m


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5

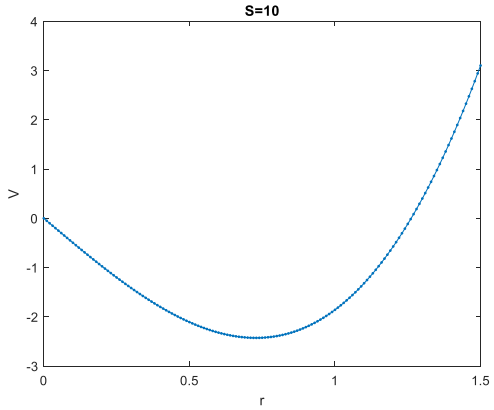

4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5

```
>> MetWyczPoszukiwaniaOS
       0    0.3000    0.6000

    0.3000    0.6000    0.9000

odcinek
    0.3000    0.6000    0.9000
```




```
> MetWyczPoszukiwaniaOS
       0    0.1500    0.3000

    0.1500    0.3000    0.4500

    0.3000    0.4500    0.6000

    0.4500    0.6000    0.7500

    0.6000    0.7500    0.9000

odcinek
    0.6000    0.7500    0.9000
```

Metoda **przyspieszonego** poszukiwania (ang. bounding phase method)

Metoda ta polega na **obraniu punktu początkowego i wybraniu kierunku poszukiwań** na podstawie porównania wartości funkcji w **punkcie początkowym oraz dwóch wartości funkcji w punktach będących w bezpośrednim sąsiedztwie punktu początkowego**. Później znajduje się drugi kraniec przedziału stosując **wykładnicza strategie poszukiwań**.
Poniżej użyty jest wykładnik równy 2, ale można używać jakakolwiek inna liczbę dodatnia. Wykładnik wyższy niż 1, powoduje ’przyspieszanie’ wykładnicze poszukiwań, co zmniejsza liczbę iteracji, ale dzieje się to kosztem uzyskanej dokładności.

metodyOptymalizacjiProcesowProdukcyjnych\ProgramOS\MetPrzyspPoszukiwaniaOS.m

Metoda **przyspieszonego** poszukiwania (ang. bounding phase method) (cd)

1) Wybierz punkt początkowy $x^{(0)}$ oraz wartość Δ. Ustal $k = 0$.

2) Jeśli $f(x^{(0)} - |\Delta|) \geqslant f(x^{(0)}) \geqslant f(x^{(0)} + |\Delta|)$, wtedy $\Delta > 0$;
Jeśli $f(x^{(0)} - |\Delta|) \leqslant f(x^{(0)}) \leqslant f(x^{(0)} + |\Delta|)$, wtedy $\Delta < 0$;
W pozostałych przypadkach wróć do kroku 1).

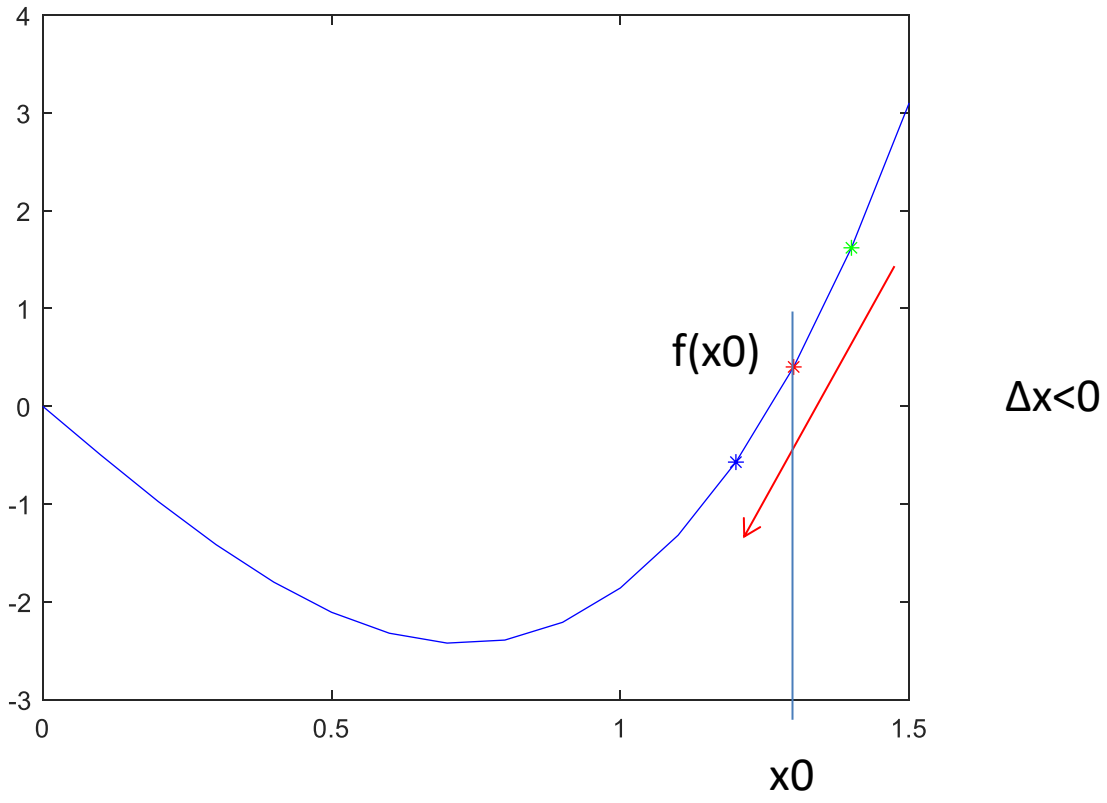
kierunek

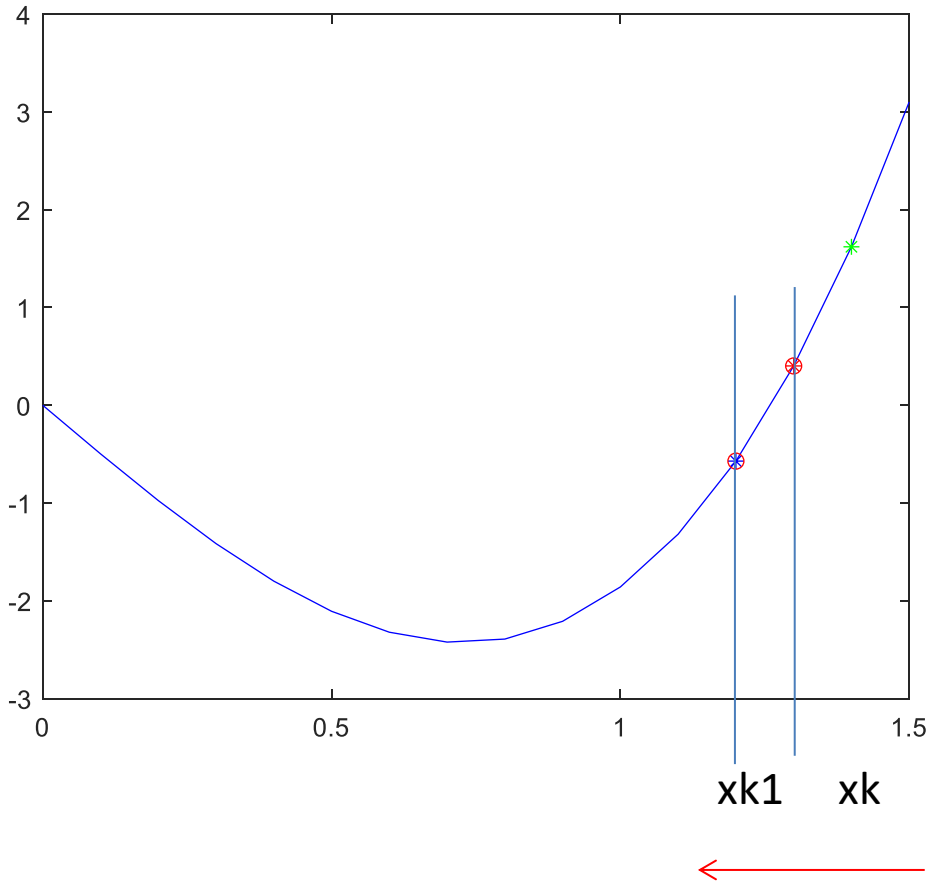
3) Ustal $x^{(k+1)} = x^{(k)} + 2^k \Delta$.

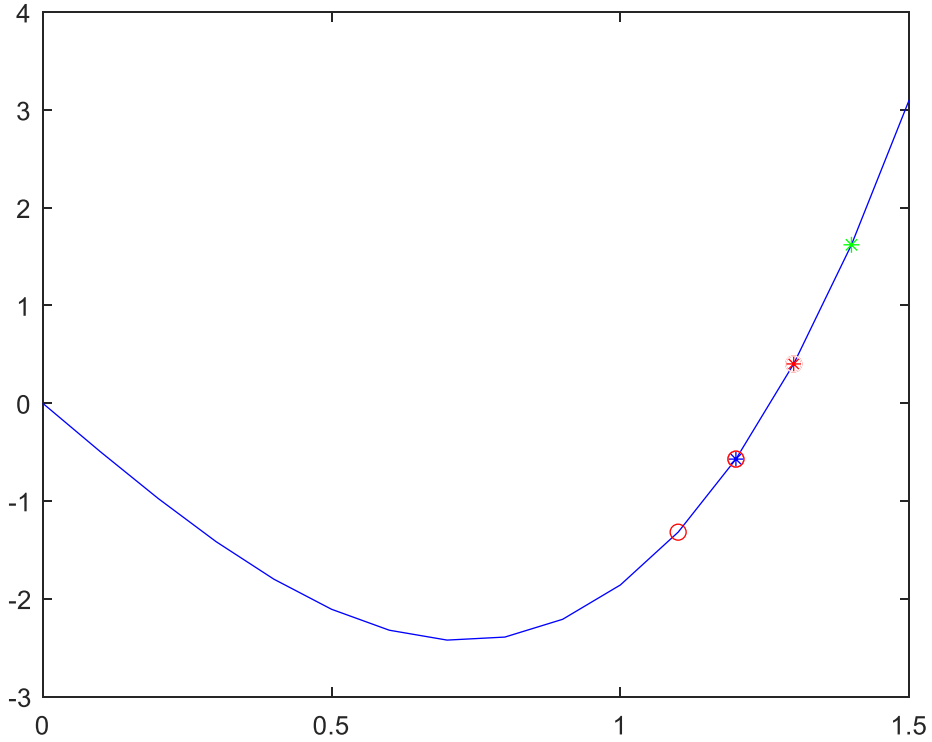
4) Jeśli $f(x^{(k+1)}) < f(x^{(k)})$, ustal $k = k + 1$ i idź do kroku 3);
W przeciwnym razie minimum znajduje się w przedziale $(x^{(k-1)}, x^{(k+1)})$,
Zakończ.

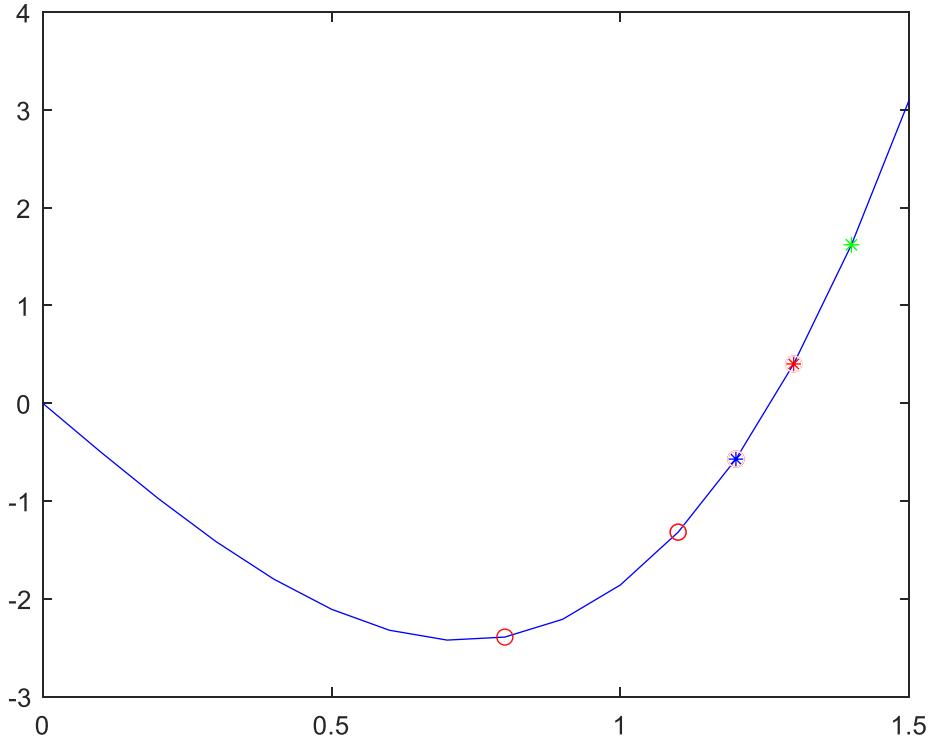
iteracje

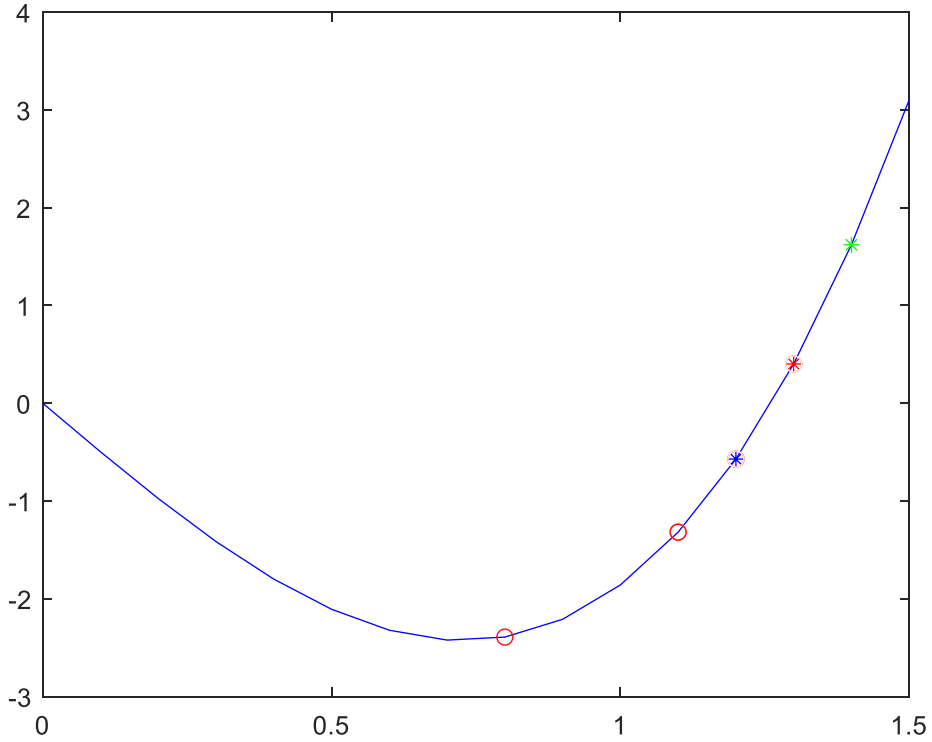
metodyOptymalizacjiProcesowProdukcyjnych\ProgramOS\MetPrzyspPoszukiwaniaOS.m

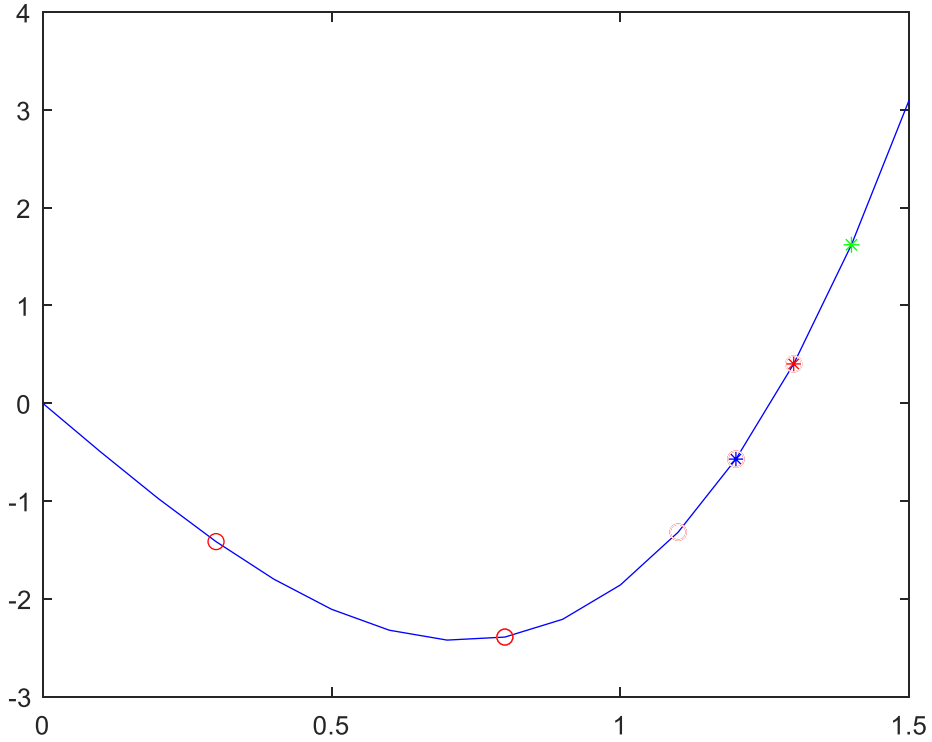

4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5
f(x0)
x0
Δx<0


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5
xk1
xk

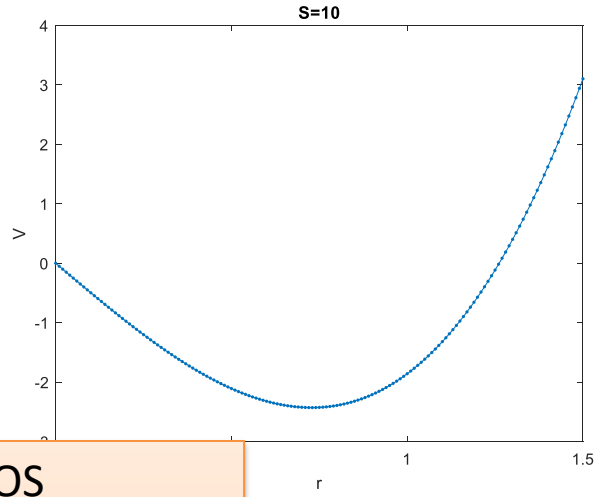

4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5

```
>> MetPrzyspPoszukiwaniaOS
   0.1000         0    0.2000    0.1000

Ptzedzial

Out =

         0    0.1000    0.2000
    1.0000    0.2000    0.3000
    2.0000    0.3000    0.6000
    3.0000    0.6000    1.1000
```

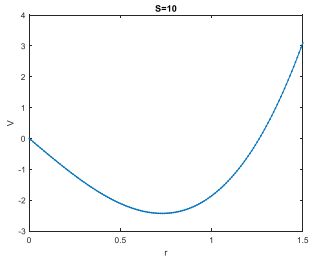



```
>> MetPrzyspPoszukiwaniaOS
   1.3000    1.2000    1.4000   -0.1000

Ptzedzial

Out =

         0    1.3000    1.2000
    1.0000    1.2000    1.1000
    2.0000    1.1000    0.8000
    3.0000    0.8000    0.3000
```

Metody znajdowania minimum z zadana dokładnością: Metody **eliminowania obszarów**

Ogólna zasada metod eliminowania obszarów jest następująca:

Rozważmy dwa punkty x_1 i x_2, które leżą w przedziale (a, b) oraz $x_1 < x_2$. Dla problemu minimalizacji funkcji unimodalnej, można wyciągnąć następujące wnioski:

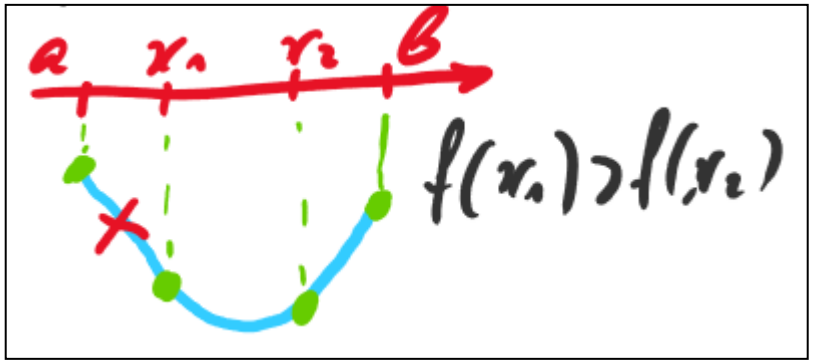


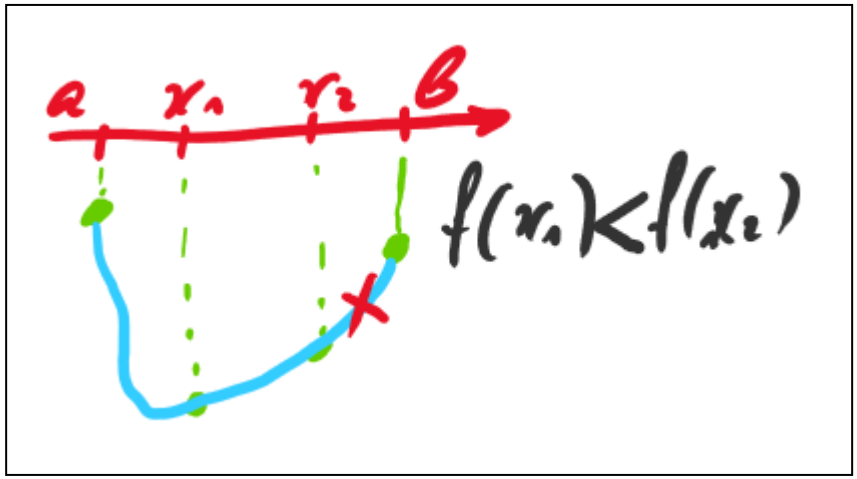
- Jeśli $f(x_1) > f(x_2)$, to minimum nie leży w (a, x_1)
- Jeśli $f(x_1) < f(x_2)$, to minimum nie leży w (x_2, b)
- Jeśli $f(x_1) = f(x_2)$, to minimum nie leży ani w (a, x_1) ani w (x_2, b)

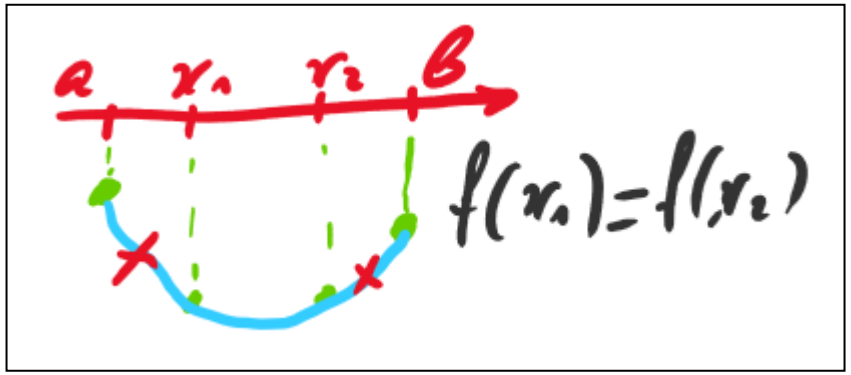
- Metoda dzielenia przedziału na połowę (ang. *interval halving method*)
- Metoda złotego podziału (ang. *golden section search*)
- Metoda liczb Fibonacciego (ang. *Fibonacci search*)

Wizualizacja eliminowania

- Jeśli $f(x_1) > f(x_2)$, to minimum nie leży w (a, x_1)
- Jeśli $f(x_1) < f(x_2)$, to minimum nie leży w (x_2, b)
- Jeśli $f(x_1) = f(x_2)$, to minimum nie leży ani w (a, x_1) ani w (x_2, b)







Metoda **dzielenia** przedziału na połowę

Metoda ta polega na wybraniu trzech punktów jednakowo odległych od siebie oraz od krańców przedziału oraz wyliczeniu wartości funkcji w tych punktach, w wyniku czego można wyeliminować połowę przedziału.

1) Wybierz dolne i górne ograniczenie przedziału a i b oraz małą liczbę ϵ. Niech $x_m = (a+b)/2$, $L_0 = L = b - a$. Wylicz $f(x_m)$.

2) Ustal $x_1 = a + L/4$, $x_2 = b - L/4$. Wylicz $f(x_1)$ oraz $f(x_2)$.

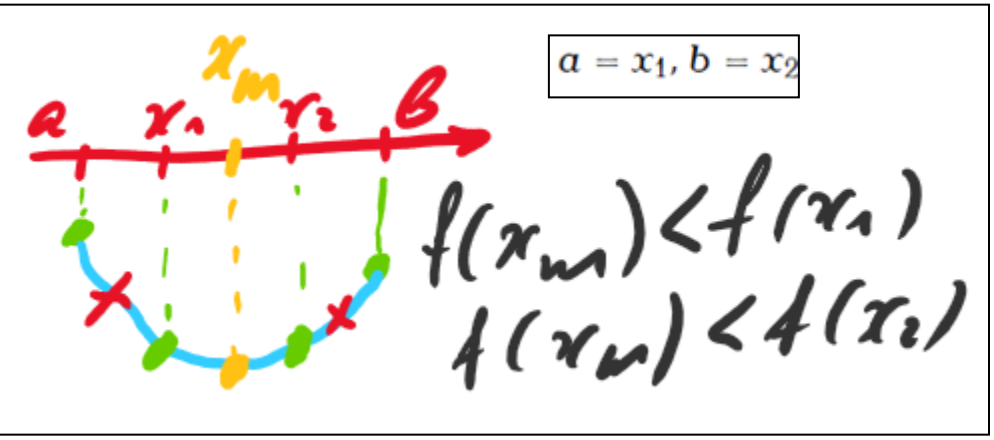
3) Jeśli $f(x_1) < f(x_m)$, ustal $b = x_m$; $x_m = x_1$; Przejdź do kroku 5);
W przeciwnym wypadku przejdź do kroku 4).

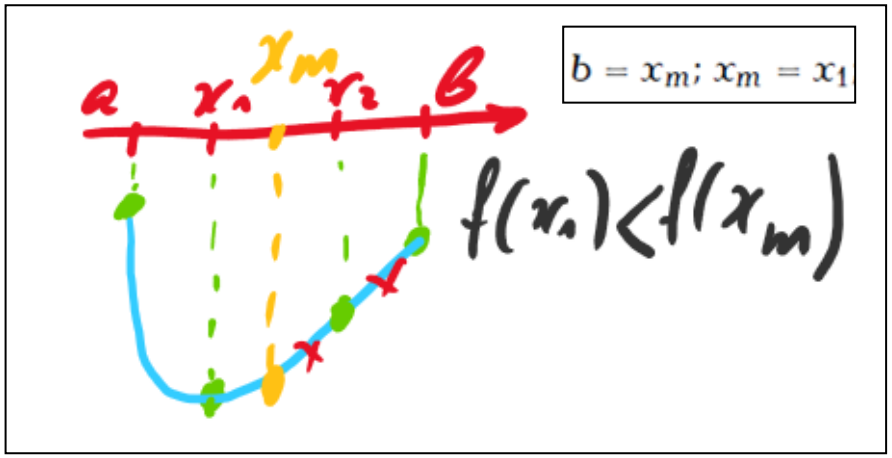
4) Jeśli $f(x_2) < f(x_m)$, ustal $a = x_m$; $x_m = x_2$; Przejdź do kroku 5);
W przeciwnym przypadku ustal $a = x_1$, $b = x_2$; przejdź do kroku 5).

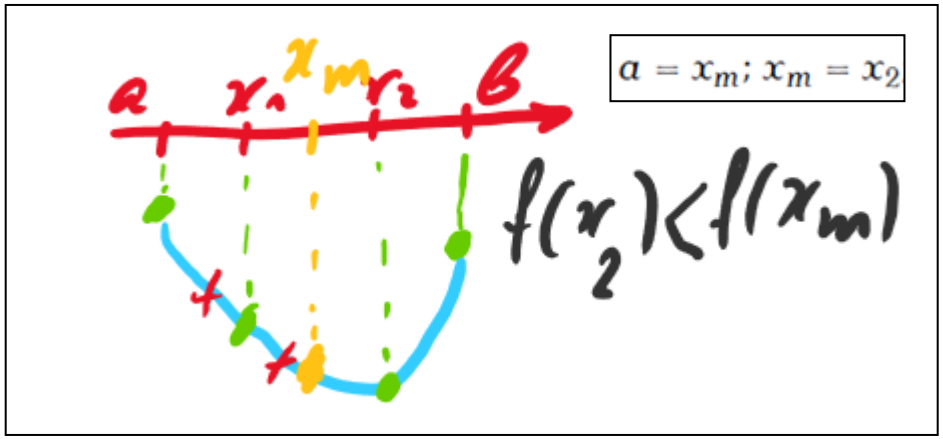
5) Wylicz $L = b - a$. Jeśli $|L| < \epsilon$, **Zakończ;**
W przeciwnym przypadku przejdź do kroku 2).

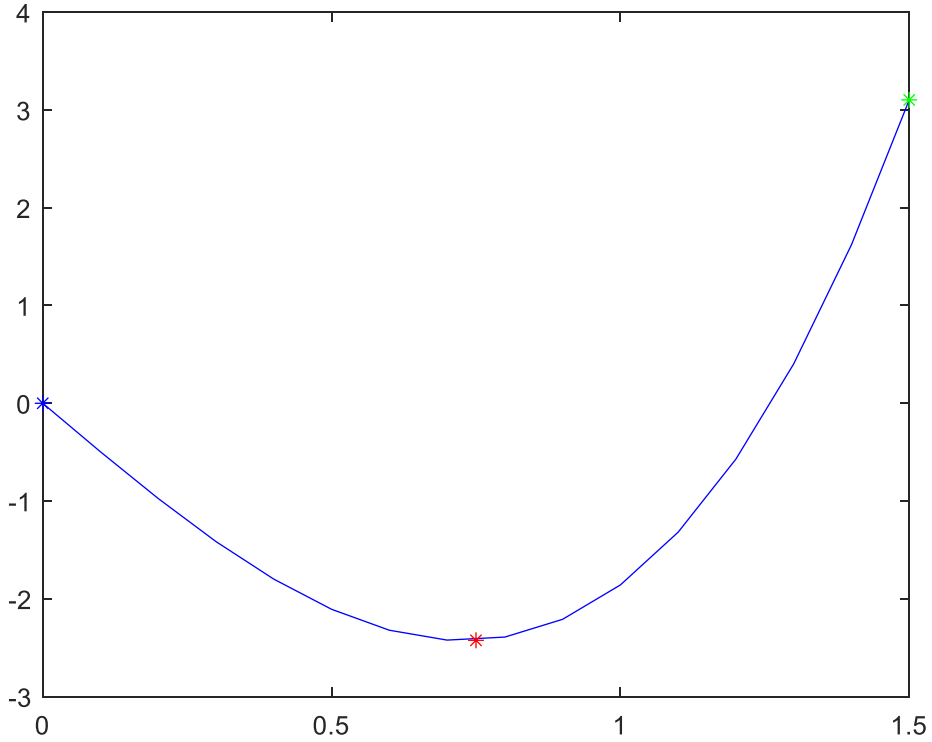
\metodyOptymalizacjiProcesowProdukcyjnych\ProgramOS\MetElimObszaruDzielDwaOS.m

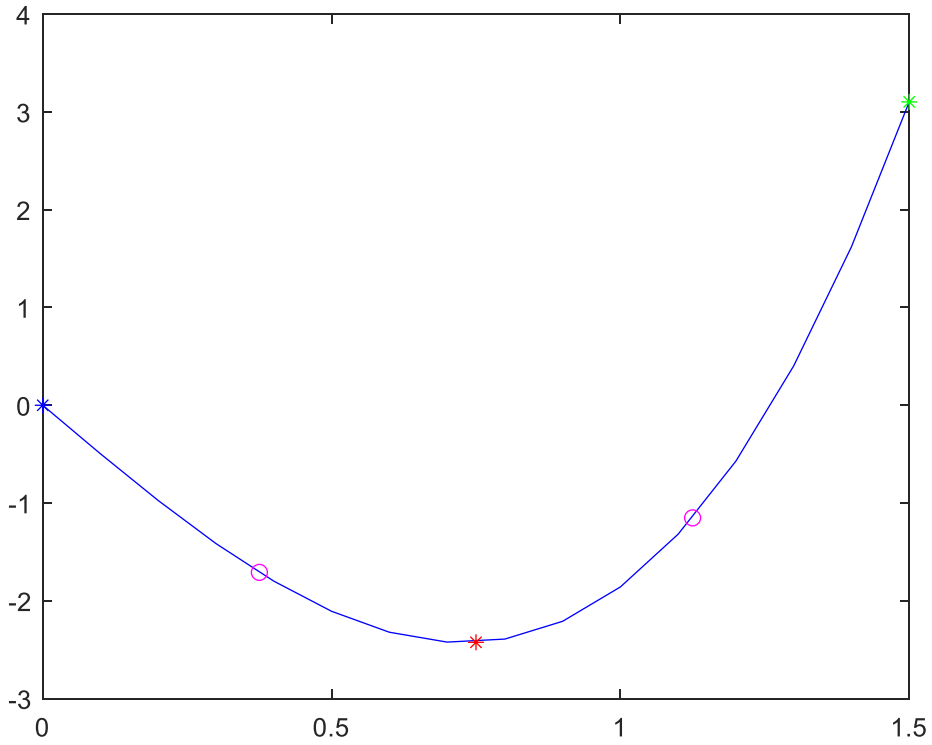
Wizualizacja metody **dzielenia** przedziału na połowę



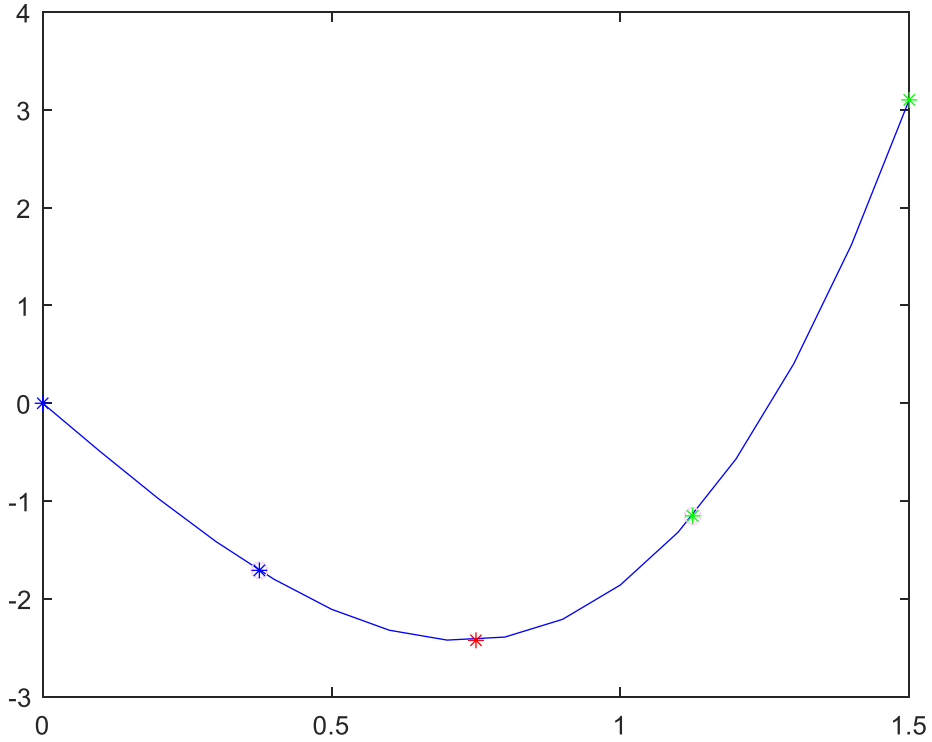


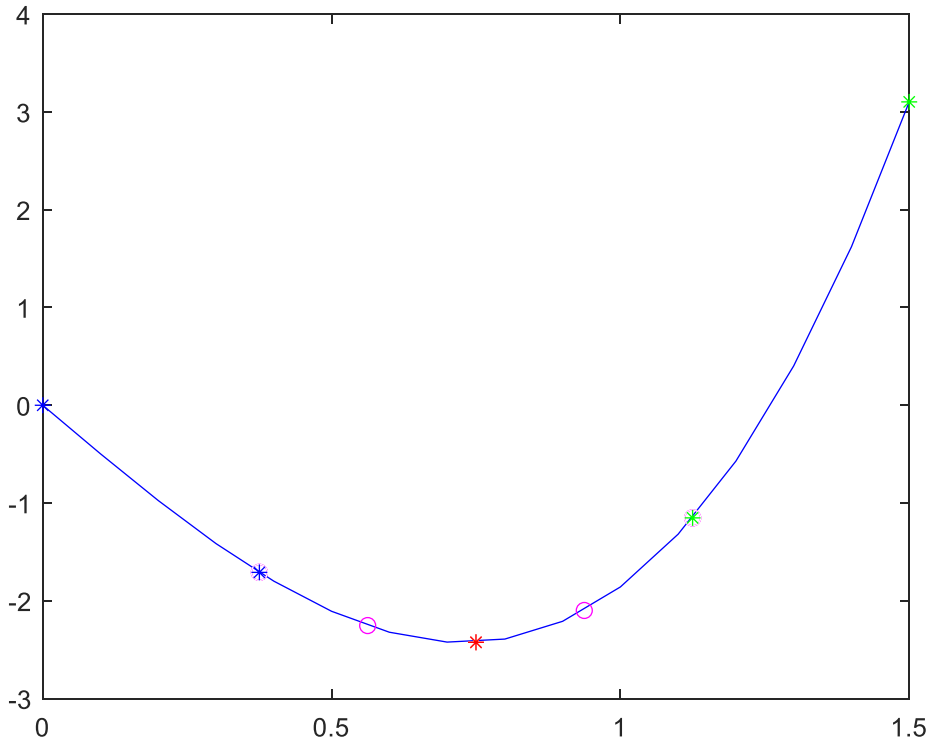



4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5



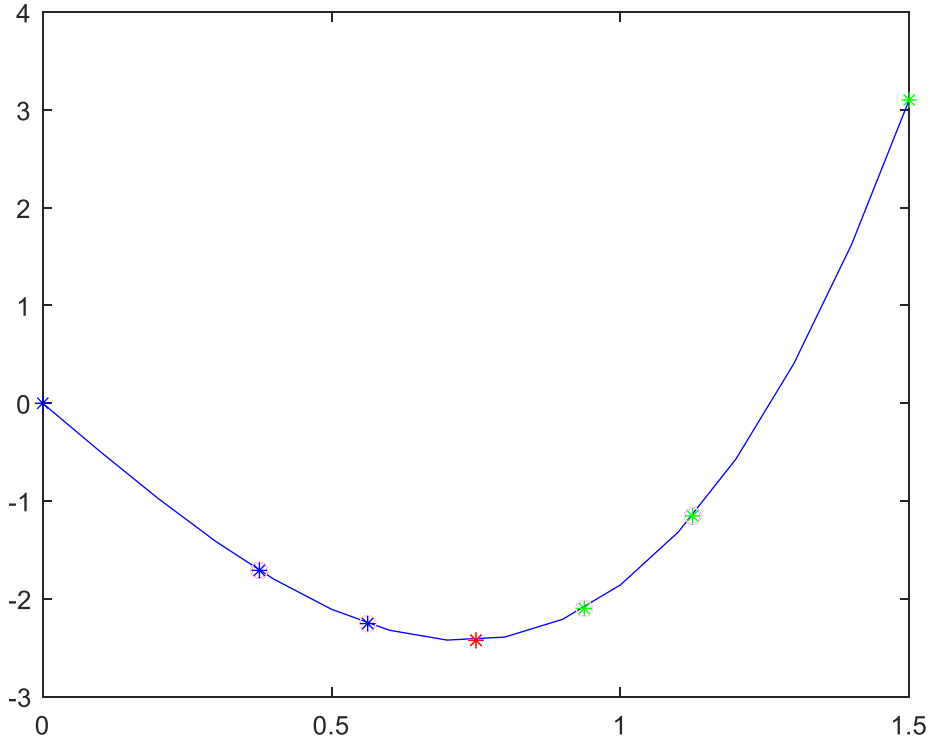
a=x1;
b=x2;

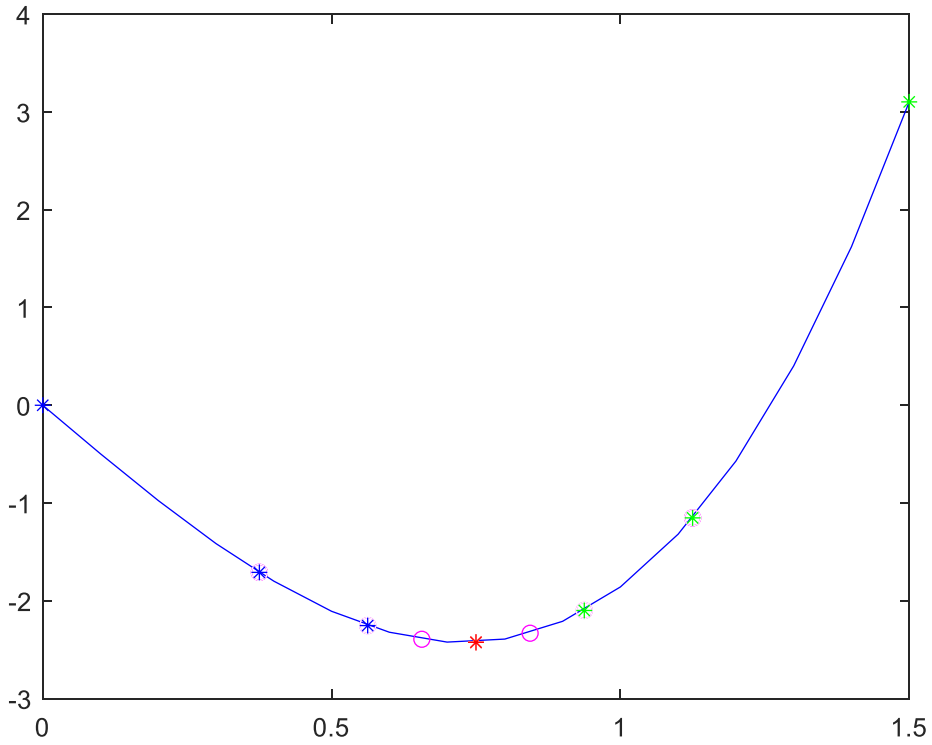

4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5



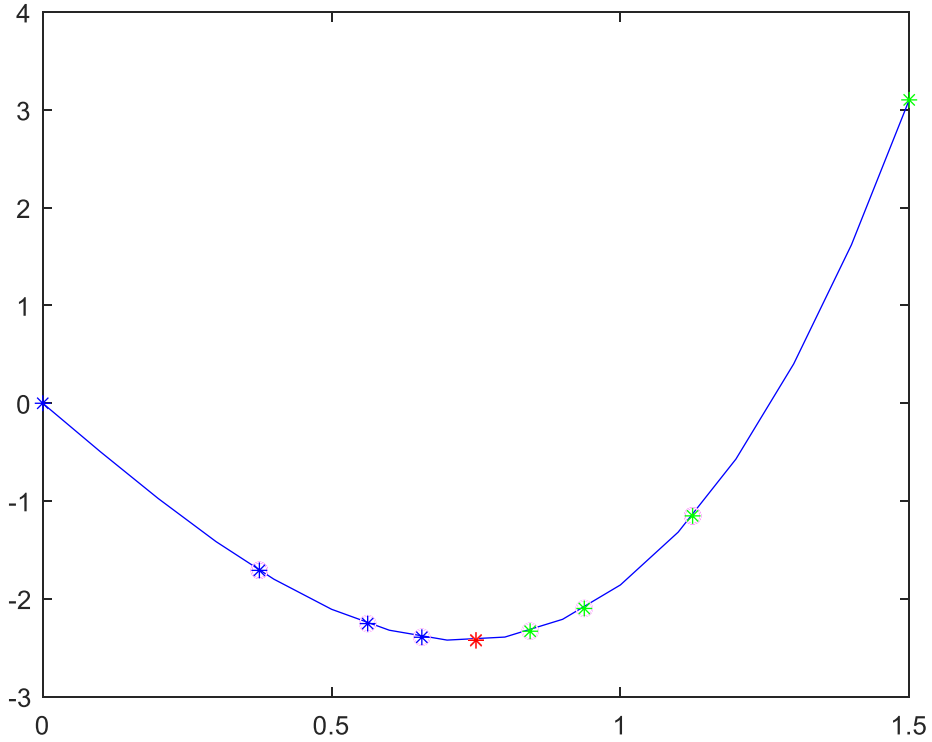
a=x1;

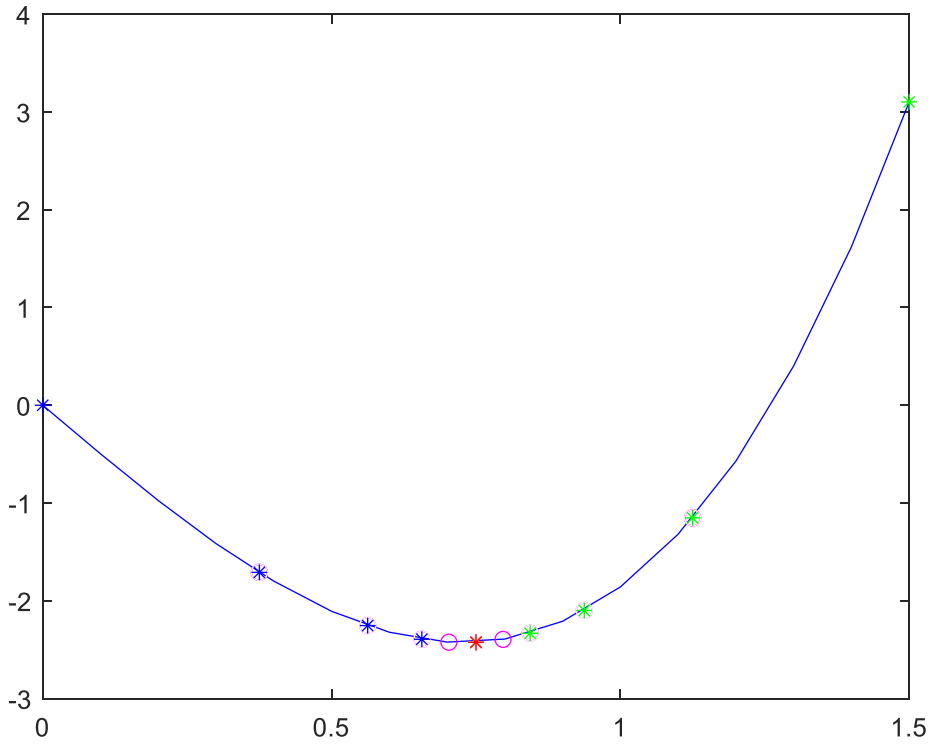
b=x2;


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5



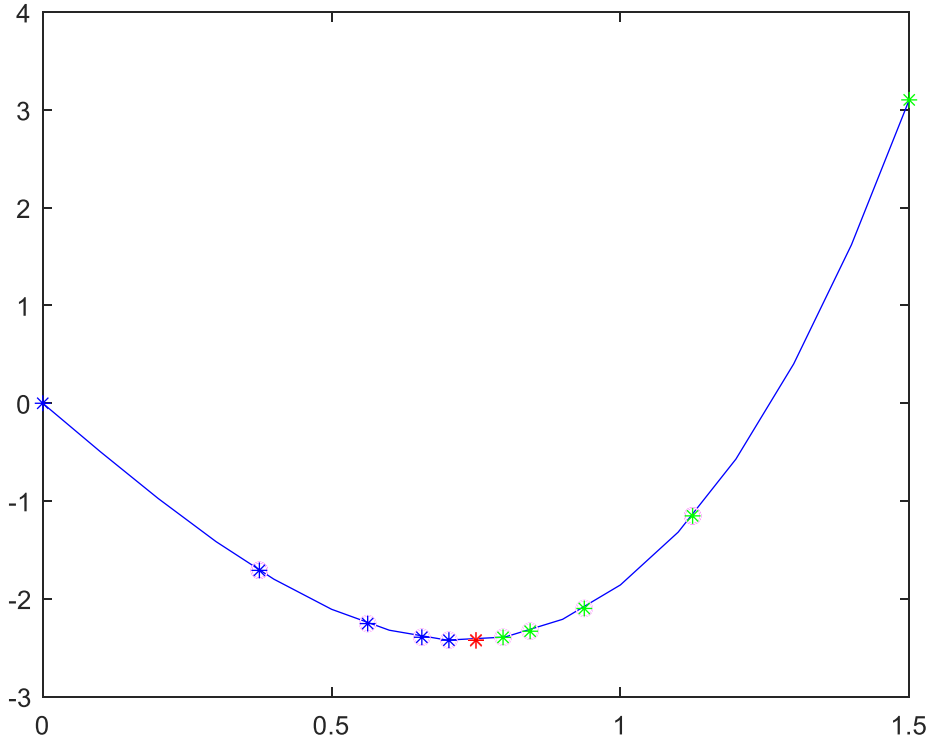
a=x1;
b=x2;

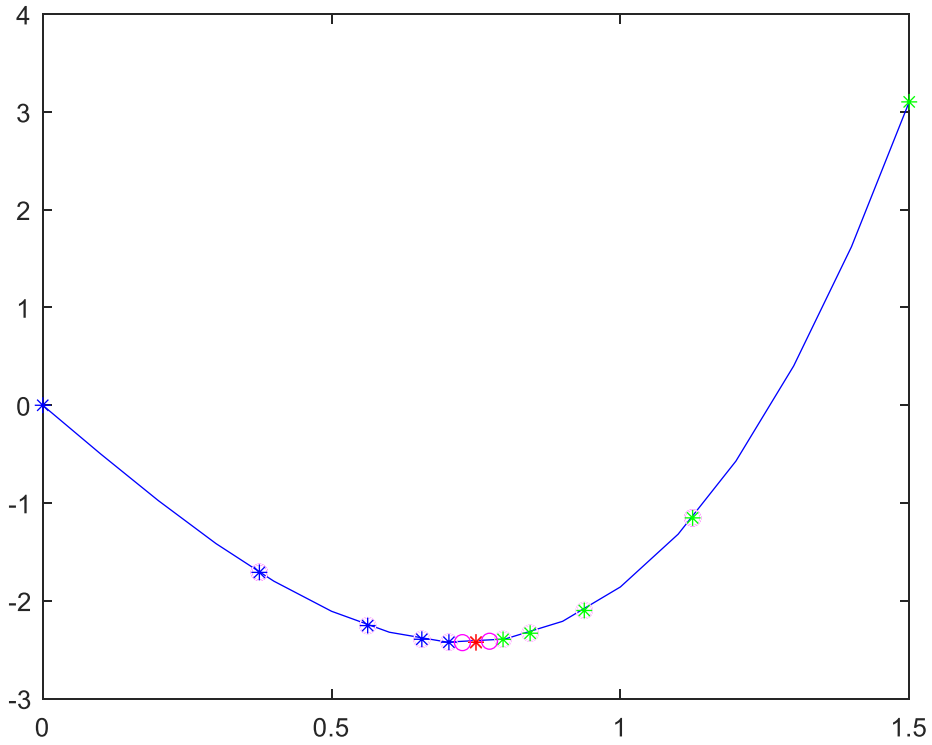

4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5



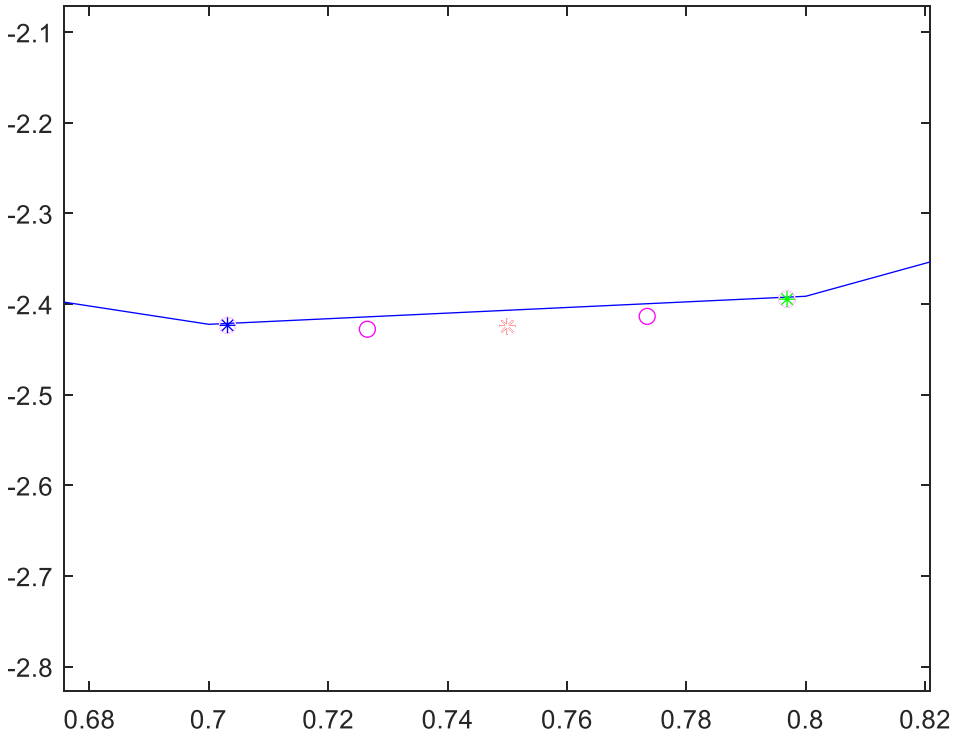
a=x1;

b=x2;

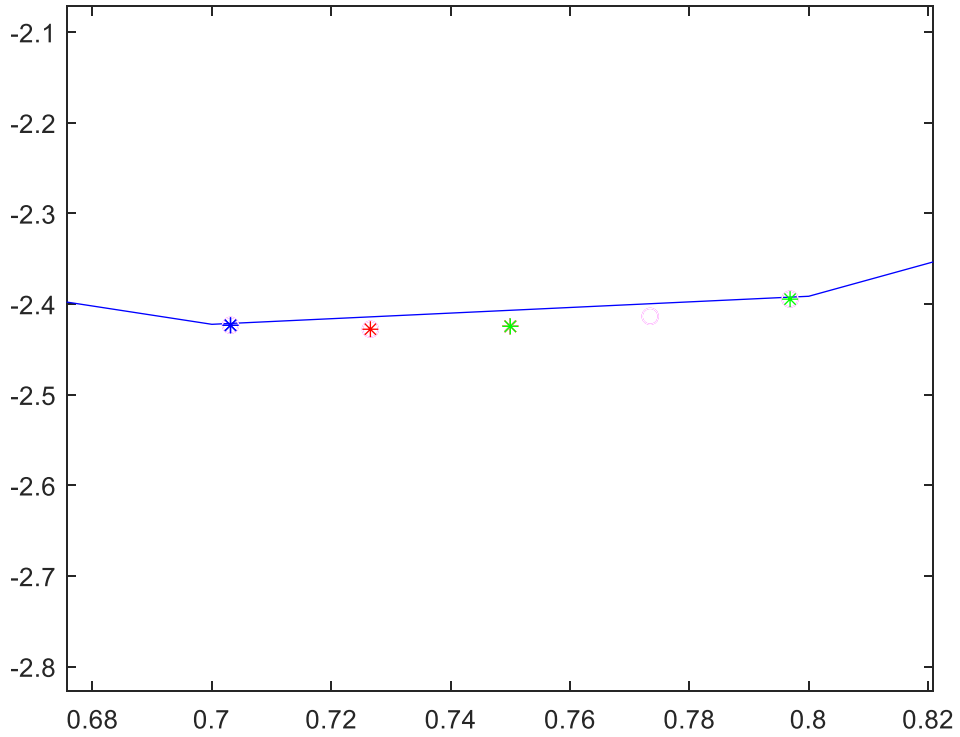

4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5

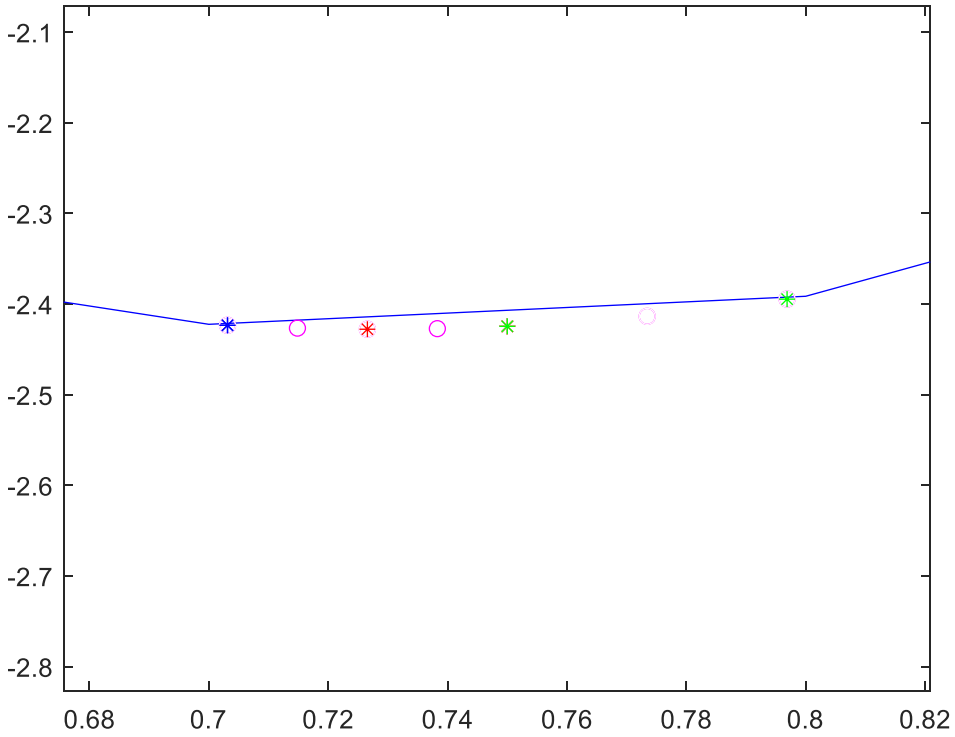



```
if puszka2(x1,S0)<puszka2(xm,S0)
    b=xm;
    xm=x1;
```



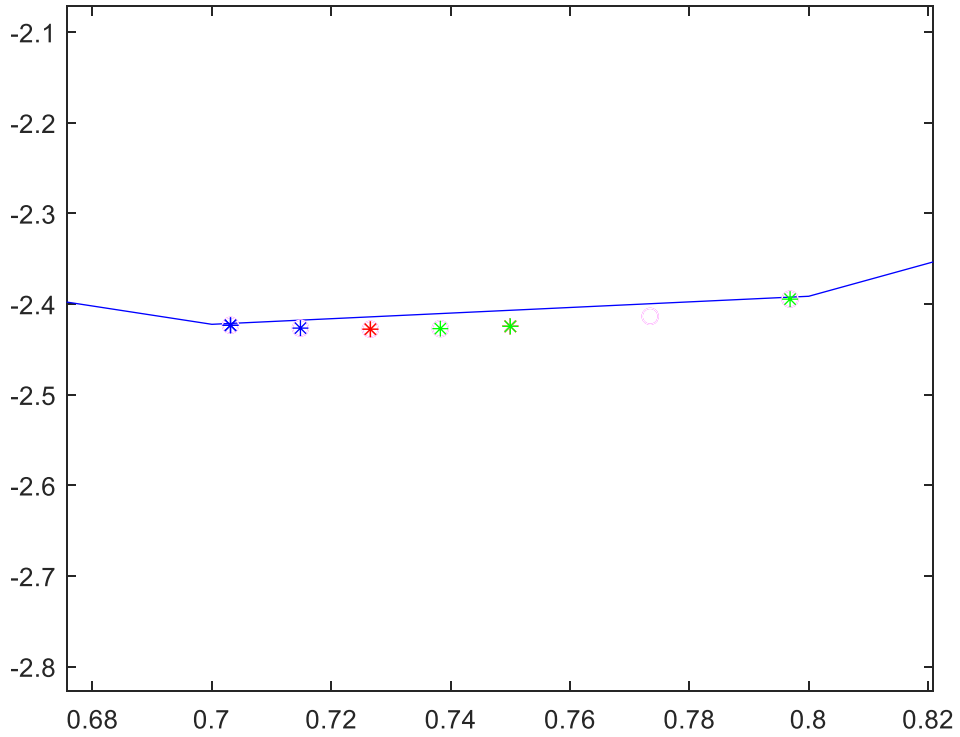

```
if puszka2(x1,S0)<puszka2(xm,S0)
    b=xm;
    xm=x1;
```

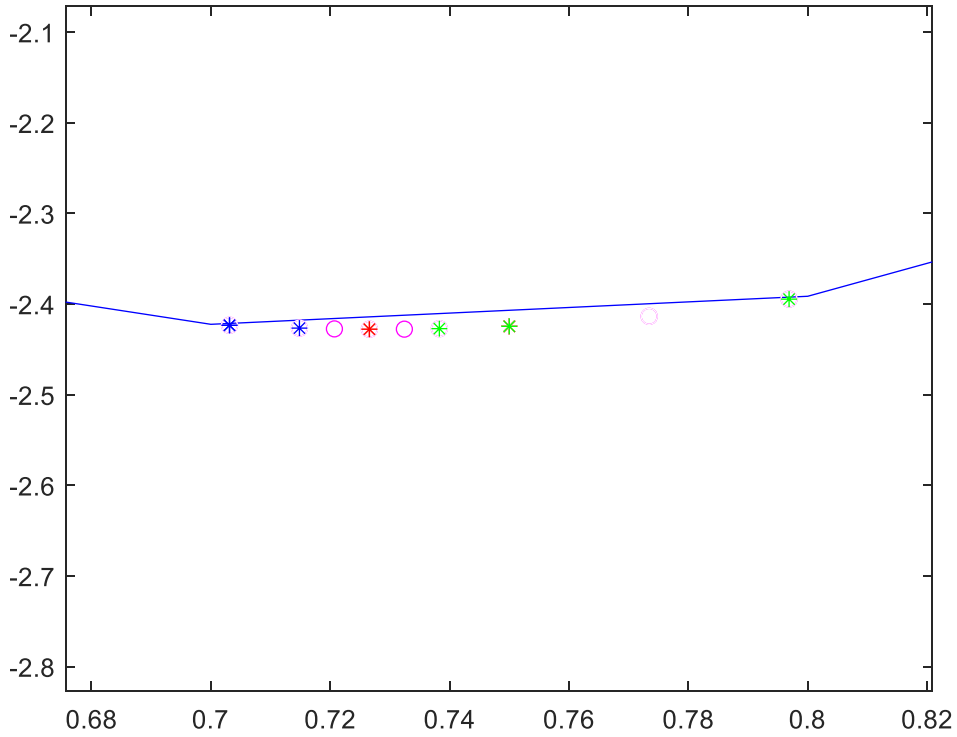

-2.1
-2.2
-2.3
-2.4
-2.5
-2.6
-2.7
-2.8
0.68
0.7
0.72
0.74
0.76
0.78
0.8
0.82



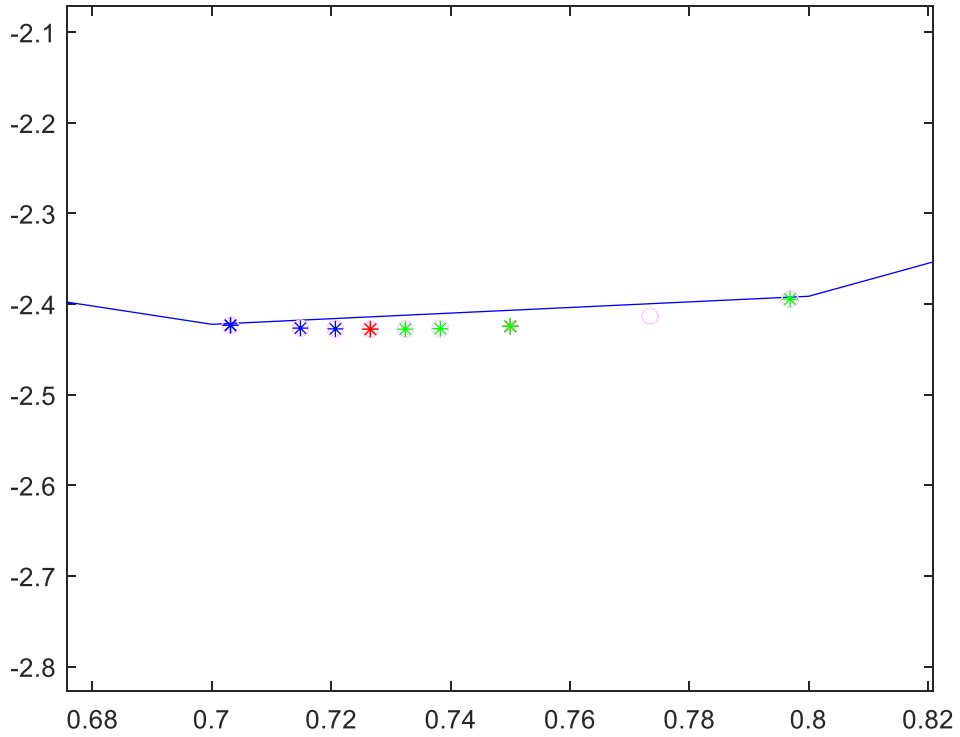
a=x1;

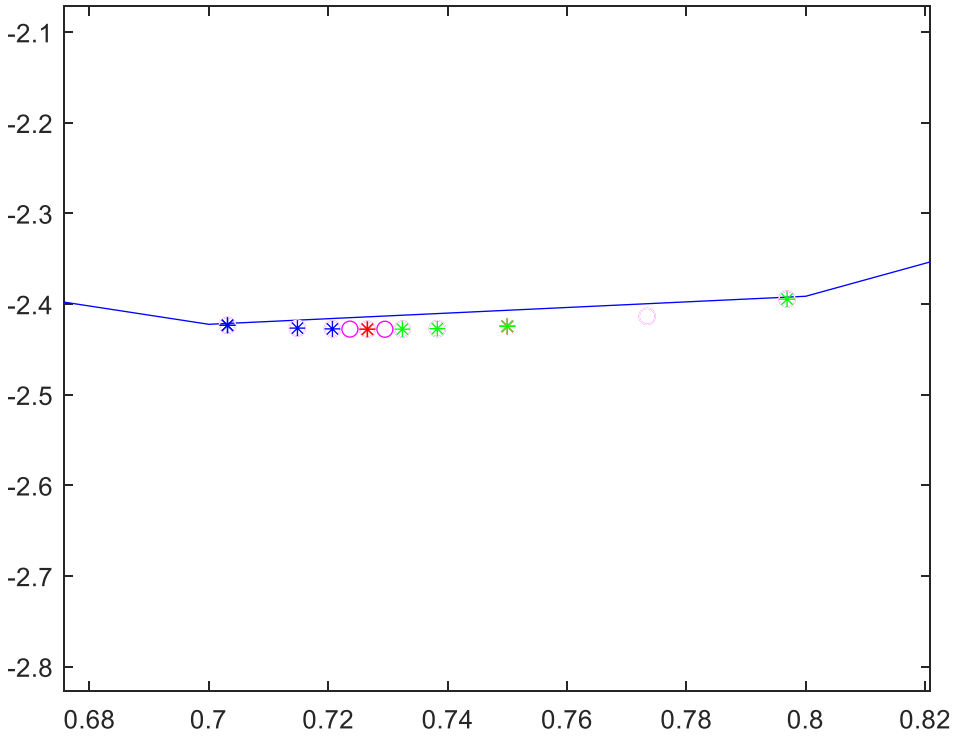
b=x2;


-2.1
-2.2
-2.3
-2.4
-2.5
-2.6
-2.7
-2.8
0.68
0.7
0.72
0.74
0.76
0.78
0.8
0.82

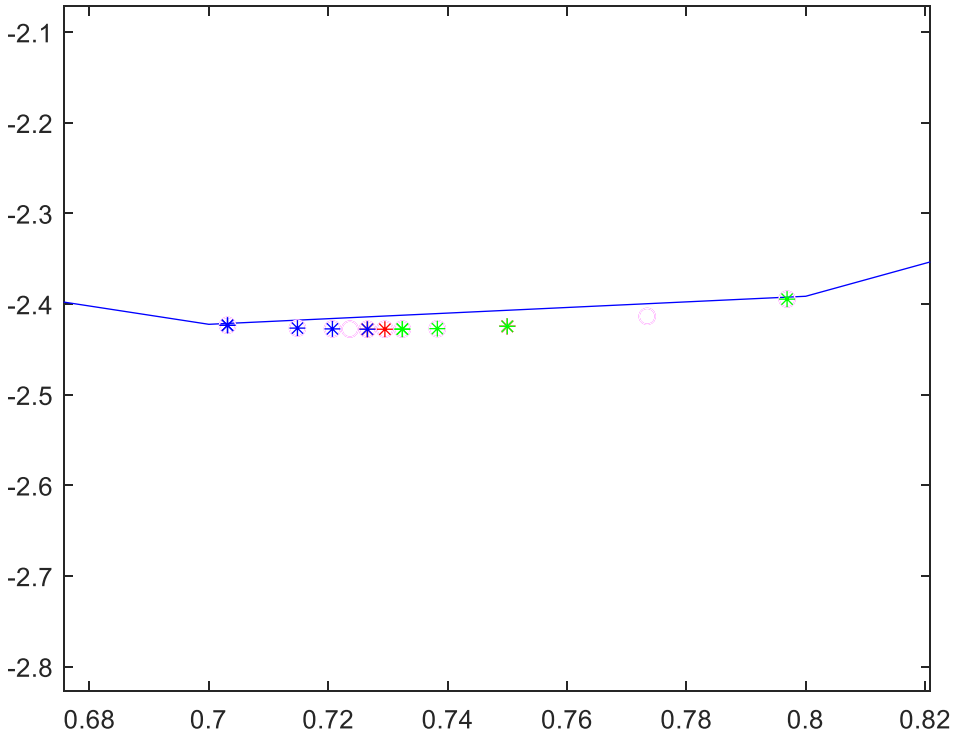


a=x1;
b=x2;


-2.1
-2.2
-2.3
-2.4
-2.5
-2.6
-2.7
-2.8
0.68
0.7
0.72
0.74
0.76
0.78
0.8
0.82



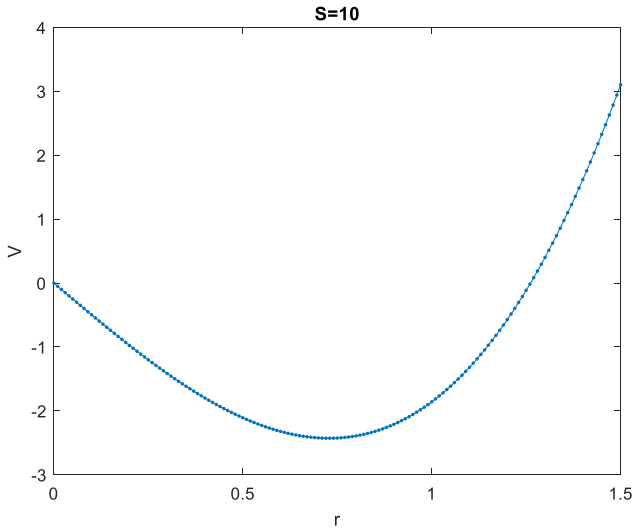

```
if puszka2(x2,S0)<puszka2(xm,S0)
a=xm;
      xm=x2;
```



L = b-a = 0.005859375

[a b]=[0.7265625 , 0.732421875]

```
>> MetElimObszaruDzielPrzezDwaOS
    0.3750    1.1250    0.7500         0    1.5000

    0.5625    0.9375    0.7500    0.3750    1.1250

    0.6563    0.8438    0.7500    0.5625    0.9375

    0.7031    0.7969    0.7500    0.6563    0.8438

    0.7266    0.7734    0.7500    0.7031    0.7969

    0.7148    0.7383    0.7266    0.7031    0.7500

    0.7207    0.7324    0.7266    0.7148    0.7383

    0.7236    0.7295    0.7266    0.7207    0.7324

    0.7266    0.7324
```

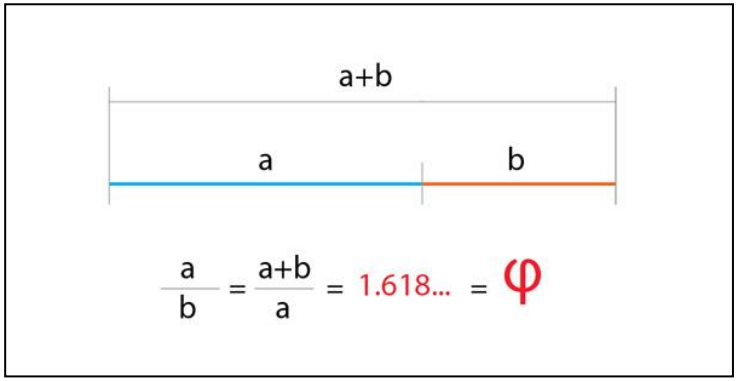


Metoda złotego podziału

W metodzie tej w każdej nowej iteracji potrzeba wyliczyć tylko jedna nowa wartość funkcji. Idea polega na tym, ze spośród dwóch punktów, które potrzebne są, aby stosować regułę eliminowania obszarów, **jeden punkt jest zawsze poprzednim a tylko drugi punkt jest nowy**. Ponadto przedział zawęża się za każda iteracja proporcjonalnie o tyle samo, czyli o wartość , która spełnia następująca zależność:

a+b=1

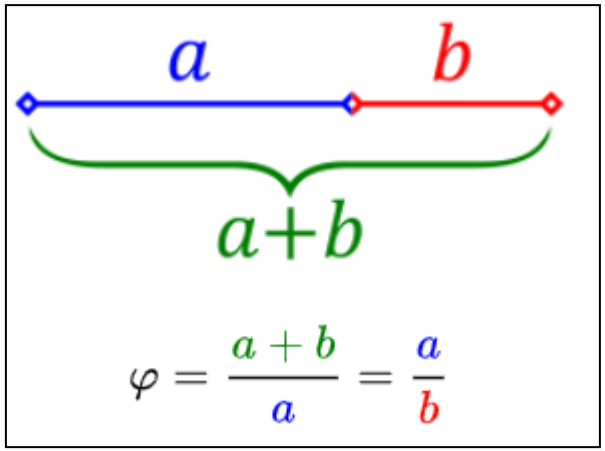
$$\frac{1-\rho}{1}=\frac{\rho}{1-\rho}\Rightarrow\rho=\frac{3-\sqrt{5}}{2}\approx 0.382$$



metodyOptymalizacjiProcesowProdukcyjnych\ProgramOS\MetElimObszaruZolotPodzialOS.m

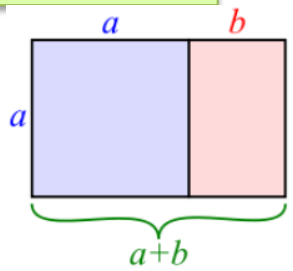
Złota liczba

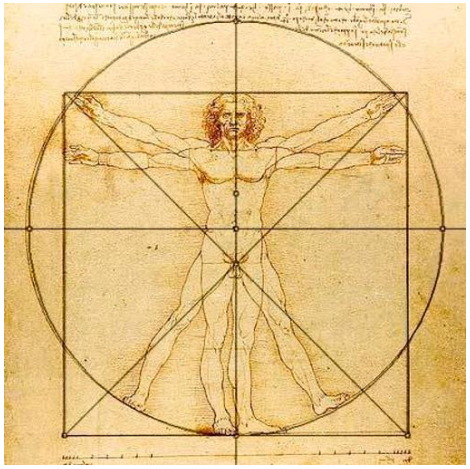
$$\varphi = \frac{1+\sqrt{5}}{2} = 1{,}61803\,39887\ldots$$

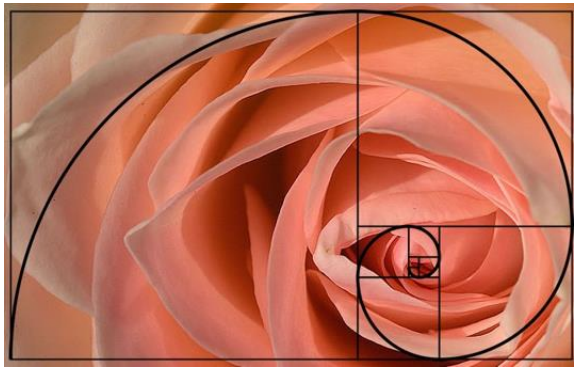


Złoty podział wykorzystuje się często w estetycznych, proporcjonalnych kompozycjach architektonicznych, malarskich, fotograficznych itp.

Złoty prostokąt może być rozcięty na kwadrat i mniejszy prostokąt o tych samych proporcjach co rozcinany.







Metoda złotego podziału (cd)

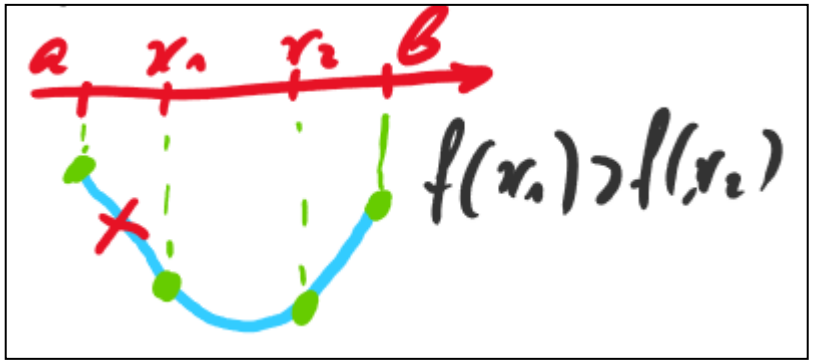
1) Wybierz dolne i górne ograniczenie przedziału a i b oraz małą liczbę ϵ. Ustal $k = 1$.

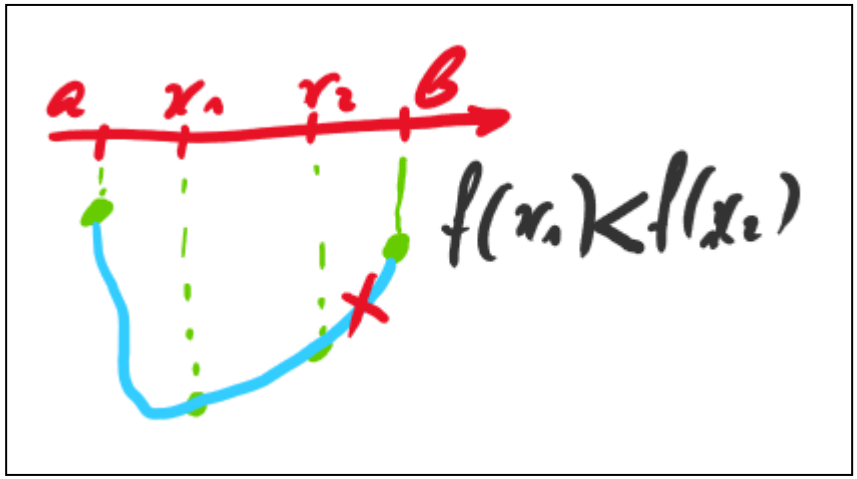
2) Ustal $w_1 = a + (1 - \rho)(b - a)$ oraz $w_2 = a + \rho(b - a)$. Wylicz $f(w_1)$ oraz $f(w_2)$, w zależności od tego, które z nich nie było wyliczone wcześniej. Wyeliminuj odpowiedni region zgodnie z regułą eliminowania obszarów. Ustal nowe a i b.

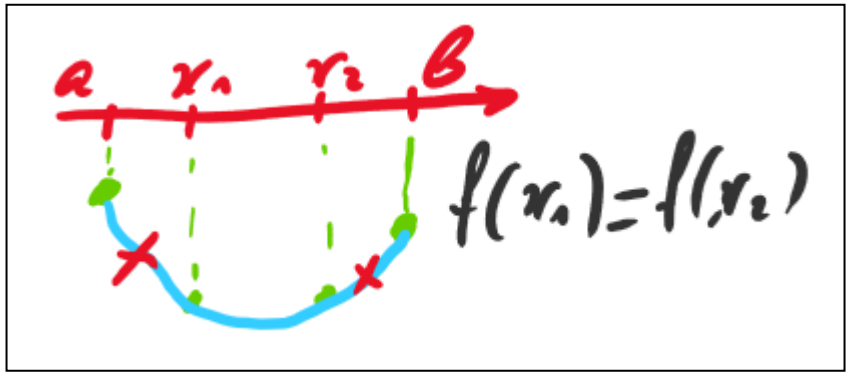
3) Czy $|a - b| < \epsilon$? Jeśli nie, ustal $k = k + 1$ i przejdź do kroku 2);
Jeśli tak, **Zakończ.**

Reguła eliminowania obszarów

- Jeśli $f(x_1) > f(x_2)$, to minimum nie leży w (a, x_1)
- Jeśli $f(x_1) < f(x_2)$, to minimum nie leży w (x_2, b)
- Jeśli $f(x_1) = f(x_2)$, to minimum nie leży ani w (a, x_1) ani w (x_2, b)



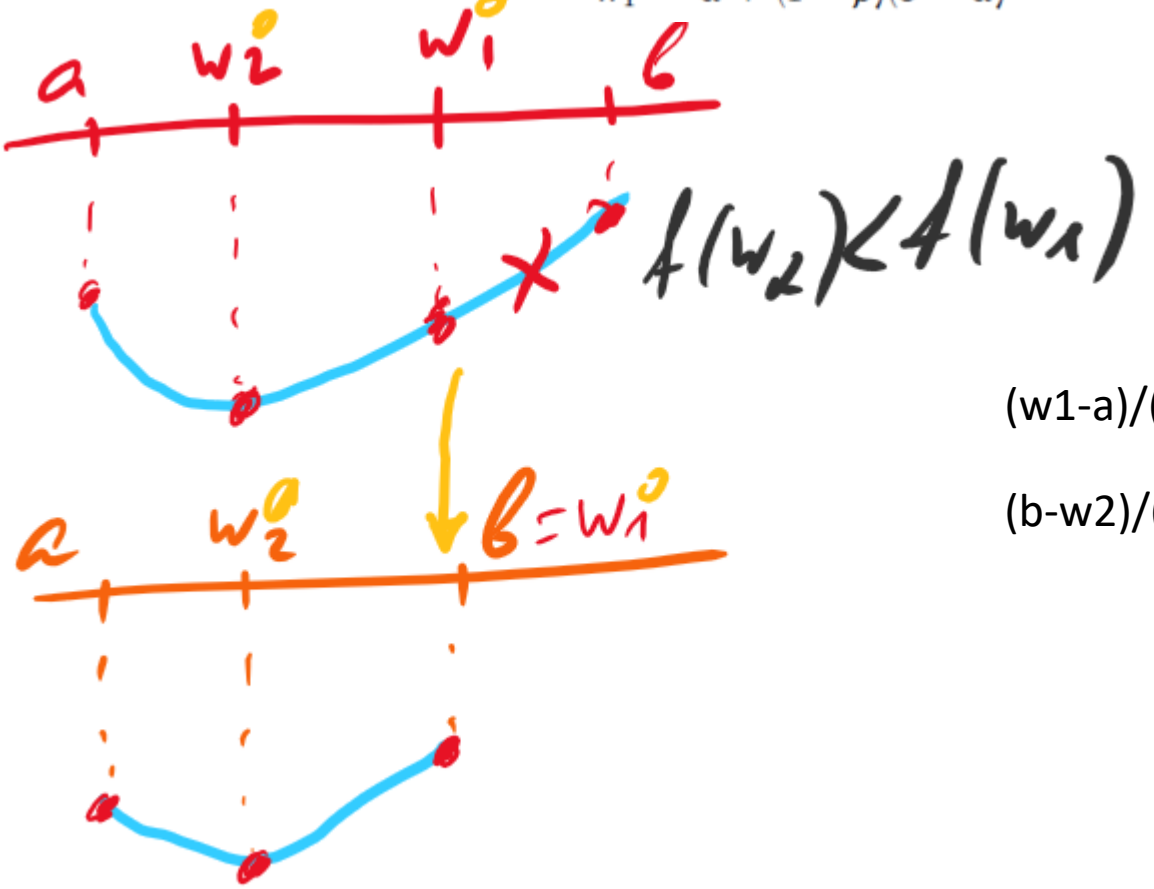




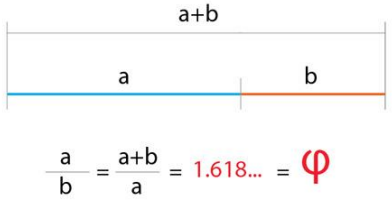
Przykład

$$w_2 = a + \rho(b - a)$$

$$w_1 = a + (1 - \rho)(b - a)$$



$$\rho = \frac{3 - \sqrt{5}}{2} \approx 0.382$$



(w1-a)/(w2-a)= 1.6180339887499

(b-w2)/(w2-a) = 1.6180339887499

Twierdzenie

Jeden punkt jest zawsze poprzednim a tylko drugi punkt jest nowy

$$b = w_1^0$$

$$w_1^1 = w_2^0$$

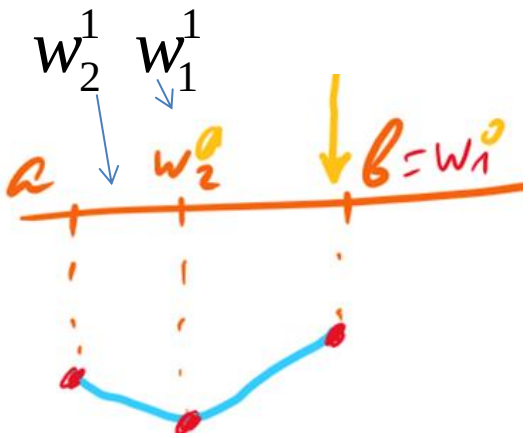
$$w_2^1 = a + \rho(b - a)$$

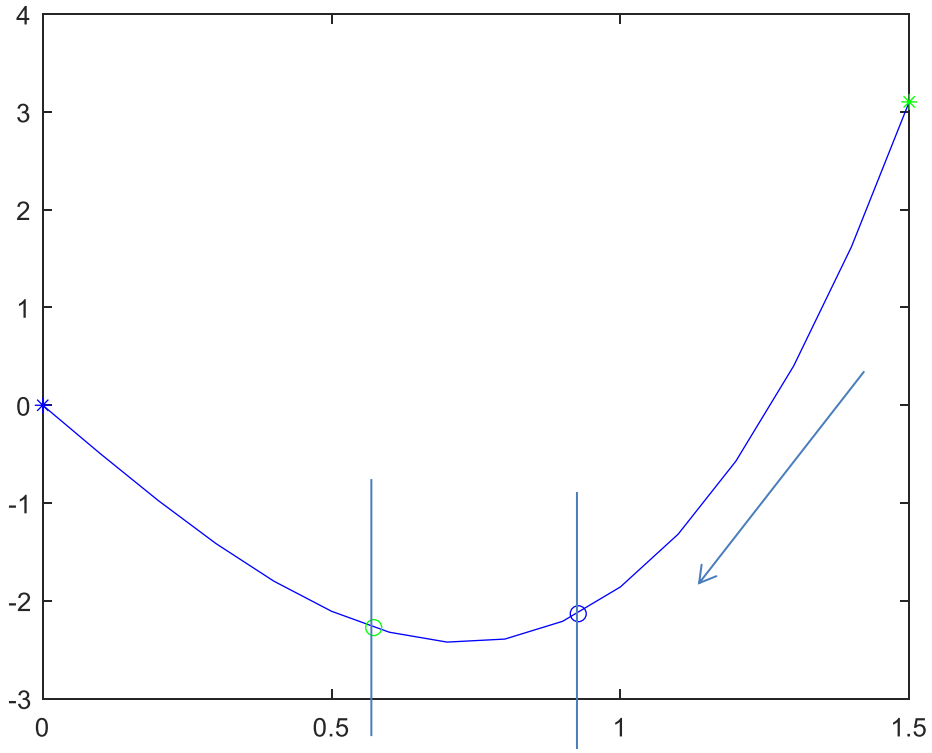
$$w_1^1 = a + (1-\rho)(b-a) = a + (1-\rho)\left(w_1^0 - a\right) =$$

$$\left\{\frac{\left(w_1^0 - a\right)}{\left(w_2^0 - a\right)} = \varphi = 1.618\right\} =$$

$$a + (1-\rho)\varphi \cdot \left(w_2^0 - a\right) =$$

$$\{(1-\rho)\varphi = 1\} = w_2^0$$



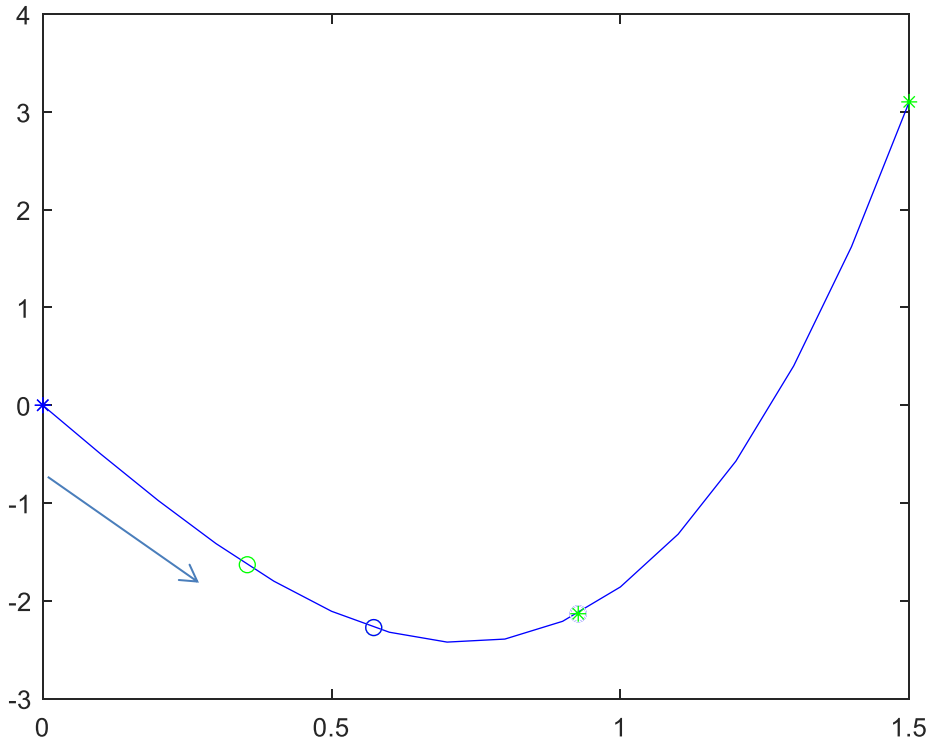


x1=min([w1 w2]);
x2=max([w1 w2]);

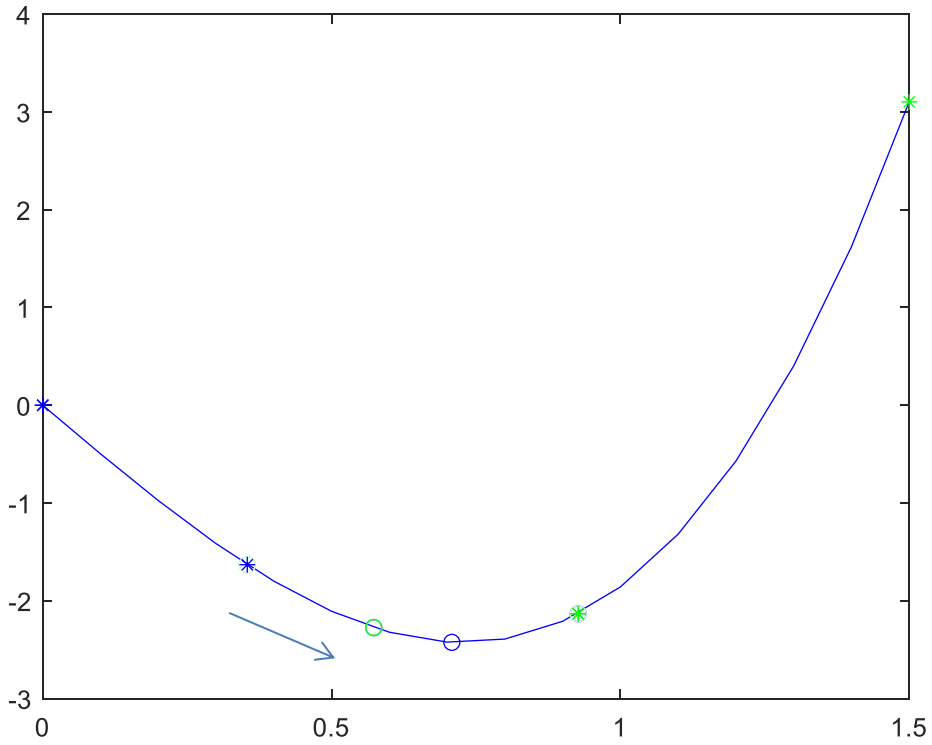
b=x2;

w2 w1

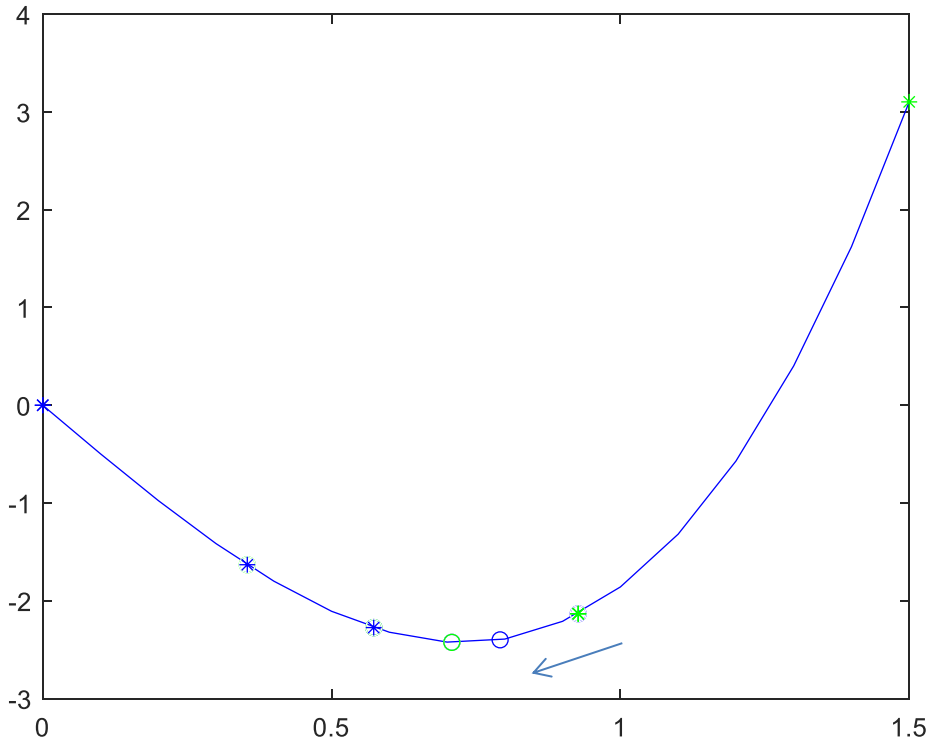
x1 x2



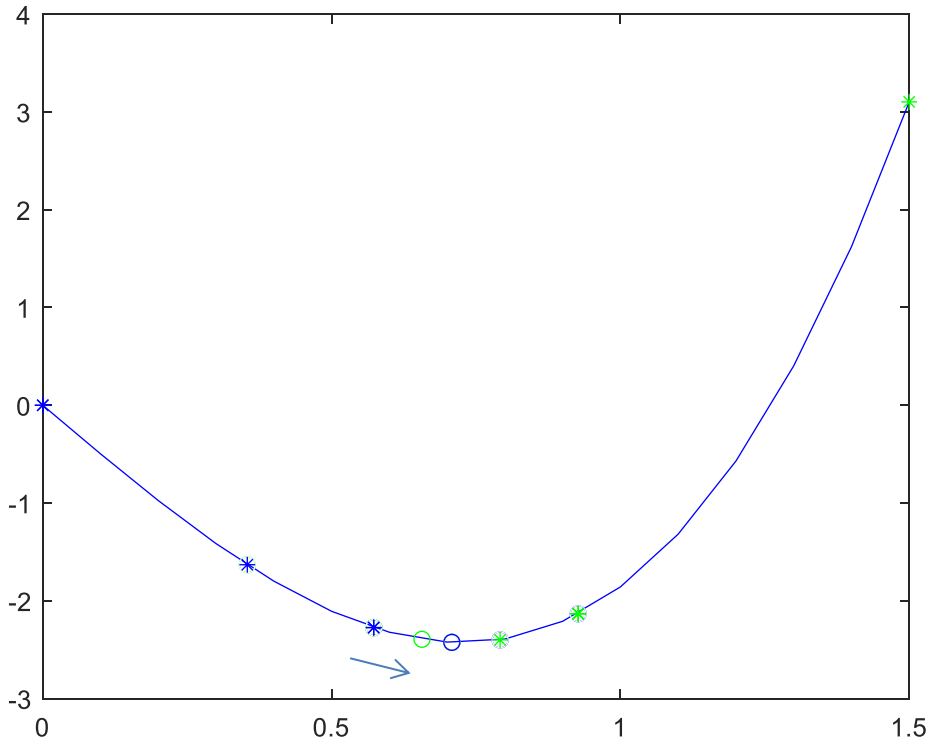
a=x1;



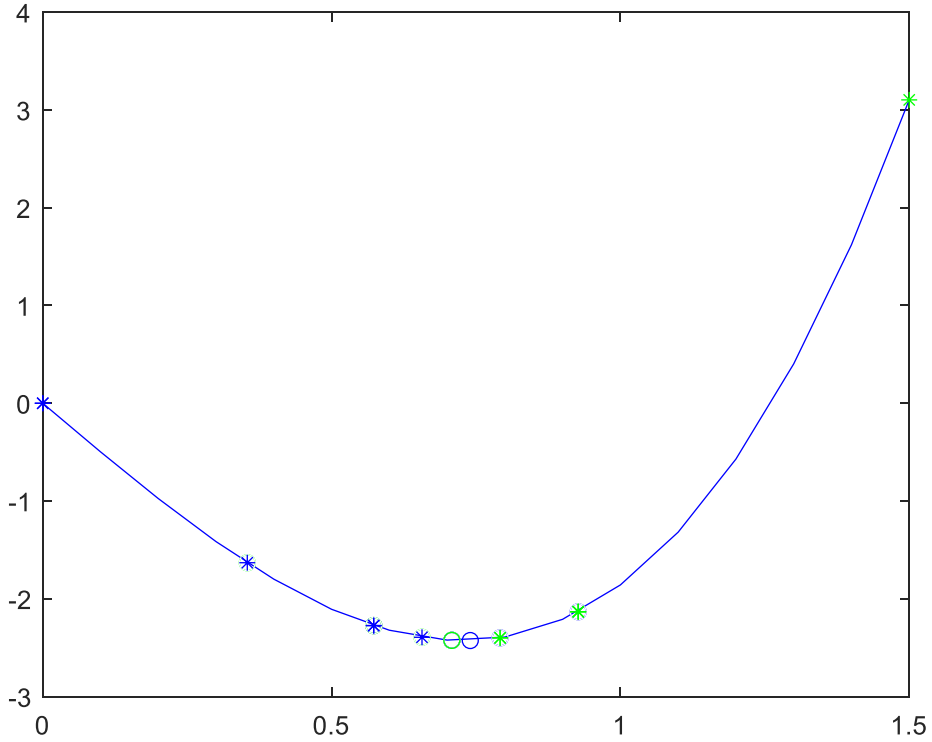
a=x1;



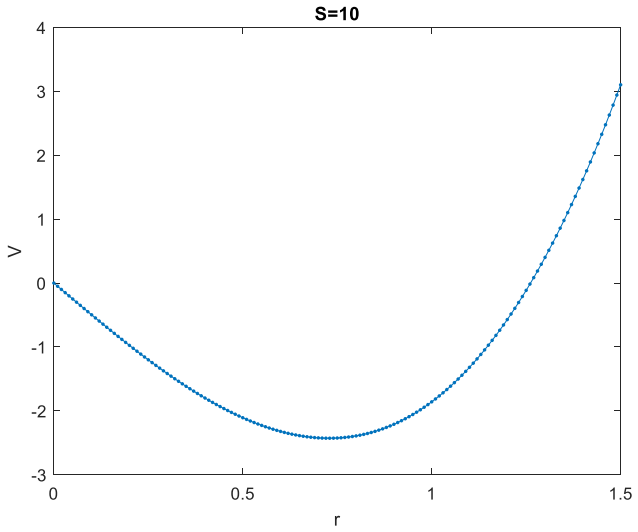
b=x2;



a=x1;


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5

```
>>
MetElimObszaruZolotPodzialOS
    0.5729    0.9271    1.5000

    0.3541    0.5729    0.9271

    0.5729    0.7082    0.5729

    0.7082    0.7918    0.3541

    0.6565    0.7082    0.2188

    0.7082    0.7401    0.1353

    0.7401    0.7599    0.0836

    0.7279    0.7401    0.0517

    0.7204    0.7279    0.0319

    0.7279    0.7326    0.0197

    0.7251    0.7279    0.0122
```



Metoda liczb Fibonacciego

W metodzie złotego podziału proporcja zmniejszania się przedziału z iteracji na iteracje pozostaje niezmienna i wynosi 0.618. W metodzie liczb Fibonacciego, idea jest taka sama jak w metodzie złotego podziału, z wyjątkiem faktu, ze w metodzie liczb Fibonacciego **proporcja zmniejszania się przedziału z iteracji na iteracje zmienia się tak, aby przedział zmniejszał się w sposób optymalny (tzn. jak najbardziej).**

$$\frac{1-\rho_{k+1}}{1} = \frac{\rho_k}{1-\rho_k}$$

$$\rho_k \in (0,1/2]$$

$$(1-\rho_1)(1-\rho_2)\mathrm{L}\ (1-\rho_N) \to \min$$

$$F_1 = 1, F_2 = 1$$

$$F_k = F_{k-1} + F_{k-2}$$

$$\rho_1 = 1 - \frac{F_N}{F_{N+1}}$$

$$\rho_2 = 1 - \frac{F_{N-1}}{F_N}\mathrm{L}$$

$$\rho_k = 1 - \frac{F_{N-k+1}}{F_{N-k+2}}$$

$$\mathrm{L}$$

$$\rho_N = 1 - \frac{F_1}{F_2}$$

Metoda liczb Fibonacciego (cd)

$$\begin{aligned} c_k &= a_k + \frac{F_{n-k-1}}{F_{n-k+1}} L_k, \\ d_k &= a_k + \frac{F_{n-k}}{F_{n-k+1}} L_k. \end{aligned}$$

$$L_{k+1} = \frac{F_{n-k}}{F_{n-k+1}} L_k$$

$$L_k = L_{k+1} + L_{k+2}, \qquad k = 1, 2, \ldots, n-3$$

Złoty podział: $L_{k+1} = \alpha L_k$

Metoda liczb Fibonacciego (cd)

1 INPUT $[a_1, b_1]$ and n
2 $L_1 = b_1 - a_1$
3 $c_1 = a_1 + \frac{F_{n-2}}{F_n} L_1$
4 $d_1 = a_1 + \frac{F_{n-1}}{F_n} L_1$
5 FOR $k = 1$ TO $n - 1$ DO
6 IF $f(c_k) > f(d_k)$
7 THEN $a_{k+1} = c_k, b_{k+1} = b_k, L_{k+1} = b_{k+1} - a_{k+1}$
8 $c_{k+1} = d_k, d_{k+1} = a_{k+1} + \frac{F_{n-k}}{F_{n-k+1}} L_{k+1}$
9 ELSE $a_{k+1} = a_k, b_{k+1} = d_k, L_{k+1} = b_{k+1} - a_{k+1}$
10 $c_{k+1} = a_{k+1} + \frac{F_{n-k-1}}{F_{n-k+1}} L_{k+1}, d_{k+1} = c_k$

MetElimObszaruFIbonacci2OS.m

a,b,L

OutD2 =

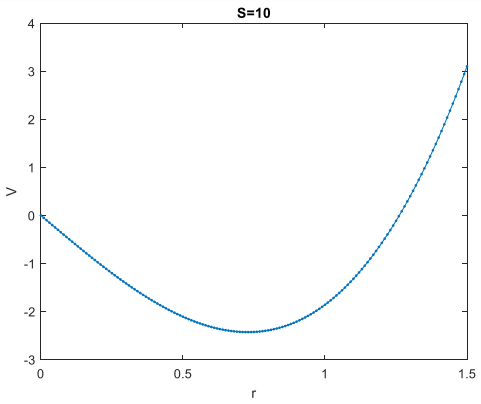
0	1.5000	1.5000
0.3750	1.1250	0.7500
0.5625	0.9375	0.3750
0.6563	0.8438	0.1875
0.7031	0.7969	0.0938
0.7031	0.7500	0.0469
0.7148	0.7383	0.0234
0.7207	0.7324	0.0117
0.7266	0.7324	0.0059

OutZP =

0	1.5000	1.5000
0	0.9271	0.9271
0.3541	0.9271	0.5729
0.5729	0.9271	0.3541
0.5729	0.7918	0.2188
0.6565	0.7918	0.1353
0.7082	0.7918	0.0836
0.7082	0.7599	0.0517
0.7082	0.7401	0.0319

OutF =

0	1.5000	1.5000
0	0.9273	0.9273
0.3540	0.9273	0.5732
0.5727	0.9273	0.3545
0.5727	0.7922	0.2195
0.6571	0.7922	0.1351
0.7081	0.7922	0.0841
0.7081	0.7586	0.0505
0.7081	0.7416	0.0335



Złoty podział vs. Fibonacci

$$L_{k+1} = \alpha L_k$$

$$L_k = L_{k+1} + L_{k+2}, \qquad k = 1, 2, \ldots, n-3$$

```
OutZP(1:10,3)./OutZP(2:11,3)

ans =

   1.6180
   1.6180
   1.6180
   1.6180
   1.6180
   1.6180
   1.6180
   1.6180
   1.6180
   1.6180
```

```
[OutF(1:7,3) OutF(2:8,3)+OutF(3:9,3)]

ans =

   1.5000   1.5005
   0.9273   0.9278
   0.5732   0.5740
   0.3545   0.3545
   0.2195   0.2192
   0.1351   0.1346
   0.0841   0.0839
```

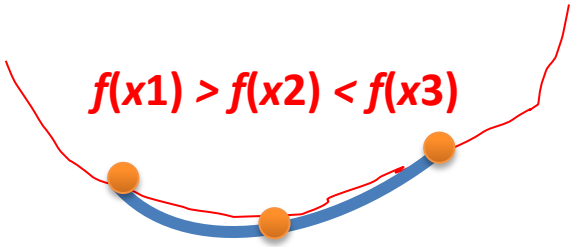
Metody estymacji punktowej. Metoda interpolacji kwadratowej Powella

20022024

Szukamy trzy punkty ***x1 < x2 < x3*** takie, ze wartości funkcji w tych punktach spełniają ***f(x1) > f(x2) < f(x3)***.

Szukamy równania **wielomianu kwadratowego** przechodzącego przez punkty (*x1; f*(*x1*)), (*x2; f*(*x2*)) i (*x3; f*(*x3*)).

$$q(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2)$$



$$q(x_1) = f(x_1)$$
$$q(x_2) = f(x_2)$$
$$q(x_3) = f(x_3)$$

MetElimObszaruInterpolacjaOS.m

Metoda interpolacji kwadratowej Powella (cd)

W tym celu zapisujemy ogólne równanie wielomianu kwadratowego przechodzącego przez punkty x_1 i x_2:

$$q(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2)$$

Następnie szukamy współczynników tego wielomianu:

$$\begin{aligned} q(x_1) &= f(x_1) = a_0 \\ q(x_2) &= f(x_2) = a_0 + a_1(x_2 - x_1) \\ q(x_3) &= f(x_3) = a_0 + a_1(x_3 - x_1) + a_2(x_3 - x_1)(x_3 - x_2) \end{aligned}$$

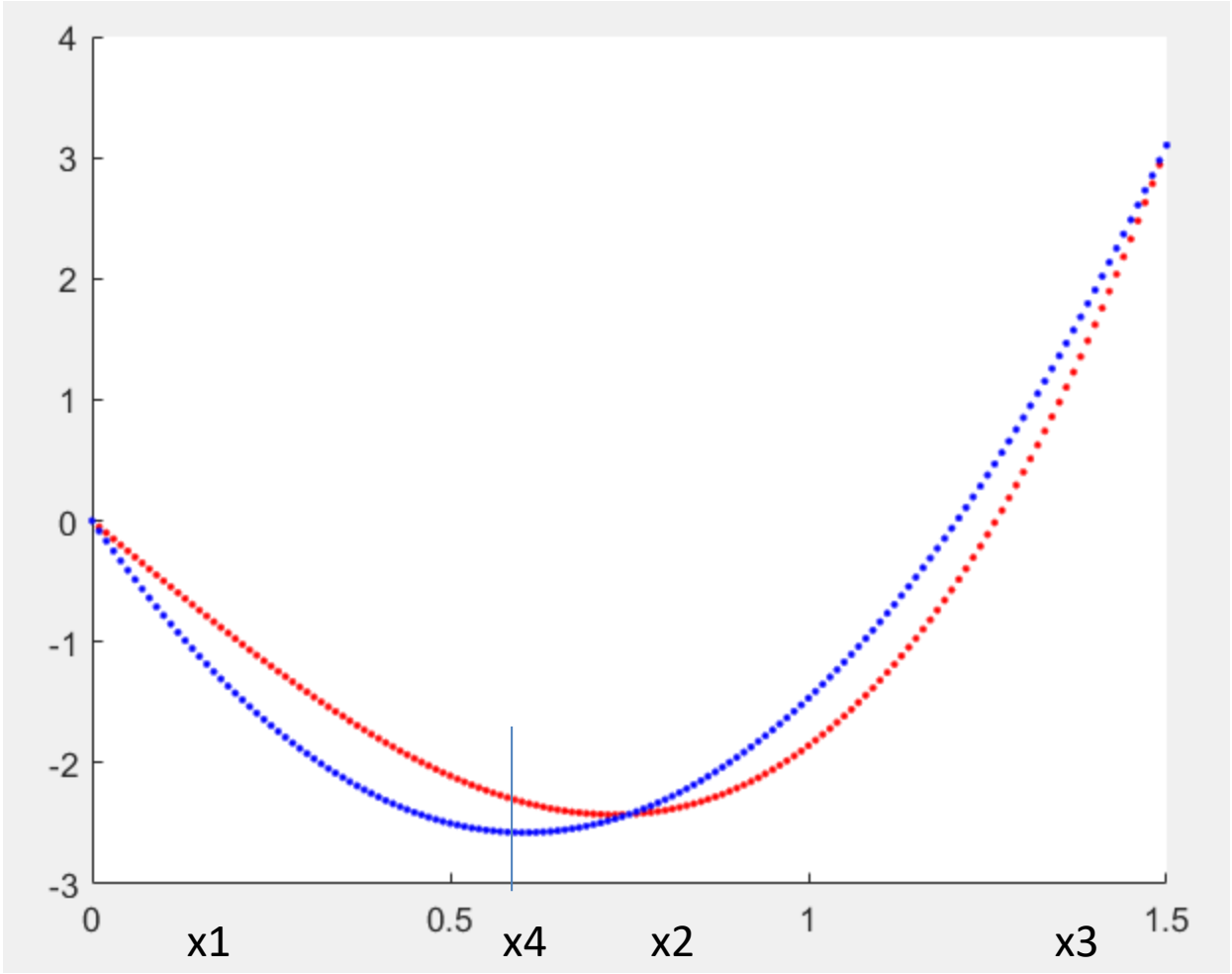
Teraz szukamy argumentu, dla którego ten wielomian kwadratowy osiąga minimum. Ponieważ zgodnie z naszymi założeniami $a_2 > 0$, minimum znajduje się tam, gdzie pochodna równa jest zero.

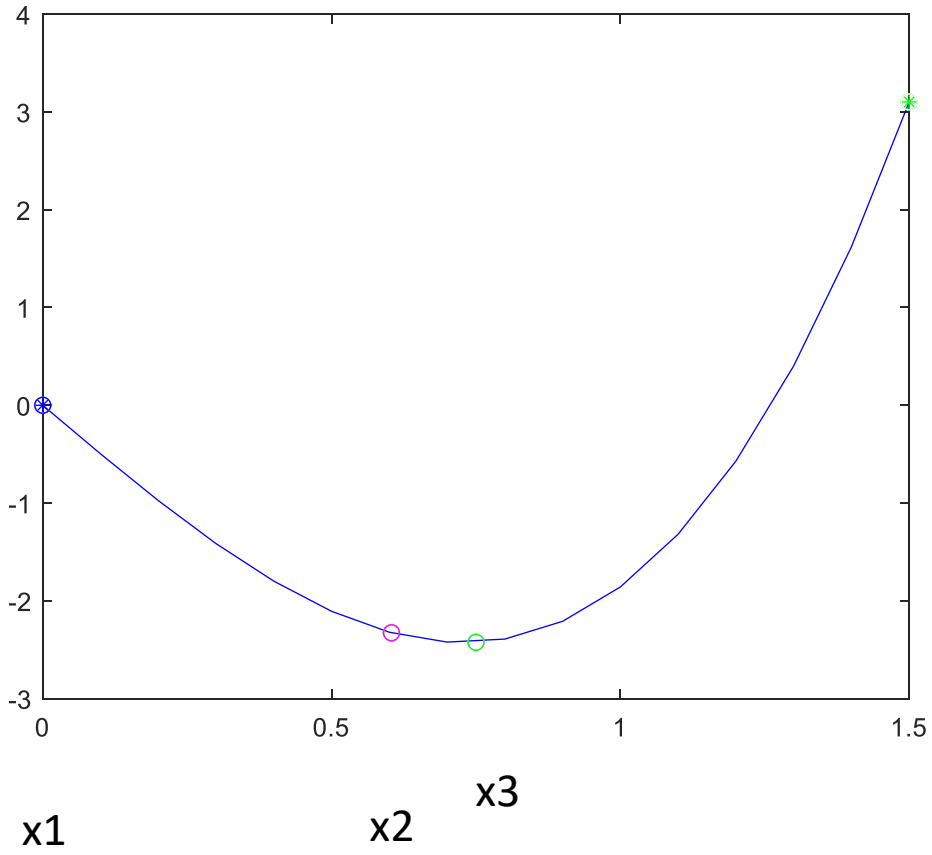
$$q'(x) = 0 \Rightarrow a_1 + a_2(x - x_2 + x - x_1) = 0 \Rightarrow x^* = \frac{x_1 + x_2}{2} - \frac{a_1}{2a_2}$$

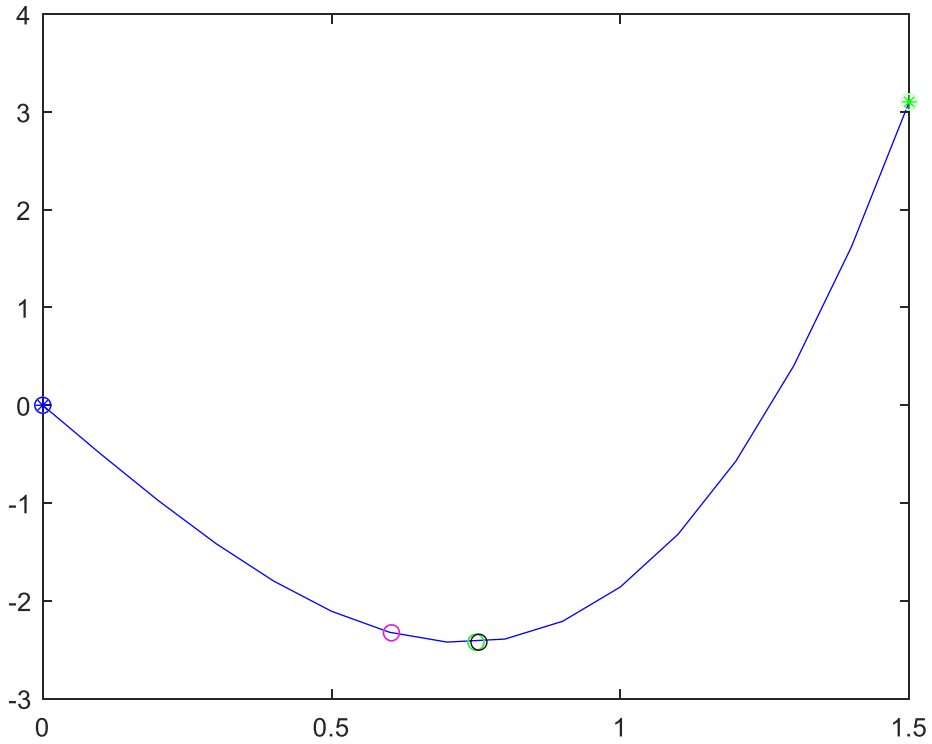
Metoda interpolacji kwadratowej Powella (cd)

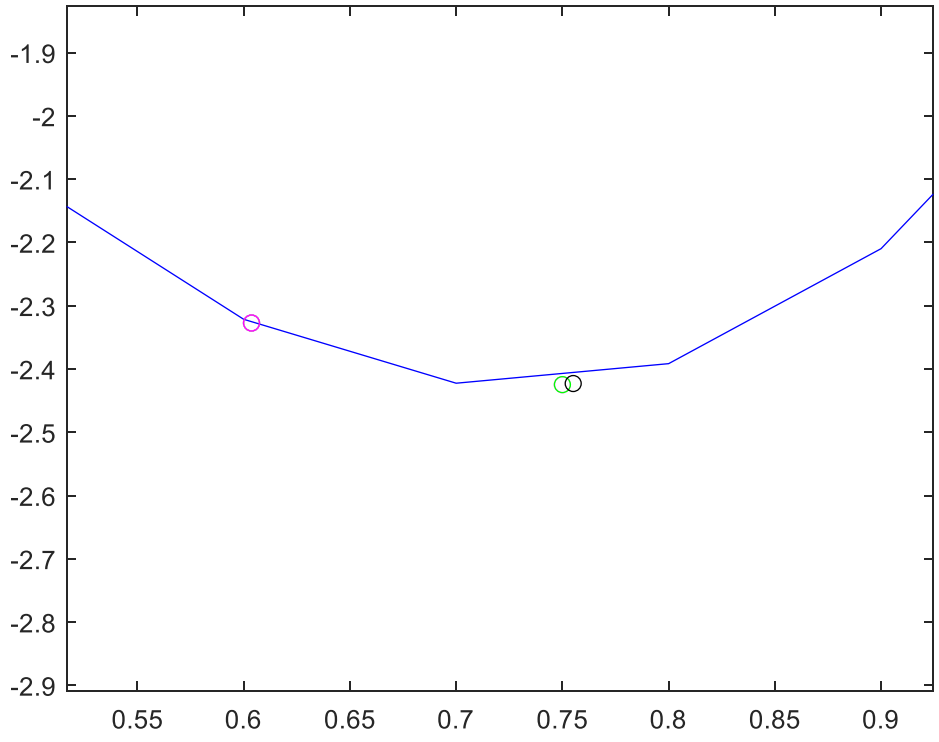
W punkcie x^* wielomian kwadratowy $q(x)$ osiąga minimum. Ponieważ $q(x)$ jest przybliżeniem funkcji $f(x)$, której minimum szukamy, x^* jest przybliżeniem wartości, w której funkcja $f(x)$ osiąga minimum. Spośród punktów (x_1, x_2, x_3, x^*), zatrzymujemy trzy najlepsze (innymi słowy wyrzucamy punkt, w którym wartość funkcji $f(x)$ jest największa) i ponownie dokonujemy interpolacji kwadratowej dla tych trzech punktów i szukamy minimum otrzymanego wielomianu. Procedura ta powtarzana jest do momentu, kiedy osiągnięta zostanie żądana dokładność.

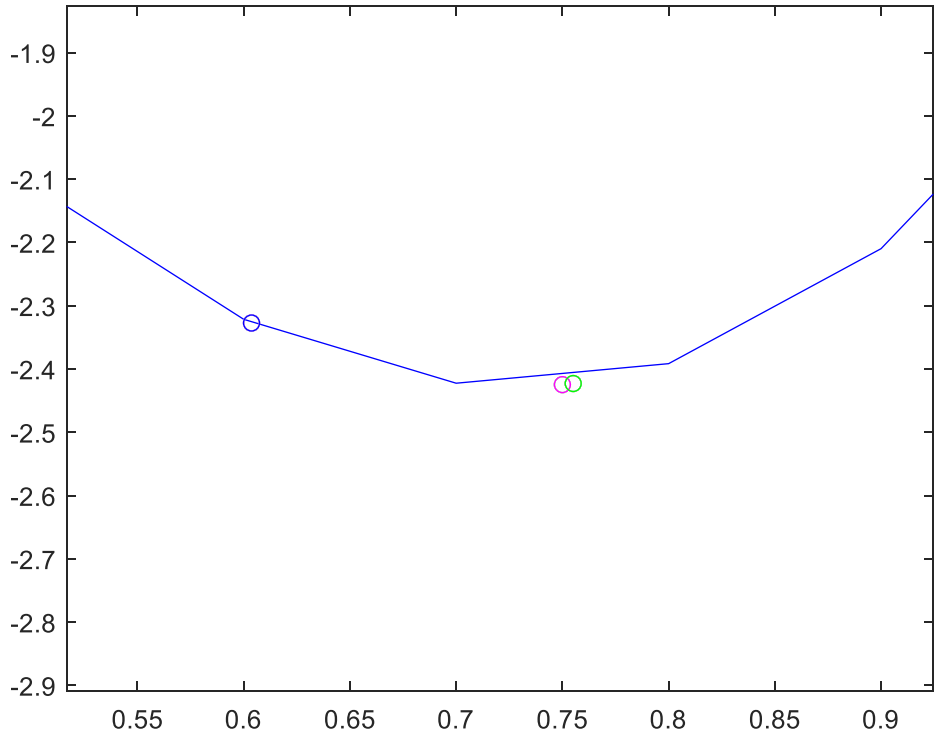
Alogorytm Powella, który został zarysowany powyżej znajduje minimum szybciej niż metoda złotego podziału, jeśli funkcja $f(x)$ nie jest skrzywiona. Dla mocno niesymetrycznych funkcji, metoda złotego podziału pozostaje jednak lepsza.

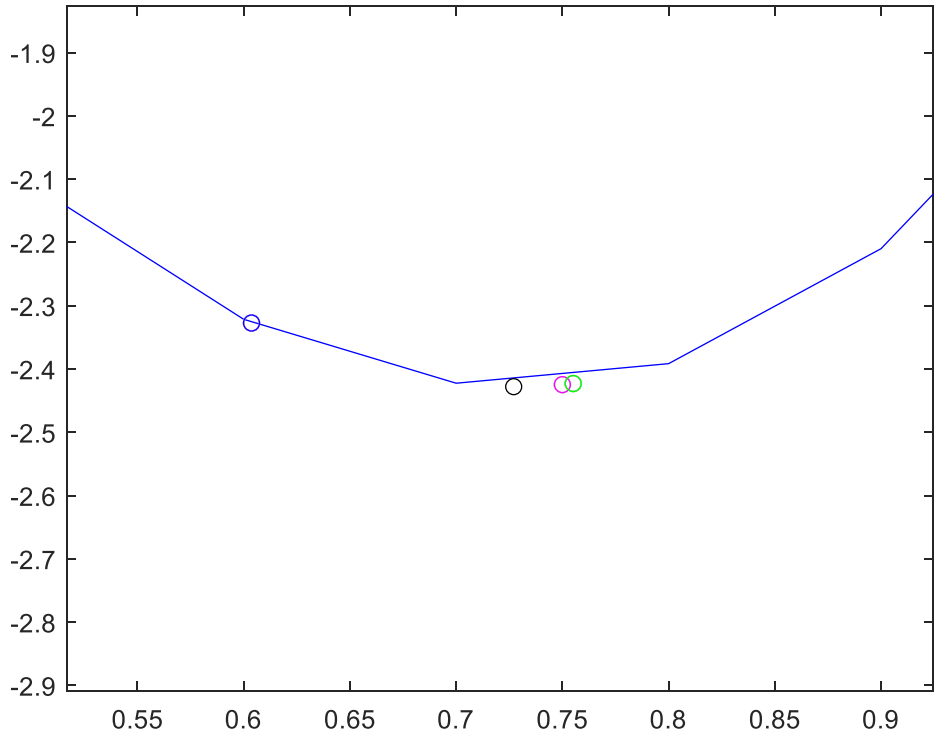

4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5
x1
x4
x2
x3


4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5
x1
x2
x3

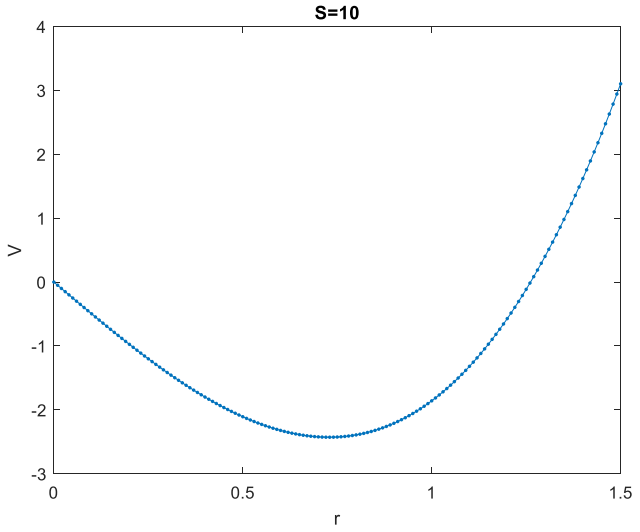

4
3
2
1
0
-1
-2
-3
0
0.5
1
1.5

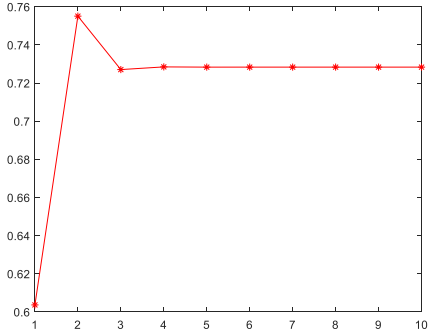

-1.9
-2
-2.1
-2.2
-2.3
-2.4
-2.5
-2.6
-2.7
-2.8
-2.9
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9


-1.9
-2
-2.1
-2.2
-2.3
-2.4
-2.5
-2.6
-2.7
-2.8
-2.9
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9

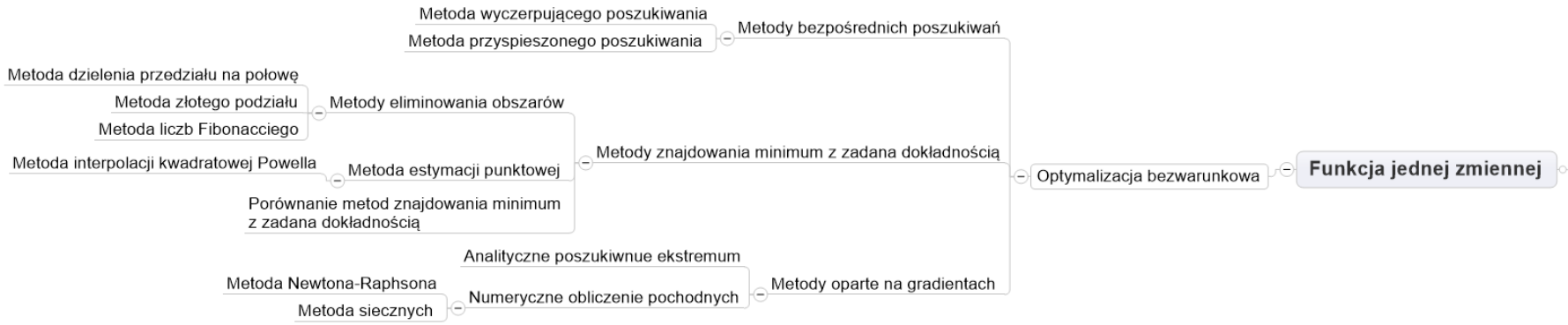

-1.9
-2
-2.1
-2.2
-2.3
-2.4
-2.5
-2.6
-2.7
-2.8
-2.9
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9

```
>> MetElimObszaruInterpolacjaOS
Sprawdzamy warunek f(x1)>f(x2)<f(x3)
         0   -2.4246    3.1029


x =

         0    0.6037    0.7500    0.6037

         0   -2.3272   -2.4246


x =

    0.6037    0.7500    0.7551    0.7551

   -2.3272   -2.4246   -2.4229
```





Schemat metod badań funkcji



Metody oparte na gradientach

Metody opisane dotychczas wykorzystywały **tylko wartości funkcji**. Metody oparte na gradientach wykorzystują natomiast **dodatkowo** informacje o pochodnych funkcji.

Ekstrema funkcji jednej zmiennej

$$y = f(x)$$

Rzeczywista ciągle różnicowana funkcja jednej zmiennej

Analityczne obliczenie pochodnej $\left.\frac{df(x)}{dx}\right|_{x=\alpha} = 0$

Metoda gradientowa (pośrednia)

$\left.\frac{d^2 f(x)}{dx^2}\right|_{x=\alpha} > 0$

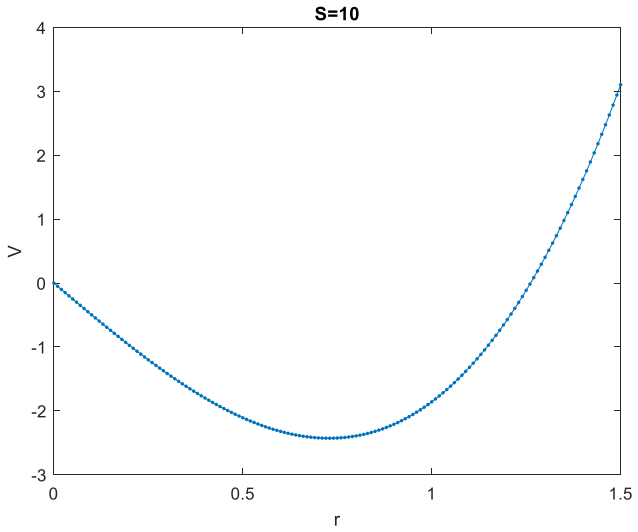
minimum

$\left.\frac{d^2 f(x)}{dx^2}\right|_{x=\alpha} < 0$

maksimum

Przykład

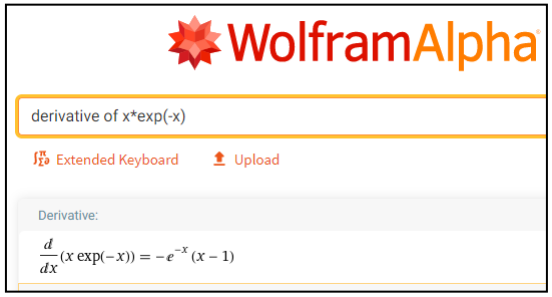
$$V(r^*) = -\left(\frac{r^*}{2} S_0 - \pi r^{*3} \right)$$




```
syms  x r S0 ;
v=-(r/2*S0-pi*r^3);
z=diff(v)
xm=solve(z)
S0 = 10;
 (6^(1/2)*S0^(1/2))/(6*pi^(1/2))

ans =0.728365620394719
```

```
>> diff(z)

ans =

6*pi*r
```

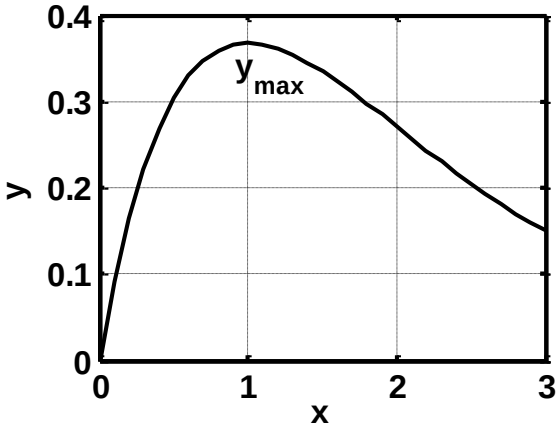
minimum



Przykład 1

$$y = xe^{-x} \qquad 0 \le x \le 3$$

```
>> x = 0 : 0.1 : 3 ;
y = x.*exp(-x) ;
plot (x,y)
grid on
xlabel('x'), ylabel('y')
text(0.9,0.34,'y_m_a_x','FontSize',12)
```



$$z = \frac{dy}{dx} \qquad z = 1/e^{x} - x/e^{x}$$

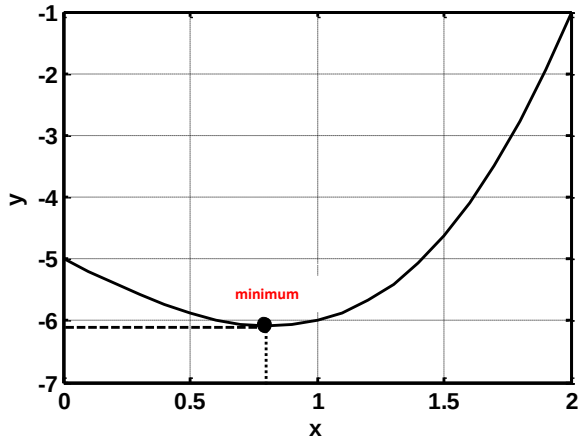
```
>>syms  x
y = x*exp(-x) ;
z=diff(y)
z =
1/exp(x) - x/exp(x)
```

```
>> xm=solve('1/exp(x) - x/exp(x)=0')
xm =
1
>>xm=1;
ymax = xm*exp(-xm)
ymax =
   0.3679
```

Przykład 2

$$y = x^3 - 2x - 5 \qquad 0 \le x \le 2$$

```
>> x = 0 : 0.1 : 2;
y = x.^3 - 2*x-5;
plot (x,y)
grid on
xlabel('x'), ylabel('y')
```



$$z = \frac{dy}{dx}$$

```
>>syms  x
y = x^3 - 2*x-5;
z=diff(y)
z =
 3*x^2 - 2
```

```
>> digits(5)
xm=vpa(solve('3*x^2 - 2=0'))
xm =
  -0.8165
  0.8165
>> xm=single(solve('3*x^2 - 2=0'))
```

$$0 \le x \le 2$$

```
>> xm = 0.8185;
ymin = xm^3-2*xm-5
 ymin =
 -6.0887
```

Numeryczne obliczenie pochodnych

Metoda różnic centralnych

$$f\left(x+\Delta x\right)=f\left(x\right)+\Delta x f'\left(x\right)+\frac{1}{2}\left(\Delta x\right)^2 f''\left(x\right)+\ldots$$

$$f\left(x-\Delta x\right)=f\left(x\right)-\Delta x f'\left(x\right)+\frac{1}{2}\left(-\Delta x\right)^2 f''\left(x\right)+\ldots$$

$$f'\left(x^{(k)}\right)=\frac{f\left(x^{(k)}+\Delta x^{(k)}\right)-f\left(x^{(k)}-\Delta x^{(k)}\right)}{2\Delta x^{(k)}}$$

$$f''\left(x^{(k)}\right)=\frac{f\left(x^{(k)}+\Delta x^{(k)}\right)-2f\left(x^{(k)}\right)+f\left(x^{(k)}-\Delta x^{(k)}\right)}{\left(\Delta x^{(k)}\right)^2}$$

Metoda Newtona-Raphsona

Załóżmy, że możemy policzyć $f(x^{(k)}), f'(x^{(k)})$ i $f''(x^{(k)})$ w każdym punkcie pomiaru funkcji $x^{(k)}$. Możemy zdefiniować wielomian kwadratowy, którego pierwsza i druga pochodna oraz wartść w punkcie $x^{(k)}$ są identyczne z tymi dla funkcji $f(x)$. Ten wielomian ma następującą postać:

$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2$$

Zamiast minimalizować funkcję $f(x)$ minimalizujemy jej przybliżenie $q(x)$. Warunek pierwszego rzędu na istnienie minimum jest następujący:

Metoda iteracyjna ⟹ $0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)})$

Nowy punkt $x = x^{(k+1)}$ spełnia zatem:

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}$$

$$f'\left(x^{(k)}\right) = \frac{f\left(x^{(k)} + \Delta x^{(k)}\right) - f\left(x^{(k)} - \Delta x^{(k)}\right)}{2\Delta x^{(k)}}$$

$$f''\left(x^{(k)}\right) = \frac{f\left(x^{(k)} + \Delta x^{(k)}\right) - 2f\left(x^{(k)}\right) + f\left(x^{(k)} - \Delta x^{(k)}\right)}{\left(\Delta x^{(k)}\right)^2}$$

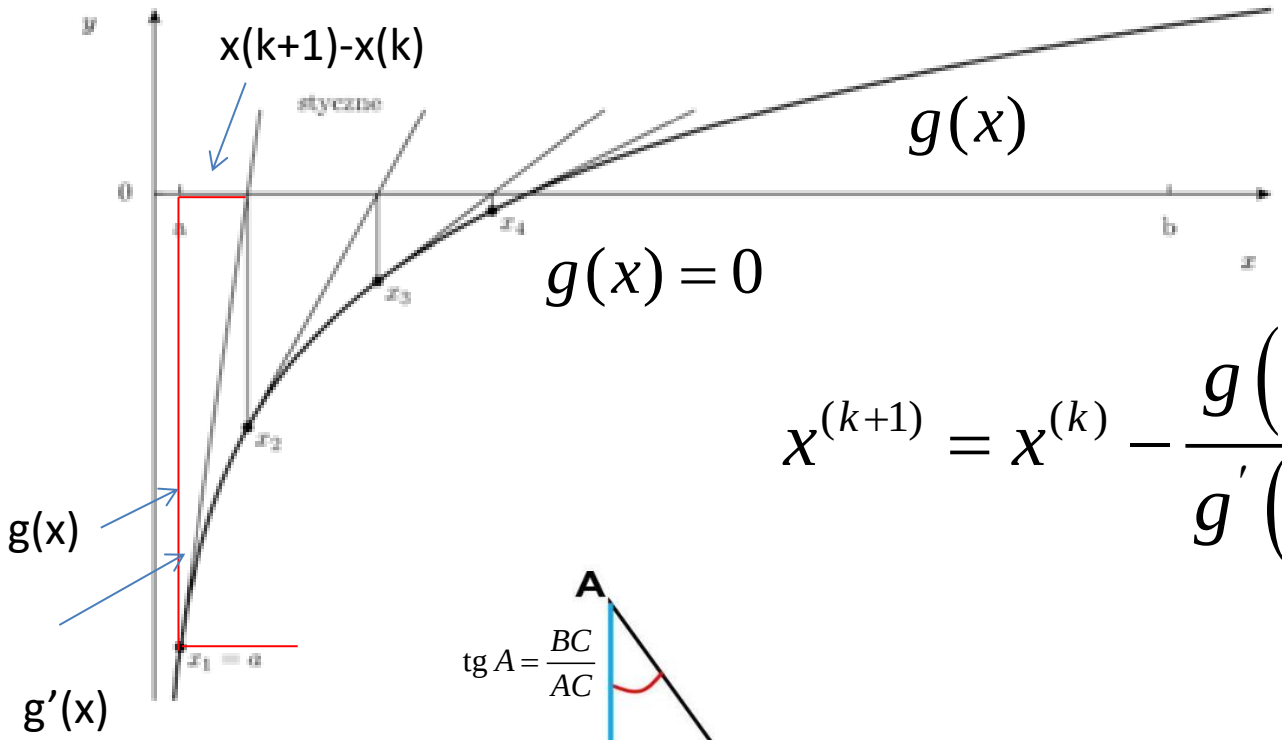
Metoda Newtona-Raphsona (cd)

Metoda Newtona-Raphsona polega na kontynuowaniu powyższej procedury do momentu, w którym pochodna $f'(x^{(k+1)})$ będzie wystarczająco blisko zera. Jeśli podstawimy $g(x) = f'(x)$, wtedy otrzymujemy formułę do iteracyjnego poszukiwania rozwiązania równania $g(x) = 0$:

$$x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)})}{g'(x^{(k)})}$$

Metoda Newtona-Raphsona działa dobrze jeśli $f''(x) > 0$ wszędzie. Jeśli natomiast $f''(x) < 0$ dla pewnego x, algorytm może nie zbiegać do minimum.

Metoda Newtona-Raphsona (cd)



$$g(x) = 0$$

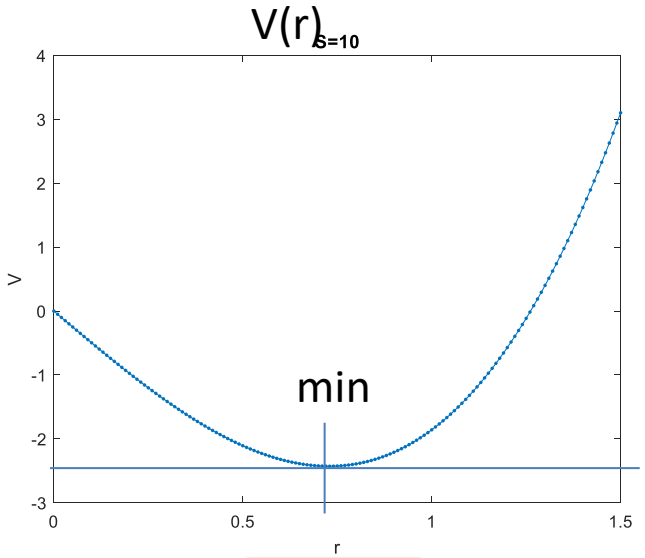
$$x^{(k+1)} = x^{(k)} - \frac{g\left(x^{(k)}\right)}{g'\left(x^{(k)}\right)}$$

$$g(x) = f'(x)$$

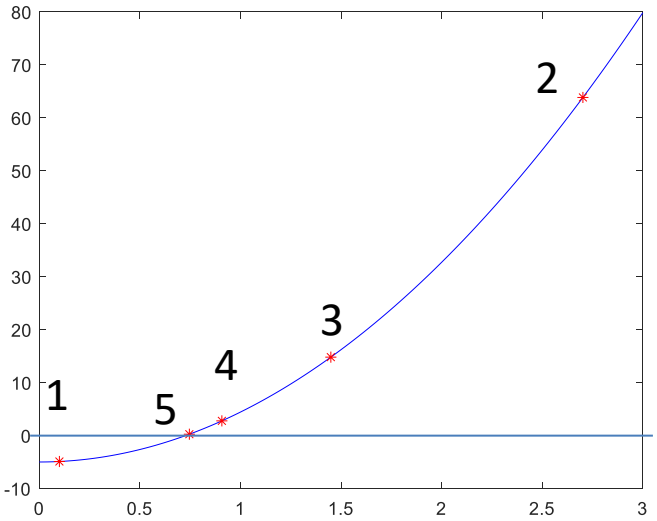
Przykład

$$V(r^*) = -\left(\frac{r^*}{2}S_0 - \pi r^{*3}\right)$$

$$g(r) = \frac{dV}{dr} = 3\pi r^2 - \frac{S_0}{2}$$



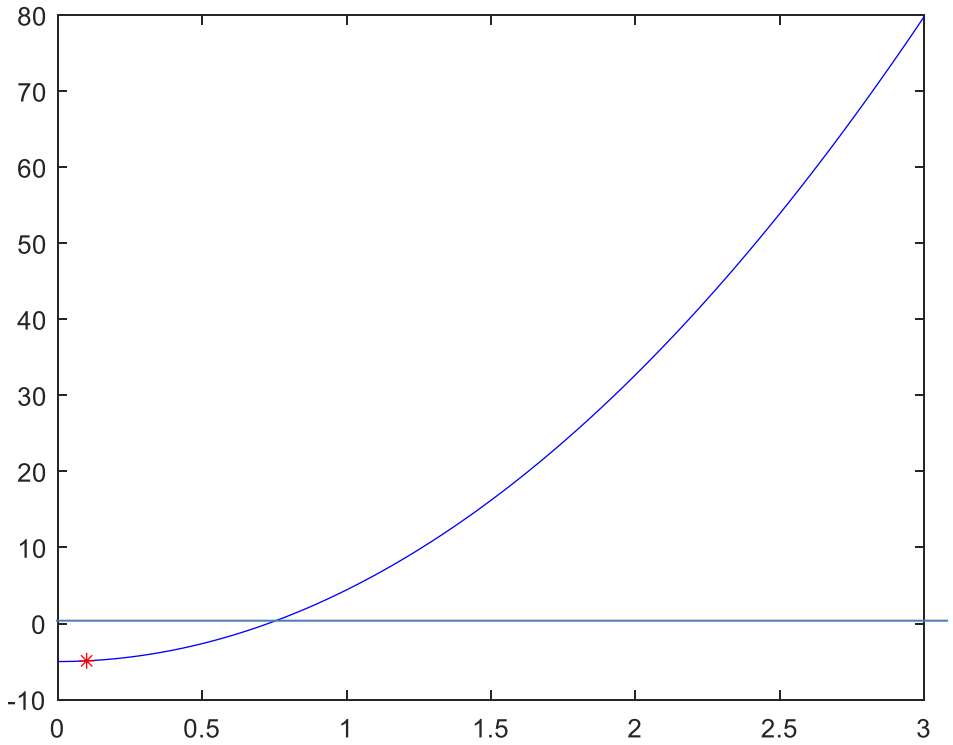
0.7283



$$g(r) = \frac{dV}{dr} = 3\pi r^2 - \frac{S_0}{2}$$

$$g(r^*) = 0$$

g(x)

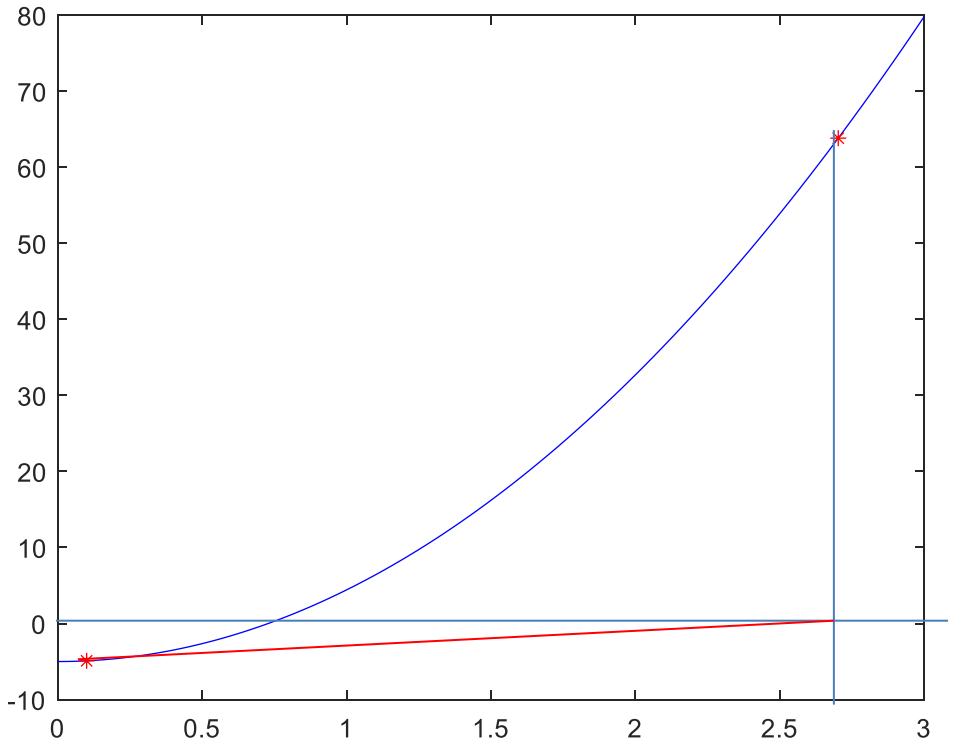



```
>> NewonOS
>> Out

Out =

         0    0.1000
    1.0000    2.7024
    2.0000    1.4494
    3.0000    0.9077
    4.0000    0.7461
    5.0000    0.7286
    6.0000    0.7283
    7.0000    0.7283
    8.0000    0.7283
    9.0000    0.7283
   10.0000    0.7283
```

$$x^{(k+1)} = x^{(k)} - \frac{g\left(x^{(k)}\right)}{g'\left(x^{(k)}\right)}$$

$$g(r) = \frac{dV}{dr} = 3\pi r^2 - \frac{S_0}{2}$$

$$g'(r) = \frac{dV}{dr} = 6\pi r$$

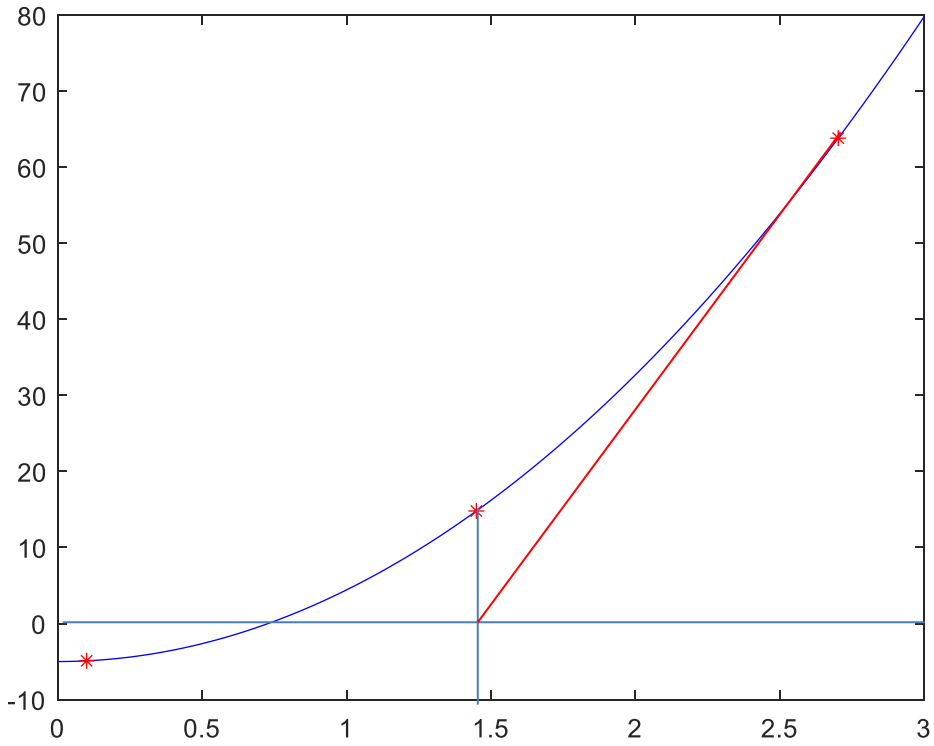



```
>> NewonOS
>> Out

Out =

         0    0.1000
    1.0000    2.7024
    2.0000    1.4494
    3.0000    0.9077
    4.0000    0.7461
    5.0000    0.7286
    6.0000    0.7283
    7.0000    0.7283
    8.0000    0.7283
    9.0000    0.7283
   10.0000    0.7283
```

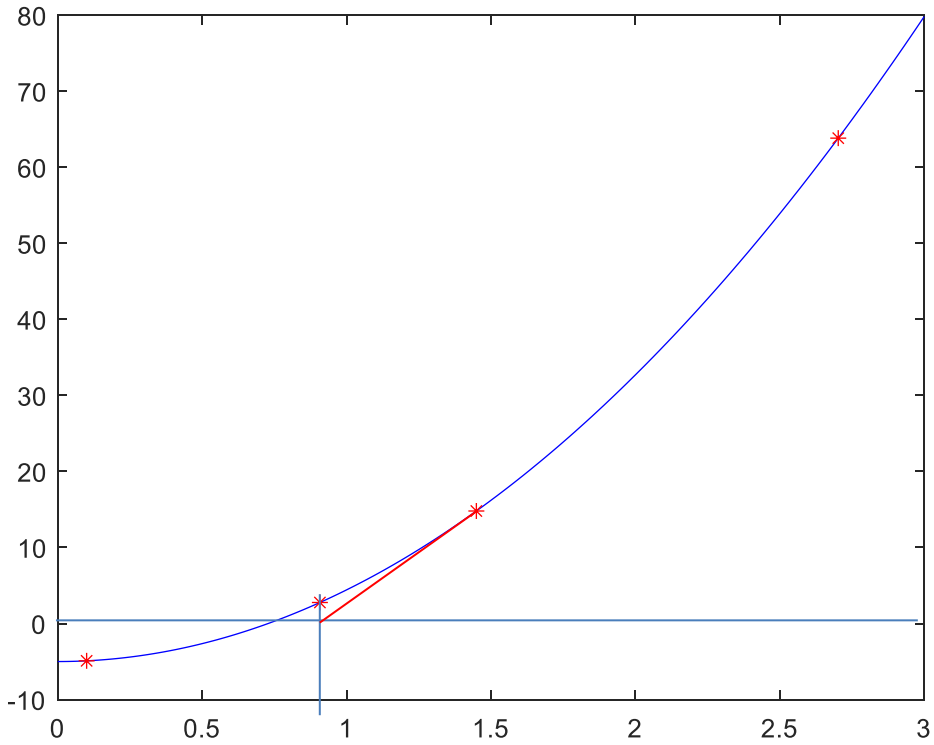
$$x^{(k+1)} = x^{(k)} - \frac{g\left(x^{(k)}\right)}{g'\left(x^{(k)}\right)}$$

$$g(0.1) = 3\pi 0.1^2 - \frac{S_0}{2}$$

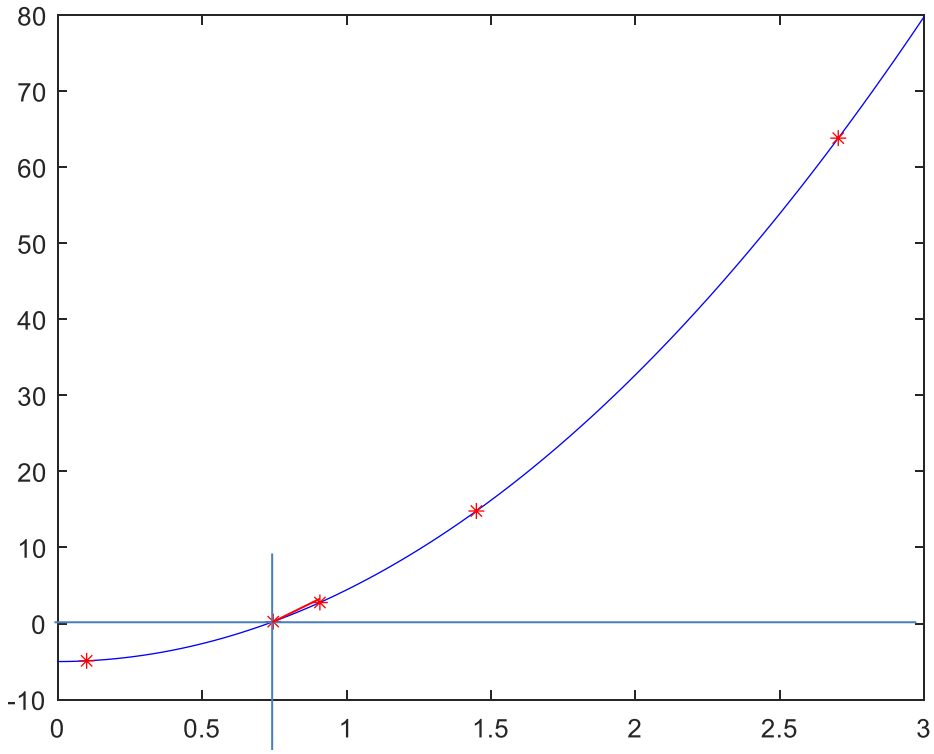
$$g'(0.1) = \frac{dV}{dr} = 6\pi 0.1$$




```
>> NewonOS
>> Out

Out =

         0    0.1000
    1.0000    2.7024
    2.0000    1.4494
    3.0000    0.9077
    4.0000    0.7461
    5.0000    0.7286
    6.0000    0.7283
    7.0000    0.7283
    8.0000    0.7283
    9.0000    0.7283
   10.0000    0.7283
```

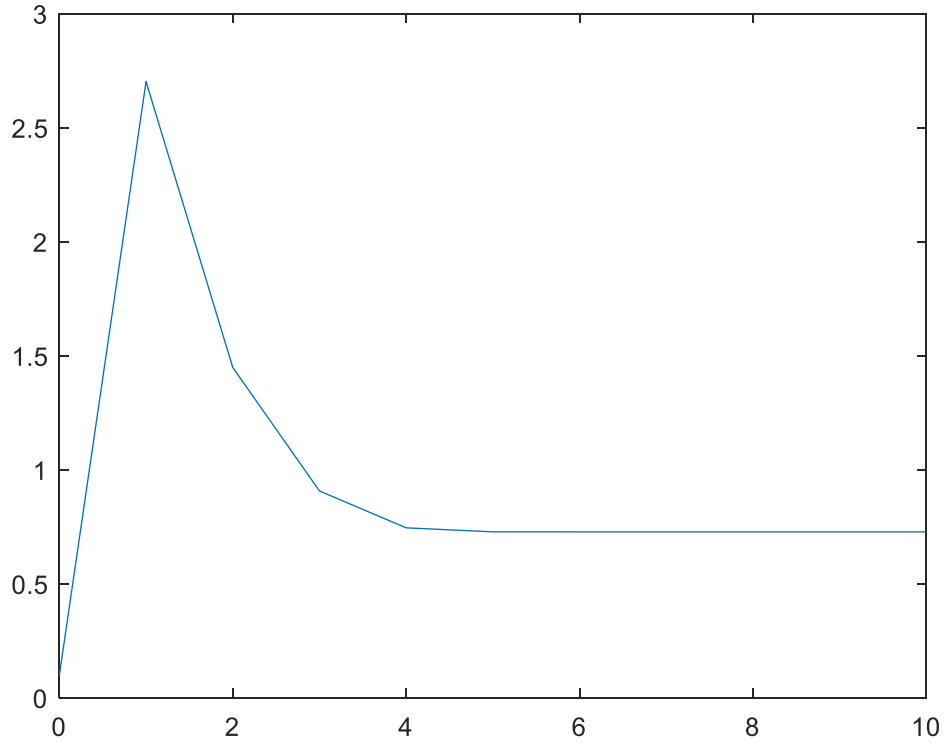



```
>> NewonOS
>> Out

Out =

         0    0.1000
    1.0000    2.7024
    2.0000    1.4494
    3.0000    0.9077
    4.0000    0.7461
    5.0000    0.7286
    6.0000    0.7283
    7.0000    0.7283
    8.0000    0.7283
    9.0000    0.7283
   10.0000    0.7283
```

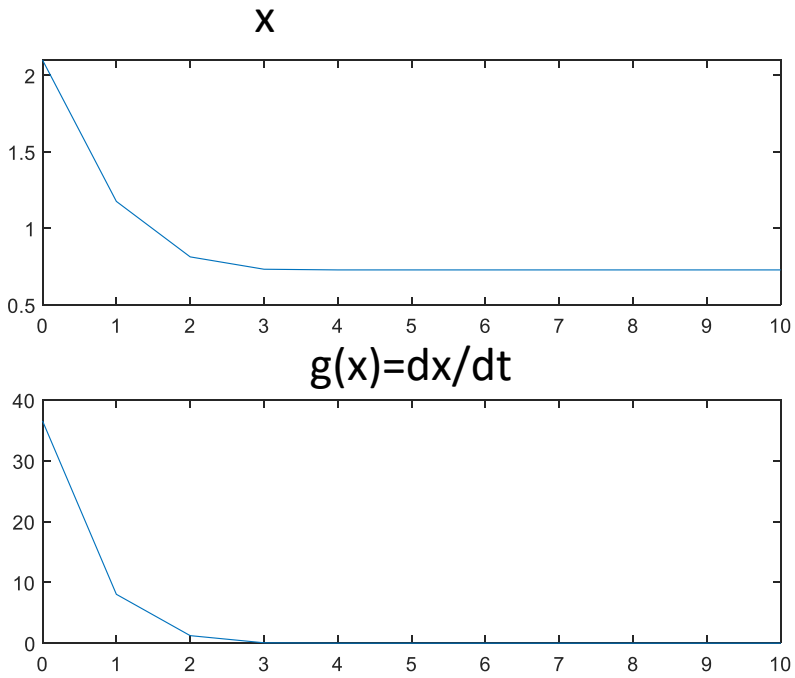


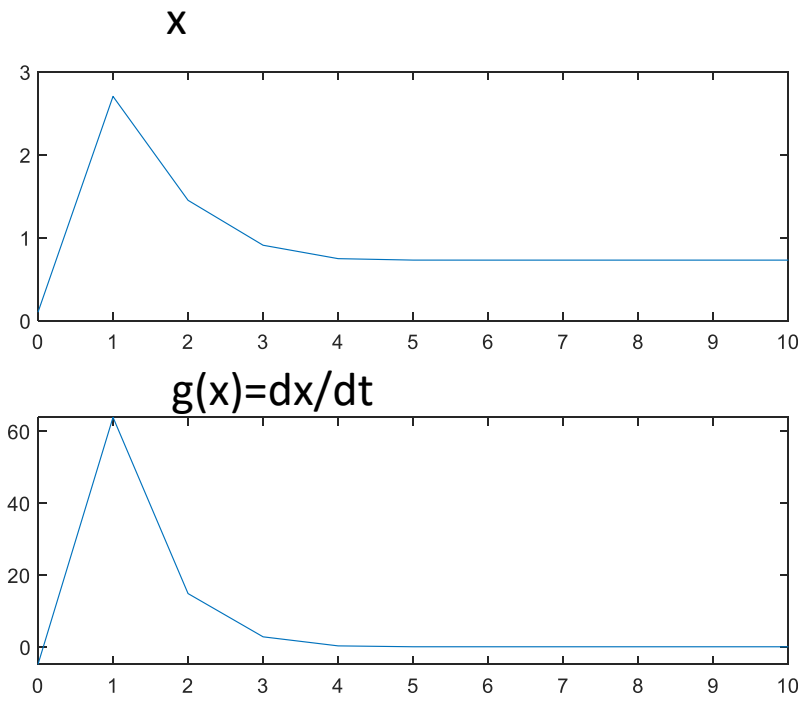

```
>> NewonOS
>> Out

Out =

         0    0.1000
    1.0000    2.7024
    2.0000    1.4494
    3.0000    0.9077
    4.0000    0.7461
    5.0000    0.7286
    6.0000    0.7283
    7.0000    0.7283
    8.0000    0.7283
    9.0000    0.7283
   10.0000    0.7283
```

```
>> NewonOS
>> Out

Out =

         0    0.1000
    1.0000    2.7024
    2.0000    1.4494
    3.0000    0.9077
    4.0000    0.7461
    5.0000    0.7286
    6.0000    0.7283
    7.0000    0.7283
    8.0000    0.7283
    9.0000    0.7283
   10.0000    0.7283
```



newtonOS.m


x
2
1.5
1
0.5
0 1 2 3 4 5 6 7 8 9 10
g(x)=dx/dt
40
30
20
10
0
0 1 2 3 4 5 6 7 8 9 10


x
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10
g(x)=dx/dt
60
40
20
0
0 1 2 3 4 5 6 7 8 9 10

Metoda siecznych (ang. secant method)

$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2}f''(x^{(k)})(x - x^{(k)})^2$$

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}$$

Jest to metoda podobna do Newtona-Raphsona. Zamiast $f''(x^{(k)})$, używa się następującego przybliżenia:

$$\frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

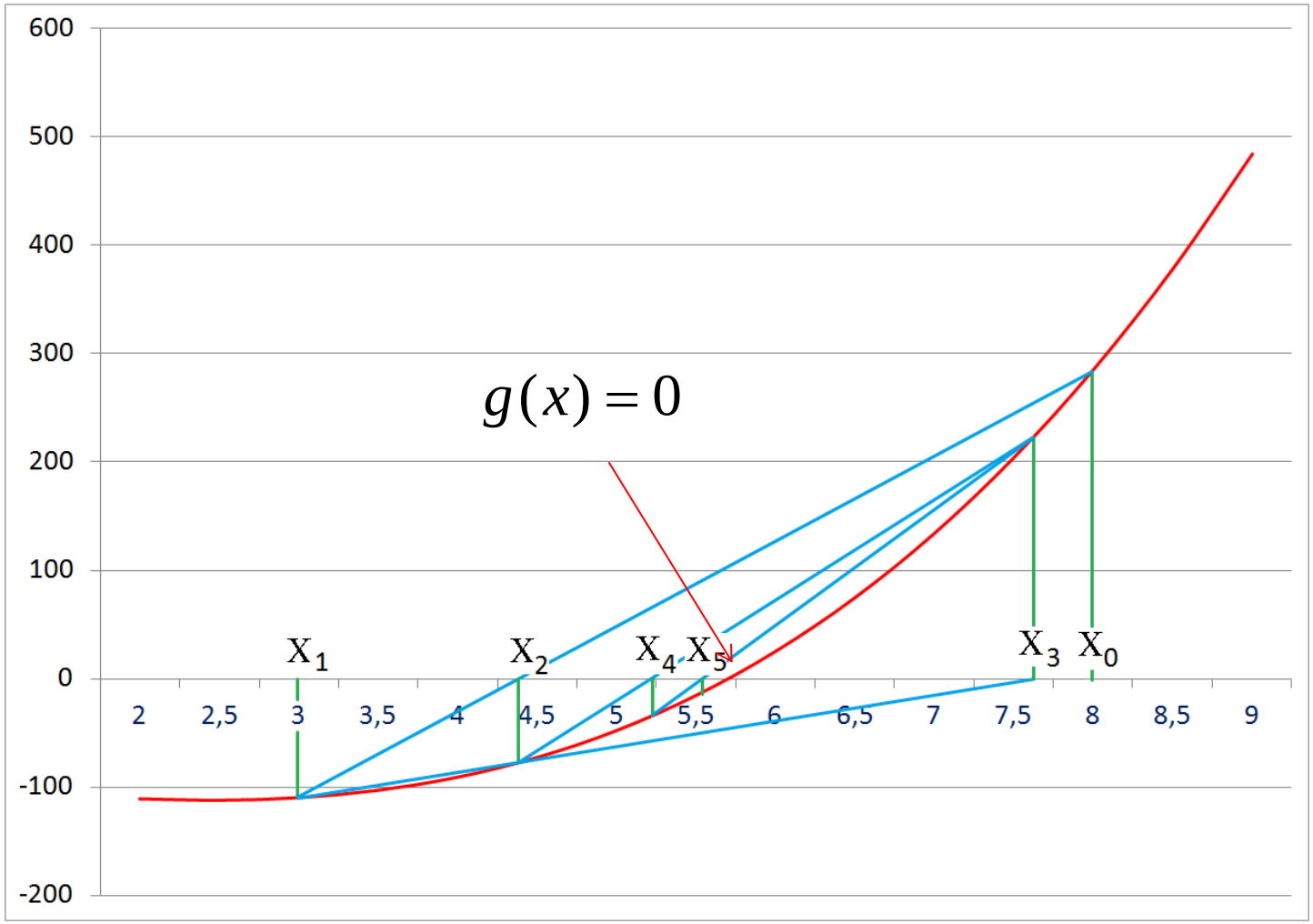
Otrzymuje się wtedy przedstawiony poniżej algorytm, który nazywa się metodą siecznych:

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f'(x^{(k)}) - f'(x^{(k-1)})} f'(x^{(k)})$$

Metoda siecznych wymaga dwóch punktów startowych $x^{(-1)}$ i $x^{(0)}$.
Tak samo, jak w przypadku metody Newtona Raphsona, metodę siecznych można wykorzystać do znajdowania pierwiastków równania $g(x) = 0$. Otrzymujemy wtedy algorytm:

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{g(x^{(k)}) - g(x^{(k-1)})} g(x^{(k)})$$

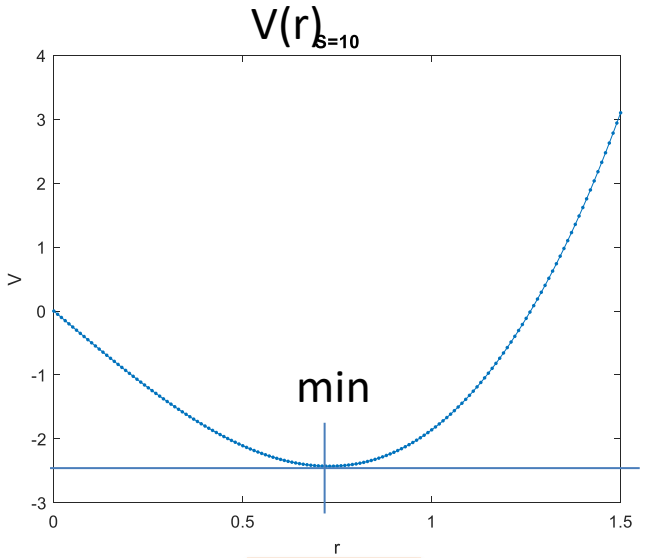
Metoda siecznych. Przykład



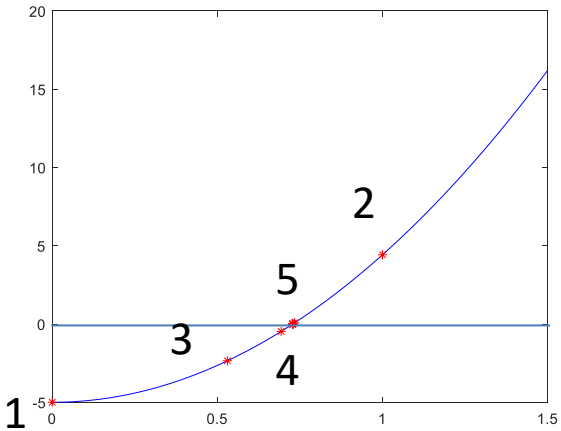
Przykład

$$V(r^*) = -\left(\frac{r^*}{2} S_0 - \pi r^{*3} \right)$$

$$g(r) = \frac{dV}{dr} = 3\pi r^2 - \frac{S_0}{2}$$

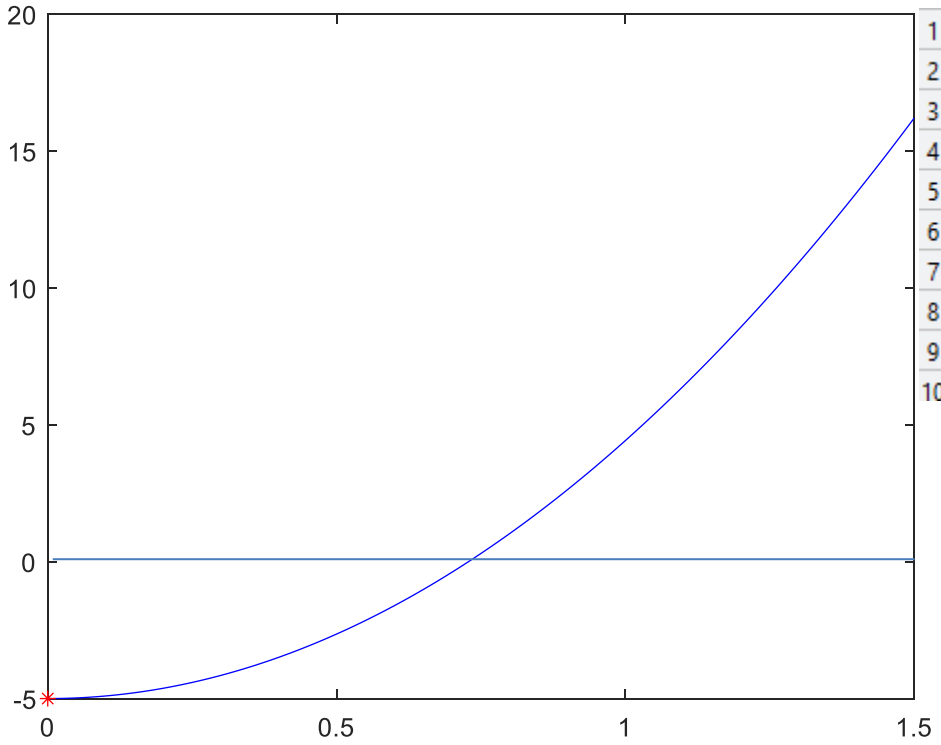


0.7283

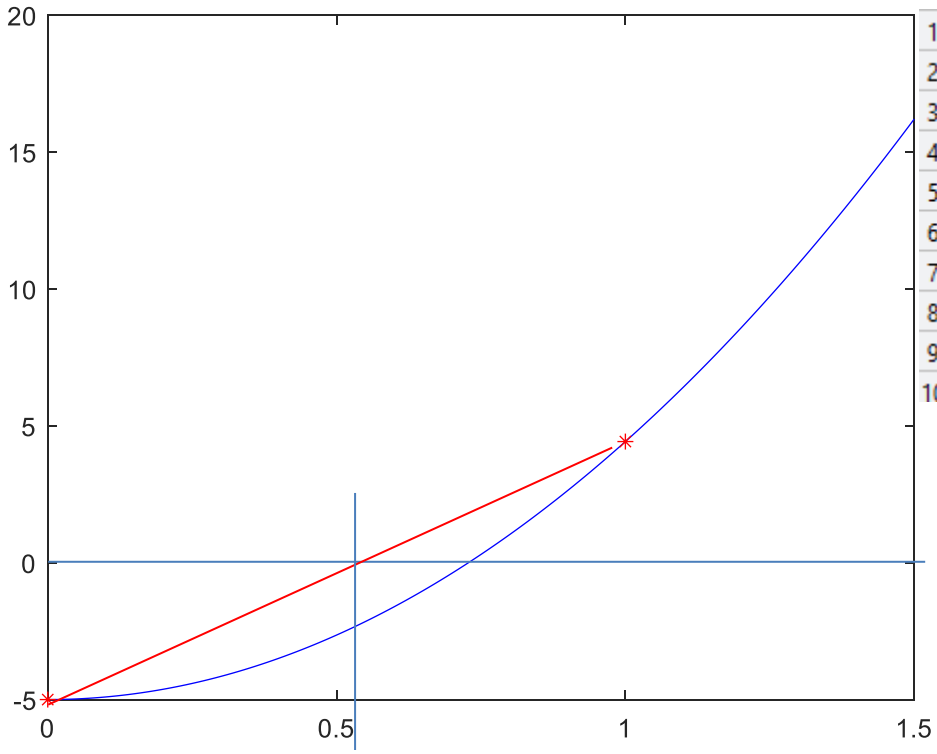


$$g(r) = \frac{dV}{dr} = 3\pi r^2 - \frac{S_0}{2}$$

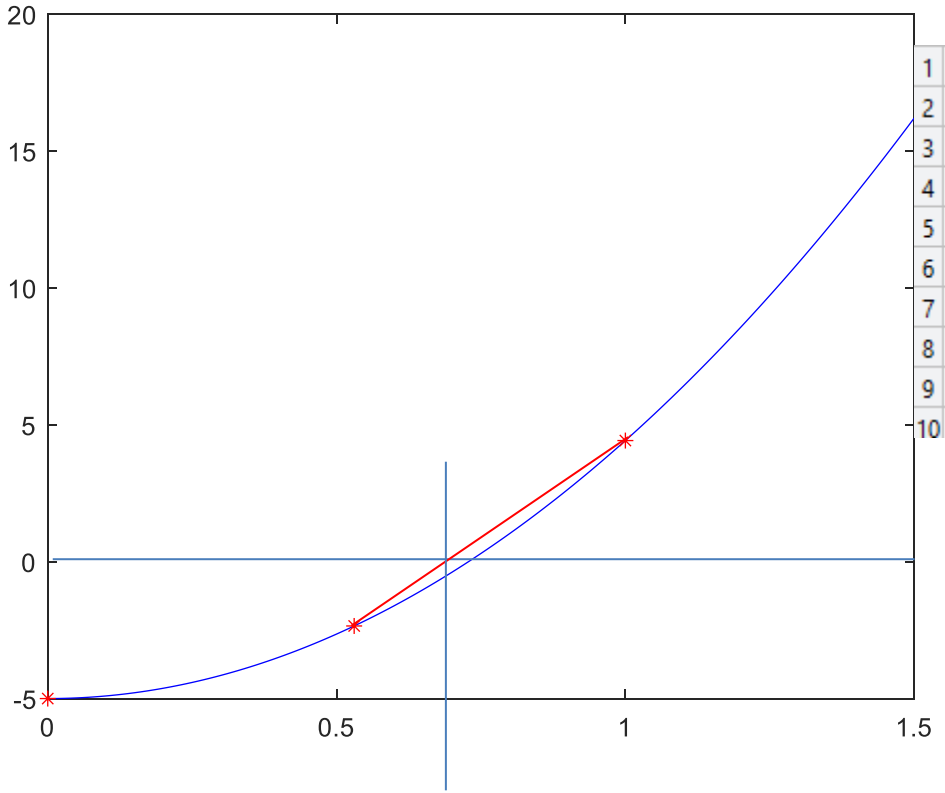
$$g(r^*) = 0$$



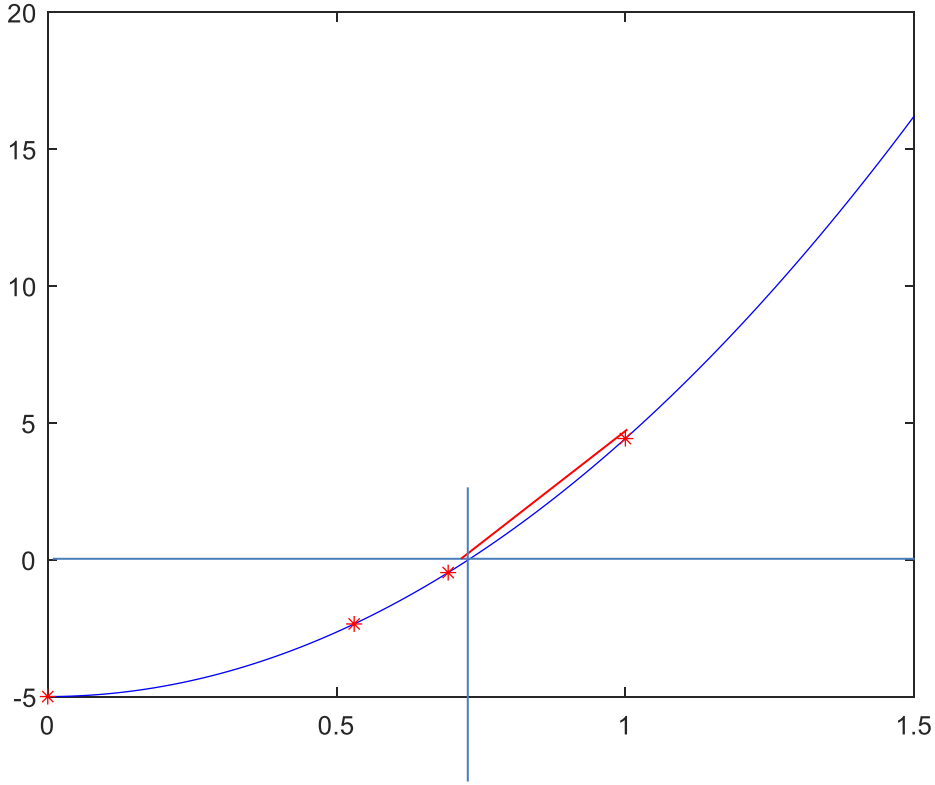
1	0	0	1
2	1	1	0.5305
3	2	0.5305	0.6932
4	3	0.6932	0.7340
5	4	0.7340	0.7282
6	5	0.7282	0.7283
7	6	0.7283	0.7283
8	7	0.7283	0.7283
9	8	0.7283	0.7283
10	9	0.7283	0.7283



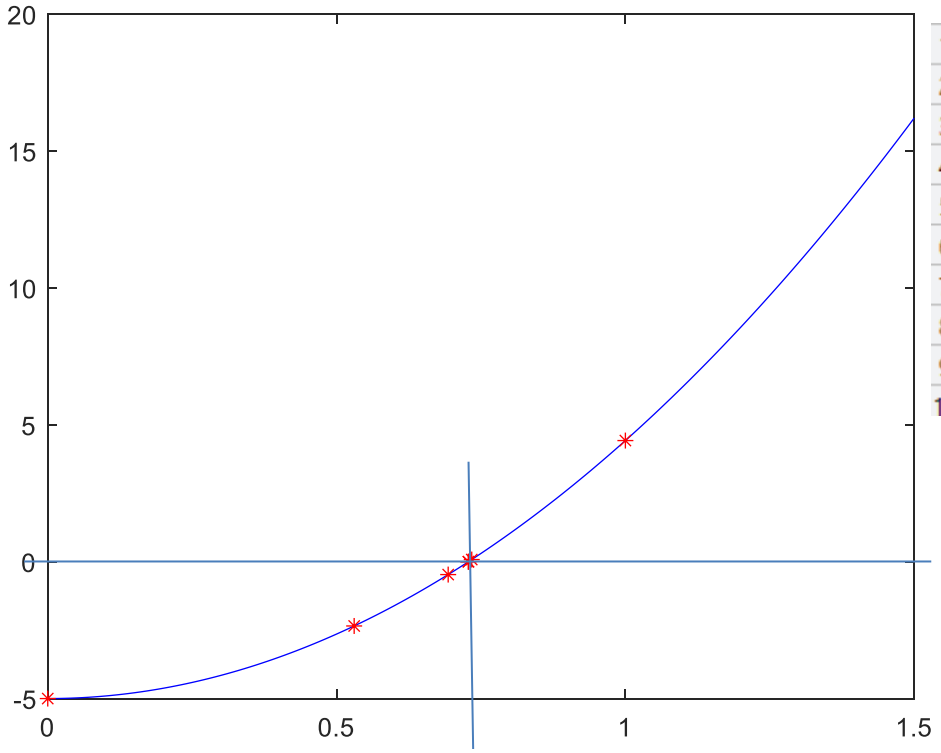
1	0	0	1
2	1	1	0.5305
3	2	0.5305	0.6932
4	3	0.6932	0.7340
5	4	0.7340	0.7282
6	5	0.7282	0.7283
7	6	0.7283	0.7283
8	7	0.7283	0.7283
9	8	0.7283	0.7283
10	9	0.7283	0.7283



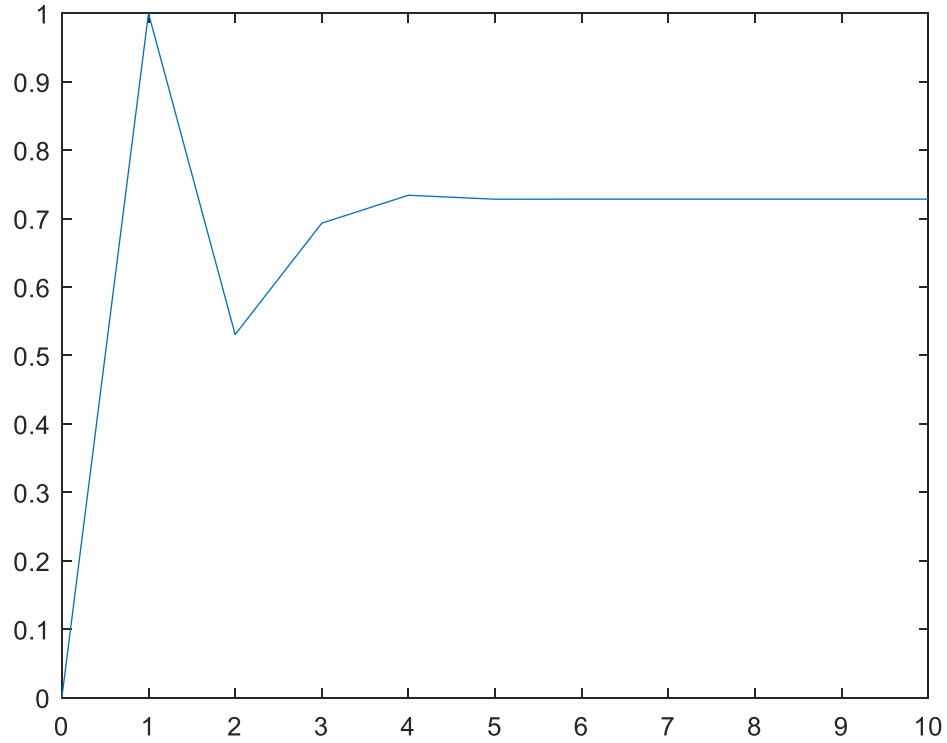
1	0	0	1
2	1	1	0.5305
3	2	0.5305	0.6932
4	3	0.6932	0.7340
5	4	0.7340	0.7282
6	5	0.7282	0.7283
7	6	0.7283	0.7283
8	7	0.7283	0.7283
9	8	0.7283	0.7283
10	9	0.7283	0.7283



1	0	0	1
2	1	1	0.5305
3	2	0.5305	0.6932
4	3	0.6932	0.7340
5	4	0.7340	0.7282
6	5	0.7282	0.7283
7	6	0.7283	0.7283
8	7	0.7283	0.7283
9	8	0.7283	0.7283
10	9	0.7283	0.7283



1	0	0	1
2	1	1	0.5305
3	2	0.5305	0.6932
4	3	0.6932	0.7340
5	4	0.7340	0.7282
6	5	0.7282	0.7283
7	6	0.7283	0.7283
8	7	0.7283	0.7283
9	8	0.7283	0.7283
10	9	0.7283	0.7283



siecznyOS.m

Porównanie metod znajdowania minimum z zadana dokładnością

Metoda liczb Fibonacciego jest najbardziej efektywna metoda eliminacji obszarów **jeśli początkowy przedział, w którym leży minimum jest znany**.

Jeśli nie znamy początkowego przedziału oraz pochodnych funkcji, wówczas najlepsza powinna być **metoda interpolacji kwadratowej Powella**.

Gdy **pierwsze pochodne są dostępne**, **metoda siecznych** powinna być najbardziej efektywna.

W końcu **metoda Newtona-Raphsona** jest najbardziej efektywna, gdy dostępne są informacje i pierwszych i drugich pochodnych funkcji.

Metody optymalizacji funkcji wielu zmiennych

Dana jest funkcja wielu zmiennych: $f : \mathbb{R}^N \rightarrow \mathbb{R}$. Mówimy, że punkt $\bar{x}$ jest punktem stacjonarnym, jeśli gradient w tym punkcie jest zerowym wektorem: $\nabla f(\bar{x}) = 0$. Punkt ten jest lokalnym minimum, jeśli Hesjan w tym punkcie $\nabla^2 f(\bar{x})$ jest dodatnio określony. Macierz jest dodatnio określona jeśli wszystkie jej wartości własne są dodatnie: $\lambda_i > 0,\ i = 1, 2, ..., N$[2].

Punkt stacjonarny

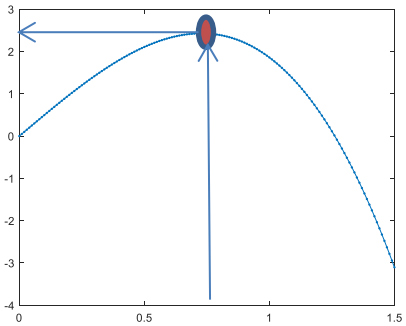
$$\nabla f(\tilde{x}) = 0$$

Warunki ekstremum funkcji jednej zmiennej

$$f\left(x^*+\Delta x\right)= f\left(x^*\right)+\Delta x f'\left(x^*\right)+\frac{1}{2}\left(\Delta x\right)^2 f''\left(x^*\right)+\ldots$$

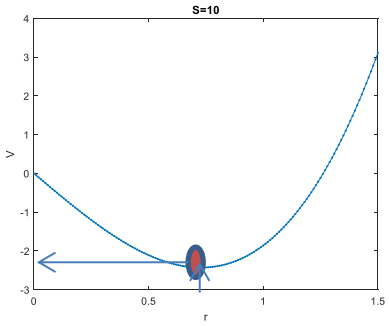
$$f\left(x^*-\Delta x\right)= f\left(x^*\right)-\Delta x f'\left(x^*\right)+\frac{1}{2}\left(-\Delta x\right)^2 f''\left(x^*\right)+\ldots$$

f(x)*



*x**

$$f\left(x^*\right)> f\left(x+\Delta x\right),\quad f\left(x^*\right)> f\left(x-\Delta x\right) \Rightarrow f'\left(x^*\right)=0,\quad f''\left(x^*\right)<0$$



f(x)*

*x**

$$f\left(x^*\right)< f\left(x+\Delta x\right),\quad f\left(x^*\right)< f\left(x-\Delta x\right) \Rightarrow f'\left(x^*\right)=0,\quad f''\left(x^*\right)>0$$

Szereg Taylora funkcji dwóch zmiennych

$$f(x,y) = f(x_0,y_0) + \frac{1}{1!}df(x_0,y_0)(x-x_0,y-y_0) + \\ + \frac{1}{2!}d^2f(x_0,y_0)(x-x_0,y-y_0) + \ldots + \\ + \frac{1}{(n-1)!}d^{n-1}f(x_0,y_0)(x-x_0,y-y_0) + \\ + R_n(x,y)$$

gdzie

$$d^n f(x_0,y_0)(x-x_0,y-y_0) = \sum_{i=0}^{n} \binom{n}{i} \frac{\partial^n f(x_0,y_0)}{\partial_x^{n-i}\partial_y^i} \cdot (x-x_0)^{n-i}(y-y_0)^i$$

Kombinacja bez powtórzeń $\binom{n}{k} = C_n^k = \frac{n!}{k!\,(n-k)!}$

Gradient funkcji skalarnej

$$f(x_1, x_2, \ldots, x_i, \ldots, x_n)$$

$$g(x^0) = \nabla f(x^0) = (\text{grad } f)(x^0) = \left(\frac{\partial f}{\partial x_1}(x^0), \ldots, \frac{\partial f}{\partial x_n}(x^0) \right)$$

Punkt krytyczny (stacjonarny) jest punktem, w którym funkcja jest różniczkowalna i jej pochodna jest równa 0.

$$(\text{grad } f)(x^0) = 0 \Rightarrow \frac{\partial f}{\partial x_i}(x^0) = 0$$

Macierz Hessego (Hesjan)

27-02-2024

$$H(x)=\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Jeżeli forma przyjmuje wartości tego samego znaku dla wszystkich punktów przestrzeni liniowej, to nazywa się ją **określoną**.

$$x^T Hx > 0,$$ **minimum**

Forma kwadratowa o współczynnikach rzeczywistych lub zespolonych zapisana w postaci symetrycznej jest dodatnio określona, jeżeli **wszystkie minory główne** jej macierzy, obliczane po przekątnej od lewego rogu, są **dodatnie**.

Wszystkie wartości własne dodatnio określonej formy są dodatnie.

Przykład 1. Macierz rzeczywista, symetryczna, **dodatnio określona**

$$\mathbf{P} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Dowód: Dla dowolnej macierzy kolumnowej $X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ jest

$$\begin{aligned} \mathbf{X}\,\mathbf{P}\,\mathbf{X}^T &= \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ &= \begin{bmatrix} 2x - y & -x + 2y - z & -y + 2z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ &= 2x^2 - xy - xy + 2y^2 - yz - yz + 2z^2. \end{aligned}$$

Oznacza to, że odpowiadająca tej macierzy forma kwadratowa P ma wzór

$$P(\mathbf{x}) = 2x^2 - 2xy + 2y^2 - 2yz + 2z^2 = x^2 + (x-y)^2 + (y-z)^2 + z^2,$$

Przykład 2. Macierz rzeczywista, symetryczna, **określona niedodatnio**

$$\mathbf{N} = \begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 0 \\ 1 & 0 & -2 \end{bmatrix}.$$

Dowód: Wykonując obliczenia jak w Przykładzie 1 łatwo przekonać się, że odpowiadająca tej macierzy forma kwadratowa N ma postać

$$N(\mathbf{x}) = -x^2 - 2xy + 2xz - 2y^2 - 2z^2 = -(x + y - z)^2 - (y + z)^2,$$

Przykład 3. Macierz rzeczywista, symetryczna, **nieokreślona**

$$\mathbf{Q} = \begin{bmatrix} 1 & 3 & -2 \\ 3 & -2 & 0 \\ -2 & 0 & 1 \end{bmatrix}.$$

Dowód: Można sprawdzić bezpośrednim rachunkiem, że macierzy $\mathbf{Q}$ odpowiada forma kwadratowa

$$Q(\mathbf{x}) = x^2 - 2y^2 + z^2 + 6xy - 4xz,$$

gdzie $\mathbf{x} = (x, y, z)$,

Forma ta jest nieokreślona, gdyż

- przyjmuje wartości zarówno dodatnie, jak i ujemne, np.:
    1. jeśli $\mathbf{x} = (0, 1, 1)$, to $Q(\mathbf{x}) = -1$,
    2. jeśli $\mathbf{x} = (2, 1, 0)$, to $Q(\mathbf{x}) = 14$,
- jest niezdegenerowana, tj. dla wszystkich $\mathbf{x} \neq \mathbf{0}$ jest $Q(\mathbf{x}) \neq 0$.

Ekstrema funkcji wielu zmiennych

$$f(x_1, x_2, \mathrm{K}, x_i, \mathrm{K}, x_n)$$

$$\min_x f(x_1, x_2, \mathrm{K}, x_n)$$

Pochodne cząstkowe

$$\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \mathrm{K}, \frac{\partial f}{\partial x_n}.$$

Przykład

$$f(x_1, x_2) = x_1^2 + x_1 x_2^3 + 3x_1 - 2x_2$$

$$\frac{\partial f}{\partial x_1} = 2x_1 + x_2^3 + 3.$$

$$\frac{\partial f}{\partial x_2} = 3x_1 x_2^2 - 2.$$

```
>>syms  x1  x2
f=x1^2+x1*x2^3+3*x1-2*x2;
fder=[diff(f,x1), diff(f,x2)]
fder =
 [ x2^3 + 2*x1 + 3, 3*x1*x2^2 - 2]
```

Pochodne cząstkowe wyższych rzędów

$$f(x_1, x_2, \mathrm{K}, x_i, \mathrm{K}, x_n)$$

$$\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \mathrm{K}, \frac{\partial f}{\partial x_n}.$$

$$\frac{\partial}{\partial x_1}\left(\frac{\partial f}{\partial x_1}\right), \frac{\partial}{\partial x_2}\left(\frac{\partial f}{\partial x_1}\right), \mathrm{K}, \frac{\partial}{\partial x_n}\left(\frac{\partial f}{\partial x_1}\right)$$

$$\frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}\right) = \frac{\partial^2 f}{\partial x_i \partial x_j} = f''_{x_i x_j}$$

$$\frac{\partial}{\partial x_1}\left(\frac{\partial f}{\partial x_2}\right), \frac{\partial}{\partial x_2}\left(\frac{\partial f}{\partial x_2}\right), \mathrm{K}, \frac{\partial}{\partial x_n}\left(\frac{\partial f}{\partial x_2}\right)$$

$$\frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k} = \frac{\partial}{\partial x_i}\left(\frac{\partial^2 f}{\partial x_j \partial x_k}\right).$$

Przykład

$$f(x_1, x_2) = x_1^2 + x_1 x_2^3 + 3x_1 - 2x_2$$

```
>>syms  x1  x2
f=x1^2+x1*x2^3+3*x1-2*x2;
derf1=[diff(f,x1) , diff(f,x2)]

derf1 =
 [ x2^3 + 2*x1 + 3, 3*x1*x2^2 - 2]

derf2=[diff(derf1,x1); diff(derf1,x2)]

derf2 =
[     2,     3*x2^2]
[ 3*x2^2, 6*x1*x2]
```

Hesjan

$$\frac{\partial^2 f}{\partial x^2} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 & 3x_2^2 \\ 3x_2^2 & 6x_1 x_2 \end{bmatrix}.$$

```
>> s=solve(derf1)
s =
  struct with fields:
    x1: [5×1 sym]
    x2: [5×1 sym]
>> s.x1
 ans =
  - root(z^5 + 3*z^2 + 4/3, z, 1)^3/2 - 3/2
 - root(z^5 + 3*z^2 + 4/3, z, 2)^3/2 - 3/2
 - root(z^5 + 3*z^2 + 4/3, z, 3)^3/2 - 3/2
 - root(z^5 + 3*z^2 + 4/3, z, 4)^3/2 - 3/2
 - root(z^5 + 3*z^2 + 4/3, z, 5)^3/2 - 3/2
 >> p=[1 0 0 3 0 4/3]
r=roots(p)
p =
  Columns 1 through 5
                1                   0                   0                   3
0
  Column 6
        1.33333333333333
r =
        -1.52845444486759 +                  0i
        0.795165920275068 +     1.16862086209307i
        0.795165920275068 -     1.16862086209307i
       -0.0309386978412746 +    0.660043801190204i
       -0.0309386978412746 -    0.660043801190204i
```

Przykład 2

$$f(x_1, x_2, x_3) = x_1^3 x_2^2 x_3^4.$$

```
>>syms  x1  x2   x3
f=x1^3*x2^2*x3^4;
 derf1=[diff(f,x1) , diff(f,x2), diff(f,x3)]
 derf1 =
[ 3*x1^2*x2^2*x3^4, 2*x1^3*x2*x3^4,
4*x1^3*x2^2*x3^3]
 >>derf2=[diff(derf1,x1); diff(derf1,x2) ;
diff(derf1,x3) ]
 derf2 =
 [    6*x1*x2^2*x3^4, 6*x1^2*x2*x3^4,
12*x1^2*x2^2*x3^3]
[    6*x1^2*x2*x3^4,    2*x1^3*x3^4,
8*x1^3*x2*x3^3]
[ 12*x1^2*x2^2*x3^3, 8*x1^3*x2*x3^3,
12*x1^3*x2^2*x3^2]
```

Hesjan

$$\left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_{i,j=1,2,3} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix} =$$

$$= \begin{bmatrix} 6x_1 x_2^2 x_3^4 & 6x_1^2 x_2 x_3^4 & 12x_1^2 x_2^2 x_3^3 \\ 6x_1^2 x_2 x_3^4 & 2x_1^3 x_3^4 & 8x_1^3 x_2 x_3^3 \\ 12x_1^2 x_2^2 x_3^3 & 8x_1^3 x_2 x_3^3 & 12x_1^3 x_2^2 x_3^2 \end{bmatrix}.$$

Przykład 2

$$f(x_1, x_2, x_3) = x_1^3 x_2^2 x_3^4.$$

obliczyć $\dfrac{\partial^3 f}{\partial x_1^2 \partial x_2}$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = 6 x_1^2 x_2 x_3^4$$

$$\frac{\partial^3 f}{\partial x_1^2 \partial x_2} = \frac{\partial}{\partial x_1}\left(\frac{\partial^2 f}{\partial x_1 \partial x_2} \right) = 12 x_1 x_2 x_3^4$$

Przykład 3

Znaleźć punkt krytyczny (stacjonarny) funkcji

$$f(x_1, x_2) = x_1^2 + 3x_1x_2 + 2x_2^2 - 4x_1 + 3x_2,$$

$$\frac{\partial f}{\partial x_1} = 2x_1 + 3x_2 - 4, \quad \frac{\partial f}{\partial x_2} = 3x_1 + 4x_2 + 3.$$

$$(\text{grad } f)(x) = (2x_1 + 3x_2 - 4,\ 3x_1 + 4x_2 + 3).$$

$$\frac{\partial f}{\partial x_1} = 2x_1 + 3x_2 - 4 = 0;$$

$$\frac{\partial f}{\partial x_2} = 3x_1 + 4x_2 + 3 = 0.$$

$$\begin{cases} x_1 = -25; \\ x_2 = 18. \end{cases}$$

```
>>
syms  x1  x2
f=x1^2+3*x1*x2+2*x2^2-4*x1+3*x2;
grad=[diff(f,x1) ; diff(f,x2)]
grad =
  2*x1 + 3*x2 - 4
 3*x1 + 4*x2 + 3
>>
[x1,x2]= solve('2*x1 + 3*x2 - 4=0 ',
'3*x1 + 4*x2 + 3=0')
 x1 =
-25
x2 =
 18
```

Przykład 4

$$f(x_1, x_2) = x_1 e^{-x_1^2 - x_2^2}.$$

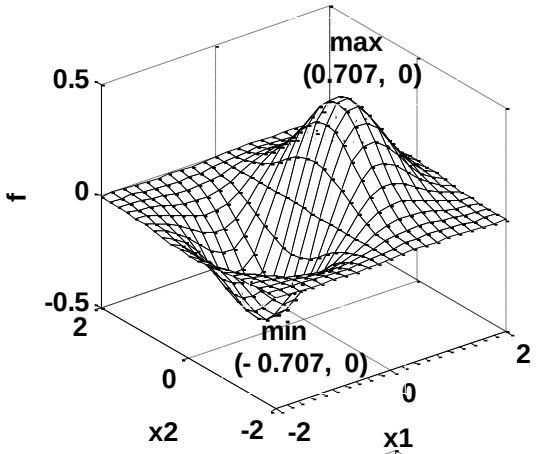
```
syms  x1  x2
f = x1*exp(-x1^2 - x2^2);
grad=[diff(f,x1) ; diff(f,x2)]
grad =
  1/exp(x1^2 + x2^2) - (2*x1^2)/exp(x1^2 +
x2^2)
                -(2*x1*x2)/exp(x1^2 + x2^2)
>>
[x1,x2]=solve('1/exp(x1^2 + x2^2) -
(2*x1^2)/exp(x1^2 + x2^2)=0', '-
(2*x1*x2)/exp(x1^2 + x2^2)=0')
x1 =
   2^(1/2)/2
 -2^(1/2)/2
  x2 =
  0
  0
```

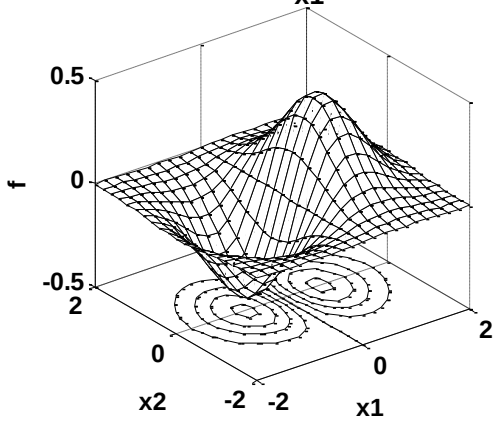
```
figure(1)
[x1,x2] = meshgrid(-2:0.2:2);
f = x1.*exp(-x1.^2 - x2.^2);
mesh(x1,x2,f)
xlabel('x1'), ylabel('x2'), zlabel('f')
figure(2)
[dx1,dx2] = gradient(f, 0.2, 0.2);
contour(x1,x2,f)
hold on
quiver(x1,x2,dx1,dx2)

colormap hsv
xlabel('x1'), ylabel('x2')
```

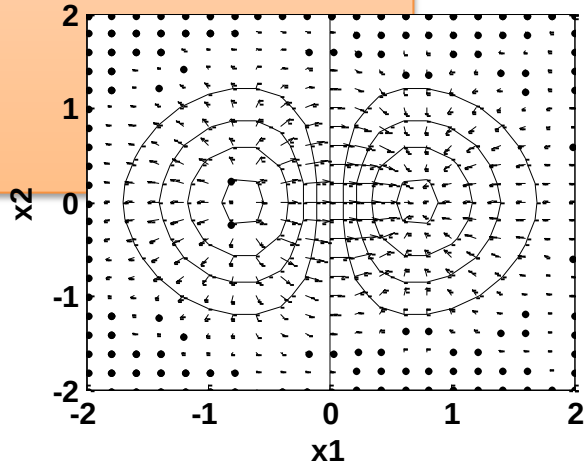
Przykład 4

$$f(x_1, x_2) = x_1 e^{-x_1^2 - x_2^2}.$$

```
figure(1)
[x1,x2] = meshgrid(-2:0.2:2);
f = x1.*exp(-x1.^2 - x2.^2);
mesh(x1,x2,f)
xlabel('x1'), ylabel('x2'), zlabel('f')
>>
figure(2)
[dx1,dx2] = gradient(f, 0.2, 0.2);
contour(x1,x2,f)
hold on
quiver(x1,x2,dx1,dx2)
colormap hsv
xlabel('x1'), ylabel('x2')
```







$$\nabla f = \left(\frac{1}{e^{x_1^2+x_2^2}} - \frac{2x_1^2}{e^{x_1^2+x_2^2}}, -\frac{2x_1 x_2}{e^{x_1^2+x_2^2}} \right)$$

$$(x_1^0, x_2^0) = (\sqrt{2}/2,\ 0) = (0.707,\ 0)$$

$$(x_1^0, x_2^0) = (-\sqrt{2}/2,\ 0) = (-0.707,\ 0)$$

Przykład 4

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1^2 + x_2^2 > 0.$$

$$\det A = \begin{vmatrix} 1 & 1 \\ -1 & 1 \end{vmatrix} = 2 \neq 0$$

Przykład 4

$$f(x_1, x_2) = x_1 e^{-x_1^2 - x_2^2}.$$

$$\nabla f = \left(\frac{1}{e^{x_1^2 + x_2^2}} - \frac{2x_1^2}{e^{x_1^2 + x_2^2}}, \; -\frac{2x_1 x_2}{e^{x_1^2 + x_2^2}} \right)$$

$$(x_1^0, x_2^0) = (\sqrt{2}/2, \; 0) = (0.707, \; 0)$$

$$(x_1^0, x_2^0) = (-\sqrt{2}/2, \; 0) = (-0.707, \; 0)$$

```
>> syms  x1  x2
f = x1*exp(-x1^2 - x2^2);
grad=[diff(f,x1) , diff(f,x2)];
hess=[diff(grad,x1); diff(grad,x2)]
hess =
 [(4*x1^3)/exp(x1^2 + x2^2) - (6*x1)/exp(x1^2
+ x2^2), (4*x1^2*x2)/exp(x1^2 + x2^2) -
(2*x2)/exp(x1^2 + x2^2)]
[ (4*x1^2*x2)/exp(x1^2 + x2^2) -
(2*x2)/exp(x1^2 + x2^2),
(4*x1*x2^2)/exp(x1^2 + x2^2) -
(2*x1)/exp(x1^2 + x2^2)]
```

Przykład 4

$$f(x_1, x_2) = x_1 e^{-x_1^2 - x_2^2}.$$

$$\nabla f = \left(\frac{1}{e^{x_1^2+x_2^2}} - \frac{2x_1^2}{e^{x_1^2+x_2^2}}, -\frac{2x_1 x_2}{e^{x_1^2+x_2^2}} \right)$$

$$(x_1^0, x_2^0) = (\sqrt{2}/2,\ 0) = (0.707,\ 0)$$

$$(x_1^0, x_2^0) = (-\sqrt{2}/2,\ 0) = (-0.707,\ 0)$$

```
>>
x1 =-2^(1/2)/2; x2 =0;
hessmin = [[(4*x1^3)/exp(x1^2 + x2^2) -
(6*x1)/exp(x1^2 + x2^2),
(4*x1^2*x2)/exp(x1^2 + x2^2) -
(2*x2)/exp(x1^2 + x2^2)]
[ (4*x1^2*x2)/exp(x1^2 + x2^2) -
(2*x2)/exp(x1^2 + x2^2),
(4*x1*x2^2)/exp(x1^2 + x2^2) -
(2*x1)/exp(x1^2 + x2^2)]]
hessmin =
   1.7155         0
        0    0.8578
 >>hessmin=[x1  x2]*hessmin*[x1;x2] %
hessmin =
   0.8578
```

Przykład 4

$$f(x_1, x_2) = x_1 e^{-x_1^2 - x_2^2}.$$

$$\nabla f = \left(\frac{1}{e^{x_1^2+x_2^2}} - \frac{2x_1^2}{e^{x_1^2+x_2^2}}, \; -\frac{2x_1 x_2}{e^{x_1^2+x_2^2}} \right)$$

$$(x_1^0, x_2^0) = (\sqrt{2}/2, \; 0) = (0.707, \; 0)$$

$$(x_1^0, x_2^0) = (-\sqrt{2}/2, \; 0) = (-0.707, \; 0)$$

```
>>
x1 =2^(1/2)/2; x2 =0;
hessmax = [[(4*x1^3)/exp(x1^2 + x2^2) -
(6*x1)/exp(x1^2 + x2^2),
(4*x1^2*x2)/exp(x1^2 + x2^2) -
(2*x2)/exp(x1^2 + x2^2)]
[ (4*x1^2*x2)/exp(x1^2 + x2^2) -
(2*x2)/exp(x1^2 + x2^2),
(4*x1*x2^2)/exp(x1^2 + x2^2) -
(2*x1)/exp(x1^2 + x2^2)]]

hessmax =
  -1.7155         0
        0   -0.8578

hessmax= [x1  x2]*hessmax*[x1;x2]; %
hessmax =
  -0.4289
```

Przykład 5

$$f(x_1, x_2) = x_1^2 e^{x_1+x_2} + x_2^2 e^{x_1-x_2}$$

$x^0 = (0, 0)$ Punkt krytyczny (stacjonarny)

$$\frac{\partial f}{\partial x_1} = (2x_1 + x_1^2)e^{x_1+x_2} + x_2^2 e^{x_1-x_2};$$

$$\frac{\partial f}{\partial x_2} = x_1^2 e^{x_1+x_2} + (2x_2 - x_2^2)e^{x_1-x_2}.$$

$$x_1 = x_2 = 0$$

$$\frac{\partial f}{\partial x_1} = (2x_1 + x_1^2)e^{x_1+x_2} + x_2^2 e^{x_1-x_2} = 0$$

$$\frac{\partial f}{\partial x_2} = x_1^2 e^{x_1+x_2} + (2x_2 - x_2^2)e^{x_1-x_2} = 0$$

Punkt krytyczny (stacjonarny)

Przykład 5

$$H(x)=\begin{bmatrix}\dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2}\\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2}\end{bmatrix},$$

$$\frac{\partial f}{\partial x_1}=(2x_1+x_1^2)e^{x_1+x_2}+x_2^2e^{x_1-x_2};$$

$$\frac{\partial f}{\partial x_2}=x_1^2e^{x_1+x_2}+(2x_2-x_2^2)e^{x_1-x_2}.$$

$$\frac{\partial^2 f}{\partial x_1^2}=(2+4x_1+x_1^2)e^{x_1+x_2}+x_2^2e^{x_1-x_2};$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2}=(2x_1+x_1^2)e^{x_1+x_2}+(2x_2-x_2^2)e^{x_1-x_2};$$

$$\frac{\partial^2 f}{\partial x_2 \partial x_1}=(2x_1+x_1^2)e^{x_1+x_2}+(2x_2-x_2^2)e^{x_1-x_2};$$

$$\frac{\partial^2 f}{\partial x_2^2}=\frac{\partial}{\partial x_2}\left(\frac{\partial f}{\partial x_2}\right)=x_1^2e^{x_1+x_2}+(2-4x_2+x_2^2)e^{x_1-x_2}.$$

$$H(0,0)=\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}.$$

$2>0$

$$\det H(0,0)=\det\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}=4>0$$

Minimum

Przykład 5

```
>>syms  x1  x2
f=x1^2*exp(x1+x2)+ x2^2*exp(x1-x2);
grad=[diff(f,x1) , diff(f,x2)] grad =
 [ x2^2*exp(x1 - x2) + 2*x1*exp(x1 + x2) +
x1^2*exp(x1 + x2), 2*x2*exp(x1 - x2) -
x2^2*exp(x1 - x2) + x1^2*exp(x1 + x2)]
```

```
>>hess=[diff(grad,x1); diff(grad,x2)]
hess =

[  2*exp(x1 + x2) + x2^2*exp(x1 - x2) +
4*x1*exp(x1 + x2) + x1^2*exp(x1 + x2),
2*x1*exp(x1 + x2) - x2^2*exp(x1 - x2) +
2*x2*exp(x1 - x2) + x1^2*exp(x1 + x2)]

[ 2*x1*exp(x1 + x2) - x2^2*exp(x1 - x2) +
2*x2*exp(x1 - x2) + x1^2*exp(x1 + x2),
2*exp(x1 - x2) + x2^2*exp(x1 - x2) -
4*x2*exp(x1 - x2) + x1^2*exp(x1 + x2)]
```

```
>> x1=0; x2=0;
hesse00 =[[2*exp(x1 + x2) + x2^2*exp(x1 - x2)
+ 4*x1*exp(x1 + x2) + x1^2*exp(x1 + x2),
2*x1*exp(x1 + x2) - x2^2*exp(x1 - x2) +
2*x2*exp(x1 - x2) + x1^2*exp(x1 + x2)]
[ 2*x1*exp(x1 + x2) - x2^2*exp(x1 - x2) +
2*x2*exp(x1 - x2) + x1^2*exp(x1 + x2),
2*exp(x1 - x2) + x2^2*exp(x1 - x2) -
4*x2*exp(x1 - x2) + x1^2*exp(x1 + x2)]]

hesse00 =
     2     0
     0     2
```

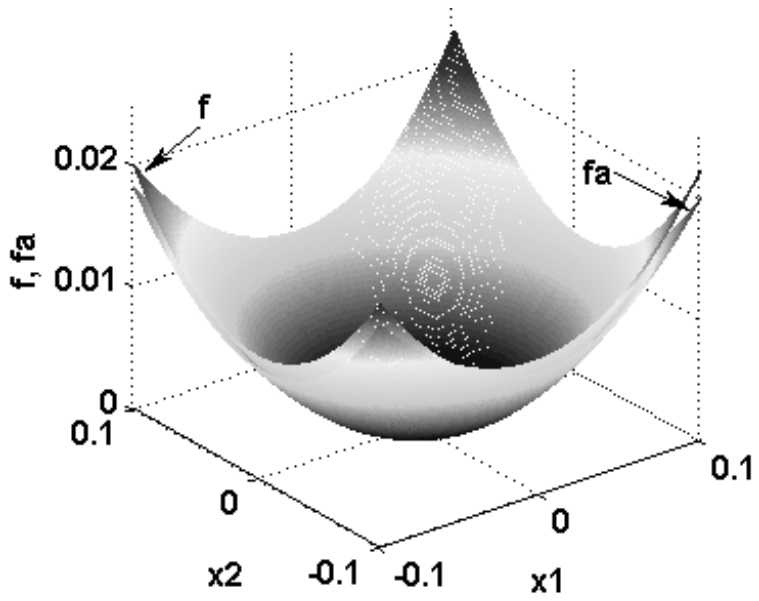
Przykład 5

```
>>syms  x1  x2
fa=1/2*[x1   x2]*[2   0; 0   2]*[x1  x2]';
pretty(fa)

  ___      ___
  x1 x1 + x2 x2
```

aproksymacja

$$\frac{1}{2}[x_1 \;\; x_2]\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = x_1^2 + x_2^2,$$

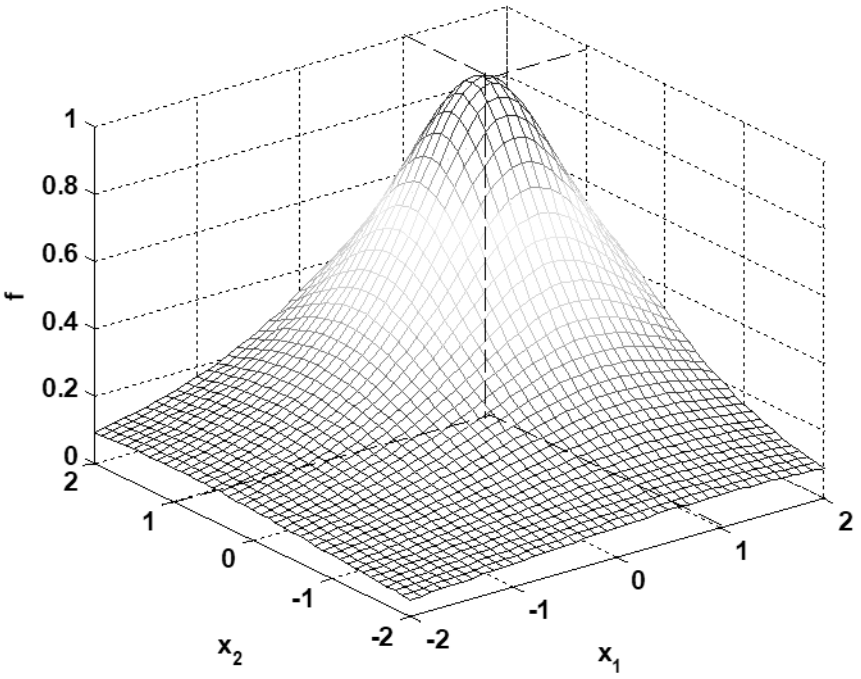
```
>>[x1,x2]=meshgrid(-0.1:0.001:0.1);
f=x1.^2.*exp(x1+x2)+ x2.^2.*exp(x1-x2);
fa=x1.^2+x2.^2; mesh(x1,x2, f)
hold on
mesh(x1,x2, fa)
 xlabel('x1'), ylabel('x2'), zlabel('f, fa ')
```



Przykład 6

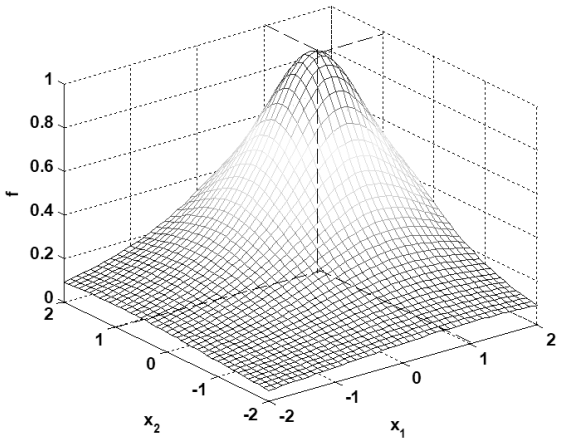
$$f(x_1, x_2) = \frac{1}{(x_1 - 1)^2 + (x_2 - 1)^2 + 1}.$$

```
>>[x1,x2]=meshgrid(-2:0.1:2);
f=1./((x1-1).^2+(x2-1).^2+1);
mesh(x1,x2, f)
 xlabel('x_1'), ylabel('x_2'), zlabel('f')
```



Przykład 6

```
>>syms x1  x2
f=1/((x1-1)^2+(x2-1)^2+1);
grad=[diff(f,x1) , diff(f,x2)]
[x1,x2]=solve(grad,x1,x2)
grad =
 [ -(2*x1 - 2)/((x1 - 1)^2 + (x2 - 1)^2 + 1)^2, -
(2*x2 - 2)/((x1 - 1)^2 + (x2 - 1)^2 + 1)^2]

 x1 =
 1
 x2 =
 1
```



Przykład 6

```
>>syms  x1  x2
hess=[diff(grad,x1) ; diff(grad,x2)]
hess =
 [ (2*(2*x1 - 2)^2)/((x1 - 1)^2 + (x2 - 1)^2 +
1)^3 - 2/((x1 - 1)^2 + (x2 - 1)^2 + 1)^2,
(2*(2*x1 - 2)*(2*x2 - 2))/((x1 - 1)^2 + (x2 - 1)^2
+ 1)^3]
[ (2*(2*x1 - 2)*(2*x2 - 2))/((x1 - 1)^2 + (x2 -
1)^2 + 1)^3, (2*(2*x2 - 2)^2)/((x1 - 1)^2 + (x2 -
1)^2 + 1)^3 - 2/((x1 - 1)^2 + (x2 - 1)^2 + 1)^2]
```

```
>> x1=1; x2=1;
hessextr =[[ (2*(2*x1 - 2)^2)/((x1 - 1)^2 + (x2 -
1)^2 + 1)^3 - 2/((x1 - 1)^2 + (x2 - 1)^2 + 1)^2,
(2*(2*x1 - 2)*(2*x2 - 2))/((x1 - 1)^2 + (x2 - 1)^2
+ 1)^3] ;
[ (2*(2*x1 - 2)*(2*x2 - 2))/((x1 - 1)^2 + (x2 -
1)^2 + 1)^3, (2*(2*x2 - 2)^2)/((x1 - 1)^2 + (x2 -
1)^2 + 1)^3 - 2/((x1 - 1)^2 + (x2 - 1)^2 + 1)^2]]

hessextr =
    -2     0
     0    -2
>>hessmax=[x1  x2]*hessextr*[x1;x2]
hessmax =
    -4
```

x = (1; 1) - maksimum

```
>> x1=1; x2=1;
fmax=1/((x1-1)^2+(x2-1)^2+1)
fmax =
     1
```

Metody bezpośrednich poszukiwań

- Metoda hipersześcienna (ang. *evolutionary optimization*)
- Metoda sympleksowa Neldera-Meada
- Metoda kierunków sprzężonych (ang. *conjugate direction*) Powella

Metoda hiperszescienna

Algorytm potrzebuje w pojedynczej iteracji 2^N+1 punktów, z czego 2^N punktów to są wierzchołki hipersześcianu scentrowanego na pozostałym punkcie. Porównuje się wartości funkcji we wszystkich tych punktach i wskazuje się najlepszy (z najmniejszą wartością). W następnej iteracji tworzy się sześcian wokół tego najlepszego punktu. Jeśli najlepszym punktem okaże się punkt, który był środkiem danego hipersześcianu, wówczas zmniejsza się rozmiar sześcianu. Proces ten kontynuuje się aż hipersześcian stanie się dostatecznie mały.

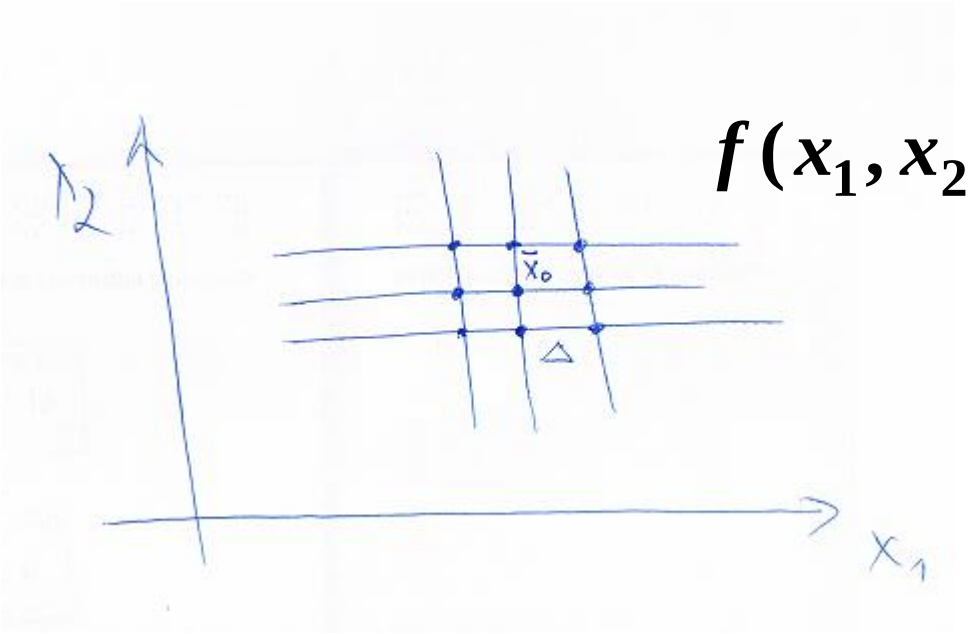
Algorytm

1) Wybierz punkt początkowy $x^{(0)}$ oraz parametry redukcji Δ_i dla każdej ze zmiennych, $i = 1, 2, ..., N$. Wybierz parametr zakończenia ϵ. Ustal $\bar{x} = x^{(0)}$

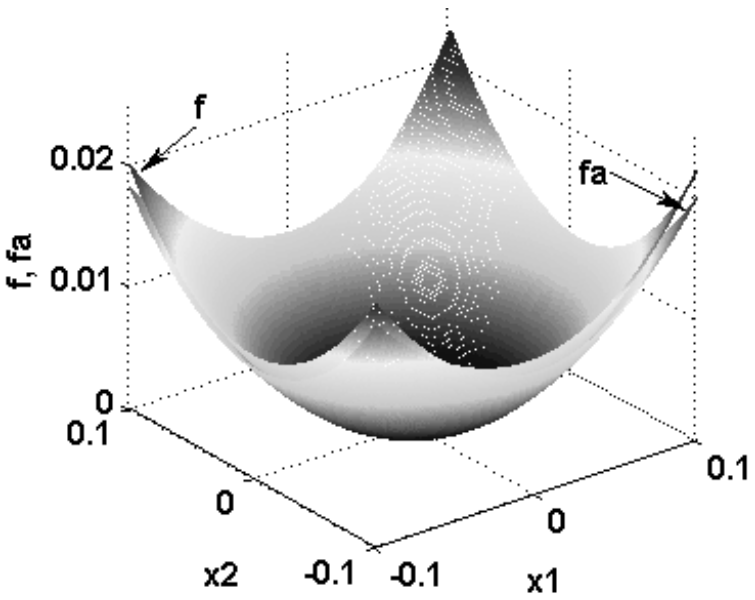
2) Jeśli $||\Delta|| < \epsilon$, **Zakończ**;
W przeciwnym przypadku stwórz 2^N punktów poprzez dodanie i odjęcie $\Delta_i/2$ do/od każdej zmiennej w punkcie $\bar{x}$.

3) Oblicz wartość funkcji dla wszystkich $(2^N + 1)$ punktów. Znajdź punkt z najmniejszą wartością. Ustal ten punkt jako $\bar{x}$.

4) Jeśli $\bar{x} = x^{(0)}$, zredukuj parametry redukcji $\Delta_i = \Delta_i/2$ i przejdź do kroku 2);
W przeciwnym przypadku ustal $x(0) = \bar{x}$ i przejdź do kroku 2)

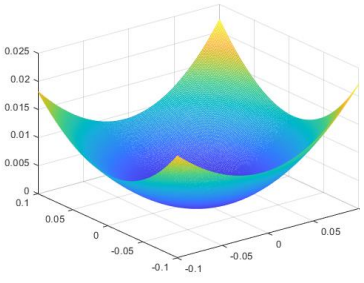


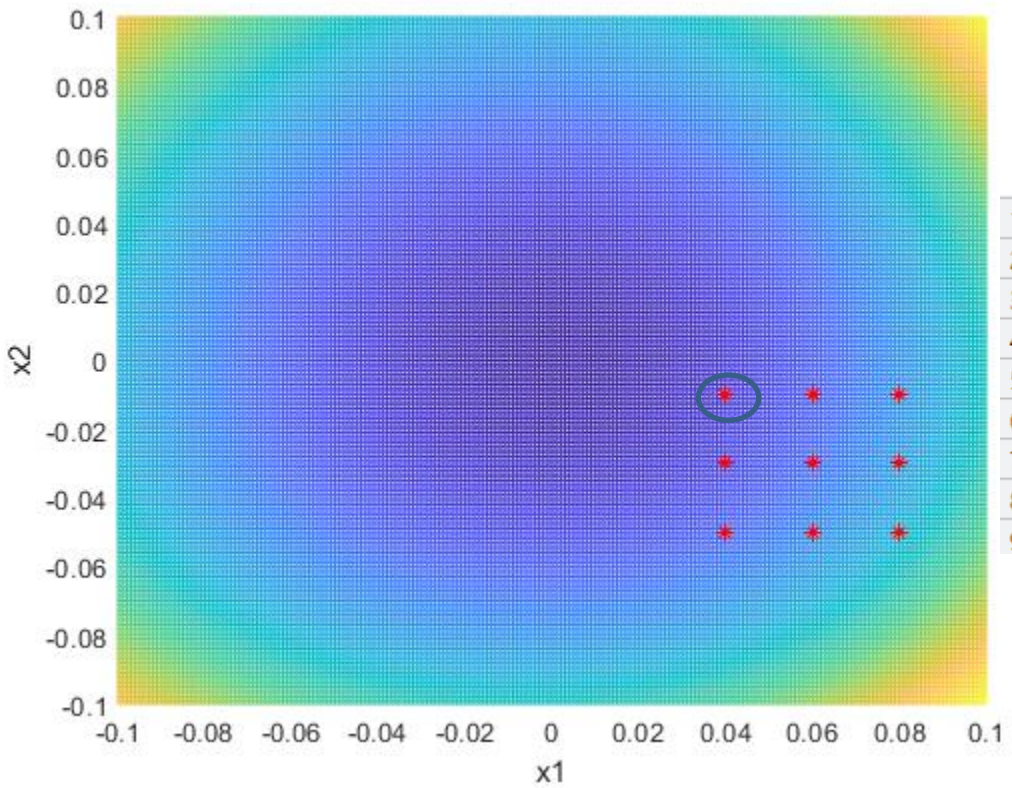
$$f(x_1, x_2) = x_1^2 e^{x_1+x_2} + x_2^2 e^{x_1-x_2}$$



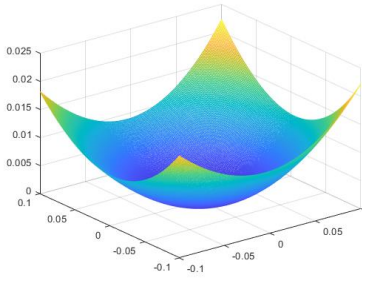
-0.01	-0.01	0.00019801986733067 6
-0.01	0	9.90049833749168e-05
-0.01	0.01	0.00019801986733067 6
0	-0.01	0.00010100501670841 7
0	0	0
0	0.01	9.90049833749168e-05
0.01	-0.01	0.00020202013400267 6
0.01	0	0.00010100501670841 7
0.01	0.01	0.00020202013400267 6

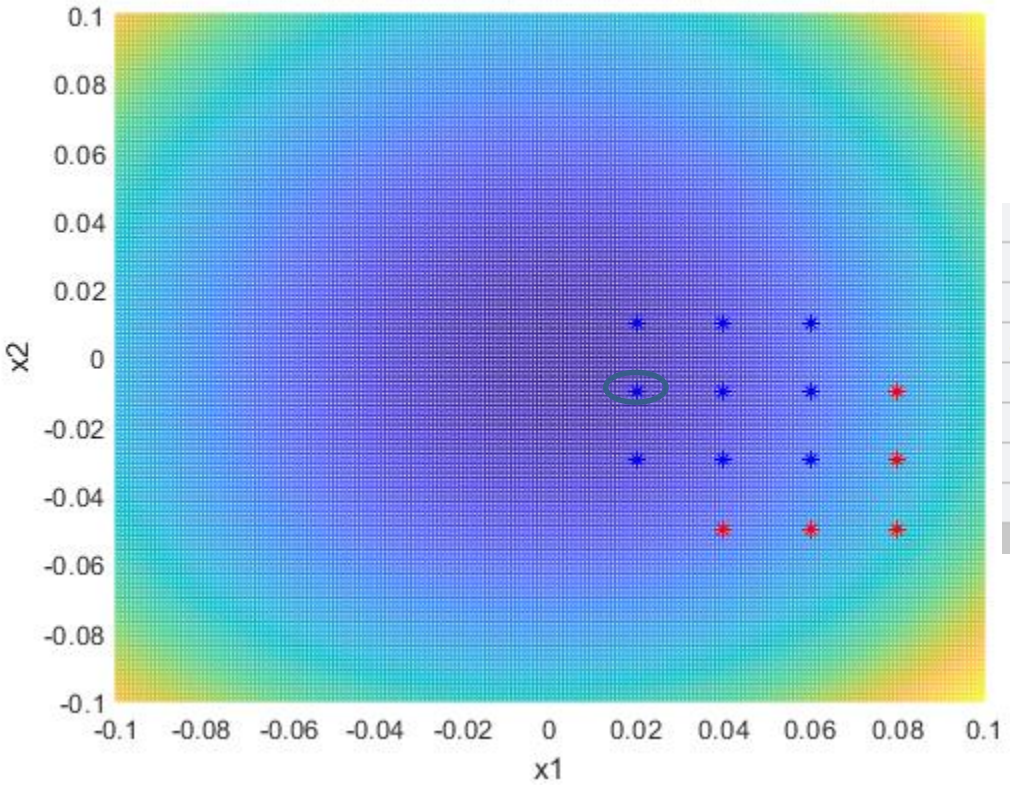
hiperPszesciennaOS.m



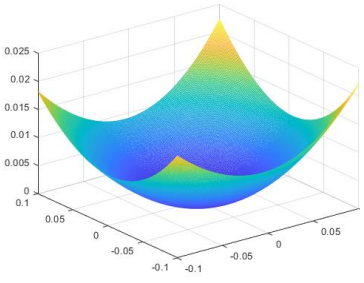


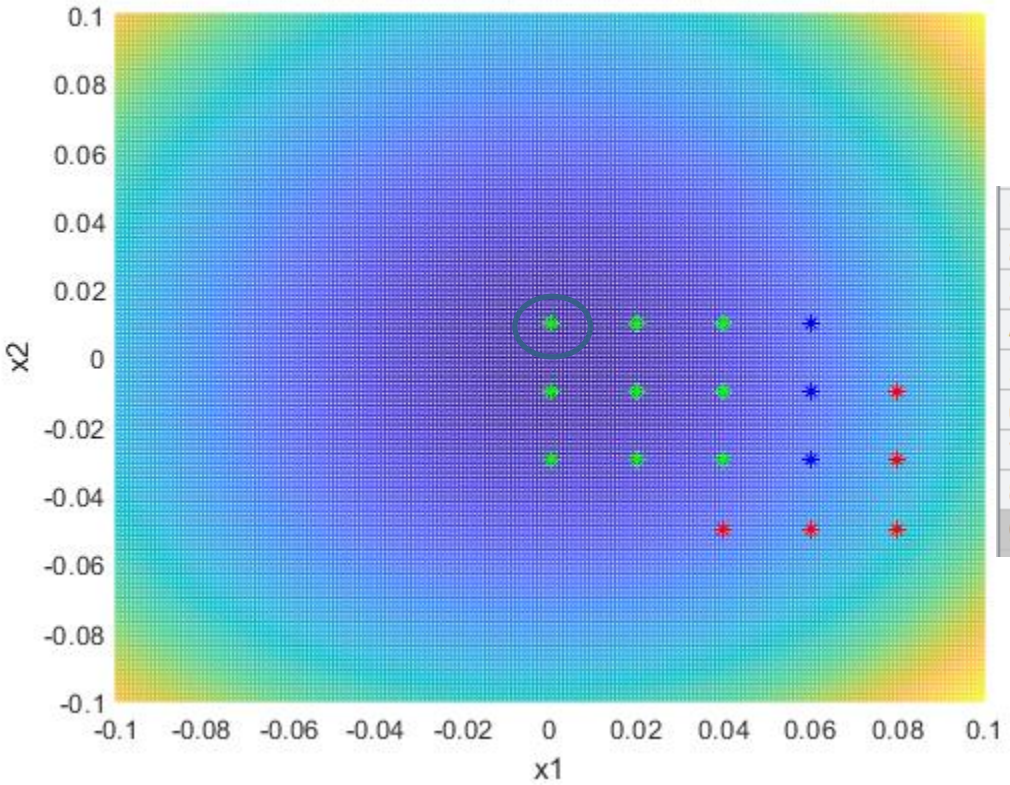
1	0.0400	-0.0500	0.0043
2	0.0400	-0.0300	0.0026
3	0.0400	-0.0100	0.0018
4	0.0600	-0.0500	0.0064
5	0.0600	-0.0300	0.0047
6	0.0600	-0.0100	0.0039
7	0.0800	-0.0500	0.0094
8	0.0800	-0.0300	0.0077
9	0.0800	-0.0100	0.0070



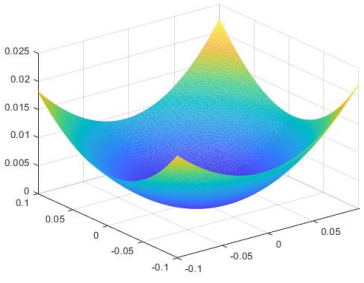


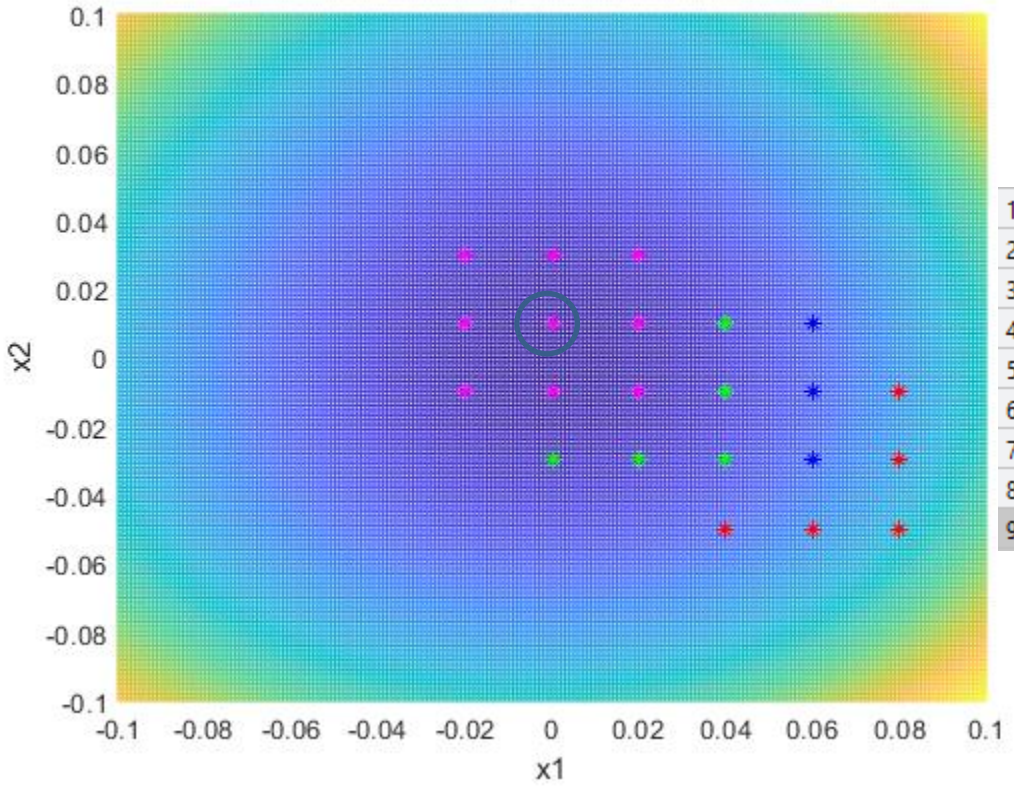
1	0.0200	-0.0300	0.0013
2	0.0200	-0.0100	5.0707e-04
3	0.0200	0.0100	5.1319e-04
4	0.0400	-0.0300	0.0026
5	0.0400	-0.0100	0.0018
6	0.0400	0.0100	0.0018
7	0.0600	-0.0300	0.0047
8	0.0600	-0.0100	0.0039
9	0.0600	0.0100	0.0040



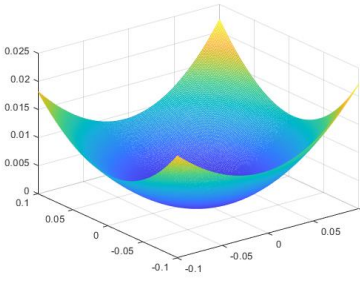


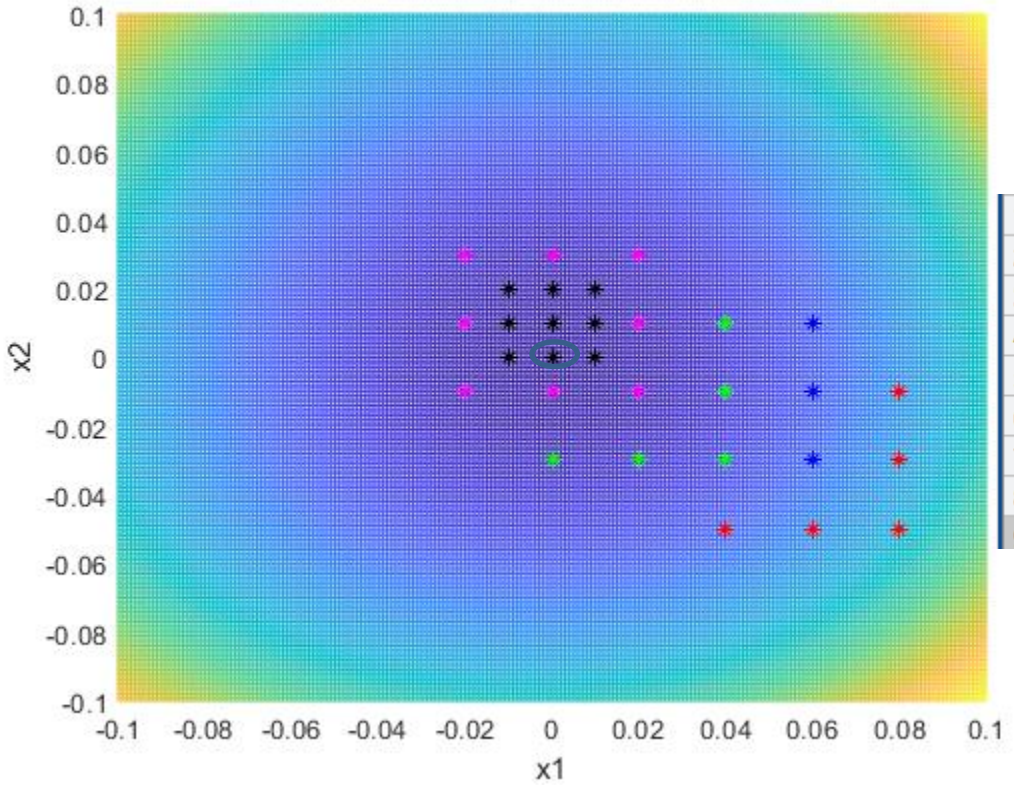
1	0	-0.0300	9.2741e-04
2	0	-0.0100	1.0101e-04
3	0	0.0100	9.9005e-05
4	0.0200	-0.0300	0.0013
5	0.0200	-0.0100	5.0707e-04
6	0.0200	0.0100	5.1319e-04
7	0.0400	-0.0300	0.0026
8	0.0400	-0.0100	0.0018
9	0.0400	0.0100	0.0018



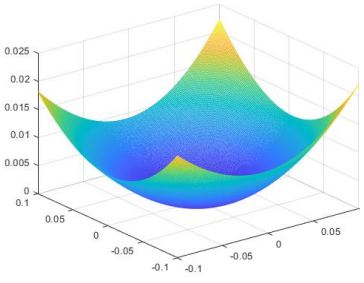


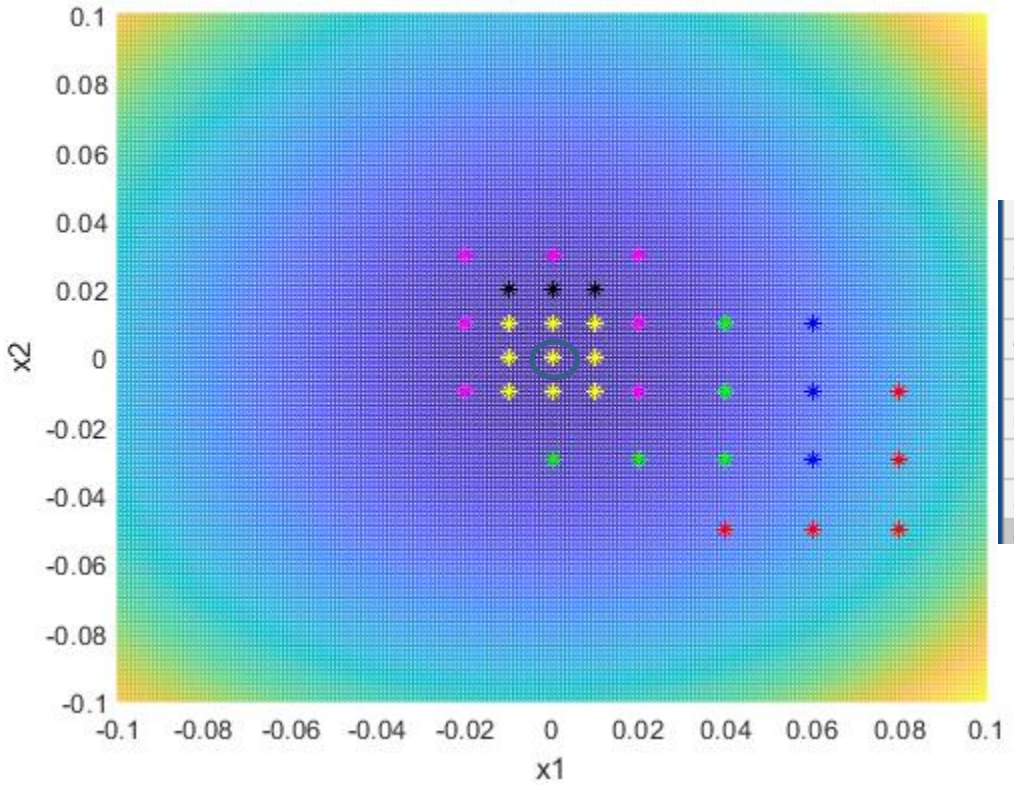
1	-0.0200	-0.0100	4.8718e-04
2	-0.0200	0.0100	4.9306e-04
3	-0.0200	0.0300	0.0013
4	0	-0.0100	1.0101e-04
5	0	0.0100	9.9005e-05
6	0	0.0300	8.7340e-04
7	0.0200	-0.0100	5.0707e-04
8	0.0200	0.0100	5.1319e-04
9	0.0200	0.0300	0.0013



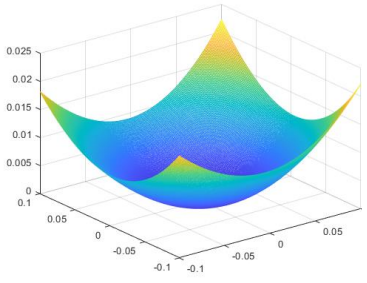


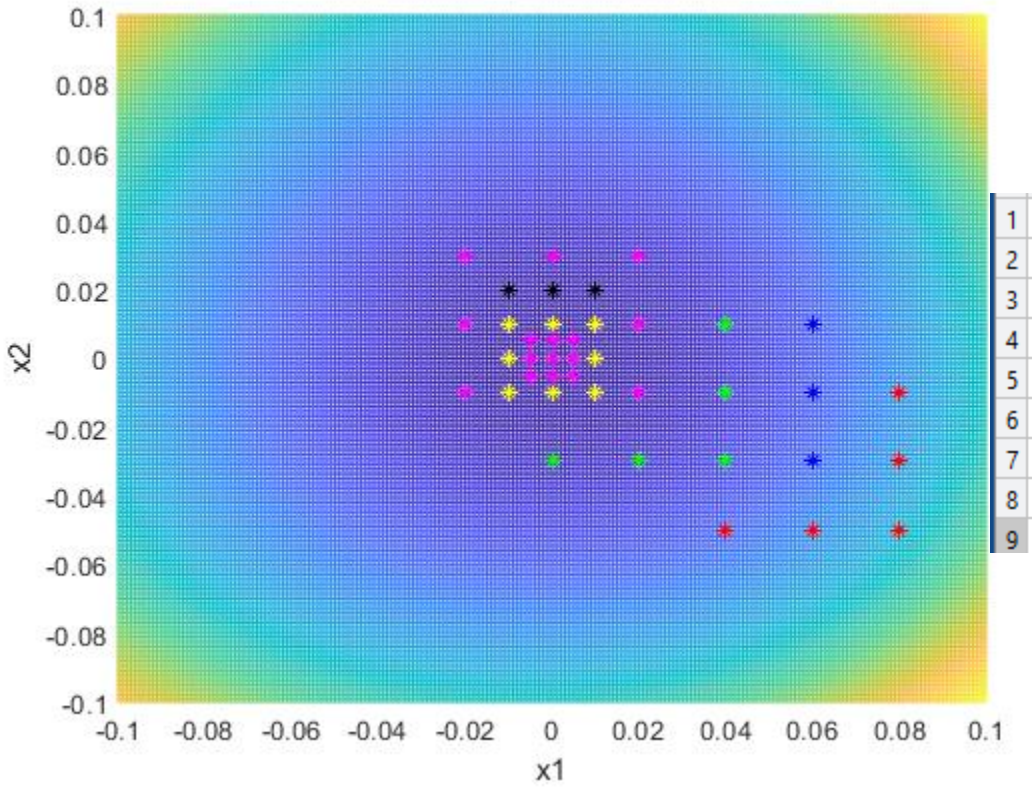
1	-0.0100	0	9.9005e-05
2	-0.0100	0.0100	1.9802e-04
3	-0.0100	0.0200	4.8918e-04
4	0	0	0
5	0	0.0100	9.9005e-05
6	0	0.0200	3.9208e-04
7	0.0100	0	1.0101e-04
8	0.0100	0.0100	2.0202e-04
9	0.0100	0.0200	4.9907e-04



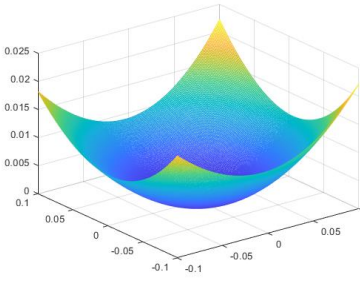


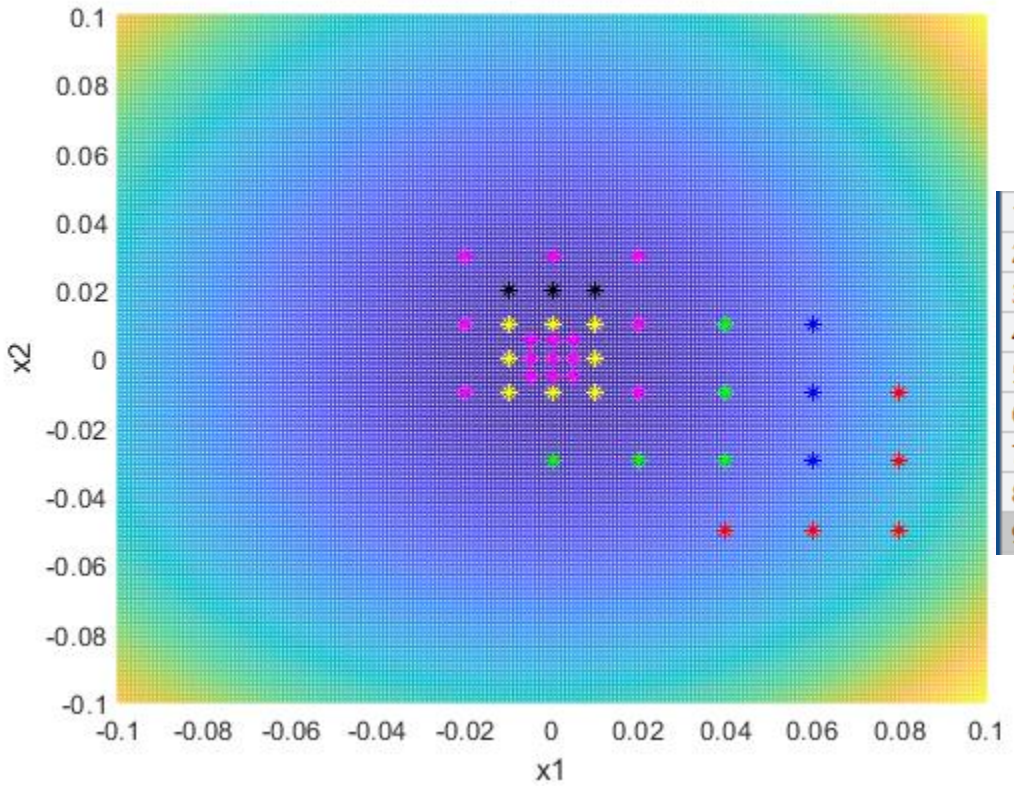
1	-0.0100	-0.0100	1.9802e-04
2	-0.0100	0	9.9005e-05
3	-0.0100	0.0100	1.9802e-04
4	0	-0.0100	1.0101e-04
5	0	0	0
6	0	0.0100	9.9005e-05
7	0.0100	-0.0100	2.0202e-04
8	0.0100	0	1.0101e-04
9	0.0100	0.0100	2.0202e-04



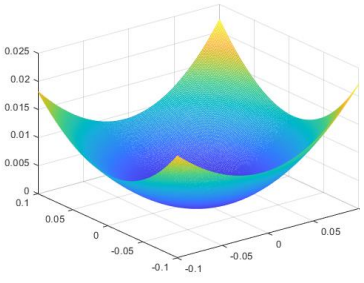


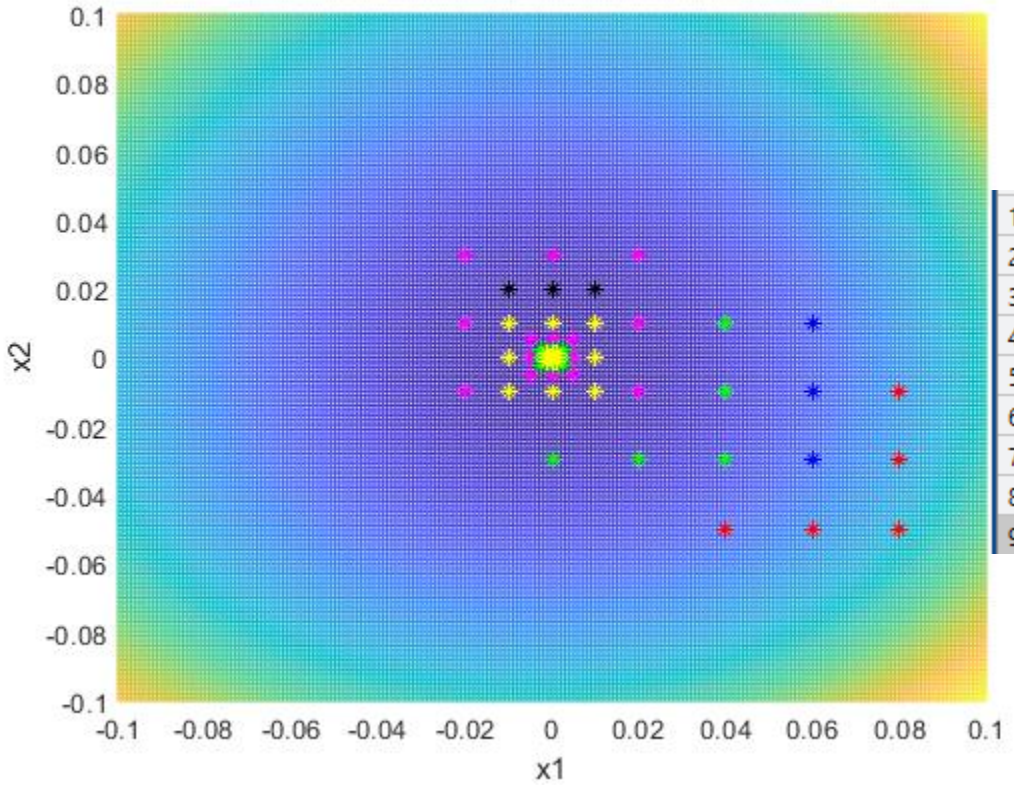
1	-0.0050	-0.0050	4.9751e-05
2	-0.0050	0	2.4875e-05
3	-0.0050	0.0050	4.9751e-05
4	0	-0.0050	2.5125e-05
5	0	0	0
6	0	0.0050	2.4875e-05
7	0.0050	-0.0050	5.0251e-05
8	0.0050	0	2.5125e-05
9	0.0050	0.0050	5.0251e-05



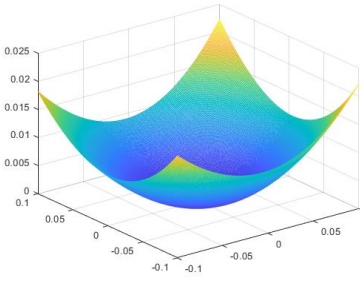


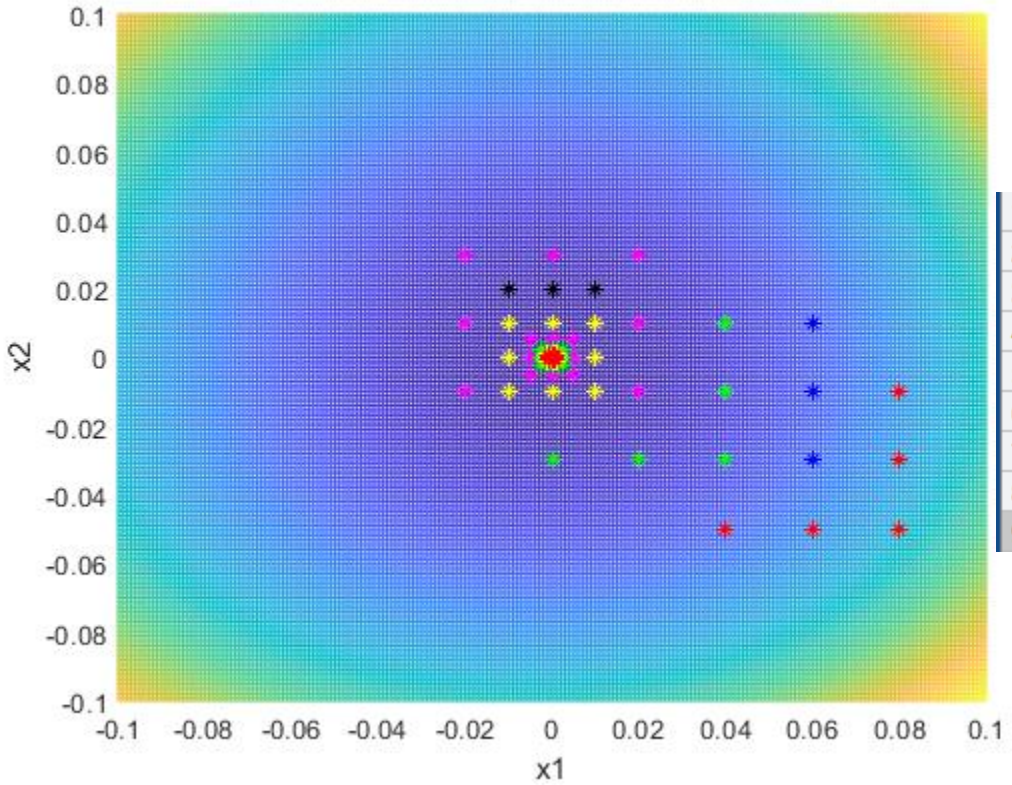
1	-0.0025	-0.0025	1.2469e-05
2	-0.0025	0	6.2344e-06
3	-0.0025	0.0025	1.2469e-05
4	0	-0.0025	6.2656e-06
5	0	0	0
6	0	0.0025	6.2344e-06
7	0.0025	-0.0025	1.2531e-05
8	0.0025	0	6.2656e-06
9	0.0025	0.0025	1.2531e-05





1	-0.0013	-0.0013	3.1211e-06
2	-0.0013	0	1.5605e-06
3	-0.0013	0.0013	3.1211e-06
4	0	-0.0013	1.5645e-06
5	0	0	0
6	0	0.0013	1.5605e-06
7	0.0013	-0.0013	3.1289e-06
8	0.0013	0	1.5645e-06
9	0.0013	0.0013	3.1289e-06





1	-6.2500e-04	-6.2500e-04	7.8076e-07
2	-6.2500e-04	0	3.9038e-07
3	-6.2500e-04	6.2500e-04	7.8076e-07
4	0	-6.2500e-04	3.9087e-07
5	0	0	0
6	0	6.2500e-04	3.9038e-07
7	6.2500e-04	-6.2500e-04	7.8174e-07
8	6.2500e-04	0	3.9087e-07
9	6.2500e-04	6.2500e-04	7.8174e-07

Metoda sympleksu Neldera-Meada

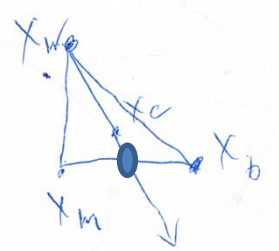
Sympleks wymaga dużo mniejszej ilości punktów niż hypersześcian, co staje się szczególnie widoczne dla wielu wymiarów. Sympleks w N-wymiarowej przestrzeni ma $N + 1$ wierzchołków. Jest to minimalna ilość wierzchołków umożliwiająca poszukiwanie we wszystkich możliwych kierunkach N-wymiarowej przestrzeni. Ważne jest jednak, aby sympleks[4] nie rozpinał figury o zerowej objętości w N-wymiarowej przestrzeni - czyli na przykład dla funkcji dwuwymiarowej trzy punkty sympleksu nie mogą leżeć na jednej linii, a w przypadku funkcji trójwymiarowej cztery punkty sympleksu nie mogą leżeć na jednej płaszczyźnie.

Sympleks – uogólnienie odcinka, trójkąta i czworościanu na dowolne wymiary. Intuicyjnie, k-wymiarowym sympleksem nazywamy k-wymiarowy wielościan, który jest **wypukłą otoczką** swoich k+1 wierzchołków.

1) Wybierz $\gamma > 1$, $\beta \in (0,1)$ oraz parametr zakończenia ϵ. Stwórz sympleks początkowy.

2) Znajdź najgorszy punkt x_w, najlepszy punkt x_b oraz drugi w kolejności najgorszych x_m. Oblicz punkt, względem którego będziemy odbijać x_w:

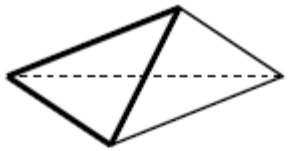
$$x_c = \frac{1}{N} \sum_{i=1, i \neq h}^{N+1} x_i.$$



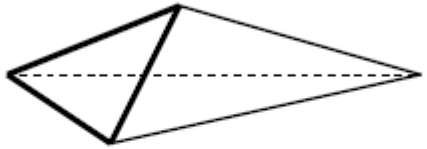
3) Oblicz punkt odbicia $x_r = 2x_c - x_w$. Ustal $x_{\text{new}} = x_r$.
Jeśli $f(x_r) < f(x_b)$, ustal $x_{\text{new}} = (1+\gamma)x_c - \gamma x_w$ (ekspansja);
Jeśli $f(x_r) \geqslant f(x_w)$, ustal $x_{\text{new}} = (1-\beta)x_c + \beta x_w$ (kontrakcja);
Jeśli $f(x_m) < f(x_r) < f(x_w)$, ustal $x_{\text{new}} = (1+\beta)x_c - \beta x_w$ (kontrakcja).
Policz $f(x_{\text{new}})$ i wymień x_w na x_{new}.

4) Jeśli $\left\{ \sum_{i=1}^{N+1} \frac{(f(x_i) - f(x_c))^2}{N+1} \right\}^{1/2} \leqslant \epsilon$, **Zakończ**;
W przeciwnym przypadku idź do kroku 2).

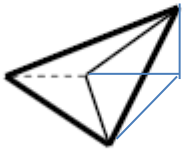
odbicie



rozciągnięcie

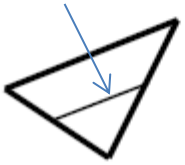


spłaszczenie

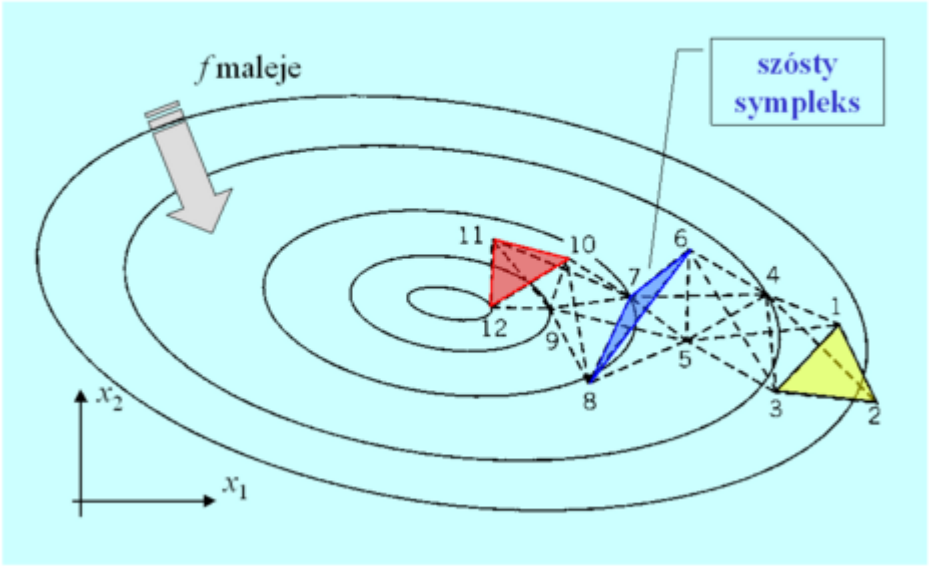


wewnętrzne/zewnętrzne

kontrakcja



Algorytm Neldera i Meada



Algorytm

1. W przestrzeni n wymiarowej tworzymy wokół punktu x_0 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})

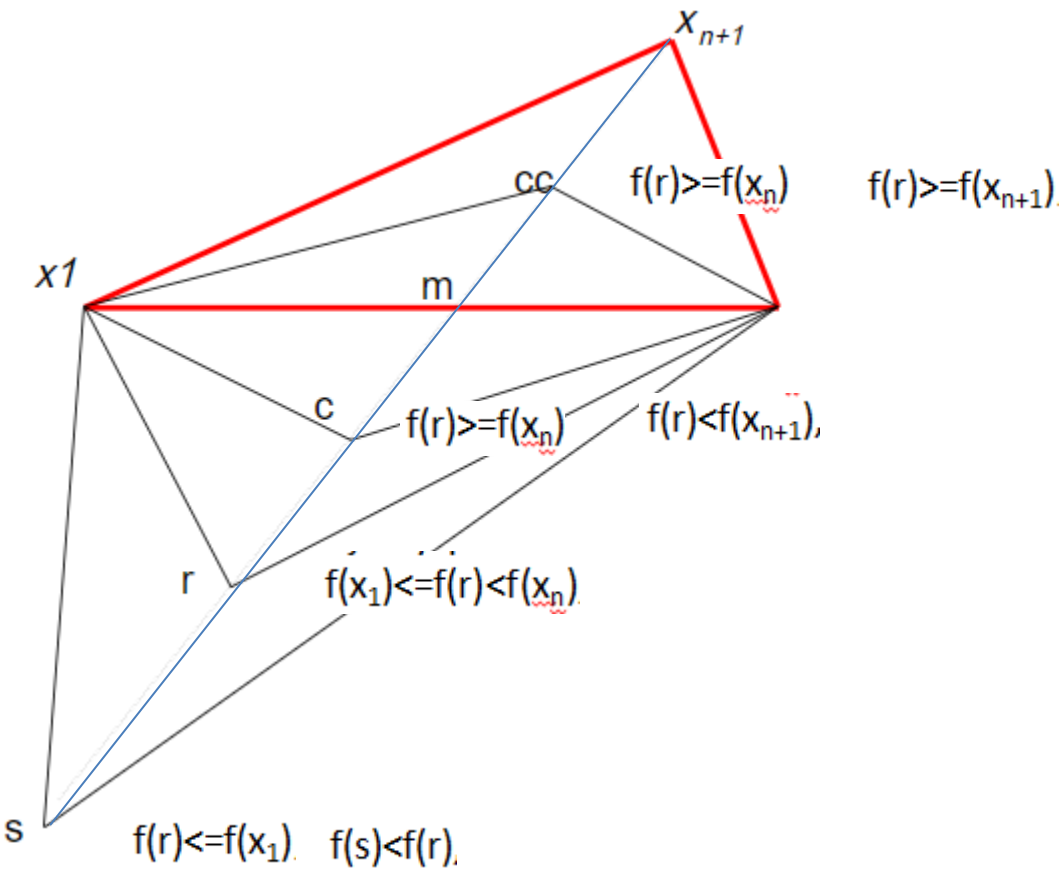
a. jeżeli $f(r)<f(x_{n+1})$, to obliczamy $c=m+(r-m)/2$,jeżeli $f(c)<f(r)$, to akceptujemy c, **contract outside**, jeżeli nie, to do punktu 7,

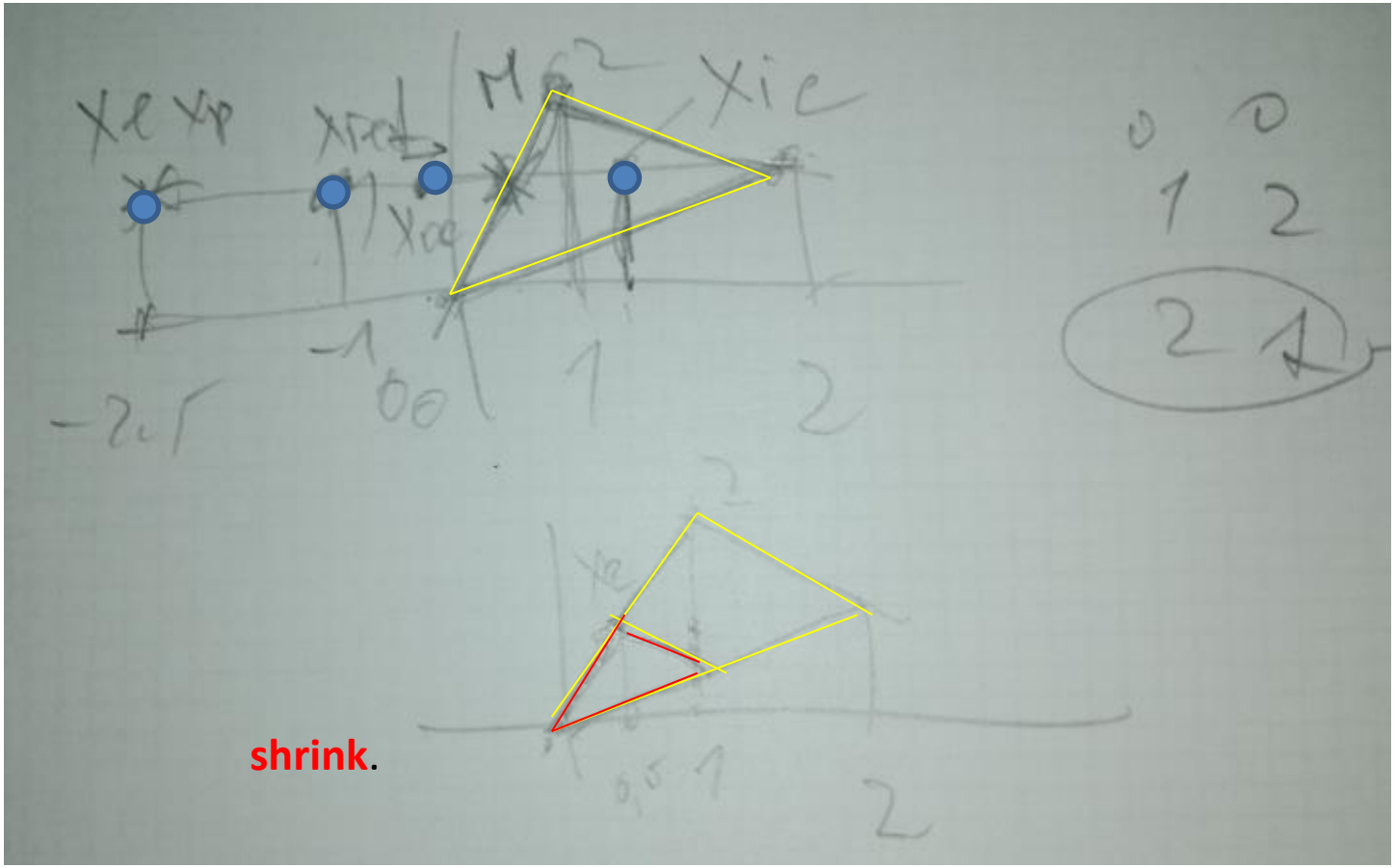
b. jeżeli $f(r)>=f(x_{n+1})$, to obliczamy $cc=m+(x_{n+1}-m)/2$,jeżeli $f(cc)<f(x_{n+1})$, to akceptujemy cc, **contract inside**, jeżeli nie, to do punktu 7.

7. Obliczamy n punktów $vi=x_1+(x_i-x_1)/2$, i=2,...,n+1, **shrink**. Ew. unikanie degradacji.

https://www.youtube.com/watch?v=j2gcuRVbwR0

http://www.scholarpedia.org/article/Nelder-Mead_algorithm


x_{n+1}
cc
f(r)>=f(x_n)
f(r)>=f(x_{n+1})
x1
m
c
f(r)>=f(x_n)
f(r)<f(x_{n+1}),
r
f(x_1)<=f(r)<f(x_n)
s
f(r)<=f(x_1)
f(s)<f(r)


shrink.

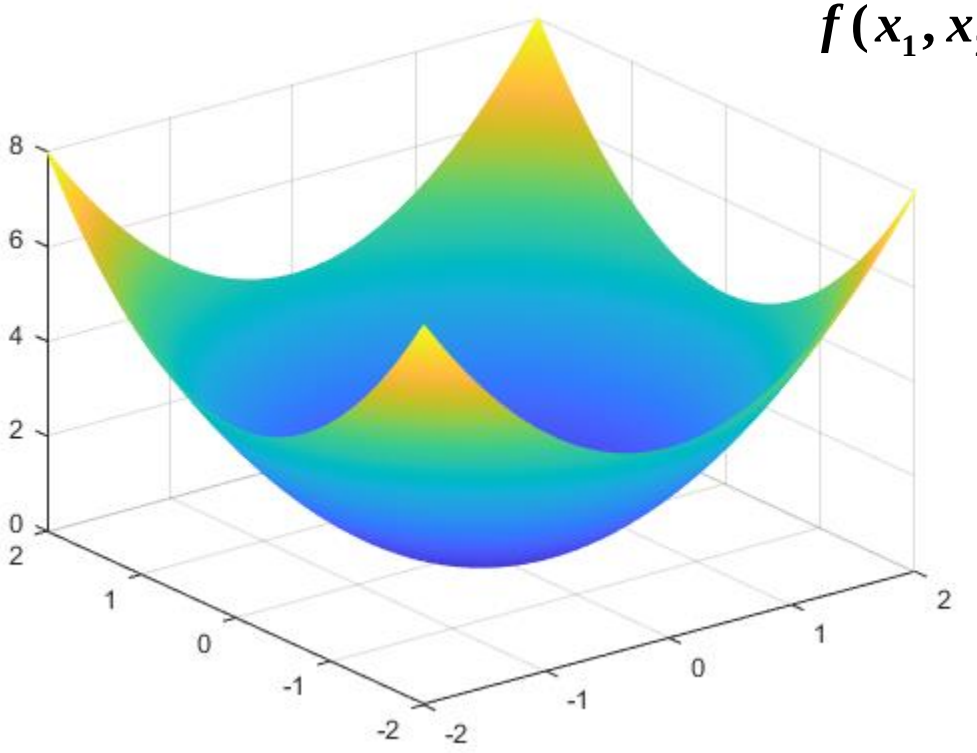
Sympleks początkowy

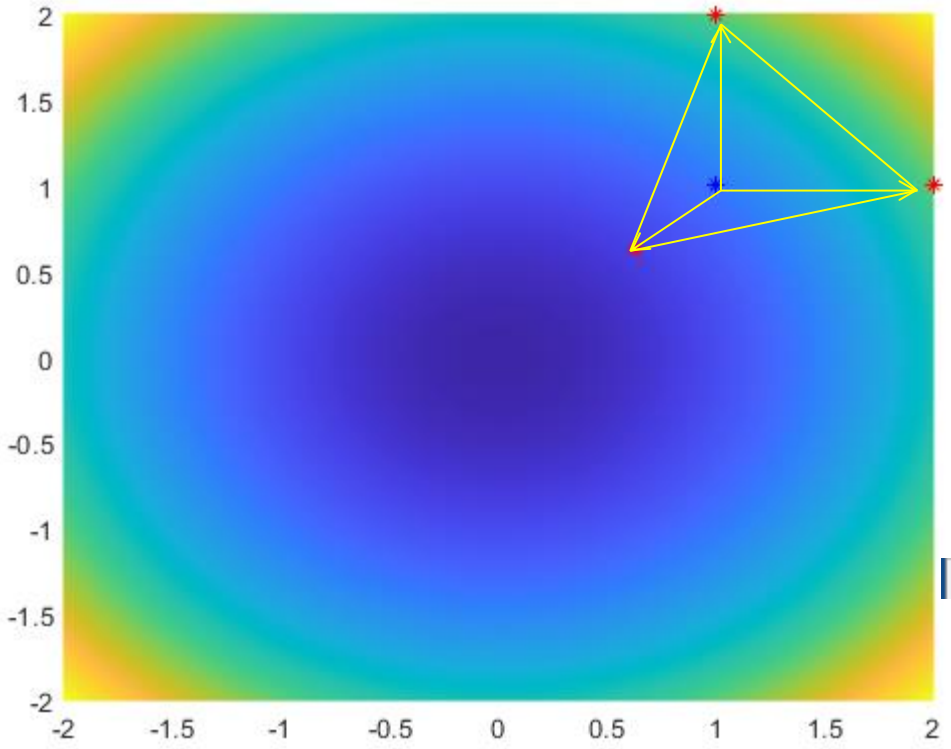
Jednym ze sposobów stworzenia sympleksu początkowego do pierwszego kroku algorytmu jest wybranie punktu bazowego $x^{(0)}$ oraz liczby C. Wówczas $N+1$ punktów sympleksu to $x^{(0)}$ oraz dla $i, j = 1, 2, ..., N$:

$$x_j^{(i)} = \begin{cases} x_j^{(0)} + C & \text{if } j = i \\ x_j^{(0)} + C\Delta & \text{otherwise,} \end{cases} \qquad \text{where } \Delta = \begin{cases} 0.25 & \text{if } N = 3 \\ \frac{\sqrt{N+1}-2}{N-3} & \text{otherwise.} \end{cases}$$

Dla zapewnienia dobrej zbieżności algorytmu, sugeruje się ustalenie $\gamma \approx 2$ i $|\beta| \approx 0.5$.

$$f(x_1, x_2) = x_1^2 + x_2^2$$





X0=[1 1]

X

M

1	0.6340	0.6340
2	2	1
3	1	2

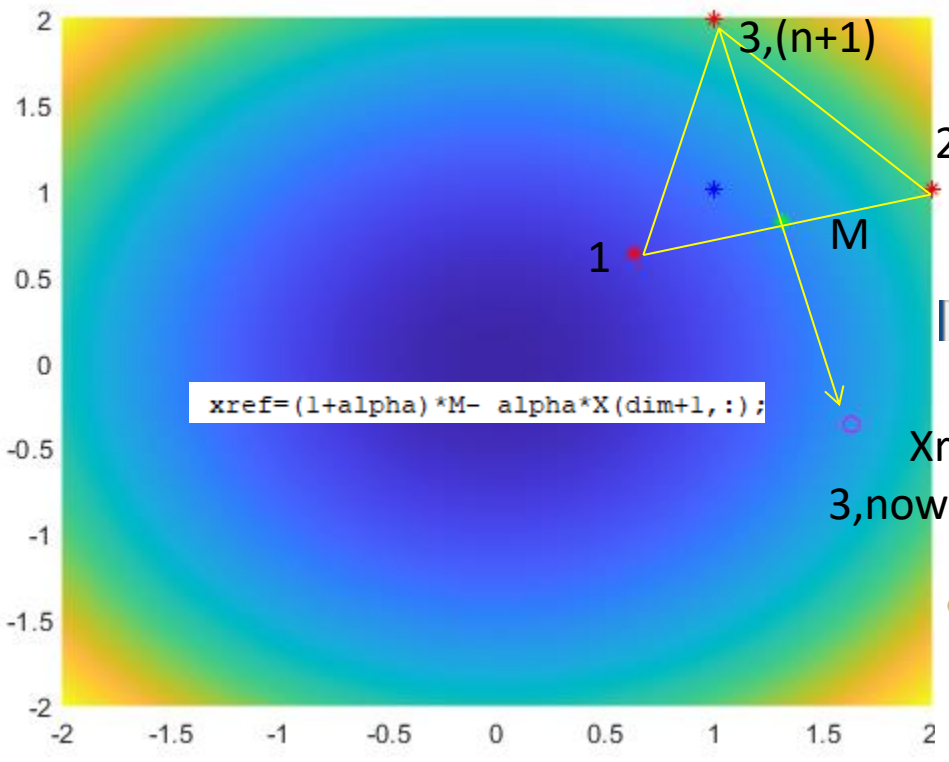
$F_{sort}(X)$

1	0.8038	5	5

```
[FX,I]=sort(FX);
X=X(I,:);
```

[xmin,fmin,ct]=ANMSsimpexOS(@par3d,[1 1],10^-4,30)

ProgramOS\par3d.m



X

1	0.6340	0.6340
2	2	1
3	1	2

$F_{sort}(X)$

1	0.8038	5	5

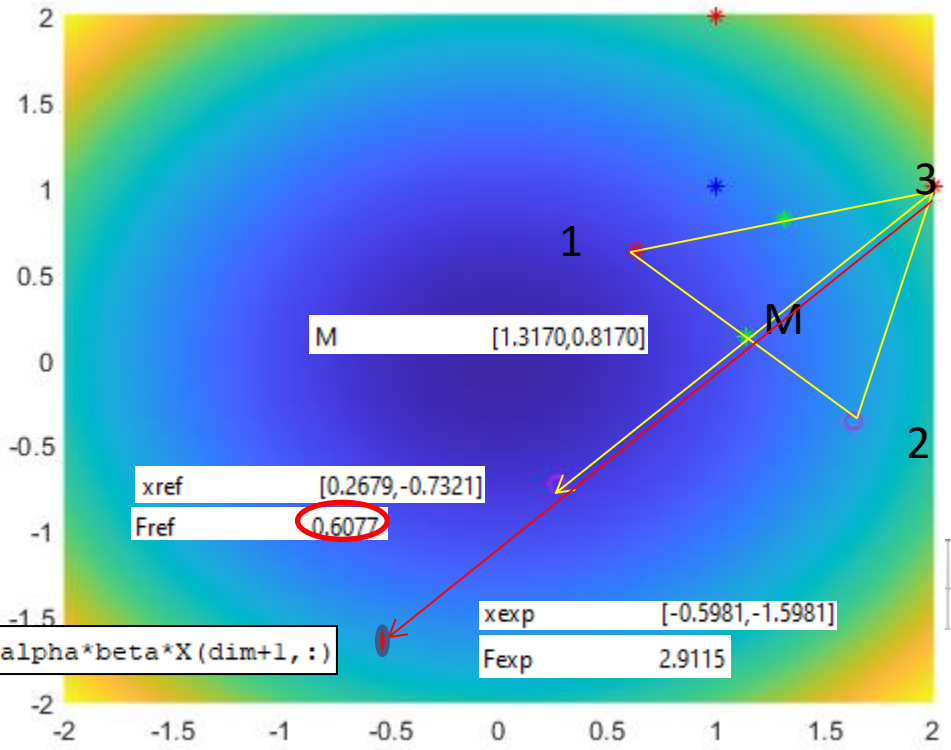
xref [1.6340,-0.3660]

Fref 2.8038

dim 2

```
if Fref<FX(dim)
    % accept reflection point
    X(dim+1,:)=xref;
    FX(dim+1)=Fref;
```

```
[FX,I]=sort(FX);
X=X(I,:);
```



X

1	0.6340	0.6340
2	1.6340	-0.3660
3	2	1

$F_{sort}(X)$

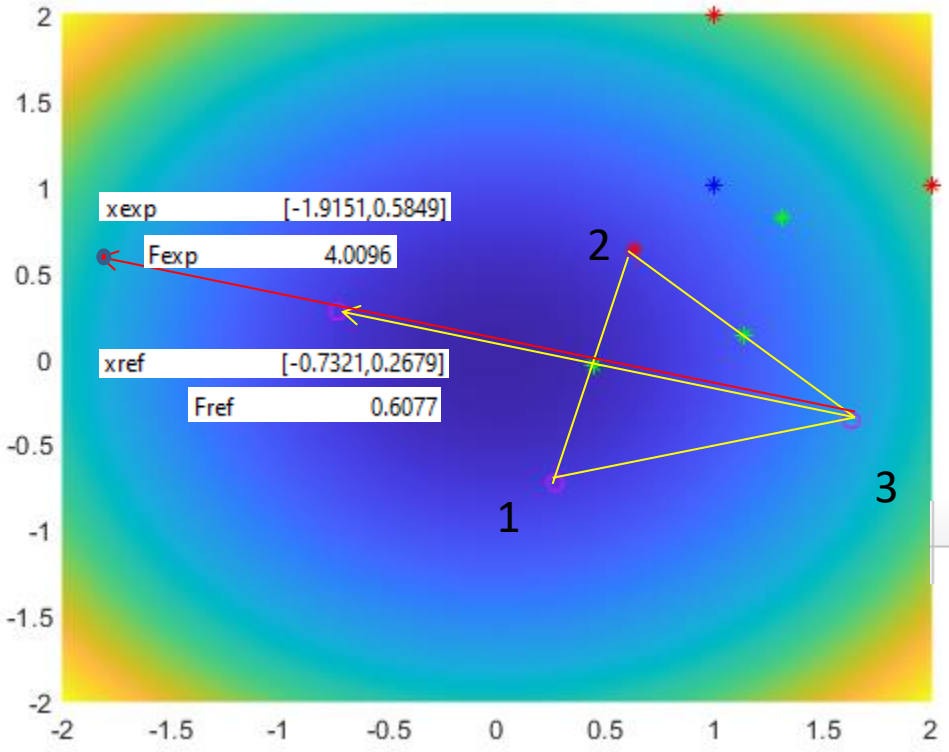
1	2	3
0.8038	2.8038	5

```
if Fref<FX(1)
    % expansion
    xexp=(1+alpha*beta)*M-alpha*beta*X(dim+1,:);
    Fexp=feval(myfunction,xexp);
```

```
if Fexp < Fref
    X(dim+1,:)=xexp;
    FX(dim+1)=Fexp;
else
    X(dim+1,:)=xref;
    FX(dim+1)=Fref;
end;
```

```
[FX,I]=sort(FX);
X=X(I,:);
```

```
Fref =

    0.607695154586736

K>> FX(1)

ans =

    0.607695154586736

K>> Fref<FX(1)

ans =

  logical

    1
```



X

1	0.2679	-0.7321
2	0.6340	0.6340
3	1.6340	-0.3660

$F_{sort}(X)$

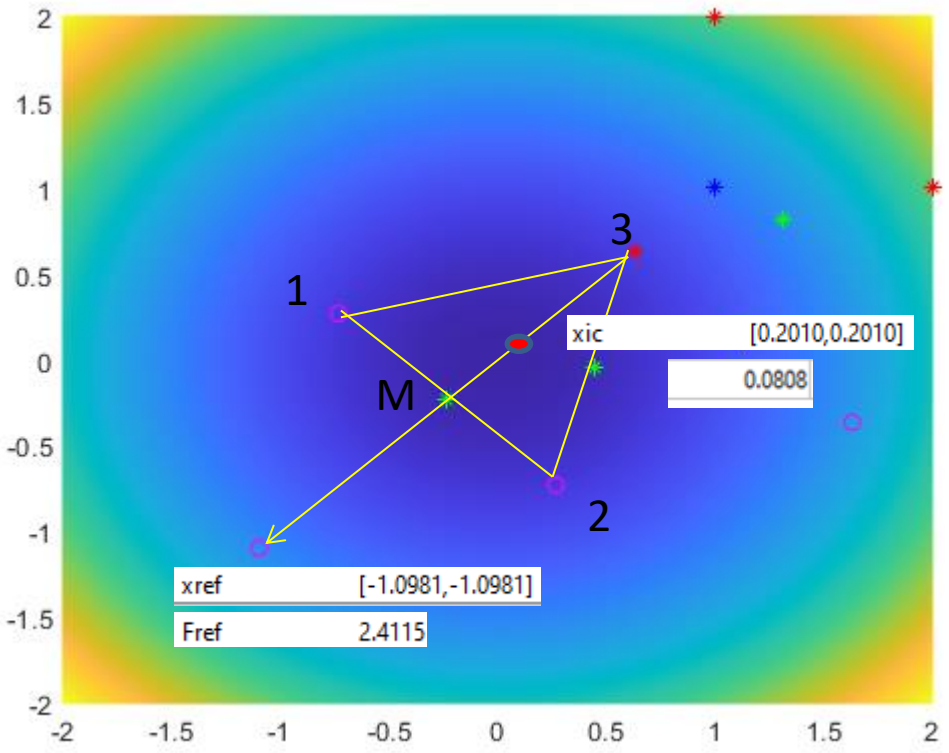
1	2	3
0.6077	0.8038	2.8038

```
[FX,I]=sort(FX);
X=X(I,:);
```

```
if Fref<FX(1)
    % expansion
    xexp=(1+alpha*beta)*M-alpha*beta*X(dim+1,:);
    Fexp=feval(myfunction,xexp);
```

```
if Fexp < Fref
 X(dim+1,:)=xexp;
 FX(dim+1)=Fexp;
else
 X(dim+1,:)=xref;
 FX(dim+1)=Fref;
end;
```

```
xic=(1-gamma)*M+gamma*X(dim+1,:);
```



X

1	-0.7321	0.2679
2	0.2679	-0.7321
3	0.6340	0.6340

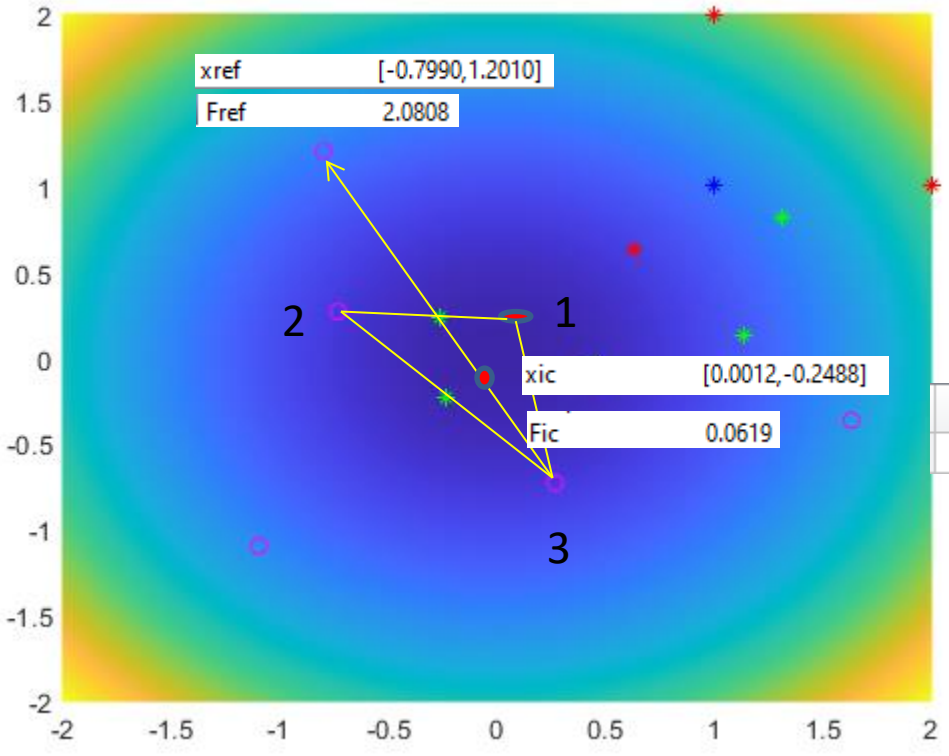
$F_{sort}(X)$

	1	2	3
1	0.6077	0.6077	0.8038

```
%inside contraction
xic=(1-gamma)*M+gamma*X(dim+1,:);
Fic=feval(myfunction,xic);
```

```
if Fic<FX(dim+1)
    X(dim+1,:)=xic;
    FX(dim+1)=Fic;
else
    % shrink
    for i=2:dim+1
        X(i,:)=X(1,:)+ delta*(X(i,:)-X(1,:));
        FX(i)=feval(myfunction,X(i,:));
    end;
```

```
[FX,I]=sort(FX);
X=X(I,:);
```



X

1	0.2010	0.2010
2	-0.7321	0.2679
3	0.2679	-0.7321

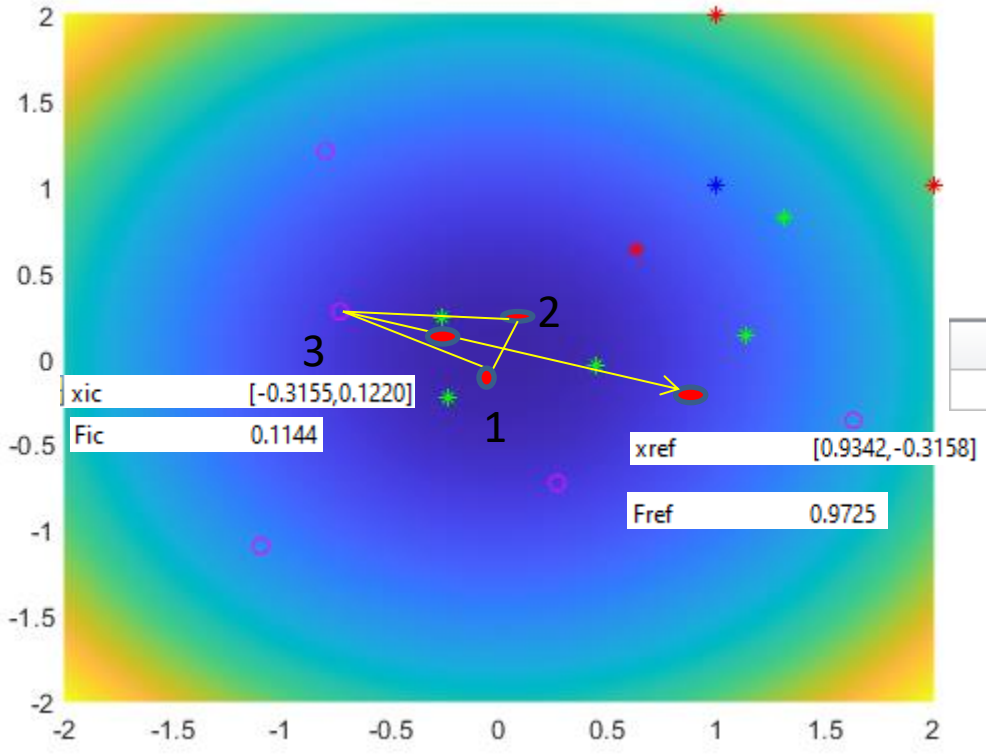
$F_{sort}(X)$

1	2	3
0.0808	0.6077	0.6077

```
%inside contraction
xic=(1-gamma)*M+gamma*X(dim+1,:);
Fic=feval(myfunction,xic);
```

```
if Fic<FX(dim+1)
    X(dim+1,:)=xic;
    FX(dim+1)=Fic;
else
    % shrink
    for i=2:dim+1
        X(i,:)=X(1,:)+ delta*(X(i,:)-X(1,:));
        FX(i)=feval(myfunction,X(i,:));
    end;
```

```
[FX,I]=sort(FX);
X=X(I,:);
```



X

1	0.0012	-0.2488
2	0.2010	0.2010
3	-0.7321	0.2679

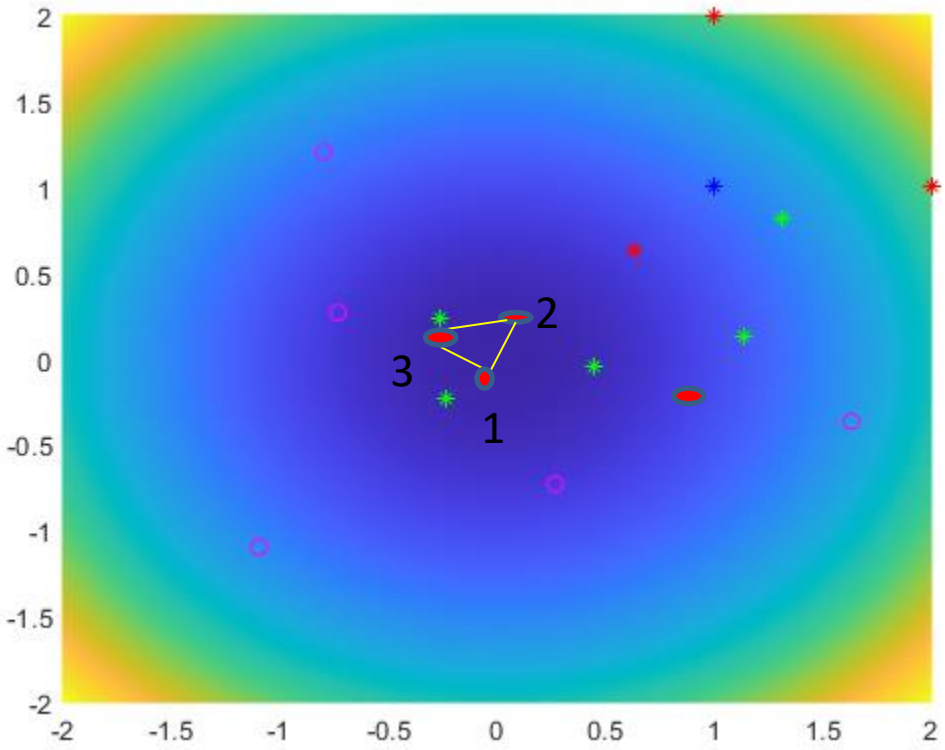
$F_{sort}(X)$

1	2	3
0.0619	0.0808	0.6077

```
%inside contraction
xic=(1-gamma)*M+gamma*X(dim+1,:
Fic=feval(myfunction,xic);
```

```
if Fic<FX(dim+1)
    X(dim+1,:)=xic;
    FX(dim+1)=Fic;
else
    % shrink
    for i=2:dim+1
        X(i,:)=X(1,:)+ delta*(X(i,:)-X(1,:));
        FX(i)=feval(myfunction,X(i,:));
    end;
```

```
[FX,I]=sort(FX);
X=X(I,:);
```

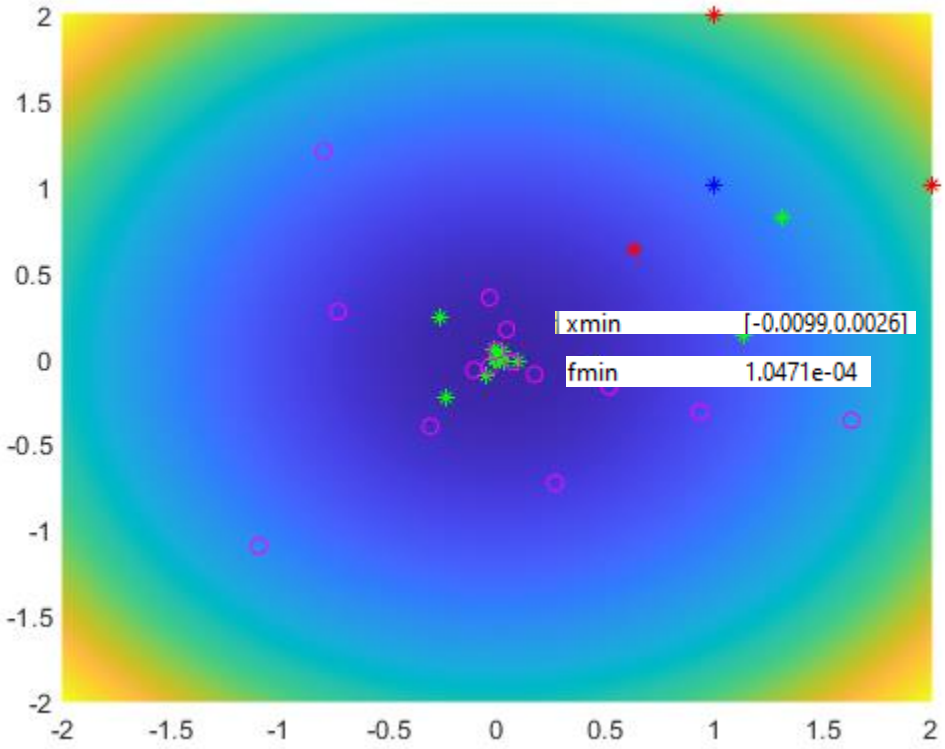


X

1	0.0012	-0.2488
2	0.2010	0.2010
3	-0.3155	0.1220

$F_{sort}(X)$

1	2	3
0.0619	0.0808	0.1144


xmin [-0.0099,0.0026]
fmin 1.0471e-04
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Metoda kierunków sprzężonych Powella

Metoda kierunków sprzężonych Powella jest chyba najbardziej popularną metodą bezpośrednich poszukiwań. Wykorzystuje ona historię poprzednich rozwiązań, aby swtorzyć nowe kierunki poszukiwań. Idea jest prosta: trzeba utworzyć N liniowo niezależnych kierunków poszukiwań i dokonać sekwencyjnie serię poszukiwań wzdłuż tych kierunków, startując za każdym razem z poprzednio znalezionego punktu.

ProgramOS\powell\powell_run.m

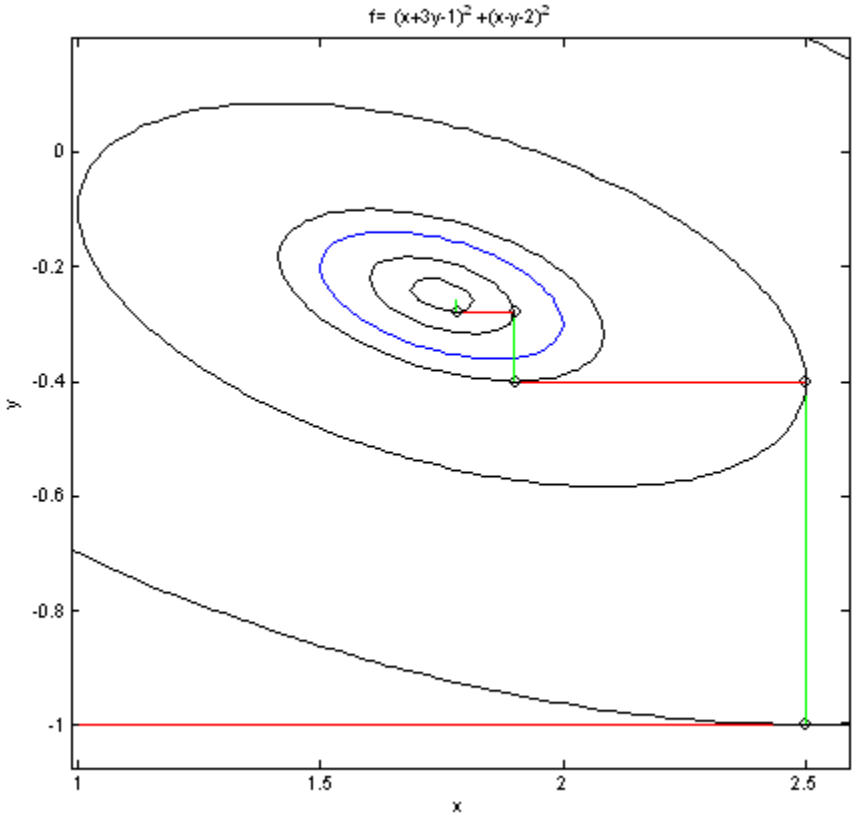
1) Wybierz punkt początkowy $x^{(0)}$ oraz zbiór N liniowo niezależnych kierunków; najlepiej $s^{(i)} = e^{(i)}$, dla $i = 1, 2, ..., N$, gdzie $e^{(i)}$ oznacza i-ty wektor bazowy bazy kanonicznej.

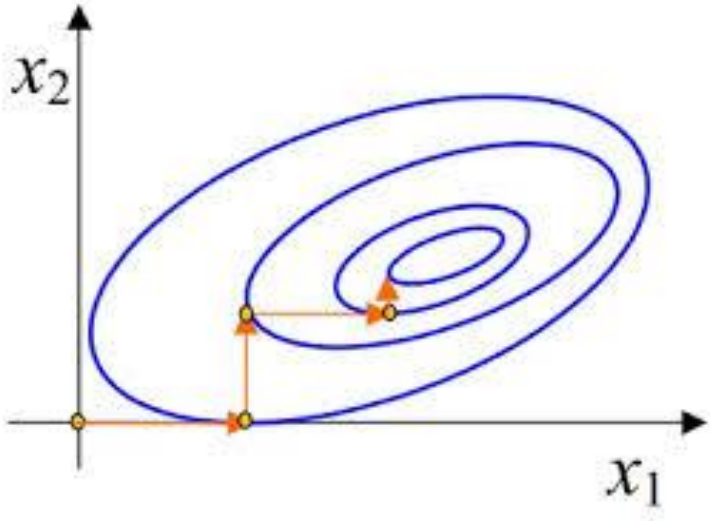
2) Szukaj minimum startując z punktu początkowego wzdłuż kierunku $s^{(1)}$. Startując z nowo znalezionego punktu (oznacz go jako $y^{(1)}$), szukaj minimum wzdłuż kierunku $s^{(2)}$. Kontynuuj szukanie wzdłuż kolejnych kierunków aż do kierunku $s^{(N)}$. Następnie ponownie szukaj minimum wdłuż kierunku $s^{(1)}$. Punkt, który został ostatecznie znaleziony oznacz jako $y^{(2)}$.

3) Oblicz $d = y^{(2)} - y^{(1)}$. Jest to kierunek sprzężony do $s^{(1)}$.

4) Jeśli $||d||$ jest małe lub kierunki $s^{(1)}, s^{(2)}, ..., s^{(N-1)}, d$ są liniowo zależne, **Zakończ**;
W przeciwnym przypadku ustal $s^{(j)} = s^{(j-1)}$ dla wszystkich $j = N, N - 1, ..., 2$. Ustal $s^{(1)} = d/||d||$ i idź do kroku 2).

Przykłady





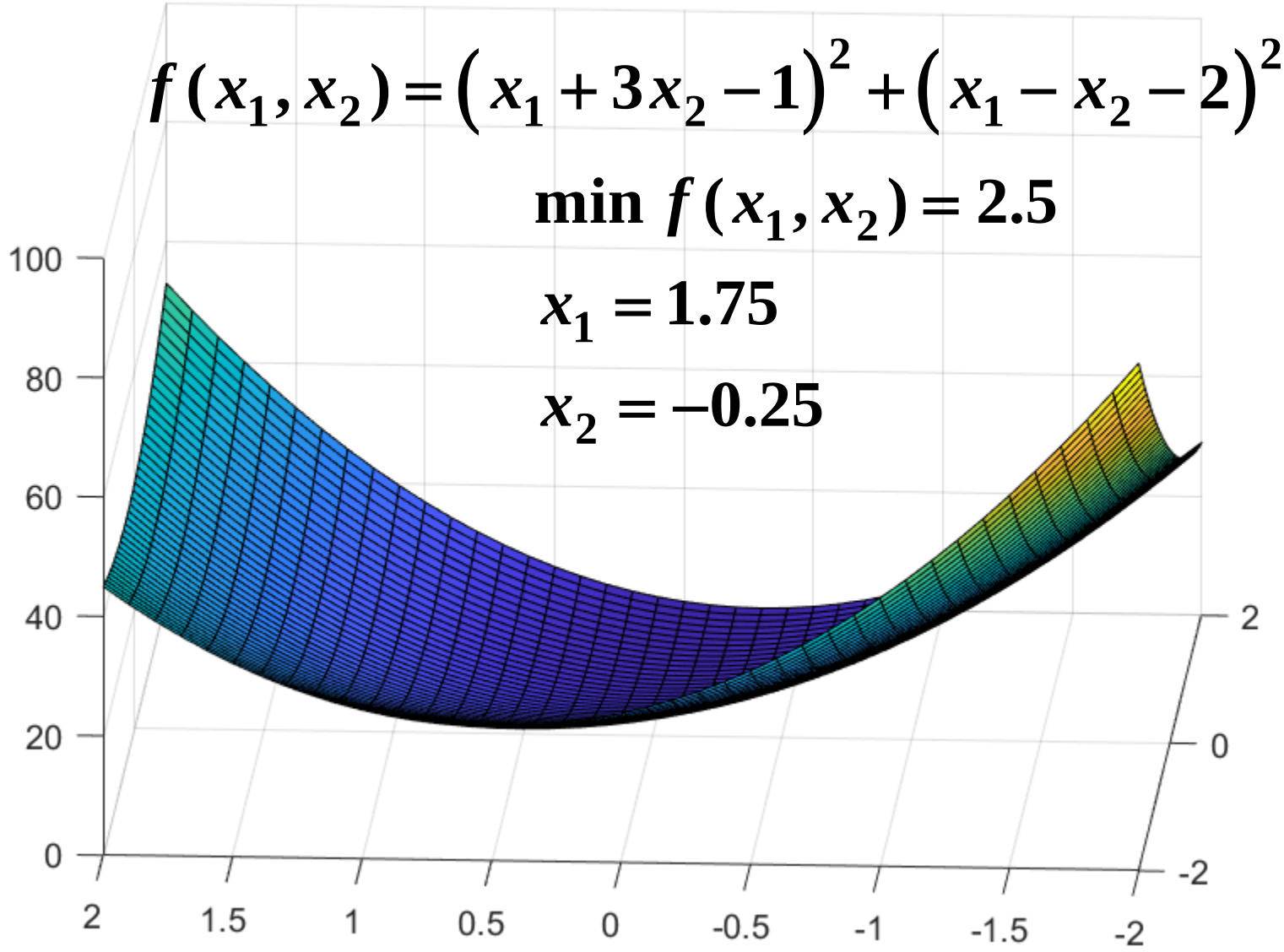
X0=[2.5 -1]

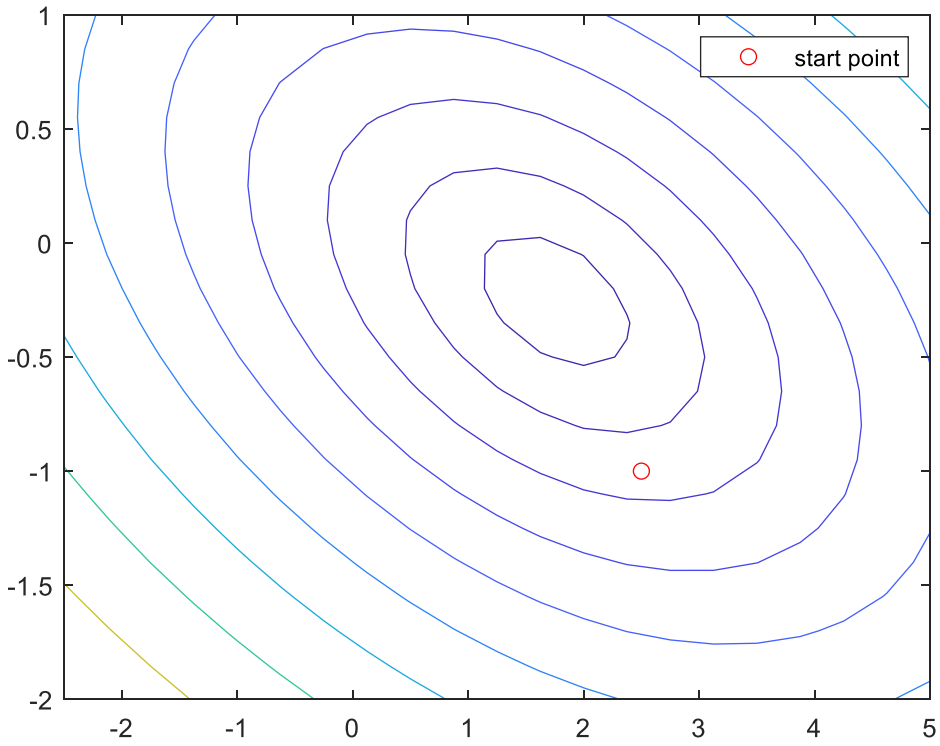
$$f(x_1, x_2) = (x_1 + 3x_2 - 1)^2 + (x_1 - x_2 - 2)^2$$

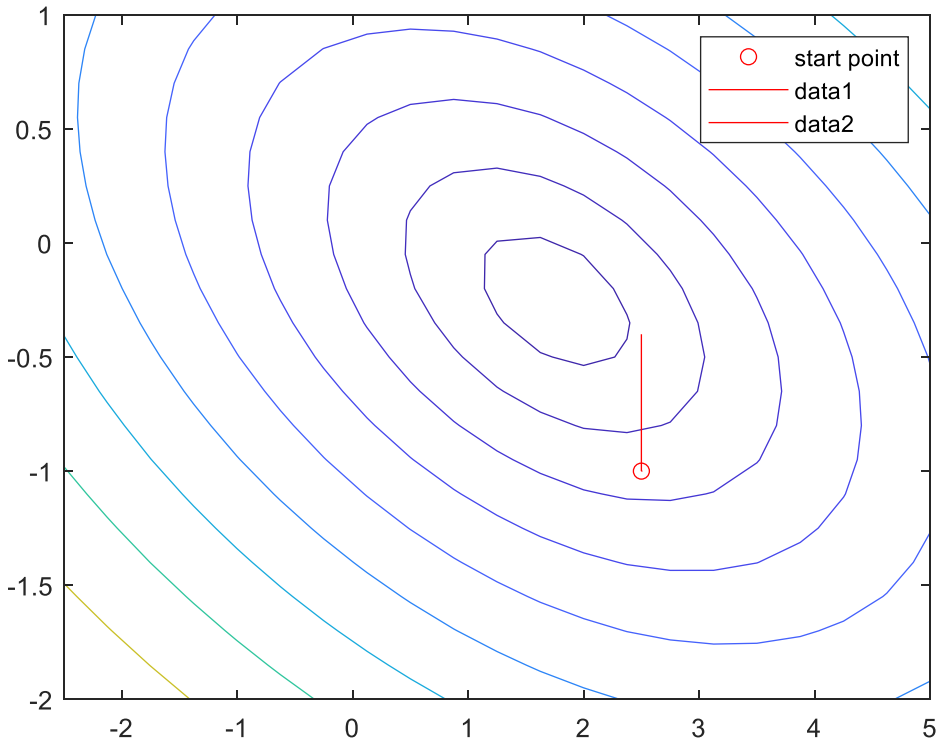
$$\min f(x_1, x_2) = 2.5$$

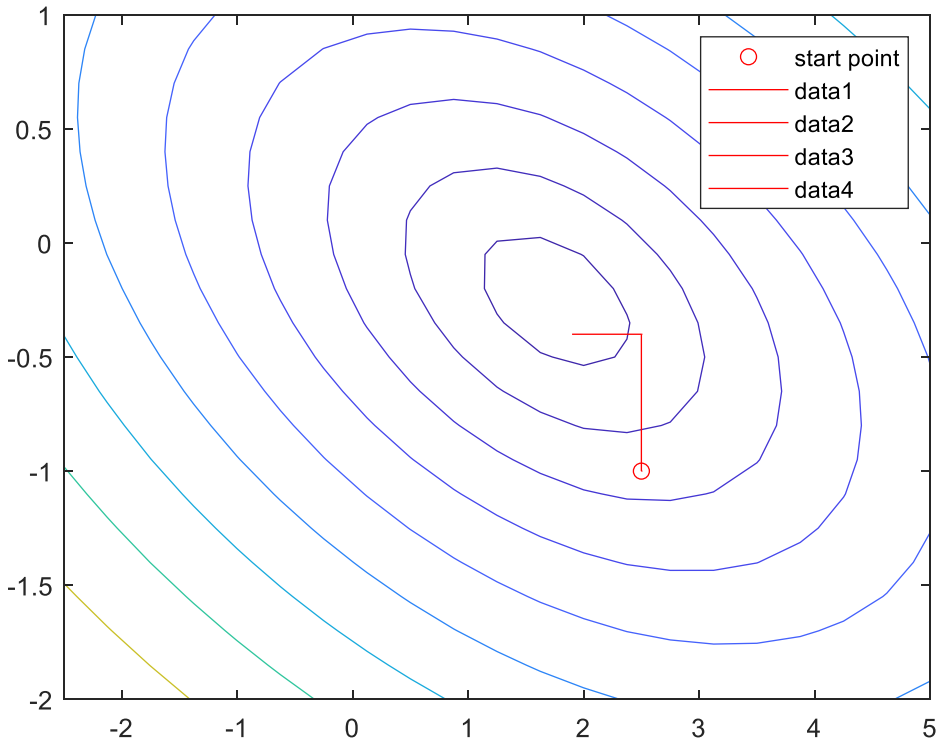
$$x_1 = 1.75$$

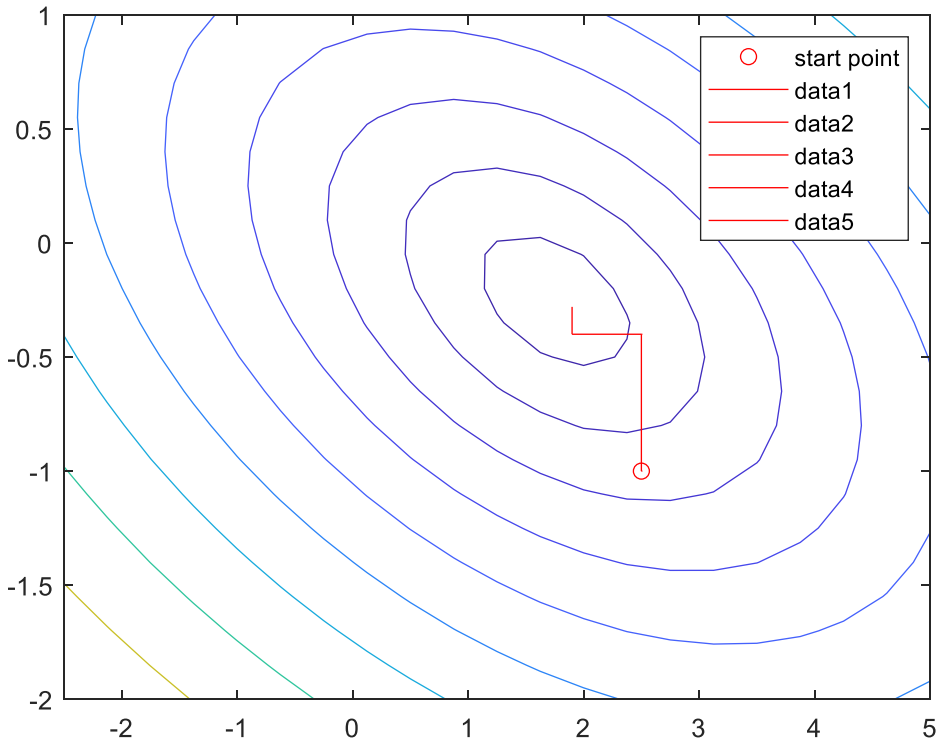
$$x_2 = -0.25$$

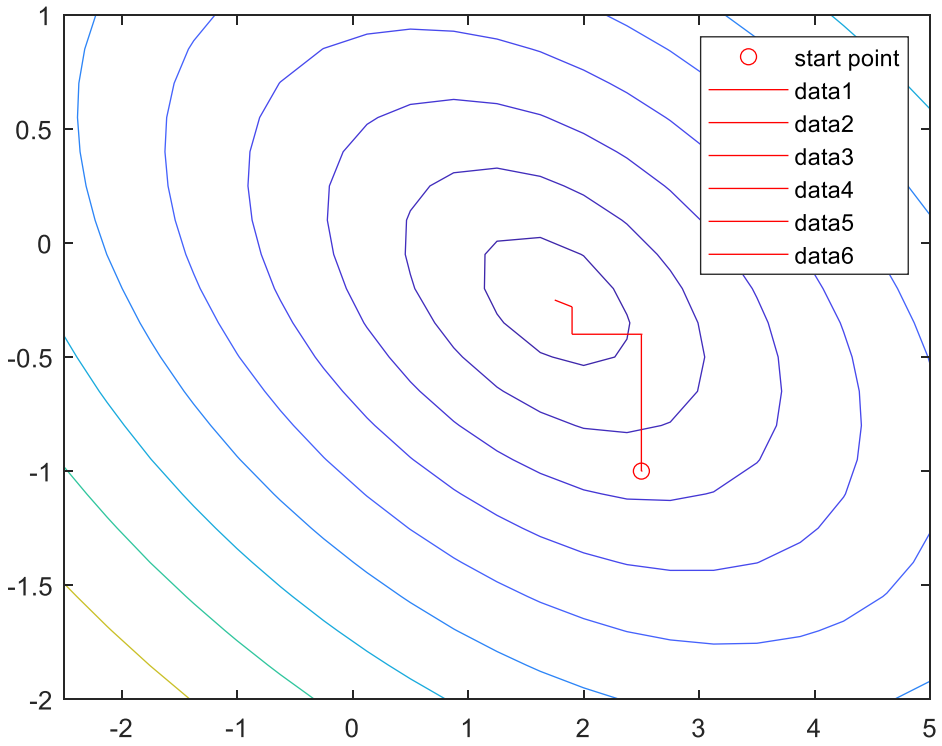



start point
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
3
4
5


start point
data1
data2
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
3
4
5


start point
data1
data2
data3
data4
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
3
4
5

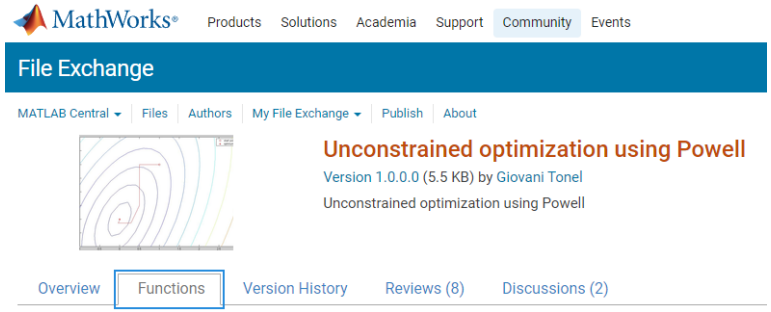

start point
data1
data2
data3
data4
data5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
3
4
5

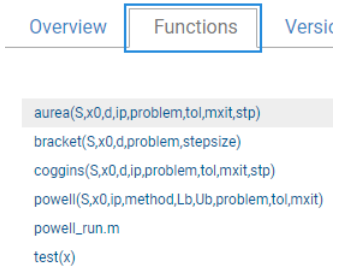


xo =

1.750000000000000
-0.250000000000000

Optymalizacja funkcji jednej zmiennej






```
133 -       if method,              % to see the linesearch plot, remove the two 0* below
134 -         [stepsize,xo,yo,nS1]=aurea(S,x,D(:,n),0*ips,problem,tol,mxit);
135 -         yo=yo*problem;
136 -       else
137 -         [stepsize,xo,yo,nS1]=coggins(S,x,D(:,n),0*ips,problem,tol,mxit);
138 -         yo=yo*problem;
139 -       end
```

```
help aurea
 Performs line search procedure for unconstrained optimization
 using golden section.

 [stepsize,xo,Ot,nS]=aurea(S,x0,d,ip,problem,tol,mxit,stp)

 S: objective function
 x0: initial point
 d: search direction vector
 ip: (0): no plot (default), (>0) plot figure ip with pause, (<0) plot figure ip
 problem: (-1): minimum (default), (1): maximum
 tol: tolerance (default = 1e-4)
 mxit: maximum number of iterations (default = 50*(1+4*~(ip>0)))
 stp: initial stepsize (default = 0.01*sqrt(d'*d))
 stepsize: optimal stepsize
 xo: optimal point in the search direction
 Ot: optimal value of S in the search direction
 nS: number of objective function evaluations
```

```
help coggins
 Performs line search procedure for unconstrained optimization
 using quadratic interpolation.

 [stepsize,xo,Ot,nS]=coggins(S,x0,d,ip,problem,tol,mxit)

 S: objective function
 x0: initial point
 d: search direction vector
 ip: (0): no plot (default), (>0) plot figure ip with pause, (<0) plot figure ip
 problem: (-1): minimum (default), (1): maximum
 tol: tolerance (default = 1e-4)
 mxit: maximum number of iterations (default = 50*(1+4*~(ip>0)))
 stp: initial stepsize (default = 0.01*sqrt(d'*d))
 stepsize: optimal stepsize
 xo: optimal point in the search direction
 Ot: optimal value of S in the search direction
 nS: number of objective function evaluations
```

Układ regulacji ekstremalnej

Układ regulacji ekstremalnej – układ regulacji, w którym regulacja przebiega tak aby wielkości regulowane przybrały wartości ekstremalne (maksymalne lub minimalne).

W układach takich steruje się obiektami, których **charakterystyki statyczne** (na wykresach typu wejście-wejście przedstawiane jako krzywe np. paraboliczne) posiadają maksima i minima.

Z pozoru sterowanie ekstremalne wydaje się prostym zadaniem (należy utrzymać wielkość wejściową na takim poziomie, dla którego wartość wyjściowa przyjmuje ekstremum) jednak **charakterystyki statyczne zwykle podlegają trudno mierzalnym zmianom w czasie** (na przykład skutkiem działania zakłóceń) i raz dobrane nastawy regulatora (wartość zadana) nie mają zastosowania w całym zakresie zmienności parametrów regulowanego obiektu. Dlatego trzeba zmieniać wartość zadaną zależnie od zmieniających się parametrów regulowanego obiektu tak aby uzyskać cel. Przy tym **poszukiwanie ekstremum odbywa się bezpośrednio na obiekcie** (co odróżnia takie układy od układów adaptacyjnych gdzie poszukiwanie ekstremum odbywa się na modelu).

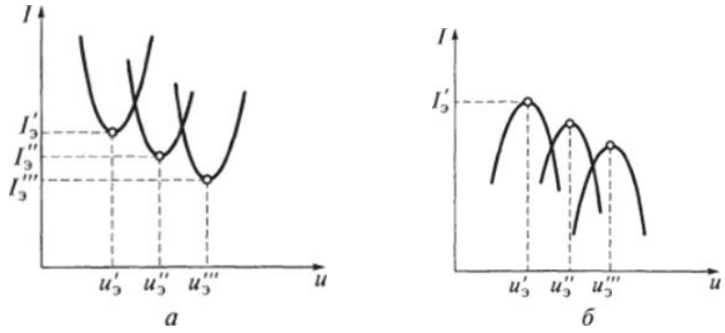
Układy regulacji ekstremalnej uzyskać można więc poprzez uzupełnienie zasadniczego układu regulacji członem optymalizującym. Zadanie takiego członu polega na automatycznym doborze wartości zadanej tak aby uzyskana wielkość sygnału sterującego zapewniała osiągnięcie ekstremum wartości wielkości regulowanej lub innej wielkości określonej przez wskaźnik jakości - określający na przykład dokładność, koszt lub sprawność (zob. też kryterium sterowania).

Ekstremum kryterium jakości

$$I = I(u_1, \ldots, u_n).$$

$$\frac{\partial I}{\partial u_1} = 0, \quad \frac{\partial I}{\partial u_2} = 0, \quad \ldots, \quad \frac{\partial I}{\partial u_n} = 0$$

$$\text{grad} I = \sum_i \bar{e}_i \frac{\partial I}{\partial u_i} = 0,$$



$$\frac{\partial I}{\partial u_i} = \frac{I(u_1, \ldots, u_i + \Delta u_i, \ldots, u_n) - I(u_1, \ldots, u_n)}{\Delta u_i} = \frac{\Delta I_i}{\Delta u_i} \quad (i = 1, 2, \ldots, n).$$

Pochodna wg czasu

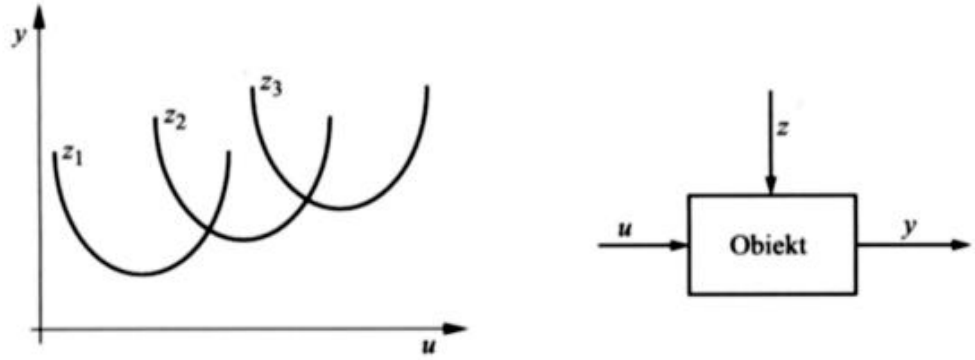
$$\dot{I} = \frac{\partial I}{\partial u_1}\dot{u}_1 + \frac{\partial I}{\partial u_2}\dot{u}_2 + \ldots + \frac{\partial I}{\partial u_n}\dot{u}_n$$

Zadania układów sterowania

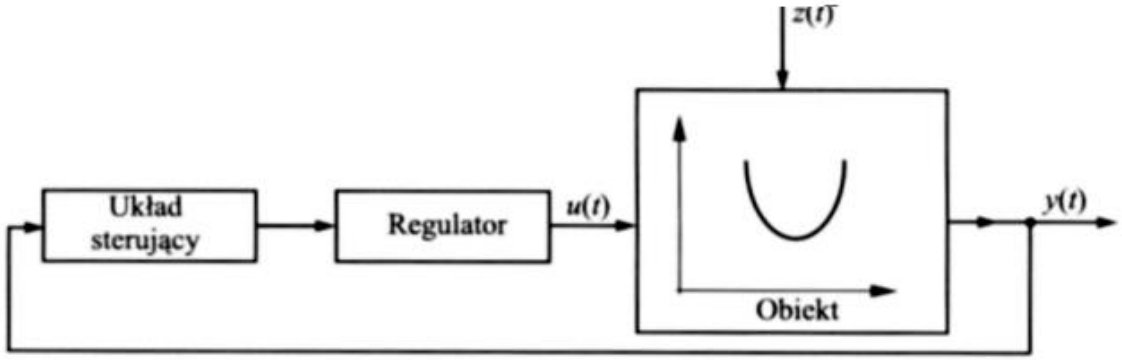
Rodzaje sterowania ze względu na charakter sygnału wartości zadanej:

- **Sterowanie stałowartościowe** (stabilizacja) ; $y_0(t)=const$;(przykłady: regulacja prędkości turbiny, temperatury pieca, prądu spawania)
- **Sterowanie programowe;** $y_0(t)=f(t)$, gdzie f(t) jest z góry znana; przykłady: sterowanie silnikiem pralki automatycznej, maszyny wytwórcze, obrabiarki sterowane numerycznie
- **Sterowanie nadążne ;** $y_0(t)=f(t)$, ale f(t) nie jest z góry znana, przykłady: śledzenie obiektu latającego (samolotu, rakiety), przez radar, system kierowania bateria przeciwlotniczą
- **Sterowanie ekstremalne;** (brak wartości zadanej); układ poszukuje punktu pracy optymalnego ze względu na minimum strat, maksimum wydajności itp.

Układ regulacji ekstremalnej

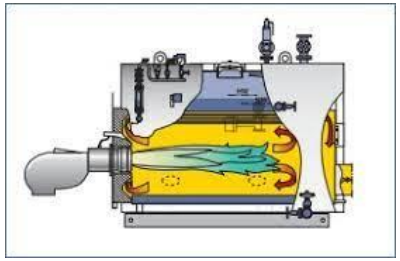


Charakterystyki statyczne obiektu regulacji nadającego się do sterowania w układzie regulacji ekstremalnej



Schemat blokowy układu regulacji ekstremalnej

Przykłady



- Układ regulacji procesu spalania (paliwa) w silniku spalinowym
- Układ regulacji procesu spalania gazu w komorze spalania pieca gazowego - tu charakterystyka statyczna (wykres temperatury w zależności od strumienia powietrza) zależy od strumienia gazu i różnej wartości opałowej gazu
- Układ regulacji stosowany w radioodbiornikach do automatycznego strojenia obwodów rezonansowych
- Kocioł okrętowy lub piec przemysłowy, do którego wprowadza się **powietrze i paliwo**. Dąży się do **maksymalizacji temperatury w piecu**, przy czym zbyt mała lub zbyt duża ilość powietrza powoduje niewykorzystanie maksymalnych możliwości palnika, a także warunki spalania zmieniają się w razie zmiany wartości opałowej paliwa, temperatury powietrza i obciążenia cieplnego pieca. Wówczas **sterowanie ekstremalne polega na odpowiednim określeniu zmiennych sterujących (dopływu paliwa i powietrza**), przy których zostanie osiągnięta wartość ekstremalna wskaźnika jakości (np. temperatury) w stanie ustalonym w danej sytuacji przy nie znanych zakłóceniach oraz wystąpi utrzymanie stanu obiektu zapewniającego zachowanie wartości ekstremalnej.

Przykład

https://www.mathworks.com/help/slcontrol/ug/esc-for-reference-model-tracking-of-uncertain-dynamical-systems.html

Adaptive Gain Tuning for Uncertain Linear Systems

For this example, consider the following first-order linear system.

$$\dot{x}(t) = a_0 x(t) + b_0 u(t)$$

Here, $x(t)$ and $u(t)$ are the state and control input of the system, respectively. The constants a_0 and b_0 are unknown.

The goal of this example is to track the performance of the following reference plant model, which defines the required transient and steady-state behavior.

$$\dot{x}_{ref}(t) = a^* x_{ref}(t) + b^* r(t)$$

Here, $x_{ref}(t)$ is the state of the reference plant and $r(t)$ is the reference signal.

The aim of the control signal $u(t)$ is to make the states $x(t)$ of the uncertain system track the reference states $x_{ref}(t)$.

$$u(t) = Kx(t) - Kr(t)$$

The designed controller contains a feedback term, $Kx(t)$, and a feedforward term, $-Kr(t)$.

Substitute this control signal into the unknown linear system dynamics.

$$\dot{x}(t) = a_0 x(t) + b_0(Kx(t) - Kr(t))$$

You can rewrite this expression as shown in the following equation.

$$\dot{x}(t) = (a_0 + b_0 K)x(t) - b_0 K r(t)$$

In the ideal case, if the coefficients a_0 and b_0 of the nominal system dynamics are known, then you can determine the controller gain K using pole-placement techniques. Doing so produces the following matching condition.

$$a_0 + b_0 K = a^*, b_0 K = b^*$$

When you use a single gain value as both the feedforward and feedback gain, this matching condition might not be satisfied for all the possible values of a_0 and b_0. For a more general solution, you can tune two different gain values (multiparameter tuning).

For this example, use the following unknown system and reference dynamics.

$$\dot{x}(t) = -1x(t) + u(t)$$

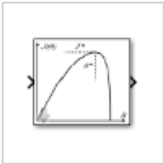
$$\dot{x}_{ref}(t) = -3x_{ref}(t) + 2r(t)$$

In this case, the ideal control gain is $K = -2$.

Extremum Seeking Control

Compute controller parameters in real time by maximizing objective function

Since R2021a



Libraries:
Simulink Control Design

Extremum Seeking Control Adaptive Gain Tuning

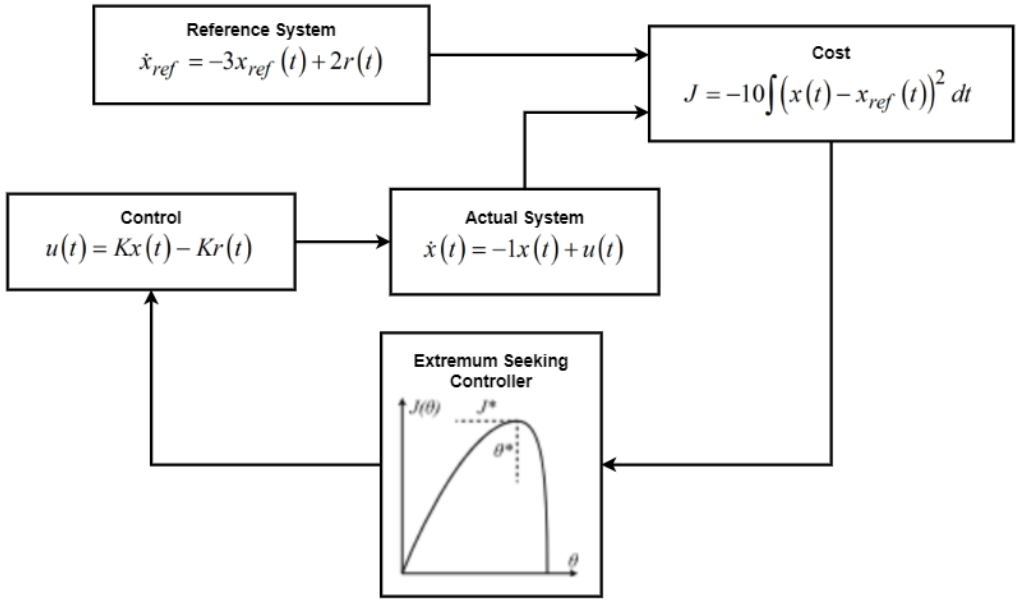
To implement an extremum seeking control (ESC) approach to the preceding problem, you define an objective function, which the ESC controller then maximizes to find the controller gain K.

For this example, use the following objective function.

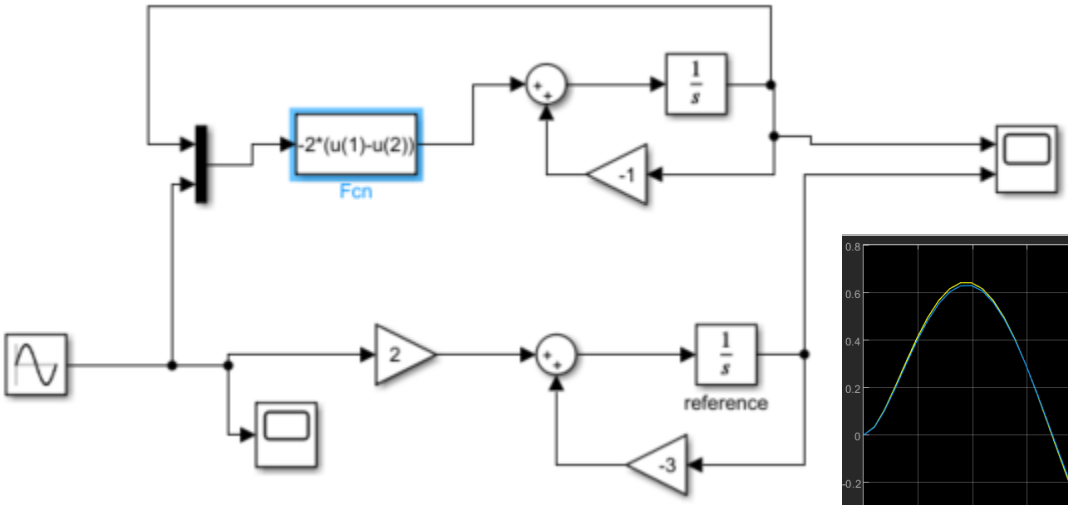
$$J = -10 \int (x(t) - x_{ref}(t))^2 dt$$

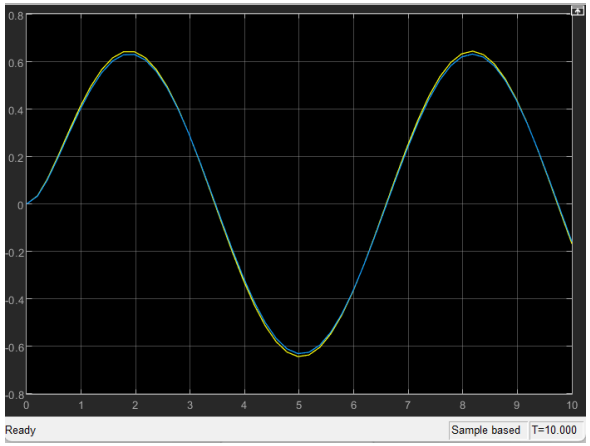
The following figure shows the setup for extremum seeking control.

- The cost function is computed from the outputs of the reference system and the actual system.
- The extremum seeking controller updates the gain parameter.
- The control action is updated using the new gain value.
- This control action is applied to the actual system.

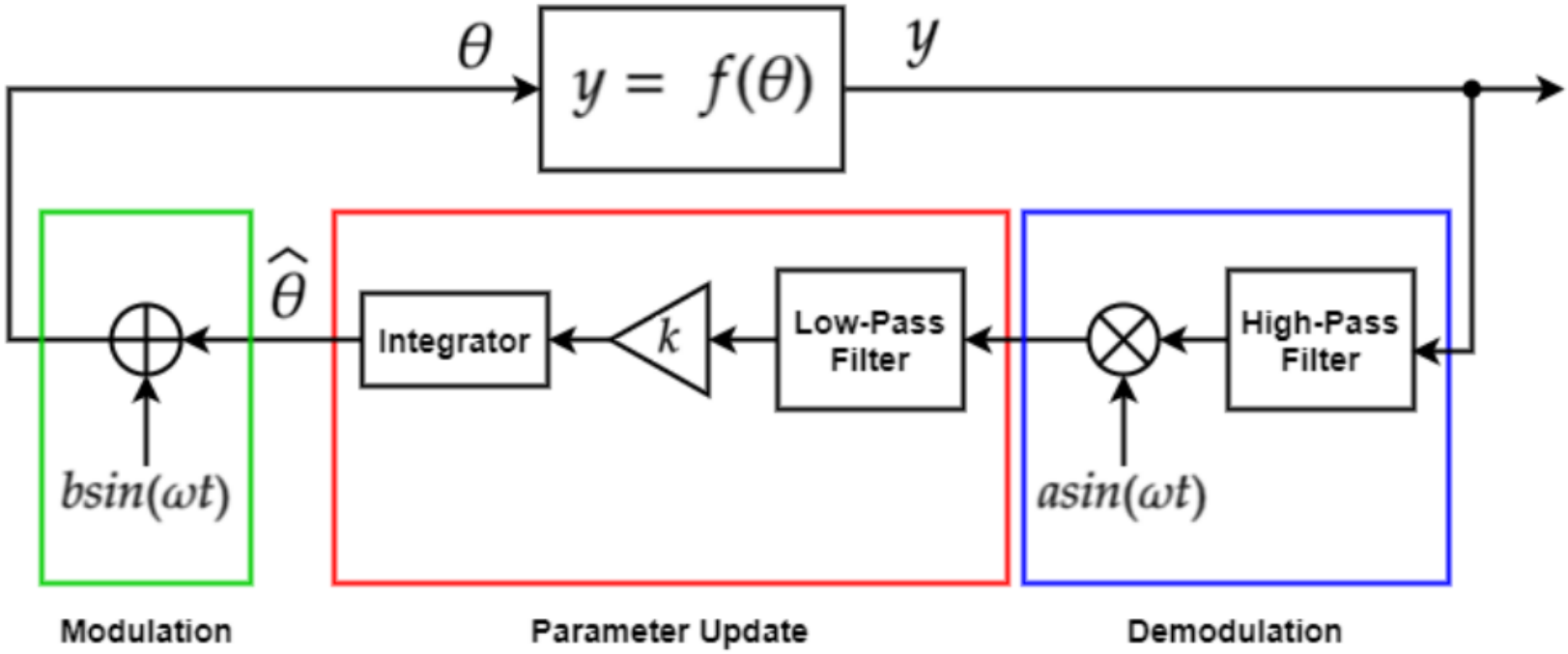


extremumControl



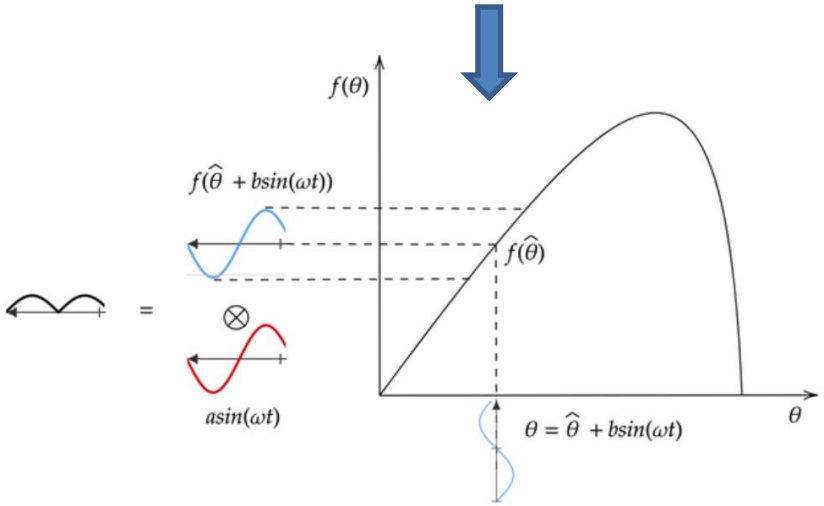


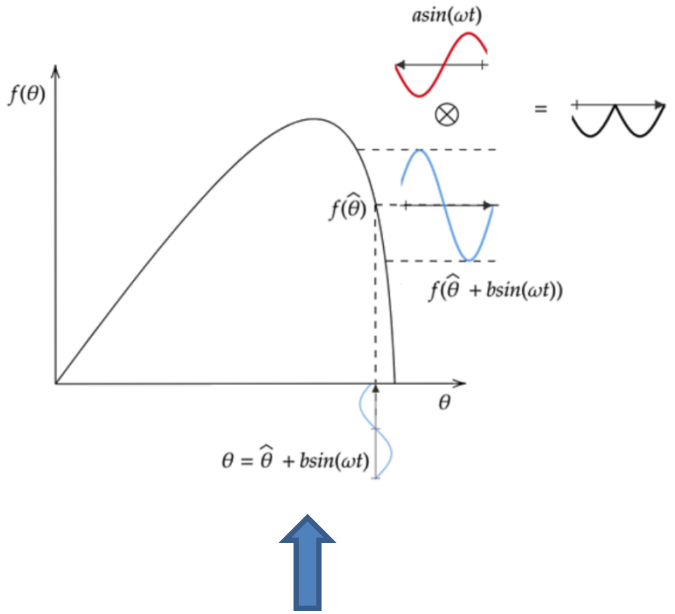
Optymalizacja statyczna



- $\hat{\theta}$ is the estimated parameter value.
- θ is the modulation signal
- $y = f(\theta)$ is the function output being maximized, that is, the objective function.
- ω is the forcing frequency of the modulation and demodulation signals.
- $b \cdot \sin(\omega t)$ is the modulation signal.
- $a \cdot \sin(\omega t)$ is the demodulation signal.
- k is the learning rate.

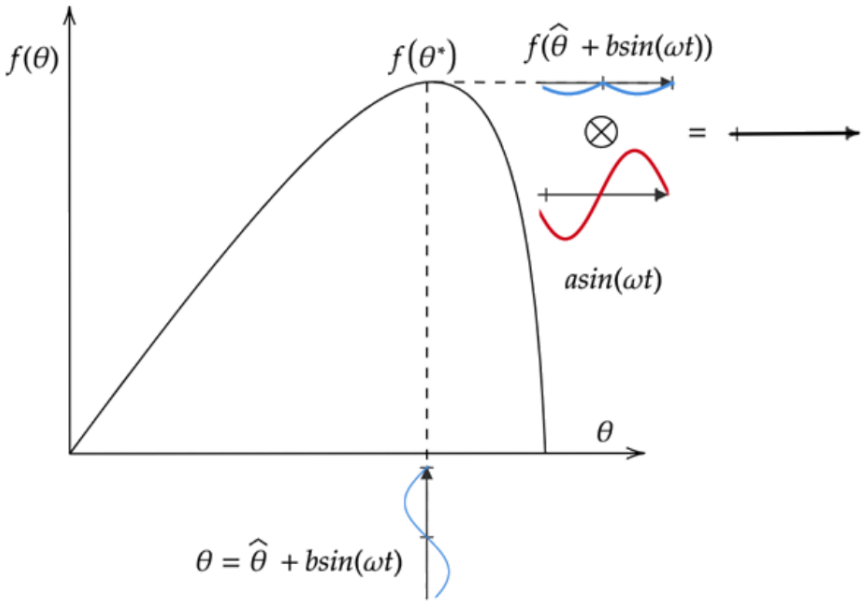
The following figure demonstrates extremum seeking for an *increasing portion* of the objective function curve. The modulated signal θ is the sum of the current estimated parameter and the modulation signal. Applying $f(\theta)$ produces a perturbed objective function with the same phase as the modulation signal. Multiplying the perturbed objective function by the demodulation signal produces a positive signal. Integrating this signal increases the value of θ, which moves it closer to the peak of the objective function.



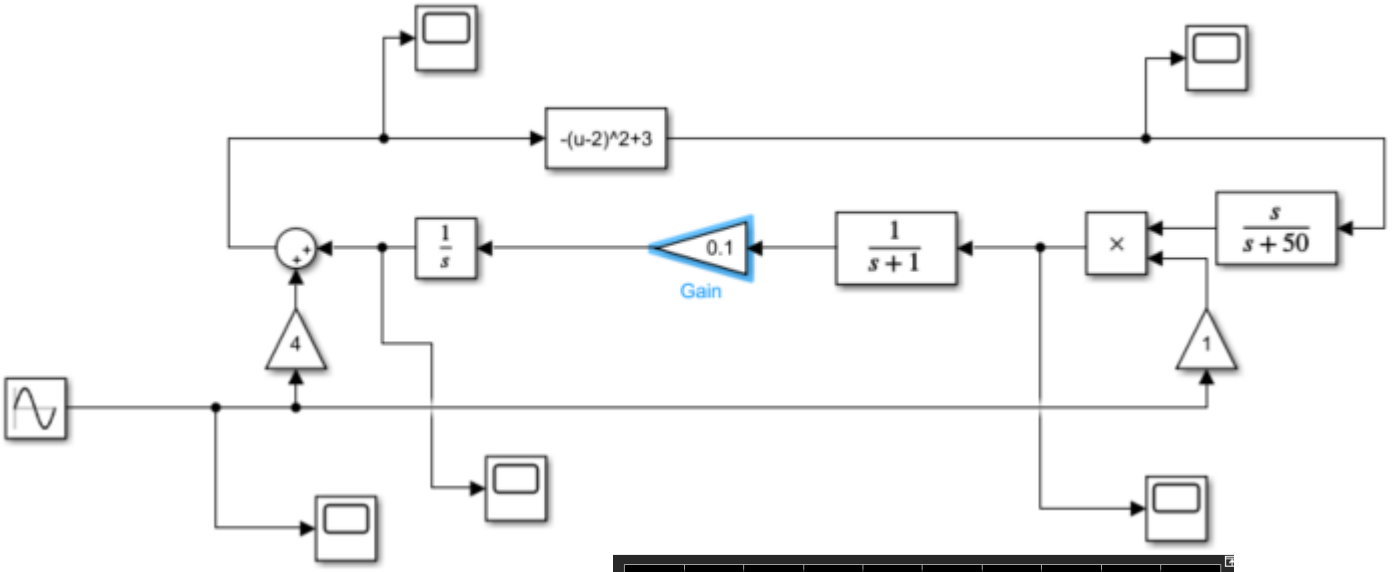


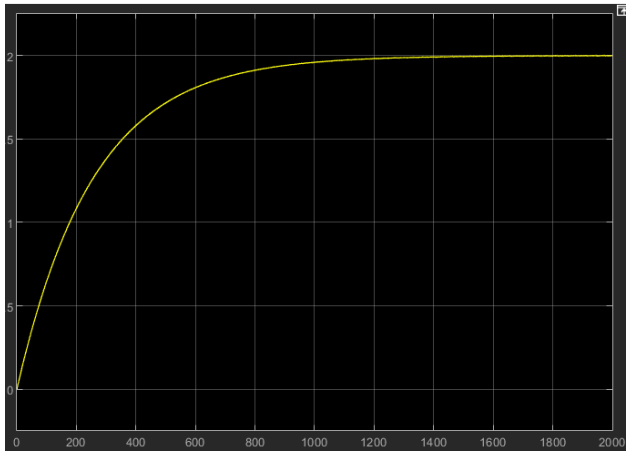
The following figure demonstrates extremum seeking for a *decreasing portion* of the objective function curve. In this case, applying $f(\theta)$ produces a perturbed objective function that is 180 degrees out of phase from the modulation signal. Multiplying by the demodulation signal produces a negative signal. Integrating this signal decreases the value of θ, which moves it closer to the peak of the objective function.

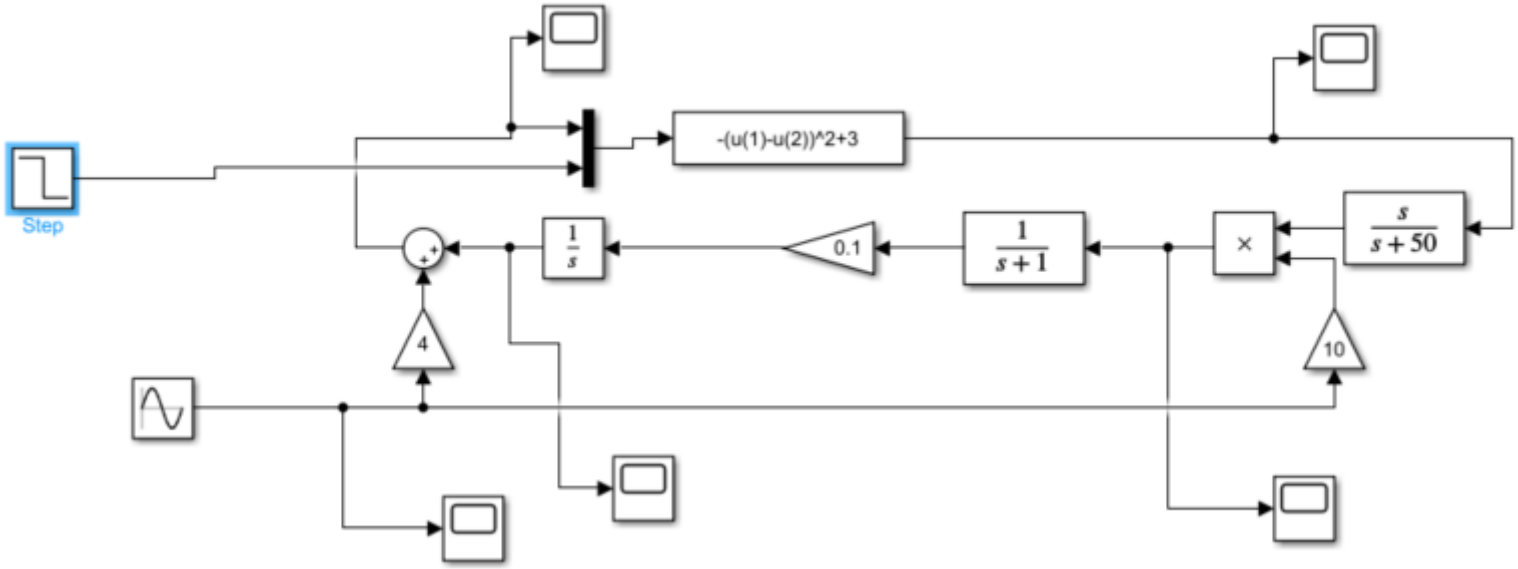
The following figure demonstrates extremum seeking for a *flat portion* of the objective function curve, that is, a portion of the curve near the maximum. In this case, applying $f(\theta)$ produces a near-zero perturbed objective function. Multiplying by the demodulation signal and integrating this signal does not significantly change the value of θ, which is already near its optimum value θ^*.



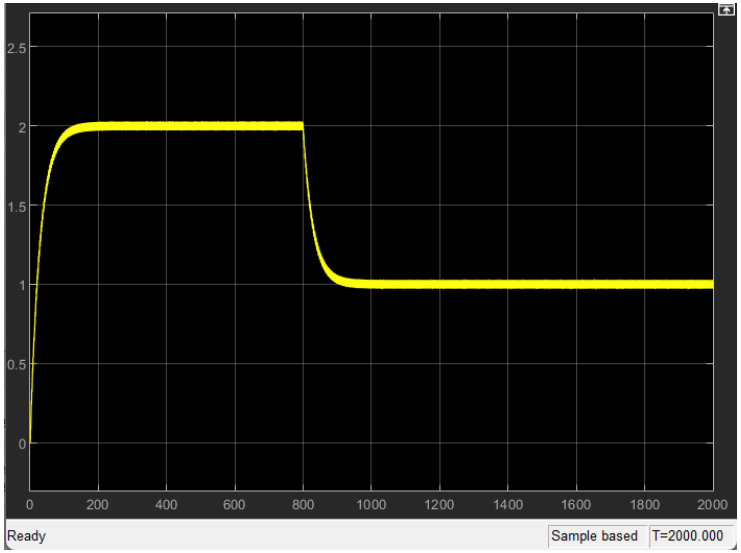
Przykład


-(u-2)^2+3
1/s
0.1
Gain
1/(s+1)
×
s/(s+50)
4
1

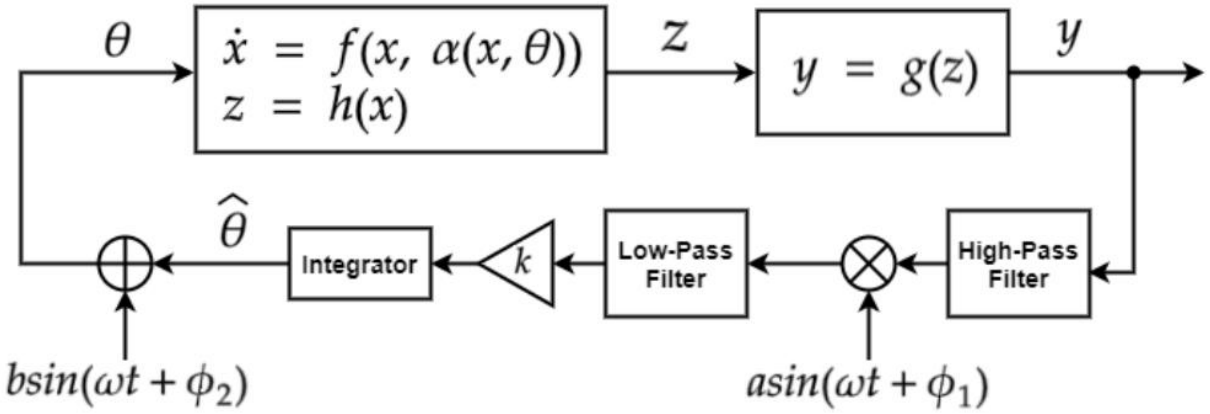

2
5
1
5
0
0
200
400
600
800
1000
1200
1400
1600
1800
2000



extrema2OS.slx



Optymalizacja dynamiczna



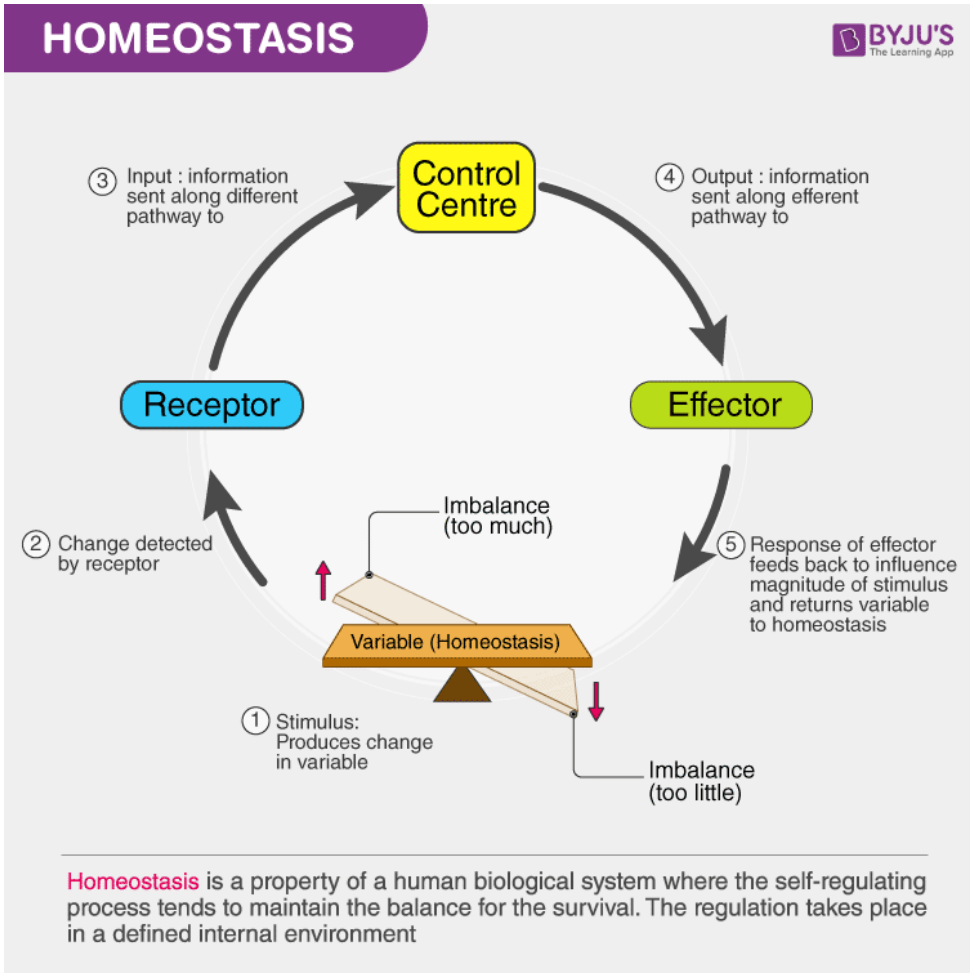
- $\dot{x} = f(x, \alpha(x, \theta))$ is the state function of the dynamic system.
- $z = h(x)$ is the output of the dynamic system.
- $y = g(z)$ is the objective function derived from the output of the dynamic system.
- ϕ_1 is the phase of the demodulation signal.
- ϕ_2 is the phase of the modulation signal.

ESC Design Guidelines

When designing an extremum-seeking controller, consider the following guidelines.

- Ensure that the system dynamics are on the fastest time scale, the forcing frequencies are on the medium time scale, and the filter cutoff frequencies are on the slowest time scale.
- Specify an amplitude for the demodulation signal that is much greater than the modulation signal amplitude ($a \gg b$).
- Select phase angles for the modulation and demodulation signals such that $\cos(\phi_1 - \phi_2) > 0$.
- When tuning multiple parameters, the forcing frequency for each tuning loop must be different.
- Try designing your system without high-pass and low-pass filters. If the performance is not satisfactory, you can then consider adding one or both filters.

Przykład. Utrzymanie homeostazy



Seminarium23012024-OS-b2.pptx

Metody gradientowe optymalizacji funkcji wielu zmiennych

- Metoda najszybszego spadku (ang. *steepest descent method*)
- Metoda Newtona
- Metoda Marquardta
- Metoda sprzężonego gradientu Fletcher-Reevesa
- Metody quasinewtonowskie:
  - Metoda Davidon-Fletcher-Powella (DFP)
  - Metoda Broyden-Fletcher-Goldfarb-Shannona (BFGS)

Metody gradientowe

Gradient w punkcie $x^{(t)}$ możemy przybliżyć numerycznie za pomocą następującej formuły:

$$\nabla f\left(x^{(t)}\right)=\begin{bmatrix}\dfrac{\partial f\left(x^{(t)}\right)}{\partial x_1}\\ \dfrac{\partial f\left(x^{(t)}\right)}{\partial x_2}\\ \mathrm{L}\\ \dfrac{\partial f\left(x^{(t)}\right)}{\partial x_N}\end{bmatrix}$$

$$\frac{\partial f\left(x^{(t)}\right)}{\partial x_i}=\frac{f\left(x_i^{(t)}+\Delta x_i^{(t)}\right)-f\left(x_i^{(t)}-\Delta x_i^{(t)}\right)}{2\Delta x_i^{(t)}}$$

```
z(x,y)
gx=∂z/∂x, gy=∂z/∂y.
gx = (z(x+hx,y)-z(x-hx,y))/2hx;
gy = (z(x,y+hy)-z(x,y-hy))/2hy.
```

numDerOS.m

Hesjan

Hesjan w punkcie $x^{(t)}$ natomiast liczymy następująco:

$$\nabla^2 f\left(x^{(t)}\right)=\begin{bmatrix} \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_1^2} & \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_2 \partial x_1} & \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_2^2} & \mathrm{L} & \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_2 \partial x_N} \\ \mathrm{L} & \mathrm{L} & \mathrm{L} & \mathrm{L} \\ \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_N \partial x_1} & \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_N \partial x_2} & \mathrm{L} & \frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_N^2} \end{bmatrix}$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j}=\frac{f(x+(e_i+e_j)h)-f(x+e_i h)-f(x+e_j h)+f(x)}{h^2}$$

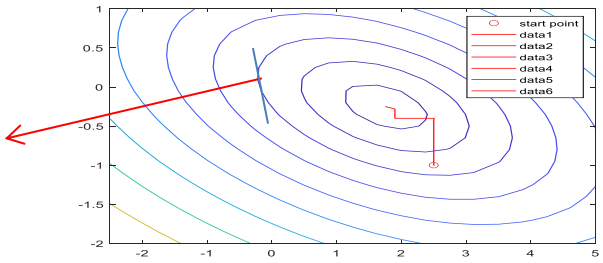
$$\frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_i^2}=\frac{f\left(x_i^{(t)}+\Delta x_i^{(t)}\right)-2f\left(x^{(t)}\right)+f\left(x_i^{(t)}-\Delta x_i^{(t)}\right)}{\left(\Delta x_i^{(t)}\right)^2}$$

$$\frac{\partial^2 f\left(x^{(t)}\right)}{\partial x_i \partial x_j}=\frac{\frac{\partial f\left(x_i^{(t)}+\Delta x_i^{(t)}\right)}{\partial x_j}-\frac{\partial f\left(x_i^{(t)}-\Delta x_i^{(t)}\right)}{\partial x_j}}{2\Delta x_i^{(t)}}$$

$$\frac{\partial f\left(x^{(t)}\right)}{\partial x_i}=\frac{f\left(x_i^{(t)}+\Delta x_i^{(t)}\right)-f\left(x_i^{(t)}-\Delta x_i^{(t)}\right)}{2\Delta x_i^{(t)}}$$

Gradient i Hesjan w punkcje ekstremum

$$f(x_1, x_2) = (x_1 + 3x_2 - 1)^2 + (x_1 - x_2 - 2)^2$$



x10=0;x20=0;
Df= [-6 -2]

$$\min f(x_1, x_2) = 2.5$$

$$x_1 = 1.75$$

$$x_2 = -0.25$$

numDerOS.m
powellOS.m

```
>> numDerOS
   1.0e-15 *

  -0.222614453595 28    0.216840434497101

   3.999999999999963   4.000000000000052
   4.000000000000052  19.999999999999989
```

```
>>  eig([4 4;4 20])
    3.0557
   20.9443
```

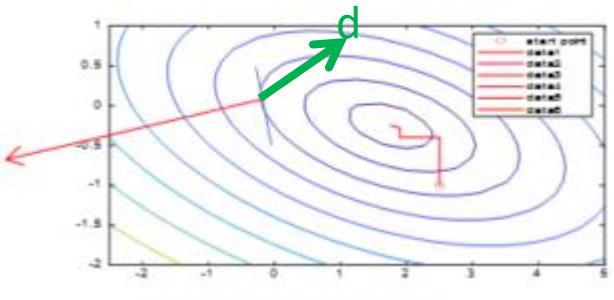
Obliczenia numeryczne

```
%num derivativs OS
dx1=0.01;
dx2=0.01;
%fun='pauel3d';
%fun='par3d';
fun='powellOS';

%x10=1;x20=1;
x10=1.75;x20=-0.25;

xt=[x10 x20];
fx=feval(fun,xt);
fpx1=feval(fun,[x10+dx1,x20]);
fpx2=feval(fun,[x10,x20+dx2]);
fmx1=feval(fun,[x10-dx1,x20]);
fmx2=feval(fun,[x10,x20-dx2]);
dfdx1=(fpx1-fmx1)./(2*dx1);
dfdx2=(fpx2-fmx2)./(2*dx2);
disp([dfdx1 dfdx2]);
%Hesjan
d2fdx1dx1=(fpx1-2*fx+fmx1)/(dx1^2);
d2fdx2dx2=(fpx2-2*fx+fmx2)/(dx2^2);
fpx1x2=feval(fun,[x10+dx1,x20+dx2]);
d2fdx1dx2=(fpx1x2-fpx1-fpx2+fx)/(dx1*dx2);
d2fdx2dx1=d2fdx1dx2;
disp([d2fdx1dx1 d2fdx1dx2;d2fdx2dx1 d2fdx2dx2]);
```

Kierunek najszybszego spadku funkcji

Ponieważ gradient jest kierunkiem najszybszego wzrostu, minus gradient jest kierunkiem najszybszego spadku funkcji. Kierunek poszukiwań (ang. search direction) $d^{(t)}$ jest kierunkiem spadku w punkcie $x^{(t)}$ jeśli w otoczeniu tego punktu spełniony jest następujący warunek:

$$\nabla f\left(x^{(t)}\right)\cdot d^{(t)}\leqslant 0$$



Oznacza to, że cosinus kąta między gradientem i kierunkiem poszukiwań jest większy niż 90°. Kierunek $d^{(t)}$ jest kierunkiem spadku, ponieważ w wyniku rozwinięcia f wokół $x^{(t)}$ otrzymujemy:

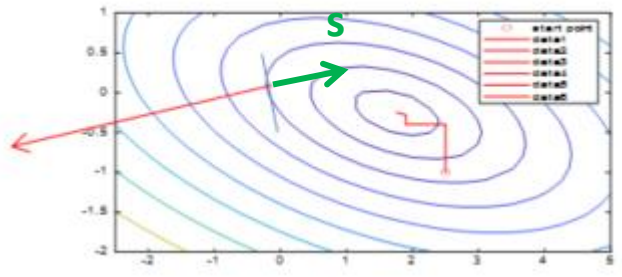
$$f\left(x^{(t+1)}\right)=f\left(x^{(t)}+\alpha d^{(t)}\right)=f\left(x^{(t)}\right)+\alpha\nabla f\left(x^{(t)}\right)\cdot d^{(t)}$$

$$x^{(k+1)}=x^{(k)}+\alpha^{(k)}s^{(k)}$$

Metoda Cauchy'ego najszybszego spadku

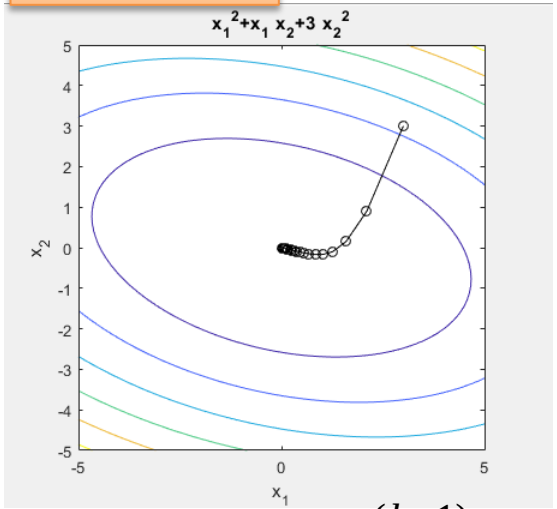
Kierunek poszukiwań w metodzie Cauchy'ego jest **kierunkiem najszybszego spadku**:

$$s^{(k)} = -\nabla f\left(x^{(k)}\right)$$

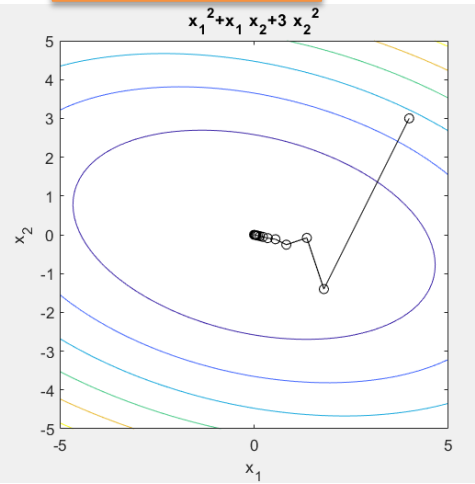


Grad_descent.m

alpha = 0.1;



alpha = 0.2;



$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} s^{(k)}$$

Algorytm

1) Wybierz maksymalną liczbę iteracji M, punkt początkowy oraz dwa parametry zakończenia ϵ_1 i ϵ_2 i ustal $k = 0$.

2) Oblicz $\nabla f(x^{(k)})$

3) Jeśli $||\nabla f(x^{(k)})|| \leqslant \epsilon_1$, **Zakończ;**
Jeśli $k \geqslant M$; **Zakończ;**
W przeciwnym razie idź do kroku 4).

$$s^{(k)} = -\nabla f\left(x^{(k)}\right)$$

4) Wykonaj poszukiwanie wzdłuż kierunku, żeby znaleźć $\alpha^{(k)}$ tak, aby $f(x^{(k+1)}) = f(x^{(k)} + \alpha^{(k)} s^{(k)})$ było minimalne. Jednym z kryteriów zakończenia jest $|\nabla f(x^{(k+1)}) \cdot \nabla f(x^{(k)})| \leqslant \epsilon_2$.

5) Jeśli $\frac{||x^{(k+1)} - x^{(k)}||}{||x^{(k)}||} \leqslant \epsilon_1$, **Zakończ;**
W przeciwnym przypadku ustal $k = k + 1$ i idź do kroku 2).

cauchyOS.m

$$f(x_1, x_2) = (x_1 + 3x_2 - 1)^2 + (x_1 - x_2 - 2)^2$$

```
>> syms x1 x2
>> y=(x1+3*x2-1)^2+(x1-x2-2)^2

y =

(x2 - x1 + 2)^2 + (x1 + 3*x2 - 1)^2

[diff(y,x1) diff(y,x2)]

ans =

[ 4*x1 + 4*x2 - 6, 4*x1 + 20*x2 - 2]
```

differentiate $(x1 - x2 - 2)^2 + (x1 + 3\,x2 - 1)^2$

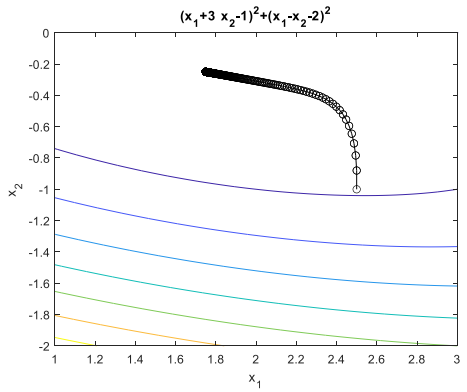
Partial derivatives:

$$\frac{\partial}{\partial x1}\left((x1 - x2 - 2)^2 + (x1 + 3\,x2 - 1)^2\right) = 4\,x1 + 4\,x2 - 6$$

$$\frac{\partial}{\partial x2}\left((x1 - x2 - 2)^2 + (x1 + 3\,x2 - 1)^2\right) = 4\,x1 + 20\,x2 - 2$$

grad_descent.m

$$\min f(x_1, x_2) = 2.5$$
$$x_1 = 1.75$$
$$x_2 = -0.25$$



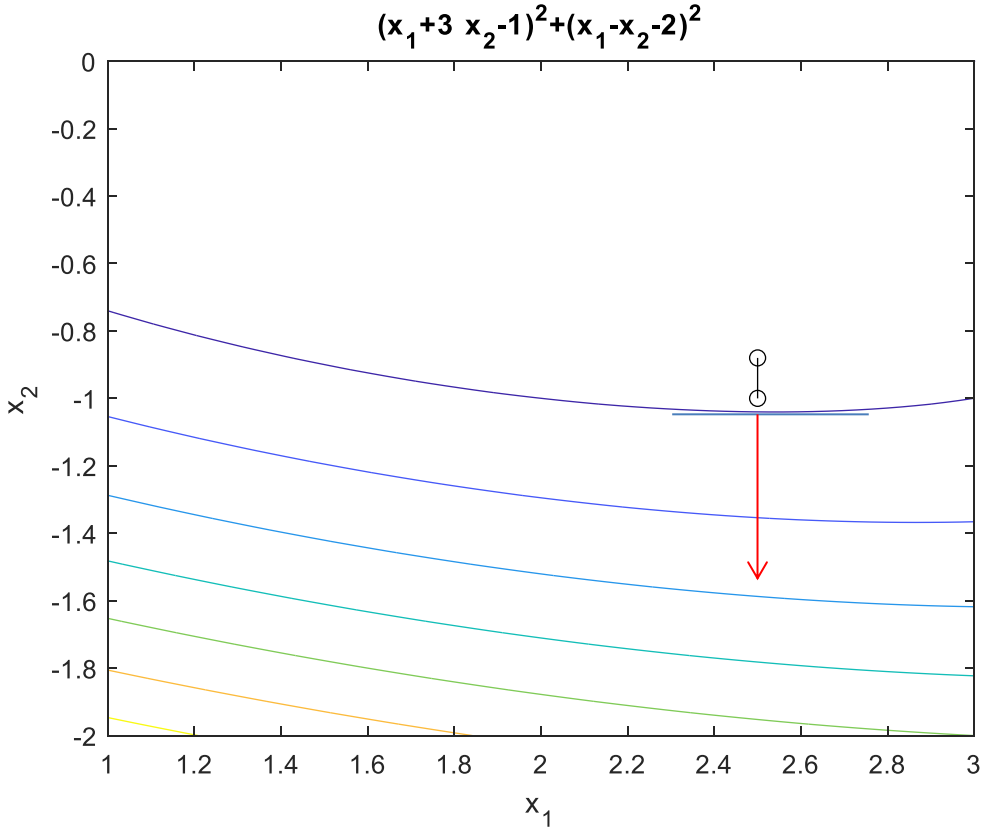
```
while and(gnorm>=tol, and(niter <= maxiter, dx >= dxmin))
    % calculate gradient:
    g = grad(x);
    gnorm = norm(g);
    % take step:
    xnew = x - alpha*g;
    % check step
    if ~isfinite(xnew)
        display(['Number of iterations: ' num2str(niter)])
        error('x is inf or NaN')
    end
    % plot current point
    plot([x(1) xnew(1)],[x(2) xnew(2)],'ko-')
    refresh
    % update termination metrics
    niter = niter + 1;
    dx = norm(xnew-x);
    x = xnew;

end
```

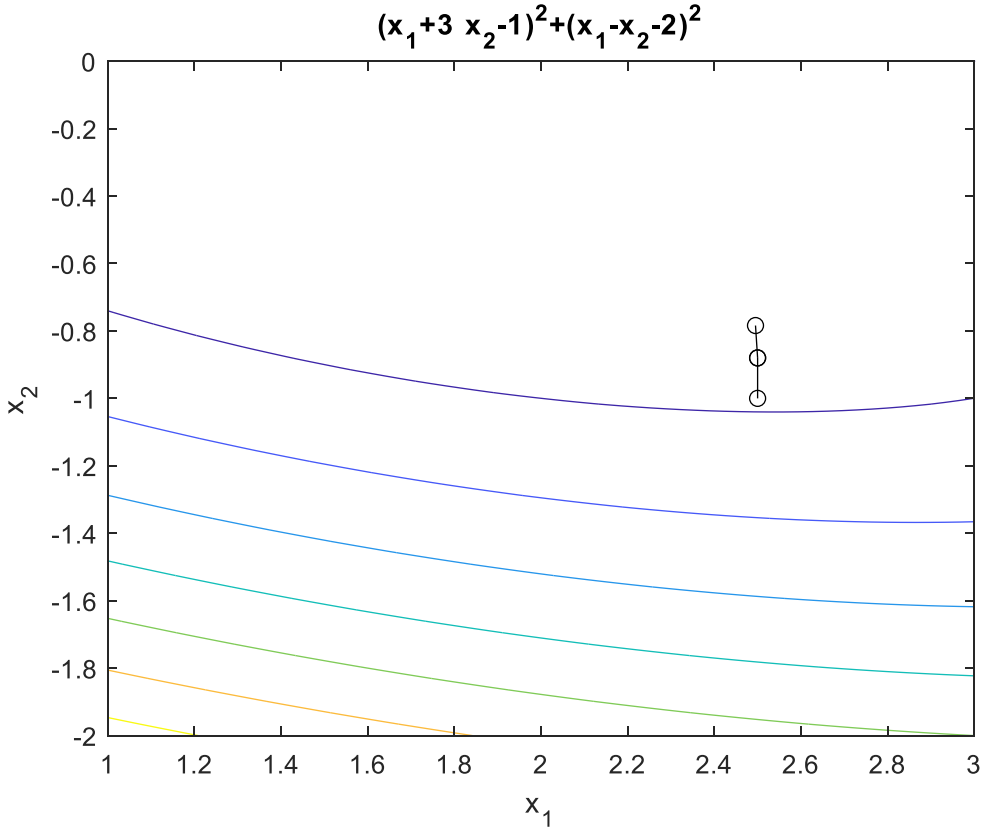
```
% define the gradient of the objective
function g = grad(x)
%g = [2*x(1) + x(2)
%       x(1) + 6*x(2)];
  g=[  4*x(1) + 4*x(2) - 6
  4*x(1) + 20*x(2) - 2
  ];
```

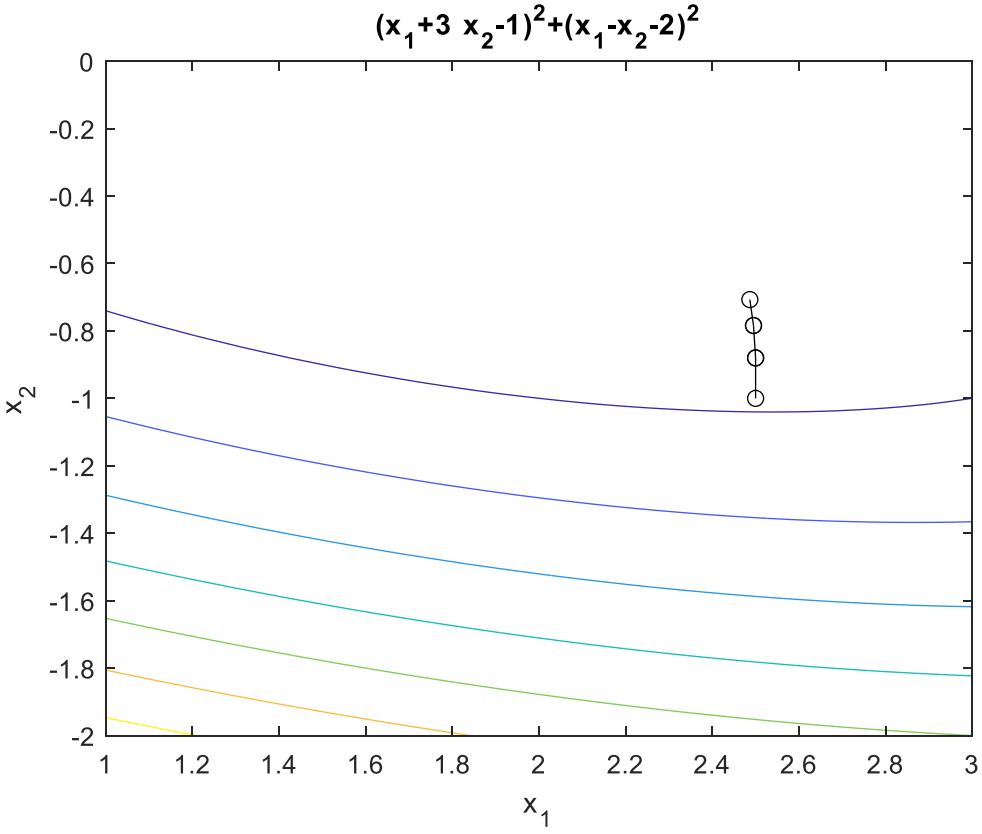
alpha = 0.01;

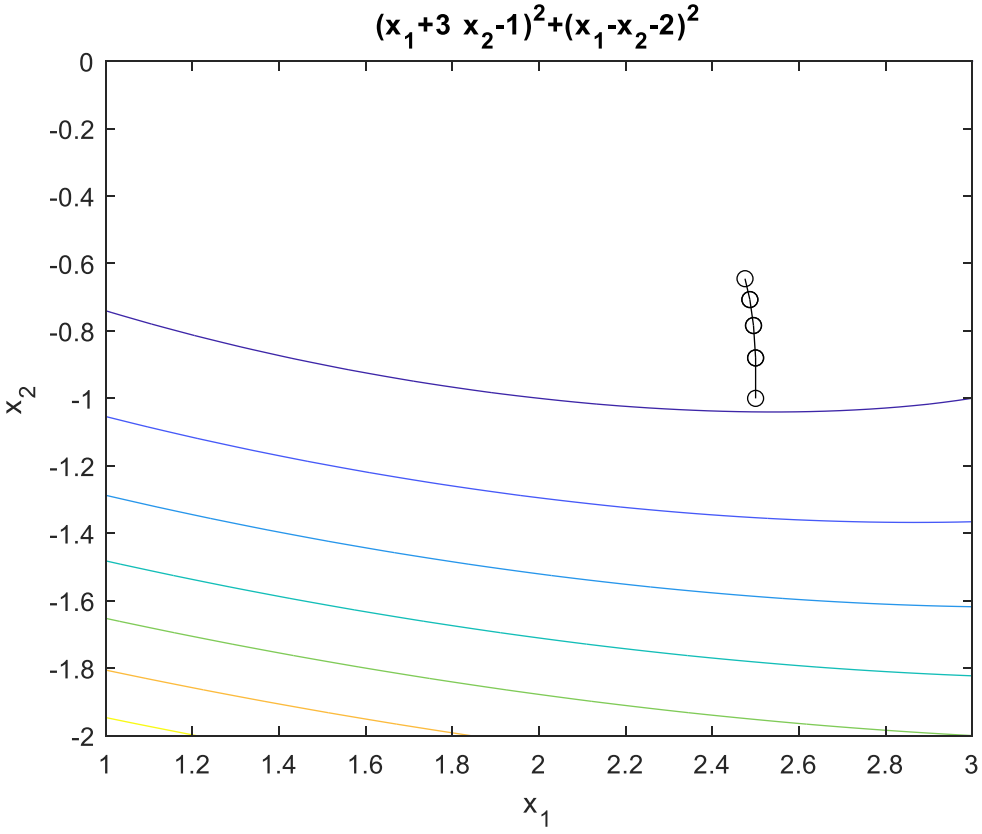
x		xnew		g	
2.5	-1	2.5	-0.88	0	-12
2.5	-0.88	2.4952	-0.784	0.48	-9.6
2.4952	-0.784	2.4868	-0.70701	0.8448	-7.6992
2.4868	-0.70701	2.4756	-0.64508	1.119	-6.1932
2.4756	-0.64508	2.4623	-0.59508	1.3219	-4.9993
2.4623	-0.59508	2.4477	-0.55456	1.469	-4.0523

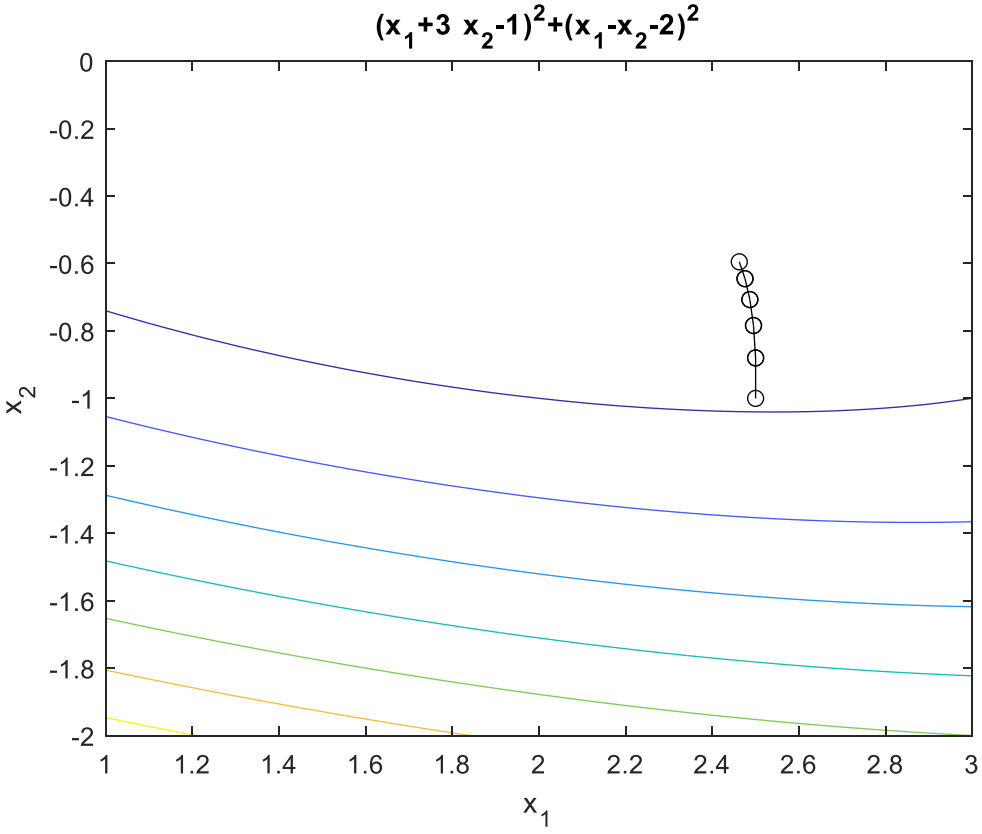
$$\min f(x_1, x_2) = 2.5$$
$$x_1 = 1.75$$
$$x_2 = -0.25$$

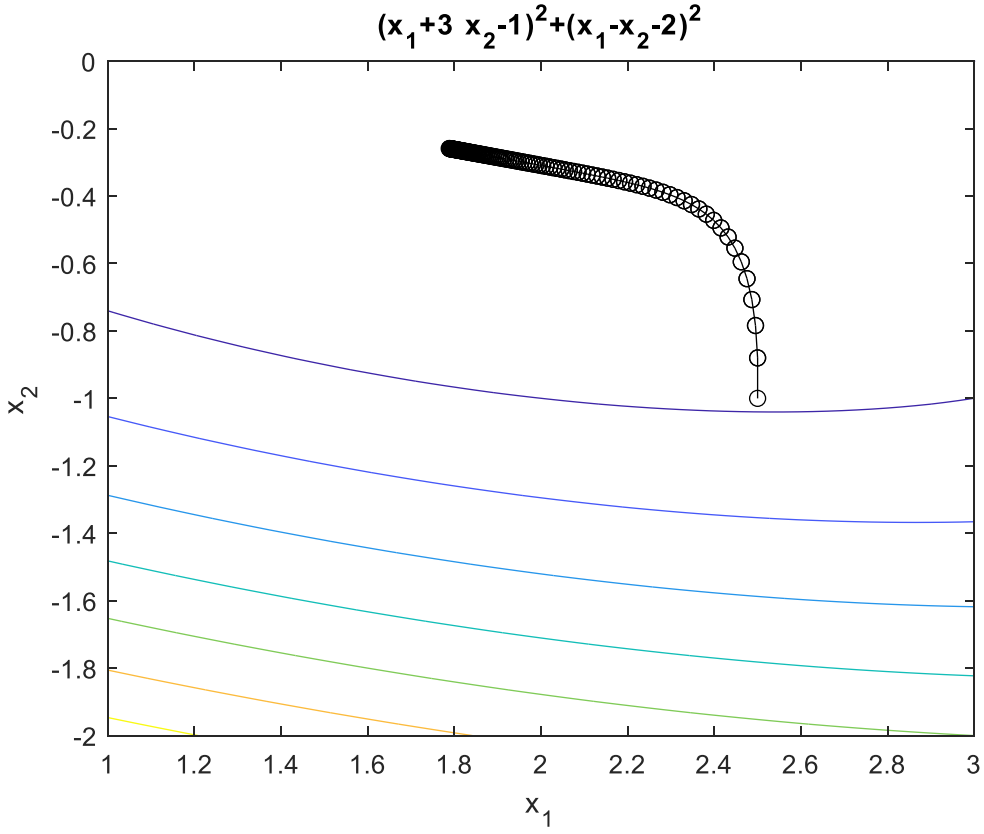

$(x_1+3\ x_2-1)^2+(x_1-x_2-2)^2$
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
x_2
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
x_1


$(x_1+3\ x_2-1)^2+(x_1-x_2-2)^2$
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
x_2
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
x_1

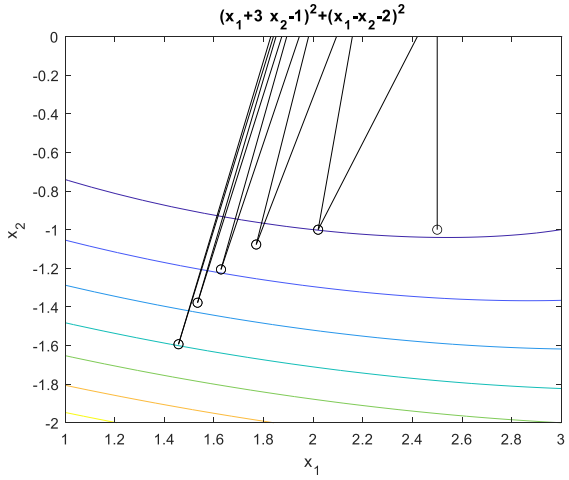

$(x_1+3\ x_2-1)^2+(x_1-x_2-2)^2$
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
x_2
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
x_1


$(x_1+3\ x_2-1)^2+(x_1-x_2-2)^2$
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
x_2
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
x_1


$(x_1+3\ x_2-1)^2+(x_1-x_2-2)^2$
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
x_2
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
x_1


$(x_1+3\ x_2-1)^2+(x_1-x_2-2)^2$
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
x_2
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
x_1

alfa=0.1



x		xnew		g	
2.5	-1	2.5	0.2	0	-12
2.5	0.2	2.02	-1	4.8	12
2.02	-1	2.212	0.392	-1.92	-13.92
2.212	0.392	1.7704	-1.0768	4.416	14.688
1.7704	-1.0768	2.093	0.56864	-3.2256	-16.454
2.093	0.56864	1.6283	-1.2058	4.6464	17.745
1.6283	-1.2058	2.0593	0.7545	-4.31	-19.603

Metoda Newtona-Raphsona

Rozwinięcie w szereg Taylora drugiego rzędu funkcji f wokół punktu $x^{(t)}$ przyjmuje następującą formę:

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

Warunek pierwszego rzędu na maksimum lokalne tej funkcji:

$$\nabla f\left(x^{(k)}\right)+\nabla^{2}f\left(x^{(k)}\right)\left(x-x^{(k)}\right)=0$$

Skąd mamy

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

$$s^{(k)}=-\left[\nabla^{2}f\left(x^{(k)}\right)\right]^{-1}\nabla f\left(x^{(k)}\right)$$

Metoda Cauchy'ego

$$s^{(k)}=-\nabla f\left(x^{(k)}\right)$$

Metoda Newtona-Raphsona(cd)

Generujemy iteracyjnie ciąg punktów $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$

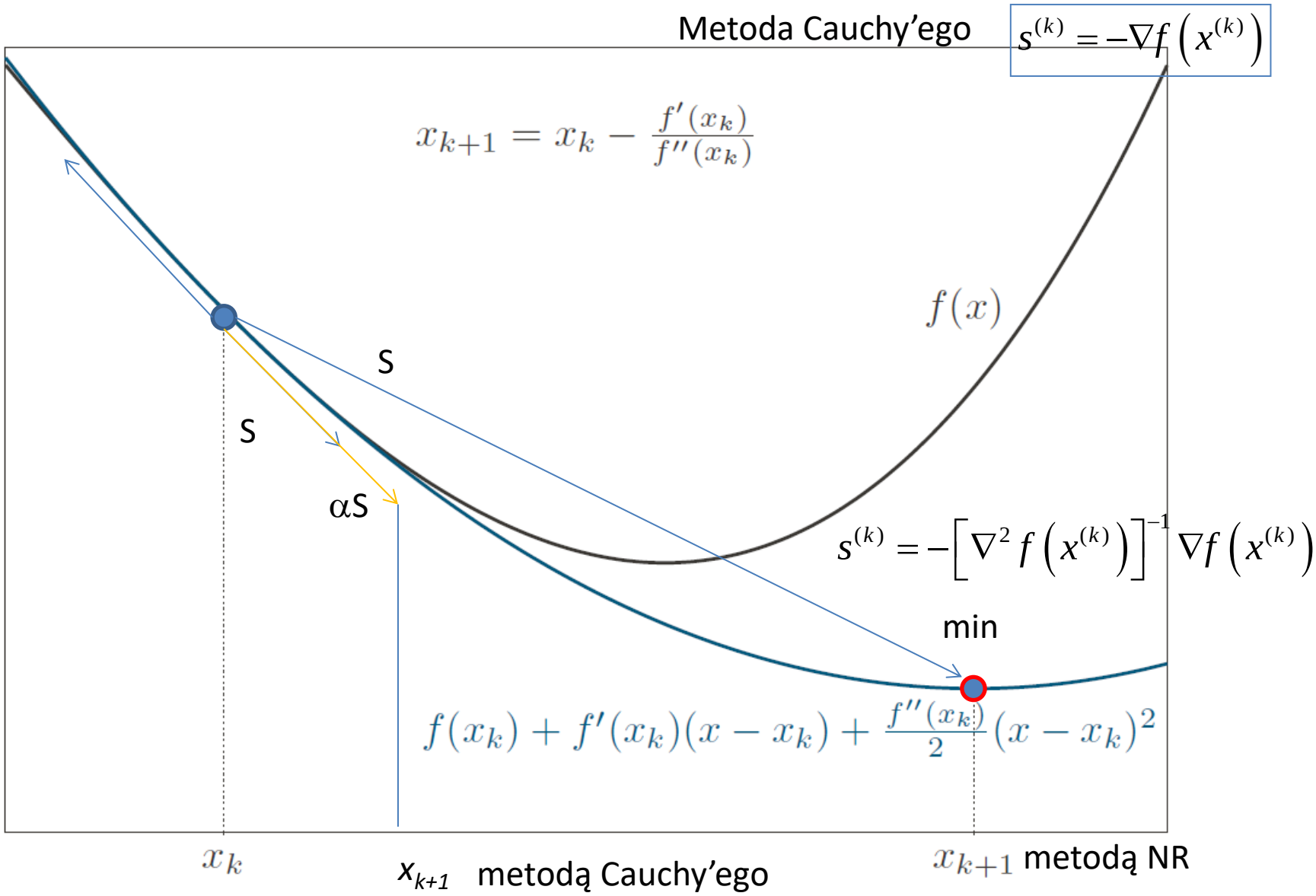
W każdej iteracji $k = 1, 2, \ldots$, przybliżamy funkcję $f(\boldsymbol{x})$ **wielomianem Taylora drugiego rzędu** w aktualnym punkcie $\boldsymbol{x}_k$:

$$P_2(\boldsymbol{x}; \boldsymbol{x}_k) \;=\; f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top (\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^\top \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k)$$

Założenie: hesjan $\nabla^2 f(\boldsymbol{x}_k)$ jest **dodatnio określony**

Jako kolejny punkt $\boldsymbol{x}_{k+1}$ bierzemy punkt minimalizujący $P_2(\boldsymbol{x}; \boldsymbol{x}_k)$

Metoda Newtona-Raphsona(cd)



Minimum funkcji kwadratowej

Przypomnienie: Minimum funkcji kwadratowej $x^\top Ax + b^\top x + c$ z dodatnio określoną macierzą A ma postać $x^* = -\frac{1}{2}A^{-1}b$

Wniosek: Minimum funkcji kwadratowej:

$$F(v) = \frac{1}{2}v^\top \nabla^2 f(x_k)v + \nabla f(x_k)^\top v + f(x_k)$$

jest w punkcie $v = -(\nabla^2 f(x_k))^{-1}\nabla f(x_k)$

Wniosek: Biorąc $v = x - x_k$, minimum wielomianu:

$$P_2(x; x_k) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2}(x - x_k)^\top \nabla^2 f(x_k)(x - x_k)$$

jest w punkcie x spełniającym:

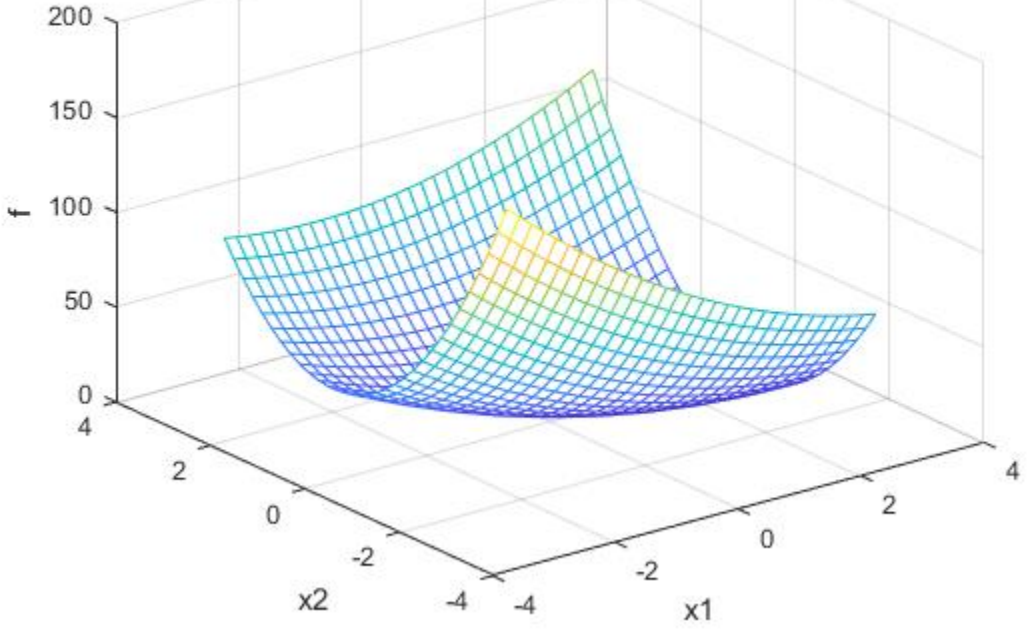
$$x - x_k = -(\nabla^2 f(x_k))^{-1}\nabla f(x_k)$$

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1}\nabla f(x_k)$$

NewtonOS1krok.m

Twierdzenie

$$f(x_1, x_2) = (x_1 + 3x_2 - 1)^2 + (x_1 - x_2 - 2)^2$$



Metoda Newtona-Rapshona znajduje minimum funkcji kwadratowej:

$$f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + c$$

w **jednym kroku**, niezależnie od punktu inicjalizacji!

Dowód

Rozważmy minimalizację funkcji funkcji kwadratowej z dodatnio określoną macierzą $\boldsymbol{A}$:

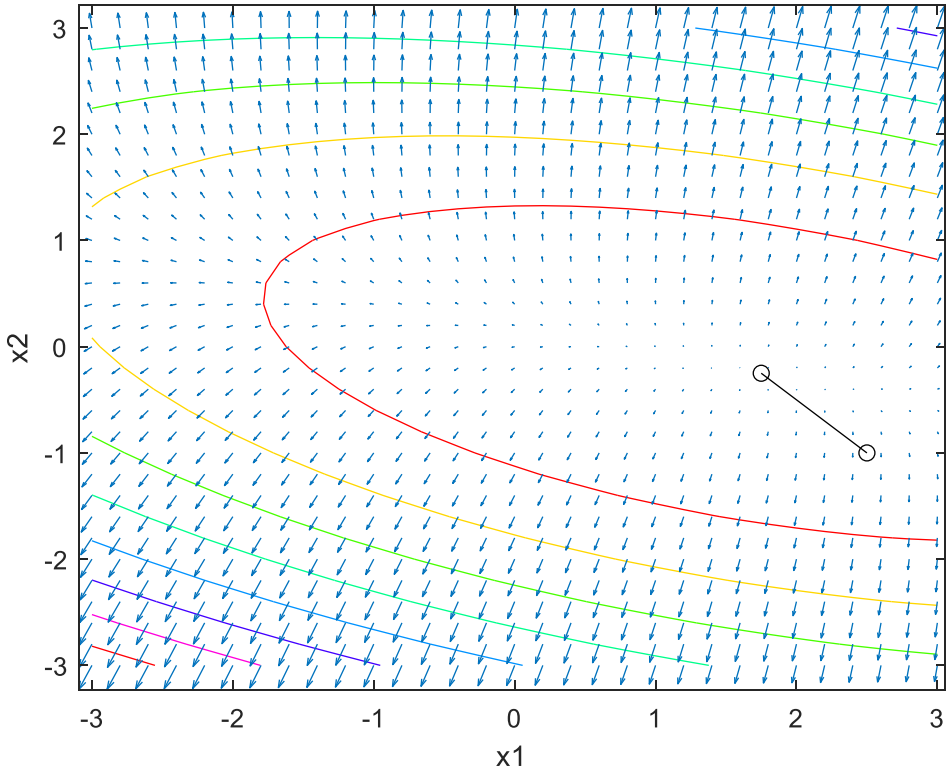
$$f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{b} + c,$$

$$\nabla f(\boldsymbol{x}) = 2\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \qquad \nabla^2 f(\boldsymbol{x}) = 2\boldsymbol{A}, \qquad (\nabla^2 f(\boldsymbol{x}))^{-1} = \frac{1}{2}\boldsymbol{A}^{-1}$$

Rozpoczynamy od dowolnego punktu $\boldsymbol{x}_1$ i wybieramy punkt $\boldsymbol{x}_2$ jako:

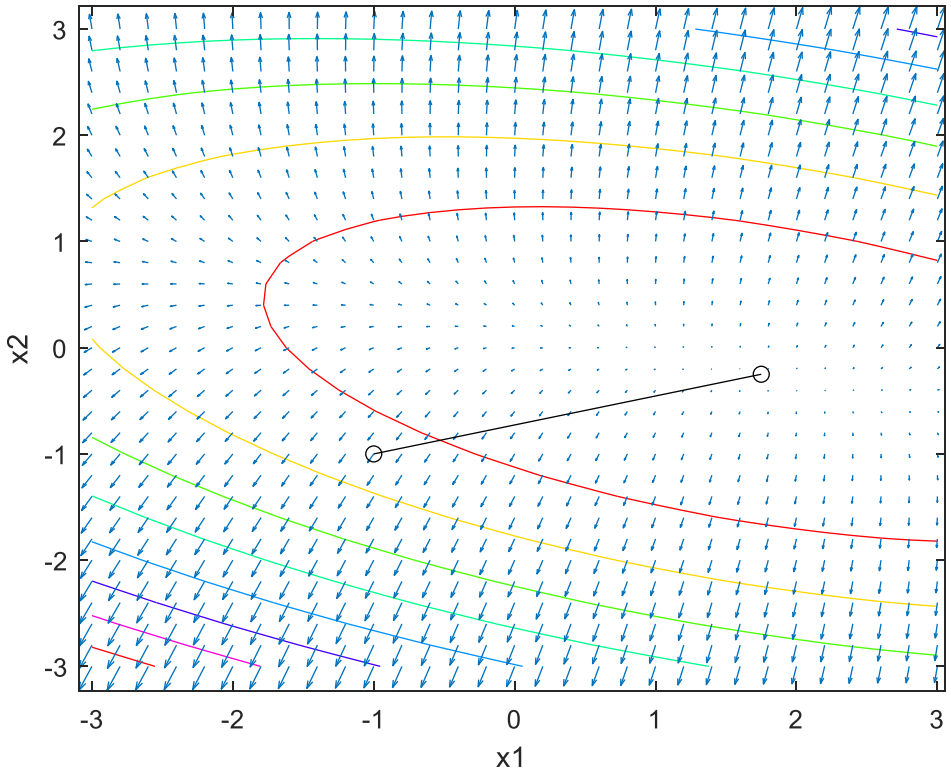
$$\begin{aligned}
\boldsymbol{x}_2 &= \boldsymbol{x}_1 - (\nabla^2 f(\boldsymbol{x}_1))^{-1} \nabla f(\boldsymbol{x}_1) \\
\boldsymbol{x}_2 &= \boldsymbol{x}_1 - \frac{1}{2}\boldsymbol{A}^{-1}(2\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{b}) \\
\boldsymbol{x}_2 &= \boldsymbol{x}_1 - \underbrace{\boldsymbol{A}^{-1}\boldsymbol{A}}_{\boldsymbol{I}}\boldsymbol{x}_1 - \frac{1}{2}\boldsymbol{A}^{-1}\boldsymbol{b} \\
\boldsymbol{x}_2 &= \boldsymbol{x}_1 - \boldsymbol{x}_1 - \frac{1}{2}\boldsymbol{A}^{-1}\boldsymbol{b} \qquad \boxed{\boldsymbol{x}_2 = -\frac{1}{2}\boldsymbol{A}^{-1}\boldsymbol{b}}
\end{aligned}$$

Ale jest to punkt minimum $\boldsymbol{x}^*$ funkcji kwadratowej!



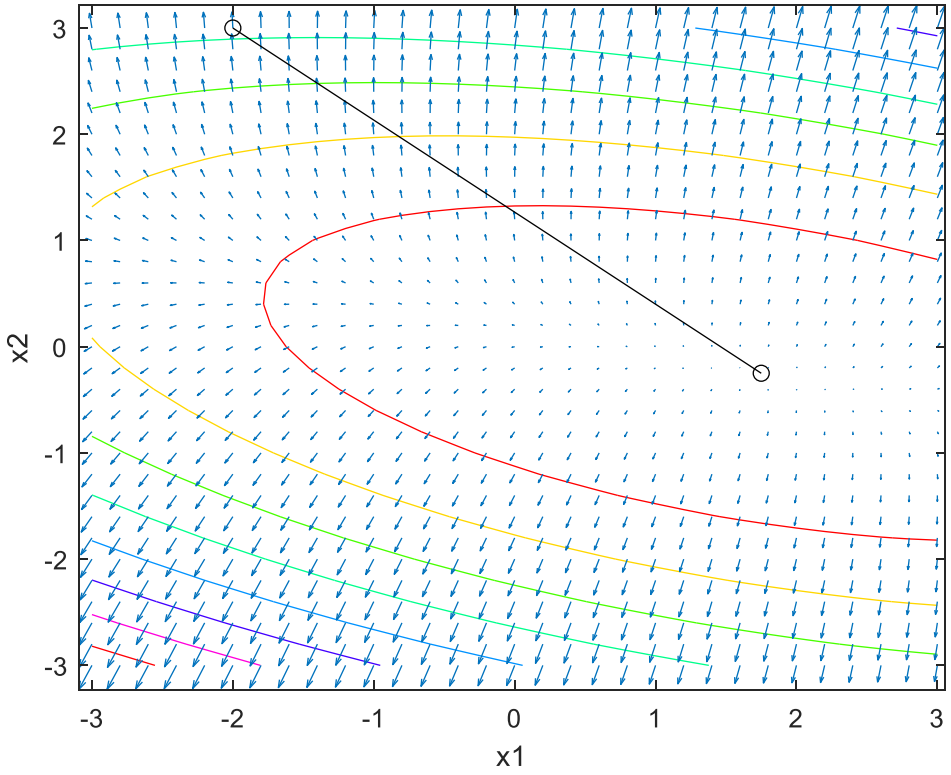

```
d=[dfdx1 dfdx2]';
s=-d2^-1*d;
xnew=xt'+s;
```

```
xnew        [1.7500;-0.2500]
xt          [2.5000,-1]
```




```
d=[dfdx1 dfdx2]';
s=-d2^-1*d;
xnew=xt'+s;
```

```
xnew        [1.7500;-0.2500]
xt          [-1,-1]
```

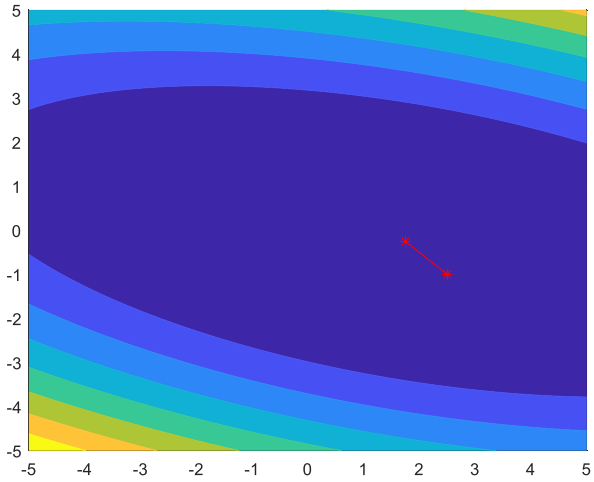



```
d=[dfdx1 dfdx2]';
s=-d2^-1*d;
xnew=xt'+s;
```

xnew	[1.7500;-0.2500]
xt	[-2,3]

Analityczne obliczenie pochodnych

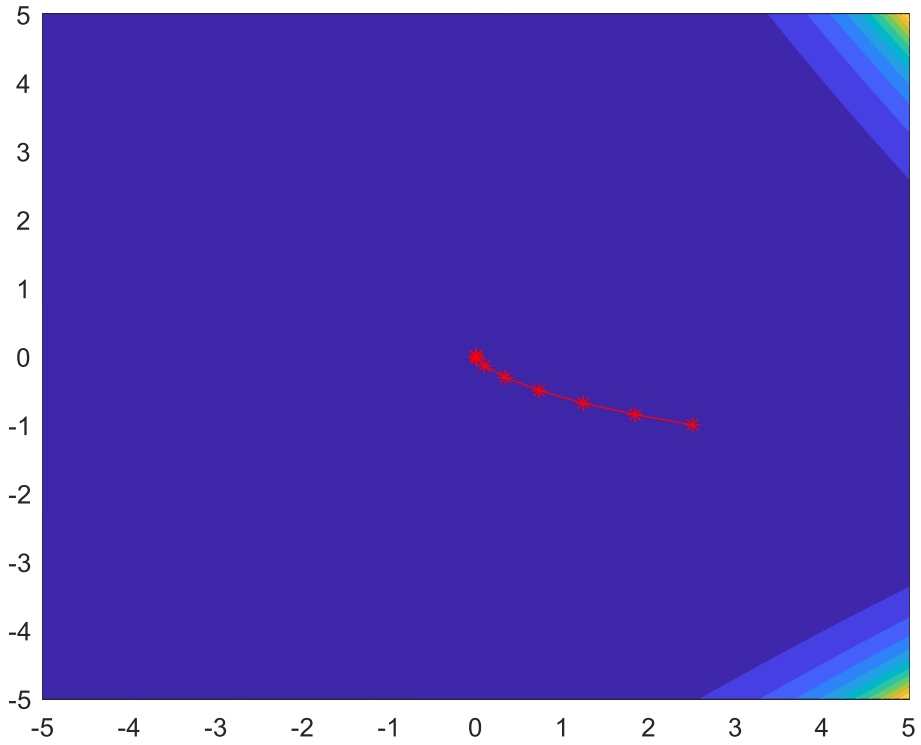
NewtonOpt.m




```
% Function Definition (Enter your Function here):
syms X Y;
%f = X - Y + 2*X^2 + 2*X*Y + Y^2;
f=X^2+X*Y+3*Y^2;
% Initial Guess (Choose Initial Guesses):
x(1) = 2;
y(1) = 3;
e = 10^(-8); % Convergence Criteria
i = 1; % Iteration Counter
% Gradient and Hessian Computation:
df_dx = diff(f, X);
df_dy = diff(f, Y);
J = [subs(df_dx,[X,Y], [x(1),y(1)]) subs(df_dy, [X,Y], [x(1),y(1)])]; % Gradient
ddf_ddx = diff(df_dx,X);
ddf_ddy = diff(df_dy,Y);
ddf_dxdy = diff(df_dx,Y);
ddf_ddx_1 = subs(ddf_ddx, [X,Y], [x(1),y(1)]);
ddf_ddy_1 = subs(ddf_ddy, [X,Y], [x(1),y(1)]);
ddf_dxdy_1 = subs(ddf_dxdy, [X,Y], [x(1),y(1)]);
H = [ddf_ddx_1, ddf_dxdy_1; ddf_dxdy_1, ddf_ddy_1]; % Hessian
S = inv(H); % Search Direction
```

```
% Optimization Condition:
while norm(J) > e
    I = [x(i),y(i)]';
    x(i+1) = I(1)-S(1,:)*J';
    y(i+1) = I(2)-S(2,:)*J';
    i = i+1;
    J = [subs(df_dx,[X,Y], [x(i),y(i)]) subs(df_dy, [X,Y], [x(i),y(i)])]; % Updated Jacobian
    ddf_ddx_1 = subs(ddf_ddx, [X,Y], [x(i),y(i)]);
    ddf_ddy_1 = subs(ddf_ddy, [X,Y], [x(i),y(i)]);
    ddf_dxdy_1 = subs(ddf_dxdy, [X,Y], [x(i),y(i)]);
    H = [ddf_ddx_1, ddf_dxdy_1; ddf_dxdy_1, ddf_ddy_1]; % Updated Hessian
    S = inv(H); % New Search Direction
end
```

f=X^2*exp(X+Y)+ Y^2*exp(X-Y);



Przykład minimalizacji funkcji 3 argumentów

$$f(x) = x_1^2 + 2x_2^2 + x_1^2 x_2^2 + x_3^2 + \exp\left(x_2^2 + x_3^2\right) - x_2 + x_3$$

$$\left|\frac{\partial f\left(x^{(k)}\right)}{\partial x_i}\right| \leq 0,001$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[j+1]} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[j]} - H^{-1}\left(f\left(x_1, x_2, x_3\right)^{[j]}\right) \nabla f\left(x_1, x_2, x_3\right)^{[j]}$$

$$H(f) = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_3} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \dfrac{\partial^2 f}{\partial x_2 \partial x_3} \\ \dfrac{\partial^2 f}{\partial x_3 \partial x_1} & \dfrac{\partial^2 f}{\partial x_3 \partial x_2} & \dfrac{\partial^2 f}{\partial x_3^2} \end{pmatrix}.$$

$$f(x) = x_1^2 + 2x_2^2 + x_1^2 x_2^2 + x_3^2 + \exp\left(x_2^2 + x_3^2\right) - x_2 + x_3$$

$$\frac{\partial f(x)}{\partial x_1} = 2x_1 + 2x_1 x_2^2$$

$$\frac{\partial f(x)}{\partial x_2} = 4x_2 + 2x_1^2 x_2 + 2x_2 \cdot \exp\left(x_2^2 + x_3^2\right) - 1$$

$$\frac{\partial f(x)}{\partial x_3} = 2x_3 + 2x_3 \cdot \exp\left(x_2^2 + x_3^2\right) + 1$$

$$\nabla f = \begin{pmatrix} 2x_1 + 2x_1 x_2^2 \\ 4x_2 + 2x_1^2 x_2 + 2x_2 \cdot \exp\left(x_2^2 + x_3^2\right) - 1 \\ 2x_3 + 2x_3 \cdot \exp\left(x_2^2 + x_3^2\right) + 1 \end{pmatrix}$$

$$\frac{\partial^2 f(x)}{\partial x_1^2} = 2 + 2x_2^2$$

$$\frac{\partial^2 f(x)}{\partial x_2^2} = 4 + 2x_1^2 + 2 \cdot \exp\left(x_2^2 + x_3^2\right) + 4x_2^2 \cdot \exp\left(x_2^2 + x_3^2\right)$$

$$\frac{\partial^2 f(x)}{\partial x_3^2} = 2 \cdot \exp\left(x_2^2 + x_3^2\right) + 4x_3^2 \cdot \exp\left(x_2^2 + x_3^2\right) + 2$$

$$\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} = 4x_1 x_2; \quad \frac{\partial^2 f(x)}{\partial x_1 \partial x_3} = 0$$

$$\frac{\partial^2 f(x)}{\partial x_2 \partial x_3} = 4x_2 x_3 \cdot \exp\left(x_2^2 + x_3^2\right)$$

$$H(f)=\begin{pmatrix} 2+2x_2^2 & 4x_1x_2 & 0 \\ 4x_1x_2 & 4+2x_1^2+2\cdot\exp\left(x_2^2+x_3^2\right)+4x_2^2\cdot\exp\left(x_2^2+x_3^2\right) & 4x_2x_3\cdot\exp\left(x_2^2+x_3^2\right) \\ 0 & 4x_2x_3\cdot\exp\left(x_2^2+x_3^2\right) & 2\cdot\exp\left(x_2^2+x_3^2\right)+4x_3^2\cdot\exp\left(x_2^2+x_3^2\right)+2 \end{pmatrix}$$

1 iteracja

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[0]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}^{[0]}$$

$$H^{[0]} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 4+2 & 0 \\ 0 & 0 & 2+2 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

$$\left(H^{[0]}\right)^{-1} = \frac{1}{48}\begin{pmatrix} 24 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 12 \end{pmatrix} \approx \begin{pmatrix} 0,5 & 0 & 0 \\ 0 & 0,16667 & 0 \\ 0 & 0 & 0,25 \end{pmatrix}$$

$$\nabla f^{[0]} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[1]} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[0]} - H^{-1}\left(f\left(x_1, x_2, x_3\right)^{[0]}\right) \nabla f\left(x_1, x_2, x_3\right)^{[0]}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}^{[0]} - \frac{1}{48} \begin{pmatrix} 24 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 12 \end{pmatrix} \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} - \frac{1}{48} \begin{pmatrix} 0 \\ -8 \\ 12 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{6} \\ -0,25 \end{pmatrix}$$

Sprawdzanie warunku stopu

$$\left|\frac{\partial f\left(x^{(k)}\right)}{\partial x_i}\right| \leq 0{,}001$$

$$\frac{\partial f\left(x^{[1]}\right)}{\partial x_i} = \begin{pmatrix} 2\cdot 0 + 2\cdot 0\cdot\frac{1}{6} \\ 4\cdot\frac{1}{6} + 2\cdot 0^2\cdot\frac{1}{6} + 2\cdot\frac{1}{6}\cdot\exp\left(\frac{1}{36}+\frac{1}{16}\right) - 1 \\ -2\cdot\frac{1}{4} - 2\cdot\frac{1}{4}\cdot\exp\left(\frac{1}{36}+\frac{1}{16}\right) + 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{3}\cdot\exp\left(\frac{13}{144}\right) - \frac{1}{3} \\ \frac{1}{2} - \frac{1}{2}\cdot\exp\left(\frac{13}{144}\right) \end{pmatrix}$$

$$\left|\frac{\partial f\left(x^{[1]}\right)}{\partial x_i}\right| = \sqrt{0^2 + \left(\frac{1}{3}\left(-1+\exp\left(\frac{13}{144}\right)\right)\right)^2 + \left(\frac{1}{2}\left(1-\exp\left(\frac{13}{144}\right)\right)\right)^2} =$$

$$= \sqrt{\frac{1}{9}\left(1-\exp\left(\frac{13}{144}\right)\right)^2 + \frac{1}{4}\left(1-\exp\left(\frac{13}{144}\right)\right)^2} =$$

$$= \sqrt{\left(\frac{1}{9}+\frac{1}{4}\right)\left(1-\exp\left(\frac{13}{144}\right)\right)^2} \approx 0,056774 \geq 0,001$$

Druga iteracja

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[1]} = \begin{pmatrix} 0 \\ \frac{1}{6} \\ -0,25 \end{pmatrix}$$

$$H^{[1]} \approx \begin{pmatrix} 2,055556 & 0 & 0 \\ 0 & 6,310565 & -0,18241 \\ 0 & -0,18241 & 4,462576 \end{pmatrix}$$

$$\left(H^{[1]}\right)^{-1} \approx \begin{pmatrix} 0,486486 & 0 & 0 \\ 0 & 0,158652 & 0,006485 \\ 0 & 0,006485 & 0,224351 \end{pmatrix}$$

$$\nabla f^{[1]} = \begin{pmatrix} 0 \\ 0,031493 \\ -0,04724 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[2]} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[1]} - H^{-1}\left(f\left(x_1, x_2, x_3\right)^{[1]}\right)\nabla f\left(x_1, x_2, x_3\right)^{[1]}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[2]} = \begin{pmatrix} 0 \\ \frac{1}{6} \\ -0.25 \end{pmatrix}^{[1]} - \begin{pmatrix} 0,486486 & 0 & 0 \\ 0 & 0,158652 & 0,006485 \\ 0 & 0,006485 & 0,224351 \end{pmatrix}\begin{pmatrix} 0 \\ 0,031493 \\ -0,04724 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[2]} = \begin{pmatrix} 0 \\ \frac{1}{6} \\ -0.25 \end{pmatrix}^{[1]} - \begin{pmatrix} 0 \\ 0,00469 \\ -0,01039 \end{pmatrix} = \begin{pmatrix} 0 \\ 0,161977 \\ -0,23961 \end{pmatrix}$$

$$\left|\frac{\partial f\left(x^{(k)}\right)}{\partial x_i}\right| \leq 0,001$$

$$\frac{\partial f\left(x^{[2]}\right)}{\partial x_i} = \begin{pmatrix} 0 \\ 0,000123 \\ -0,00023 \end{pmatrix}$$

$$\left|\frac{\partial f\left(x^{[1]}\right)}{\partial x_i}\right| = \sqrt{0^2 + 0,000123^2 + 0,00023^2} \approx 0,000264 < 0,001$$

Rozwiązanie

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{[2]} = \begin{pmatrix} 0 \\ 0,161977 \\ -0,23961 \end{pmatrix}; \; f(x) \approx 0,795547$$

Inne metody

Metoda najszybszego spadku (ang. steepest descent method)
Metoda Newtona
Metoda Marquardta
Metoda sprzężonego gradientu Fletcher-Reevesa
Metody quasinewtonowskie:

* **Metoda Davidon-Fletcher-Powella (DFP)**
* **Metoda Broyden-Fletcher-Goldfarb-Shannona (BFGS)**

Metoda Cauchy'ego działa dobrze, gdy punkt początkowy jest daleko od minimum, podczas, gdy metoda Newtona działa dobrze, gdy punkt początkowy jest blisko minimum. W metodzie Maruardta metodę Cauchy'ego stosuje się na początku by następnie zaadoptować metodę Newtona.

Algorytm metody Maquarta

1) Wybierz maksymalną liczbę iteracji M, punkt początkowy oraz parametr zakończenia ϵ. Ustal $k = 0$ oraz $\lambda^{(0)} = 10^4$ (duża liczba).

2) Oblicz $\nabla f(x^{(k)})$

3) Jeśli $||\nabla f(x^{(k)})|| \leqslant \epsilon$, **Zakończ;**
Jeśli $k \geqslant M$; **Zakończ;**
W przeciwnym razie idź do kroku 4).

4) Oblicz $x^{(k+1)} = x^{(k)} - \left[\nabla^2 f(x^{(k)}) + \lambda^{(k)} I\right]^{-1} \nabla f(x^{(k)})$

5) Jeśli $f(x^{(k+1)}) < f(x^{(k)})$, idź do kroku 6);
W przeciwnym przypadku idź do kroku 7).

6) Ustal $\lambda^{(k+1)} = \frac{1}{2}\lambda^{(k)}$, $k = k + 1$ i idź do kroku 2).

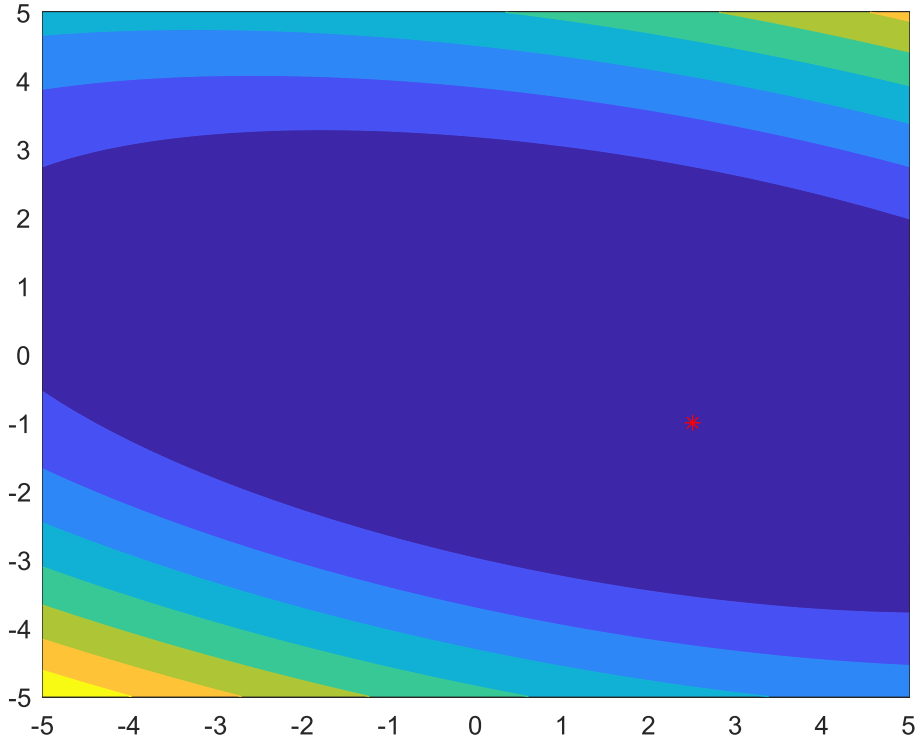
7) Ustal $\lambda^{(k+1)} = 2\lambda^{(k)}$ i idź do kroku 4).

```
%Insert starting points
x01=2.5;
x02=-1;
 lambda=1
```

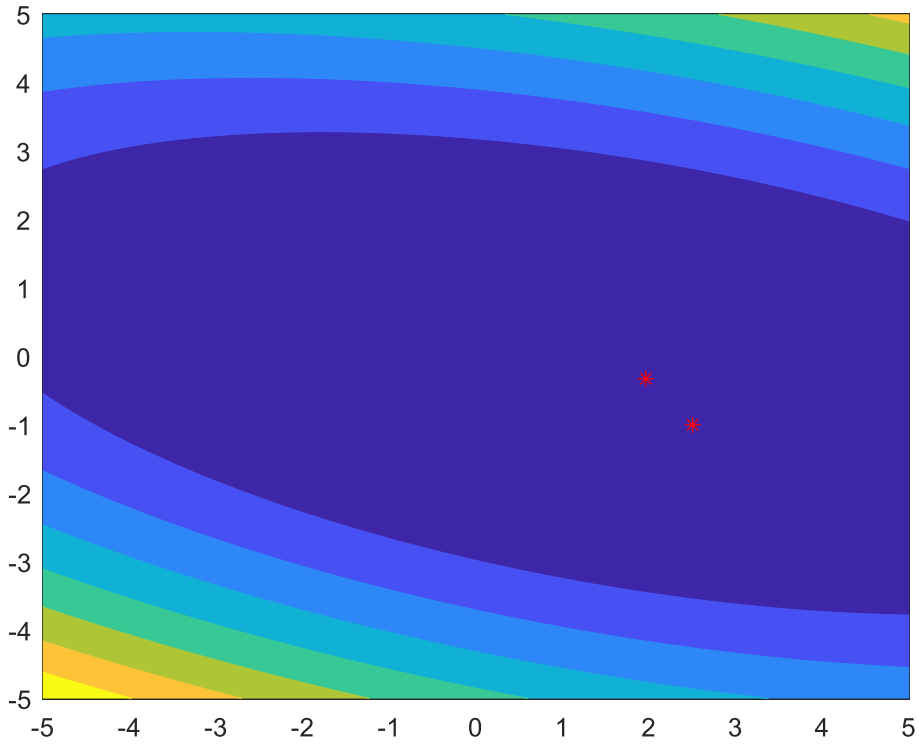
K	x1	x2	f	λ
1.000000000000000	1.960674157303371	-0.325842696629214	0.082375962630981	0.500000000000000
2.000000000000000	1.780309449254006	-0.257763860747836	0.001498827391873	0.250000000000000
3.000000000000000	1.752300880607975	-0.250550345067598	0.000008551786904	0.125000000000000
4.000000000000000	1.750090476292853	-0.250021401158006	0.000000013206825	0.062500000000000
5.000000000000000	1.750001813618231	-0.250000428263940	0.000000000005306	0.031250000000000

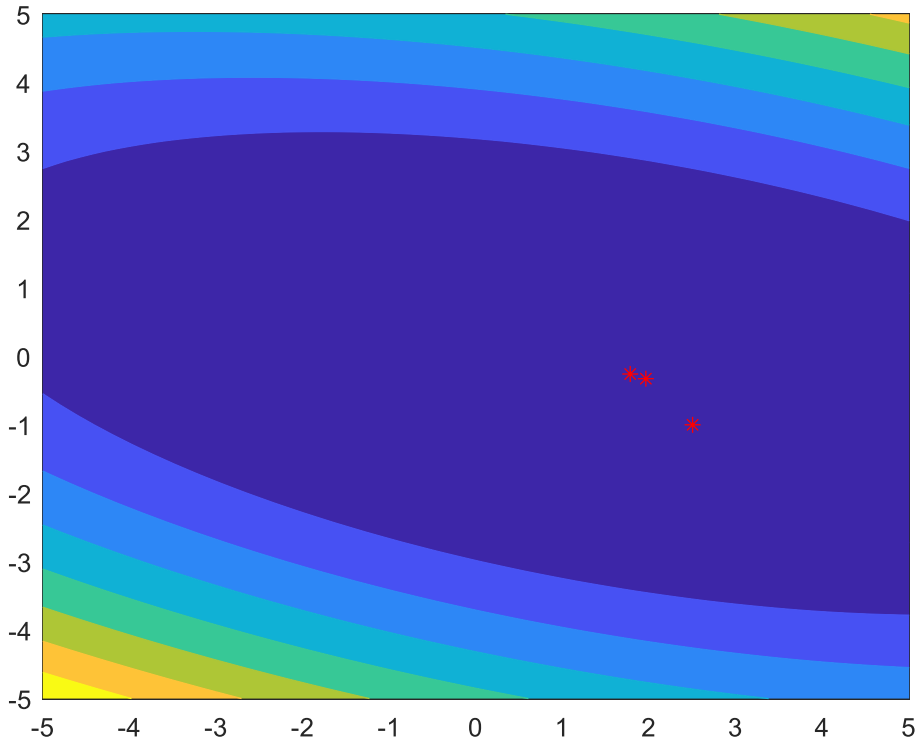
Maquard.m

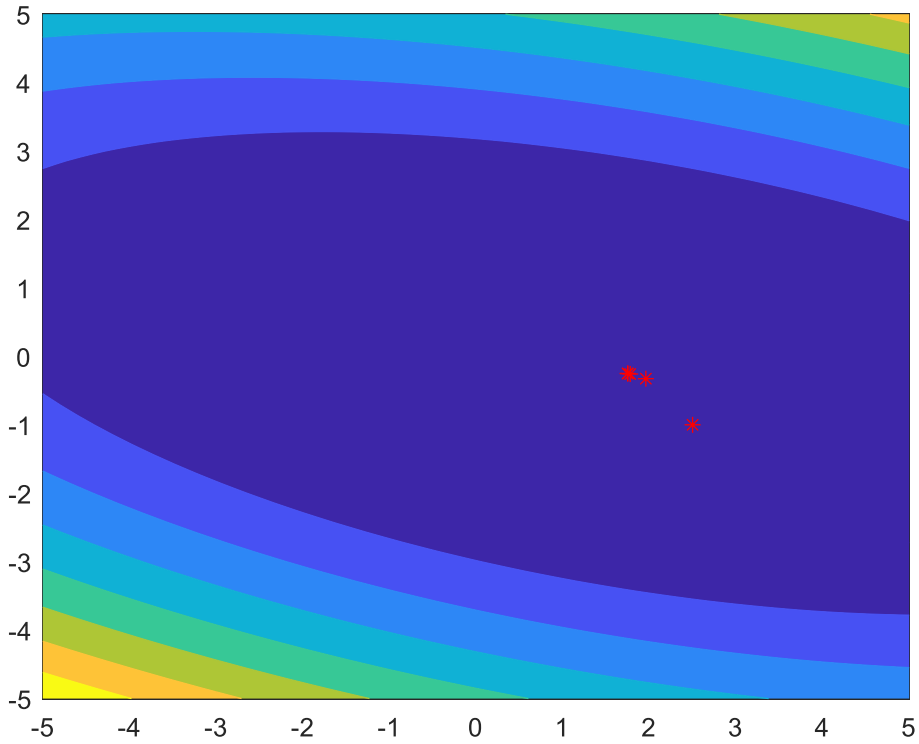
$$f(x_1, x_2) = (x_1 + 3x_2 - 1)^2 + (x_1 - x_2 - 2)^2$$




```
%Insert starting points
x01=2.5;
x02=-1;
```


5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5


5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5


5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

Metoda Fletcher-Reeves gradientu sprzężonego

Forma kwadratowa $$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{b}, \quad \mathbf{x} \in \mathbf{R}^n.$$

błąd $$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k.$$

kierunek $$\mathbf{p}_{k+1} = \mathbf{r}_k - \frac{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{r}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k}\mathbf{p}_k.$$

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
  $\alpha_k := \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{r}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k}$
  $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
  $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
  if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
  $\beta_k := \frac{\mathbf{r}_{k+1}^{\mathrm{T}}\mathbf{r}_{k+1}}{\mathbf{r}_k^{\mathrm{T}}\mathbf{r}_k}$
  $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
  $k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

Przykład

$$f(x_1, x_2) = (x_1 + 3x_2 - 1)^2 + (x_1 - x_2 - 2)^2$$

```
A =

     4     4
     4    20

>> b

b =

     6
     2
```

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{b},$$

```
>> syms x1 x2
>> expand((x1+3*x2-1)^2+(x1-x2-2)^2)
 2*x1^2 + 4*x1*x2 - 6*x1 + 10*x2^2 - 2*x2 + 5
```

fletcherReev.m

	1	2	3	4	5	6	7	8	9
1	0	1	1	0.0486	-2	-22	-2	-22	0.0146
2	1	0.9029	-0.0685	0.3217	2.6624	-0.2420	2.6331	-0.5642	7.2958e-29
3	2	1.7500	-0.2500	0.0477	-5.3291e-15	-2.2204e-14	-5.3291e-15	-2.2204e-14	1.0101e-05
4	3	1.7500	-0.2500	0.3273	-7.0572e-17	1.6937e-17	-7.0626e-17	1.6713e-17	2.3183e-25
5	4	1.7500	-0.2500	0.0477	-8.0365e-30	-3.4007e-29	-8.0365e-30	-3.4007e-29	4.0748e-08
6	5	1.7500	-0.2500	0.3273	-6.8648e-33	1.6223e-33	-6.8651e-33	1.6209e-33	1.1078e-22
7	6	1.7500	-0.2500	0.0477	-1.7059e-44	-7.2257e-44	-1.7059e-44	-7.2257e-44	3.0436e-10
8	7	1.7500	-0.2500	0.3273	-1.2606e-48	2.9761e-49	-1.2606e-48	2.9759e-49	1.2285e-21
9	8	1.7500	-0.2500	0.0477	-1.0431e-59	-4.4184e-59	-1.0431e-59	-4.4184e-59	2.2649e-11
10	9	1.7500	-0.2500	0.3273	-2.1028e-64	4.9641e-65	-2.1028e-64	4.9640e-65	3.4214e-21

Fletcher- Reeves algorytm:

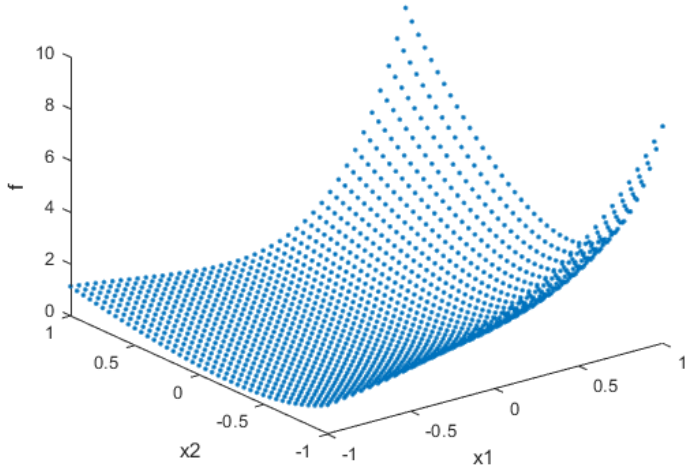
1: $p_0 = -\nabla f(x_0),\ k = 0$
2: **while** $\|\nabla f(x_k)\| > \epsilon$ **do**
3: $\quad \alpha_k = \text{linesearch for } f \text{ along } p_k$
4: $\quad x_{k+1} = x_k + \alpha_k p_k \qquad \alpha_k = \arg\min_\alpha f(x_k + \alpha p_k)$
5: $\quad \beta_{k+1}^{\mathrm{FR}} = \|\nabla f(x_{k+1})\|^2 / \|\nabla f(x_k)\|^2$
6: $\quad p_{k+1} = -\nabla f(x_{k+1}) + \beta_{k+1}^{\mathrm{FR}} p_k$
7: $\quad k = k + 1$
8: **end while**

$$f(x_1, x_2) = x_1^2 e^{x_1+x_2} + x_2^2 e^{x_1-x_2}$$

$x^0 = (0, 0)$ Punkt krytyczny (stacjonarny)

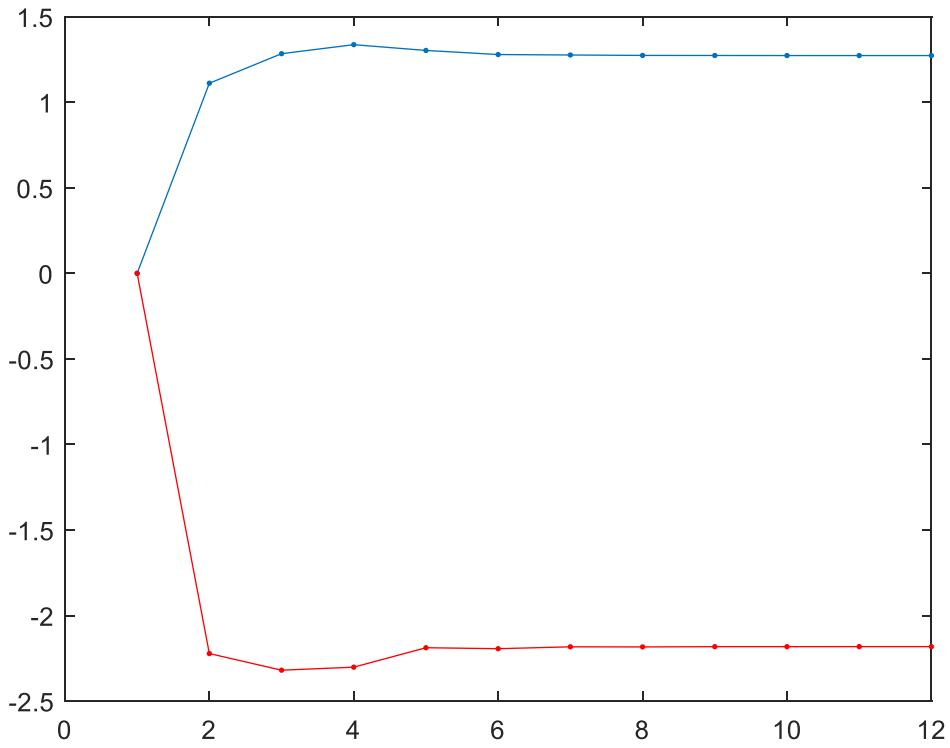
$$\frac{\partial f}{\partial x_1} = (2x_1 + x_1^2)e^{x_1+x_2} + x_2^2 e^{x_1-x_2};$$

$$\frac{\partial f}{\partial x_2} = x_1^2 e^{x_1+x_2} + (2x_2 - x_2^2)e^{x_1-x_2}.$$



Gradi.m

```
clc,clear
% Méthode du gradient conjugué
A=[5 2;2 3]; b=[-2;4];c=0;
X=[0;0];  % Solution initiale
r=b-A*X;
d=r;
Out=[X'];
while norm(r)>0.0001
    alpha=r'*r./(d'*A*d);
    X=X-alpha*d;
    ri=r;
    r=ri-alpha*A*d;
    beta=r'*r./ri'*r;
    d=r+beta*d;
    Out=[Out;X'];
end
disp('la solution est:')
disp(X)
plot(Out(:,1),'.-');
hold on;
plot(Out(:,2),'r.-');
```



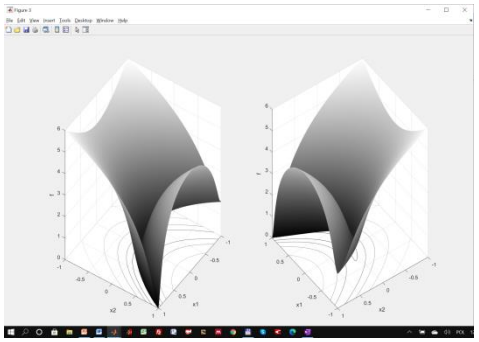
Optymalizacja w Matlabie. Funkcja Rosenbrock

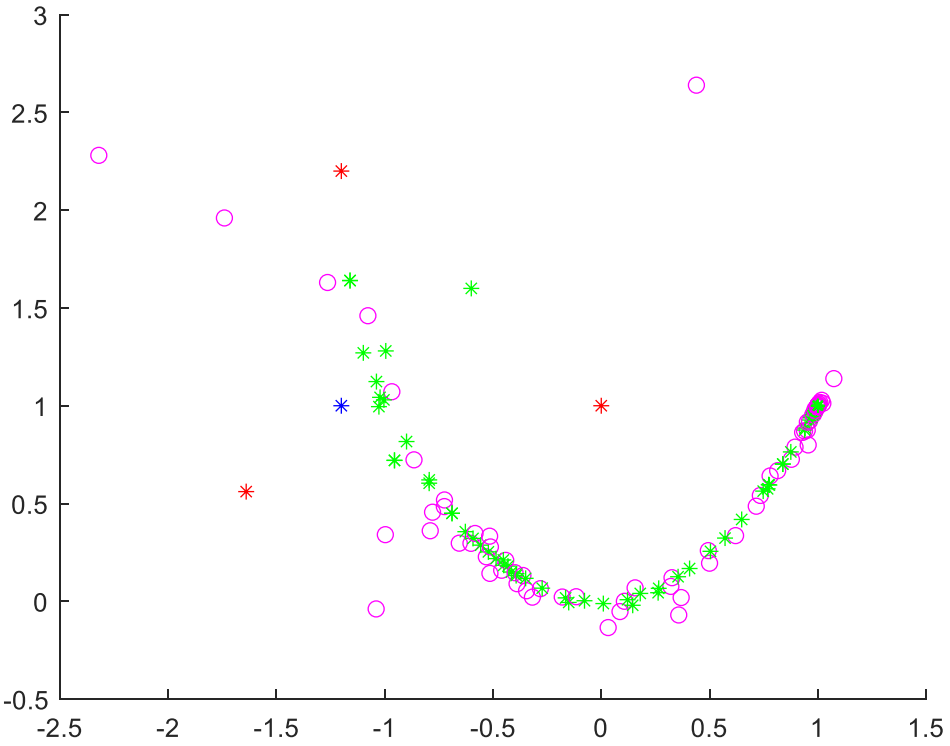
$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Rozenbrock folder

[x,fval] = **fminsearch**(@banana,[-1.2, 1])

[xmin,fmin,ct]=ANMSsimpexOS(@banana,[-1.2 1],10^-4,300)





fminsearch
Find minimum of unconstrained multivariable function using derivative-free method
algorithm: 'Nelder-Mead simplex direct search'

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

Iteration	Func-count	min f(x)	Procedure
0	1	-6.70447	
1	3	-6.89837	initial simplex
2	5	-7.34101	expand
3	7	-7.91894	expand
4	9	-9.07939	expand
5	11	-10.5047	expand
6	13	-12.4957	expand
7	15	-12.6957	reflect
8	17	-12.8052	contract outside
9	19	-12.8052	contract inside
10	21	-13.0189	expand
11	23	-13.0189	contract inside
12	25	-13.0374	reflect
13	27	-13.122	reflect
14	28	-13.122	reflect
15	29	-13.122	reflect
16	31	-13.122	contract outside
17	33	-13.1279	contract inside
18	35	-13.1279	contract inside
19	37	-13.1296	contract inside
20	39	-13.1301	contract inside
21	41	-13.1305	reflect
22	43	-13.1306	contract inside
23	45	-13.1309	contract inside
24	47	-13.1309	contract inside
25	49	-13.131	reflect
26	51	-13.131	contract inside
27	53	-13.131	contract inside
28	55	-13.131	contract inside

ıb/ref/fminsearch.html#d120e380389

Optymalizacja w Matlabie

fminfunOS.m $f(x)=e^{x_1}(4x_1^2+2x_2^2+4x_1x_2+2x_2+1).$

x = 0.5000 -1.0000
fval = 3.6609e-15

```
%function f = objfun(x)
objfun =@(x)exp(x(1))*(4*x(1)^2 + 2*x(2)^2 + 4*x(1)*x(2) + 2*x(2) + 1);
 x0 = [-1,1];
options = optimset('LargeScale','off');
[x,fval,exitflag,output] = fminunc(objfun,x0,options)
```

Algorithms

collapse all

Quasi-Newton Algorithm

The quasi-newton algorithm uses the BFGS Quasi-Newton method with a cubic line search procedure. This quasi-Newton method uses the BFGS ([1],[5],[8], and [9]) formula for updating the approximation of the Hessian matrix. You can select the DFP ([4],[6], and [7]) formula, which approximates the inverse Hessian matrix, by setting the HessUpdate option to 'dfp' (and the Algorithm option to 'quasi-newton'). You can select a steepest descent method by setting HessUpdate to 'steepdesc' (and Algorithm to 'quasi-newton'), although this setting is usually inefficient. See fminunc quasi-newton Algorithm.

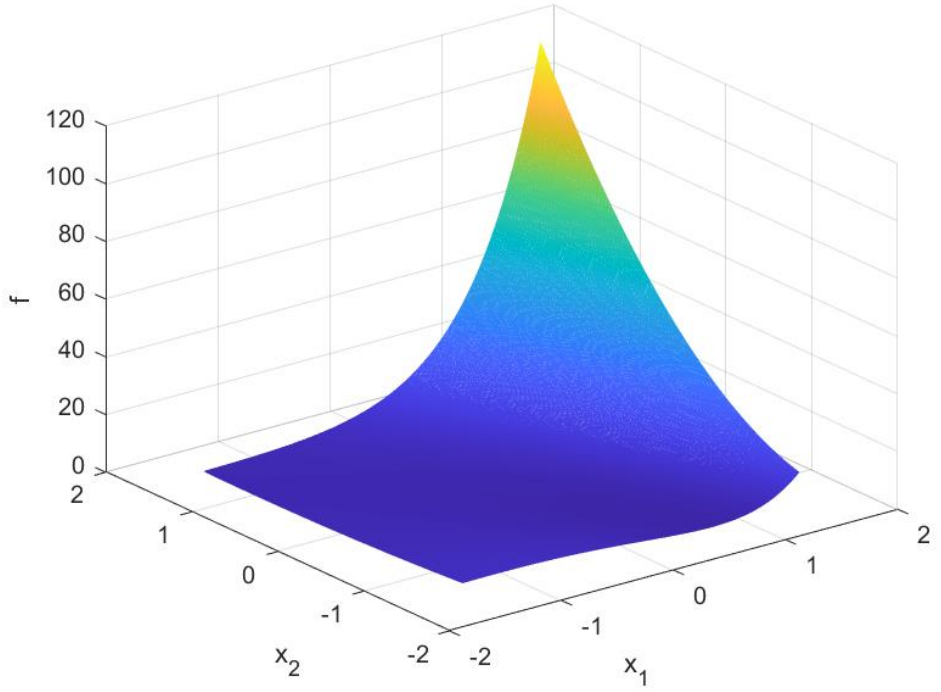
Trust Region Algorithm

The trust-region algorithm requires that you supply the gradient in fun and set SpecifyObjectiveGradient to true using optimoptions. This algorithm is a subspace trust-region method and is based on the interior-reflective Newton method described in [2] and [3]. Each iteration involves the approximate solution of a large linear system using the method of preconditioned conjugate gradients (PCG). See fminunc trust-region Algorithm, Trust-Region Methods for Nonlinear Minimization and Preconditioned Conjugate Gradient Method.

$$f(x)=e^{x_1}(4x_1^2+2x_2^2+4x_1x_2+2x_2+1).$$

x = 0.5000 -1.0000

fval = 3.6609e-15



output =

struct with fields:

iterations: 8
funcCount: 66
stepsize: 6.3361e-07
lssteplength: 1
firstorderopt: 1.2284e-07
algorithm: 'quasi-newton'
message: '↲Local minimum found.↲↲Optimization completed because the size of the gradient is less than↲the value of the optimality tolerance.↲↲<stopping criteria details>↲↲Optimization completed: The first-order optimality measure, 7.076980e-08, is less ↲than options.OptimalityTolerance = 1.000000e-06.↲↲'

Lokowanie biegunów układu dynamicznego

$$\begin{aligned}\dot{x}_1 &= -0{,}5x_1 + u_1,\\ \dot{x}_2 &= -2x_2 + 10x_3 - 2u_1 + 2u_2,\\ \dot{x}_3 &= x_2 - 2x_3 + u_2.\end{aligned}$$

$$\frac{dx}{dt} = Ax(t) + Bu(t),$$

$$A = \begin{bmatrix} -0{,}5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2 \end{bmatrix};\quad B = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix};\quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix};\quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

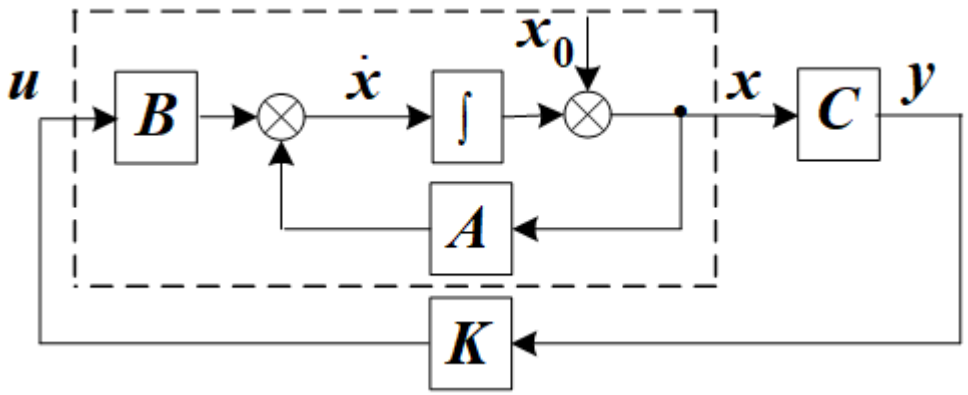
$$y_1 = x_1,\ y_2 = x_3.$$

$$y(t) = Cx(t),$$

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix};\quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$$u(t) = Ky(t),$$

$$\frac{dx}{dt} = Ax(t) + BKCx(t) = (A + BKC)x(t).$$




```
>>A=[-0.5  0  0; 0  -2  10; 0  1  -2];
s=eig(A)
s =
   1.1623
  -5.1623
  -0.5000
```

$$A + BKC$$

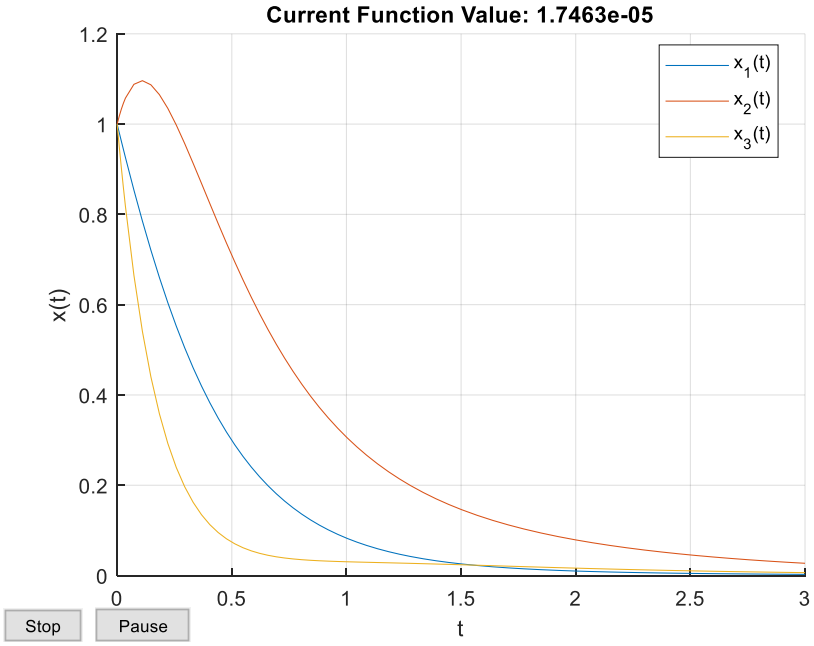
$$[-5, -3, -1]$$

```
function y=bieg(x)
A =  [ -0.5  0  0;  0  -2  10;  0  1  -2 ];
B =  [ 1  0;  -2  2;  0  1 ];
C =  [ 1  0  0;  0  0  1 ];
K=[x(1) x(2);x(3) x(4)];
eigfun =  sort(eig(A+B*K*C));
goal = [-5,-3,-1]';
y=norm(goal-eigfun);
```

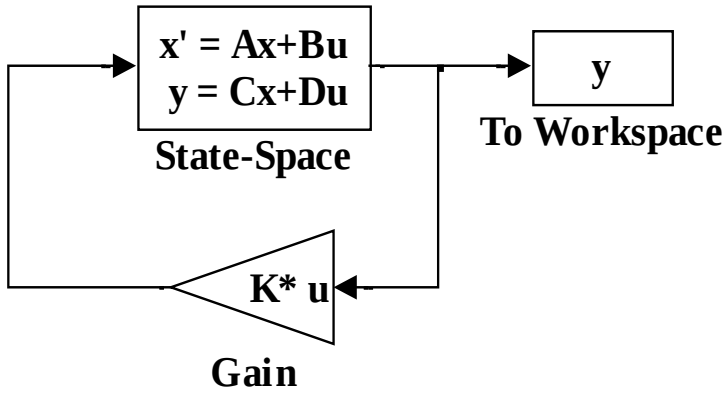
```
A =  [ -0.5  0  0;  0  -2  10;  0  1  -2 ];
B =  [ 1  0;  -2  2;  0  1 ];
C =  [ 1  0  0;  0  0  1 ];
K=[-1 -1 ;-1 -1];
options =
optimset('Display','iter','PlotFcns',@optimplotf
val);
[x,fval,e,o] = fminsearch(@bieg,[-1 -1 -1 -
1],options)
```

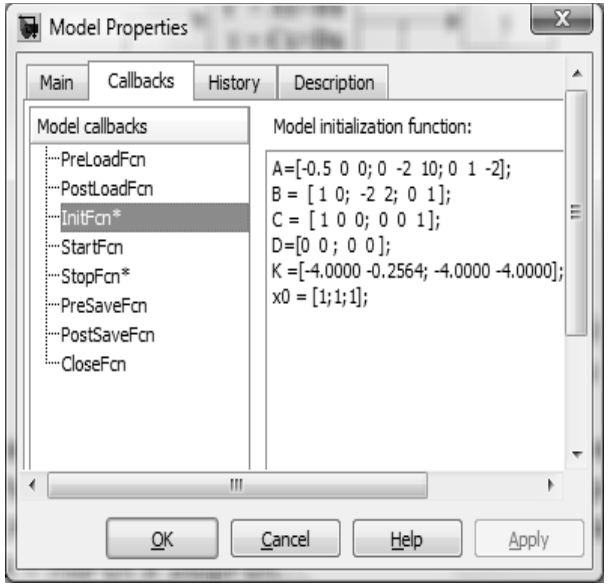
Iteration	Func-count	min f(x)	Procedure
0	1	2.52224	
1	5	2.49652	initial simplex
2	7	2.44186	expand
3	8	2.44186	reflect
4	10	2.37396	expand
5	11	2.37396	reflect
6	13	2.24397	expand
7	14	2.24397	reflect
8	16	2.06286	expand
9	17	2.06286	reflect
10	19	1.72847	expand
11	20	1.72847	reflect
12	22	1.22076	expand
13	23	1.22076	reflect
14	25	0.339008	expand

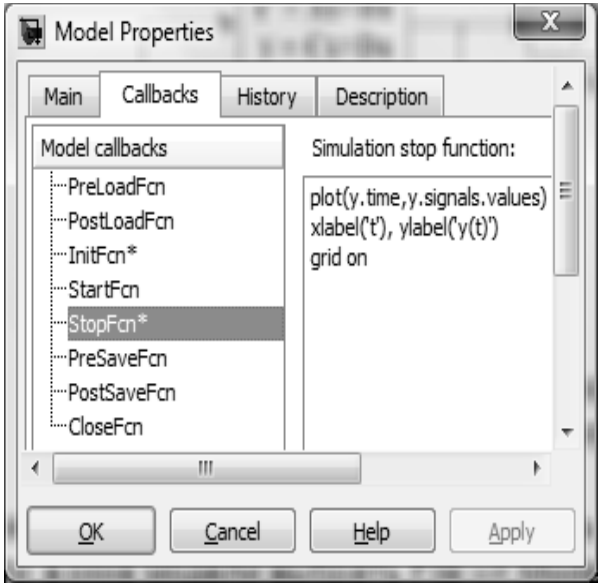
```
x0 = [1;1;1];
[Times, xvals] = ode45(@(u,x)((A +
B*K*C)*x),[0, 3],x0);
plot(Times,xvals)
legend('x_1(t)','x_2(t)','x_3(t)','Location','best')
grid on
xlabel('t');
ylabel('x(t)');
```



Model w Simulinku





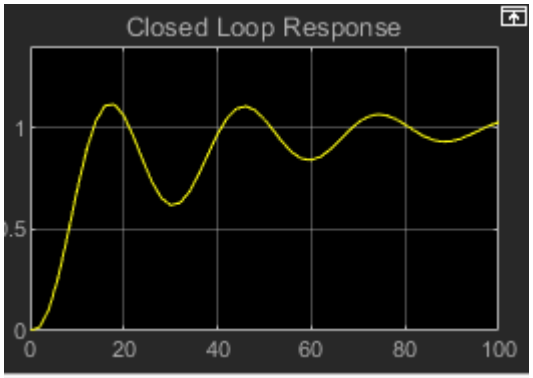


-

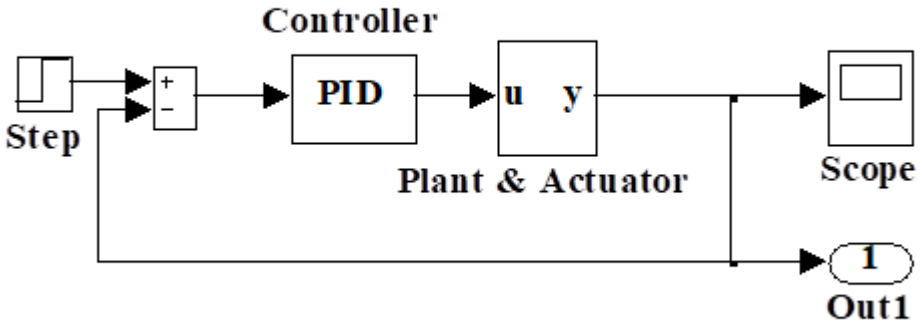
PID sterownik

$$W_o(s) = \frac{Y(s)}{U(s)} = \frac{1,5}{50s^3 + a_2 s^2 + a_1 s + 1},$$

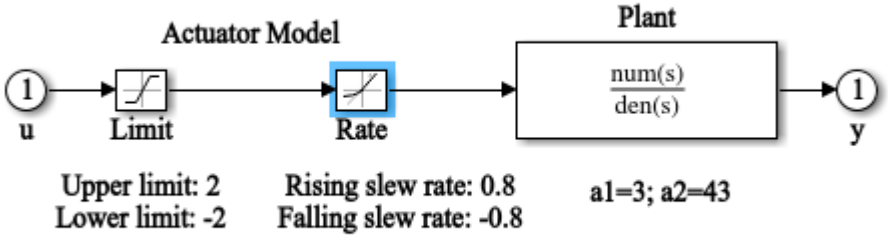
$$W_p(s) = K_p + K_i \frac{1}{s} + K_d s.$$

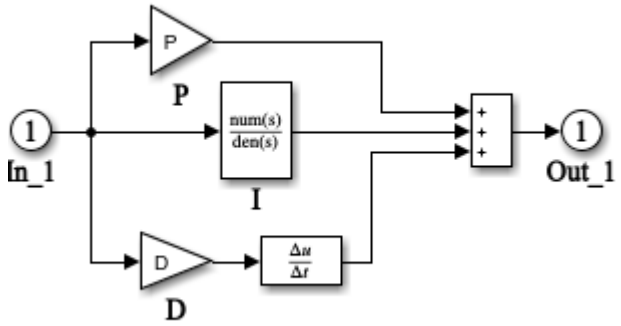


Tunable Variables are PID gains, Kp, Ki, Kd.



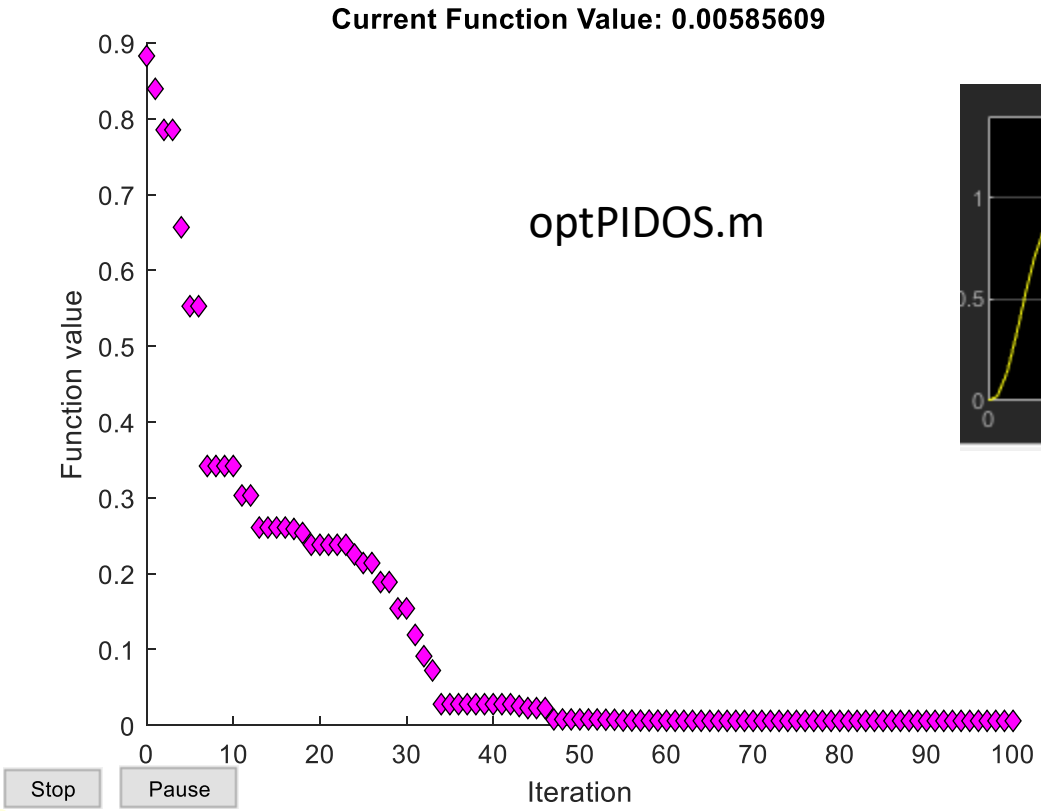
Optsim.mdl


Actuator Model
Plant
1
u
Limit
Rate
num(s)
den(s)
1
y
Upper limit: 2
Lower limit: -2
Rising slew rate: 0.8
Falling slew rate: -0.8
a1=3; a2=43


P
P
1
In_1
num(s)
den(s)
I
1
Out_1
D
D

```
function [Kp,Ki,Kd] = optpidOS
%optsim
pid0 = [0.6 0.05 2];
a1 = 3; a2 = 43;
options = optimset('Display','iter','PlotFcns',@optimplotfval);
[x,fval,e,o] = fminsearch(@optpid1,pid0,options)
Kp = x(1);
Ki = x(2) ;
Kd = x(3) ;
   function F = optpid1(pid)
Kp = pid(1);
     Ki = pid(2);
     Kd = pid(3);
simopt = simset('solver','ode5',...
                  'SrcWorkspace','Current');
     [tout,xout,yout] = sim('optsim',[0 100],simopt);
     F = norm(yout-1);
     %F = norm(yout(10:51)-1);


   end
end
```

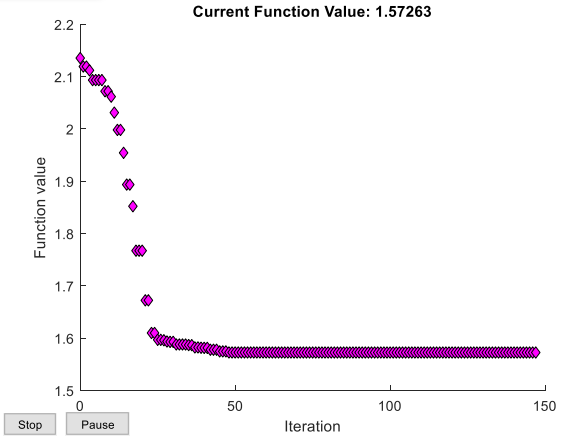
F = norm(yout(10:51)-1);



optPIDOS.m

[Kp Ki Kd]=[0.851496432319671,0.083222330933200,7.016770273098319]

F = norm(yout-1);



x=[0.851496432319671,0.083222330933200,7
.016770273098319]
Kp=x(1)
Ki=x(2)
Kd=x(3)