

Data oddania: _____

Ocena: _____

Autor 1 nr albumu 1

Autor 2 nr albumu 2

Zadanie Numer: Temat*

1. Cel

Optymalizacja wielowymiarowa bez ograniczeń

Czas na realizację zadania: 3 tygodnie

Zaimplementować jedną spośród następujących metod optymalizacji:

- pełzający sympleks
- metoda Fletchera-Reevesa
- **metoda quasi-Newtona w wersji Davidon-Fletcher-Powell (DFP)**

Przedstawiany jako rozwiązanie program powinien prezentować kolejne kroki poszukiwania rozwiązania.

2. Rozwiązanie zadania

Metody „quasi-newtonowskie” są metodami wyznaczania lokalnych ekstremów funkcji. Metoda ta oblicza postać przybliżoną macierzy Hessego, przy pomocy gradientu. Kolejne przybliżenia rozwiązania wyznaczamy w następujący sposób.

Wiadomość o wartości macierzy Hessego w punkcie daje nam informacje o krzywiznie funkcji. Jeśli macierze ta jest dodatnio określona w punkcie to znaczy że jest to lokalne minimum, natomiast gdy jest ujemnie oznaczona oznacza to że w danym punkcie znajduje się lokalne maksimum. W przypadku gdy macierze jest nieokreślona ma miejsce punkt przegięcia.

* SVN: <http://serce.ics.p.lodz.pl/svn/labs/moo/adres@rewizja>

1. Inicjalizacja

- ustal dodatnio określoną macierz S_0
- punkt początkowy x_0
- licznik $k = 0$

2. Iteracja

- oblicz $g_k = \nabla f(x_k)$
- oblicz $d_k = -S_k g_k$
- wyznacz α_k metodą Wolfe
- przypisz $x_{k+1} = x_k + \alpha_k d_k$
- Aktualizacja przybliżenia macierzy Hessego
 - oblicz $p_k = \alpha_k d_k$; $g_{k+1} = \nabla f(x_{k+1})$
 - oblicz $q_k = g_{k+1} - g_k$
 - oblicz $S_{k+1} = S_k + \frac{p_k p_k^T}{\langle p_k, q_k \rangle} - \frac{S_k q_k q_k^T S_k}{\langle q_k, S_k q_k \rangle}$
- przypisz $k = k + 1$

Gdzie warunkiem stopu algorytmu jest albo liczba iteracji, albo wielkość kroku.

Kryterium Wolfa, jest rozszerzeniem kryterium poprawy Armijo

$$f(x^k + \lambda^k d^k) \leq f(x^k) + c \lambda^k \nabla f^T(x^k) d^k \quad (1)$$

o kryterium krzywizny:

- Słabe kryterium krzywizny

$$\nabla f^T(x^k + \lambda^k d^k) d^k \geq c \nabla f^T(x^k) d^k \quad (2)$$

- Silne kryterium krzywizny

$$\|\nabla f^T(x^k + \lambda^k d^k) d^k\| \leq \|c \nabla f^T(x^k) d^k\| \quad (3)$$

Gdzie :

- x^k - kolejne przybliżenia rozwiązania
- λ^k - długość kroku
- d^k - kierunek zmiany
- c - stałe używane w kryteriach
- $f(x)$ - wartość funkcji podstawowej w punkcie
- $\nabla f(x)$ - wartość gradientu w punkcie

Dobór długości kroku na podstawie kryterium Wolfe odbywa się przy pomocy algorytmu backtrakingu:

1. Sprawdź czy zachodzi kryterium dla zadanej długości kroku
2. Jeśli kryterium zachodzi, zwróć długość kroku λ
3. Jeśli kryterium nie zachodzi, oblicz $\lambda = \lambda \rho$, gdzie $\rho \in (0, 1)$, wróć do punktu 1

3. Opis programu

Program jest implementacją wyżej opisanej metody.

Wyjaśnienie znaczenia zmiennych :

- **f0** - funkcja podstawowa
- **f1** - gradient funkcji
- **d** - kierunek zmiany
- **x** - kolejne przybliżenia
- **a** - długość kroku
- **c** - parametr kryterium
- **Sk** - kolejne przybliżenia macierzy Hessego
- **rho** - parametr backtrackingu

3.1. Metoda Quasi-newtona DFP

```

1 function x=dfp(x0,fn,a0,rho,c,kMax,eps)
  f=inline(fn);
3 f0=@(x)f(x(1),x(2));
  syms x y
5 t=[x;y];
  w=fn;
7 gfn=matlabFunction(jacobian(w,t).');
  f1=@(x)gfn(x(1),x(2));
9 sl='wolfe';
  sl=@(x,d)backtk(sl,a0,x,d,c,f0,f1,rho);
11 x=x0;
  k=0;
13
  Sk = eye(length(x0),length(x0));
15 err = 1.0;
  X=x(1);
17 Y=x(2);
  fk=0;
19 while (k<kMax) && (eps < err)
      g = f1(x);
21 d = -Sk*g;
      a=sl(x,d);
23 fk = f0(x);
      x=x+a*d;
25 fprintf('f(%.3f,%.3f)=%.3f\n',x(1),x(2),fk);
      fk_new = f0(x);
27 p = a*d;
      q= f1(x) - g;
29 Sk = Sk + (p*(p'))/(p'*q) - (Sk*q*(q')*Sk)/(q'*(Sk*q));
      k=k+1;
31 err = max(abs(fk - fk_new),norm(p));
end

```

metoda quasi-newtona DFP

3.2. Kryterium Wolfe

```

1 if c(1)*f1(x)'*p<=f1(x+a*p)'*p &&
2   f0(x+a*p)<=f0(x)+c(2)*a*f1(x)'*p
   test=1;
4 else
   test=0;
6 end

```

Slabe kryterium Wolfa

```
1 if abs(c(1)*f1(x)*p)>=abs(f1(x+a*p)*p) &&  
2   f0(x+a*p)<=f0(x)+c(2)*a*f1(x)*p  
   test=1;  
4 else  
   test=0;  
6 end
```

Silne kryterium Wolfa

3.3. Algorytm backtrackingu

```
switch cond  
2 case 'wolfe'  
   check=@(a) wolfe(a,x,p,c(1),f1) &&  
   armijo(a,x,p,c(2),f0,f1);  
4 case 'wolfes'  
   check=@(a) wolfes(a,x,p,c(1),f1) &&  
   armijo(a,x,p,c(2),f0,f1);  
8 end  
a=a0;  
10 while check(a)~=1 && abs(a) > 0.0000001  
   a=rho*a;  
12 end
```

Algorytm backtrackingu

4. Wyniki

Testowana funkcja : $x^2 + y^2$.

Dla Argumentów : $[2; 1]$, $x^2 + y^2$, 0.6, 0.99, $[0.6; 0.4]$, 100, 0.001
kolejne przybliżenia:

f(-0.400,-0.200)=5.000

f(-0.160,-0.080)=0.200

f(-0.064,-0.032)=0.032

f(-0.026,-0.013)=0.005

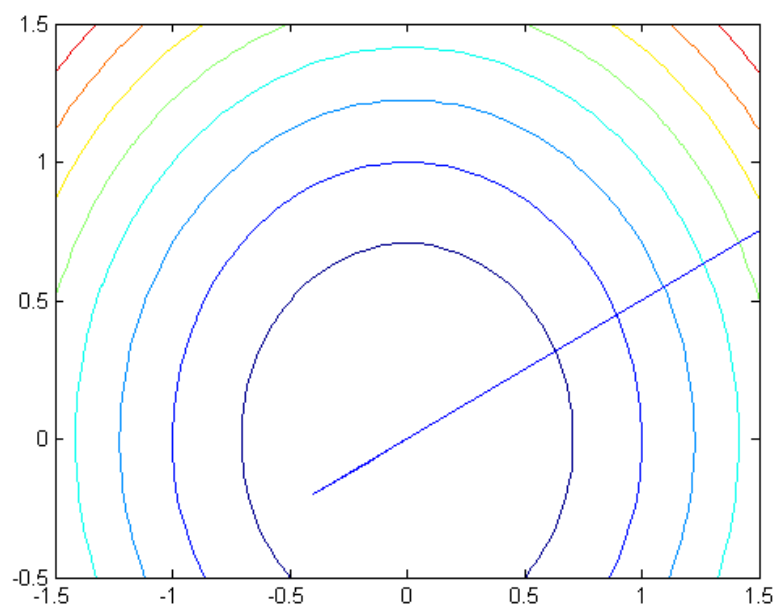
f(-0.010,-0.005)=0.001

f(-0.004,-0.002)=0.000

f(-0.002,-0.001)=0.000

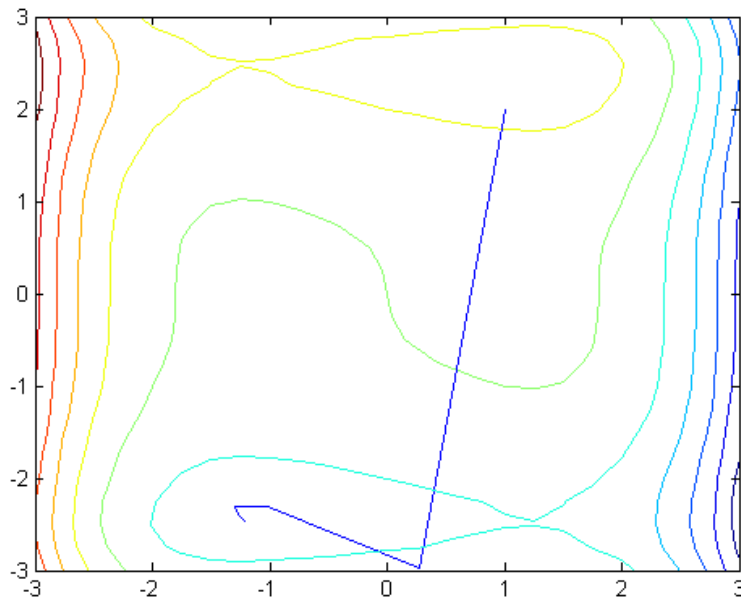
f(-0.001,-0.000)=0.000

f(-0.000,-0.000)=0.000



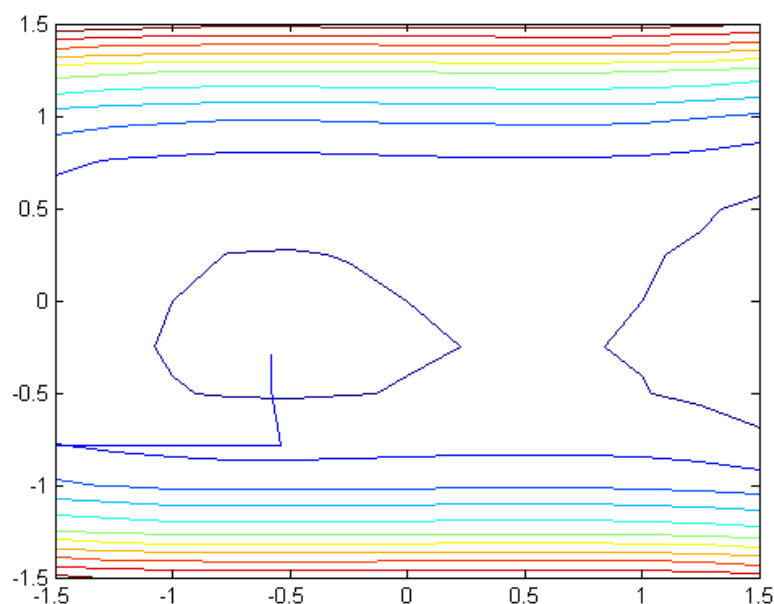
Rysunek 1: Wykres kolejnych przybliżeń dla funkcji $x^2 + y^2$.

Testowana funkcja : $x + 10 * x - x^5 + y + 10 * y^3 - y^5$
Dla Argumentów : $[1; 2], 'x + 10 * x - x^5 + y + 10 * y^3 - y^5', 0.6, 0.99, [0.6; 0.4], 15, 0.00001$
kolejne przybliżenia:
 $f(0.272, -2.977) = 60.000$
 $f(-1.014, -2.301) = -30.053$
 $f(-1.296, -2.313) = -69.709$
 $f(-1.296, -2.313) = -70.454$
 $f(-1.254, -2.415) = -70.454$
 $f(-1.234, -2.444) = -71.809$
 $f(-1.225, -2.452) = -71.942$
 $f(-1.221, -2.454) = -71.955$
 $f(-1.219, -2.456) = -71.958$
 $f(-1.218, -2.456) = -71.958$
 $f(-1.218, -2.456) = -71.958$
 $f(-1.218, -2.456) = -71.958$
 $f(-1.218, -2.456) = -71.958$
 $f(-1.218, -2.456) = -71.958$
 $f(-1.218, -2.456) = -71.958$



Rysunek 2: Wykres kolejnych przybliżeń dla funkcji
 $x + 10 * x - x^5 + y + 10 * y^3 - y^5$.

Testowana funkcja : $x - x^3 + y + 10 * y^4$
Dla Argumentów : $[-2; -2], 'x - x^3 + y + 10 * y^4', 0.6, 0.99, [0.6; 0.4], 15, 0.00001$
kolejne przybliżenia:
 $f(-1.958, -0.792) = 164.000$
 $f(-0.538, -0.784) = 8.694$
 $f(-0.538, -0.784) = 2.608$
 $f(-0.558, -0.635) = 2.608$
 $f(-0.570, -0.544) = 0.608$
 $f(-0.577, -0.466) = -0.055$
 $f(-0.581, -0.406) = -0.380$
 $f(-0.581, -0.362) = -0.519$
 $f(-0.580, -0.332) = -0.575$
 $f(-0.579, -0.313) = -0.596$
 $f(-0.578, -0.302) = -0.602$
 $f(-0.578, -0.297) = -0.604$
 $f(-0.578, -0.294) = -0.604$
 $f(-0.577, -0.293) = -0.604$
 $f(-0.577, -0.293) = -0.604$



Rysunek 3: Wykres kolejnych przybliżeń dla funkcji $x - x^3 + y + 10 * y^4$.

$f(x, y)$	punkt	wartość w punkcie
$x^2 + y^2$	(0,0)	0
$x + 10 * x - x^5 + y + 10 * y^3 - y^5$	(-1.218,-2.456)	-71.958
$x - x^3 + y + 10 * y^4$	(-0.577,-0.293)	-0.604

Tablica 1: Zestawienie wyników uzyskanych przy pomocy WolframAlpha

5. Wnioski

W problemie optymalizacji funkcji wielu zmiennych, możemy mieć przypadki trudne do rozstrzygnięcia posiadając jedynie informacje o przebiegu wartości pierwszej pochodnej w punkcie. Przypadkiem takim może być punkt przebiegu który jest punktem ekstremalnym ale nie jest on ani minimum ani maksimum. Wartość przybliżona macierzy Hessego w punkcie rozstrzyga ten problem. Jednakowo wprowadza nową informację dotyczącą przebiegu badanej funkcji - jej krzywiznę.

Zaprezentowana tutaj metoda wyznacza wartość przybliżoną macierzy Hessego metodą Davidon-Fletcher-Powell [2], jak widać na prezentowanych rzutach funkcji udało znaleźć się lokalne minima i z zadowalającą dokładnością (zgodnie w wynikami wzorcowymi zawartymi w tabeli 1.

Literatura

- [1] http://en.wikipedia.org/wiki/Quasi-Newton_method

- [2] http://en.wikipedia.org/wiki/DFP_updating_formula
- [3] http://en.wikipedia.org/wiki/Wolfe_conditions