Informatyka, studia dzienne, mgr II st. semestr I

Metody obliczeniowe optymalizacji **2011/2012**

Prowadzący: mgr inż. Łukasz Chomątek czwartek, 16:00

Data oddania: _____ Ocena: _____

Paweł Musiał 178726
Łukasz Michalski 178724

Zadanie 2: Optymalizacja kierunkowa*

1. Cel

Celem zadania było napisać program, który dla dowolnej funkcji dwóch zmiennych rozwiąże zadanie optymalizacji na odcinku. Optymalizacja kierunkowa musi być przeprowadzona z wykorzystaniem kryteriów:

- Armijo
- Wolfa
- Goldsteina

Przedstawiany jako rozwiązanie program powinien pozwolić wprowadzić funkcję oraz odcinek, w którym poszukiwane będzie rozwiązanie.

2. Rozwiązanie zadania

2.1. Metoda najszybszego spadku

Metoda najszybszego spadku jest iteracyjnym algorytmem wyszukiwania minimum zadanej funkcji celu f. Założenia dla metody są następujące:

- $f \in C^1$ (funkcja jest ciągła i różniczkowalna),
- f jest ściśle wypukła w badanej dziedzinie.

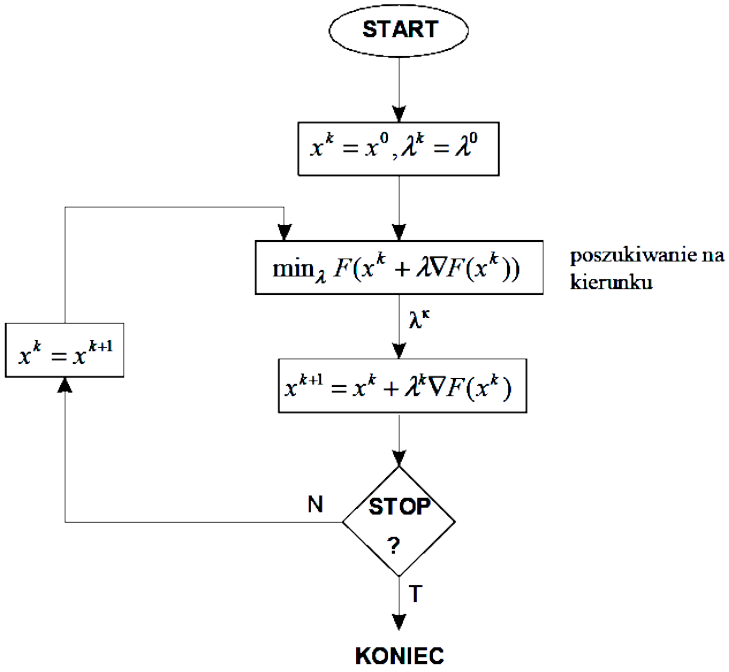
Poszukiwania te odbywają się na podstawie gradientu tej funkcji. Wiadomo, że gradient $\bigtriangledown f = [\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n}]^T$ ma ważną własność mówiącą że poruszając

* SVN: `https://serce.ics.p.lodz.pl/svn/labs/moo/lc_cz1600/lmpm`

się z dowolnego punktu x „w kierunku gradientu" osiągamy (lokalnie) najszybszy przyrost funkcji. W myśl tej zasady, jeśli $\bigtriangledown f$ wyznacza najszybszy wzrost, to $-\bigtriangledown f$ (antygrandient) wyznacza najszybszy spadek. Na tym spostrzeżeniu opiera się metoda najszybszego spadku, którą w skrócie można ja opisać następująco:

1. Znajdź najlepszy kierunek (kierunek najszybszego spadku),
2. Określ jak daleko chcesz „zrobić krok" w tym kierunku,
3. Zrób krok i sprawdź warunek stopu.

Problemem występującym przy zastosowaniu metody najszybszego spadku jest jej „spowolnienie", gdy zbliża się do minimum (zmiany zmiennych zależna od wielkości gradientu, a gradient dąży do zera w otoczeniu punktu minimum). Algorytm działania tej metody został przedstawiony na diagramie:

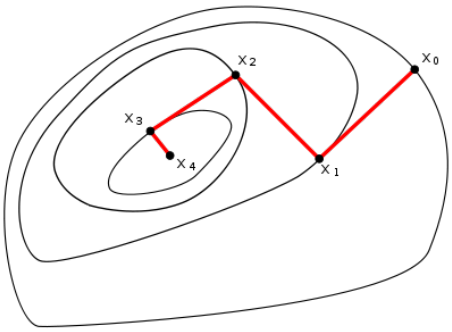


Rysunek 1: Schemat działania metody najszybszego spadku.
Źródło: Wykład 4 Wojciecha Grega: Metody Optymalizacji

Jak można zauważyć ważnym elementem tego algorytmu jest wybór odpowiedniej długość użytego kroku oraz kryterium stopu. Ma to bowiem wpływ na szybkość i stabilność działania oraz na osiągnięte wyniki. W tym zadaniu skupimy się na trzech kryteriach doboru optymalnego kroku:

- Armijo
- Wolfa
- Goldsteina

Na rysunku 2 można zobaczyć przykład działania metody najszybszego spadku dla dwuwymiarowej funkcji celu. W każdym kroku, w zadanym kierunku wyszukiwana jest najmniejsza wartość funkcji celu.



Rysunek 2: Przebieg działania metody najszybszego spadku.
Źródło: `http://pl.wikipedia.org/wiki/Plik:Metoda_najszybszego_spadku.svg`

W opisach kryteriów zbieżności posługiwać będziemy się poniższymi oznaczeniami :

- 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

2.2. Kryterium Armijo

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

Warunek ten oznacza, że redukcja wartości funkcji celu powinna być proporcjonalna do długości kroku jak również do pochodnej kierunkowej. Metoda ta jednak nie zapewnia nam szybkiej zbieżności, ponieważ zawsze może być spełniony dla dostatecznie małej wielkości λ.

2.3. Kryterium Wolfa

Kryterium Wolfa, jest rozszerzeniem kryterium poprawy Armijo, o kryterium krzywizny:

- Słabe kryterium krzywizny

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

- Silne kryterium krzywizny

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

Kryterium to zapewnia nam, że gdy w punkcie początkowym jest duży spadek, nie będzie wykonany mały krok.

2.3.1. Słabe kryterium Wolfa

$$\nabla f^T(x^k+\lambda^k d^k)d^k \geq c\nabla f^T(x^k)d^k \;\; \&\& \;\; f(x^k+\lambda^k d^k) \leq f(x^k)+c\lambda^k\nabla f^T(x^k)d^k \tag{4}$$

Jest to kryterium Armijo z dodatkowym założeniem słabego kryterium krzywizny.

2.3.2. Silne kryterium Wolfa

$$\|\nabla f^T(x^k+\lambda^k d^k)d^k\| \leq \|c\nabla f^T(x^k)d^k\| \;\; \&\& \;\; f(x^k+\lambda^k d^k) \leq f(x^k)+c\lambda^k\nabla f^T(x^k)d^k \tag{5}$$

Jest to kryterium Armijo z dodatkowym założeniem silnego kryterium krzywizny.

2.4. Kryterium Goldsteina

$$f(x^k) + (1-c)\lambda^k\nabla f(x^k)d^k \leq f(x^k+\lambda^k d^k) \leq f(x^k) + c\lambda^k\nabla f(x^k)d^k \tag{6}$$

Kryterium Goldsteina, podobnie jak w przypadku kryterium Wolfa, zabezpiecza nas przed zbyt małym krokiem, który mógłby niepotrzebnie zwiększyć ilość potrzebnych iteracji. Lewa i prawa część nierówności zapewnia nam ograniczenie odpowiednio dolnej i górnej granicy kroku.

2.5. Algorytm backtrackingu

Jest metodą wyznaczania długości kroku na podstawie zadanego kryterium.

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 w całości jest implementacją powyżej opisanych metod. Wyjaśnienie znaczenia zmiennych :

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

3.1. Metoda najszybszego spadku

```
function x=steepest_descent(x0,sl,fn,a0,rho,c)
f=inline(fn);
f0=@(x)f(x(1),x(2));
t=[x;y];
w=fn;
gfn=matlabFunction(jacobian(w,t).');
f1=@(x)gfn(x(1),x(2));
while k<10 %warunek zakonczenia, liczba iteracji
%kierunek
    p=-f1(x);
    %dlugosc kroku spelniajaca kryterium
    a=kryterium(x,p);
    %kolejne przyblizenie
    x=x+a*p
    %licznik iteracji
    k=k+1;
end
```

Metoda najszybszego spadku

3.2. Kryterium Armijo

```
if f0(x+a*p)<=f0(x)+c*a*f1(x).'*p
    test=1;
else
    test=0;
end
```

Kryterium Armijo

3.3. Kryterium Wolfa

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

Slabe kryterium Wolfa

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

Silne kryterium Wolfa

3.4. Kryterium Goldsteina

```
if  f0(x+a*p)<=f0(x)+c*a*f1(x)'*p &&
 f0(x+a*p)>=f0(x)+(1.0-c)*a*f1(x)'*p
    test=1;
else
    test=0;
end
```

Kryterium Goldsteina

3.5. Algorytm backtrackingu

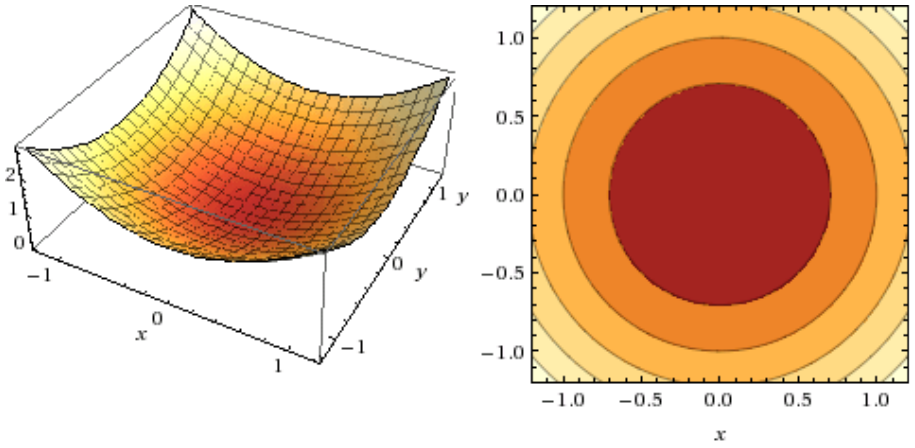
```
switch cond
    case 'armijo'
        check=@(a)armijo(a,x,p,c,f0,f1);
    case 'goldstein'
        check=@(a)goldstein(a,x,p,c,f0,f1);
    case 'wolfe'
        check=@(a) wolfe(a,x,p,c(1),f1) &&
                armijo(a,x,p,c(2),f0,f1);
    case 'wolfes'
        check=@(a) wolfes(a,x,p,c(1),f1) &&
                armijo(a,x,p,c(2),f0,f1);
end
a=a0;
while check(a)~=1  %poszukiwanie dlugosci kroku dla ktorej
                   %zachodzic bedzie dane kryterium
    a=rho*a;
end
```

Algorytm backtrackingu

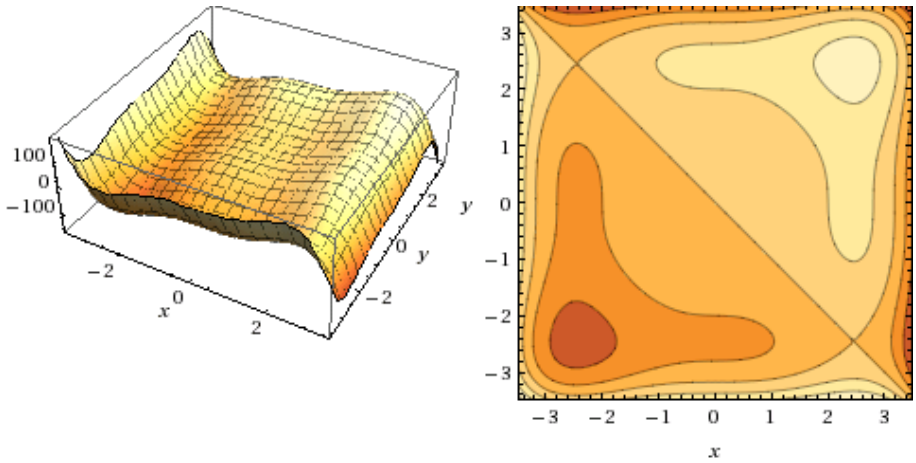
4. Wyniki

W sekcji tej zaprezentowano wyniki badań poszczególnych metod. Zastosowane zostały skróty odnośnie funkcji testowych :

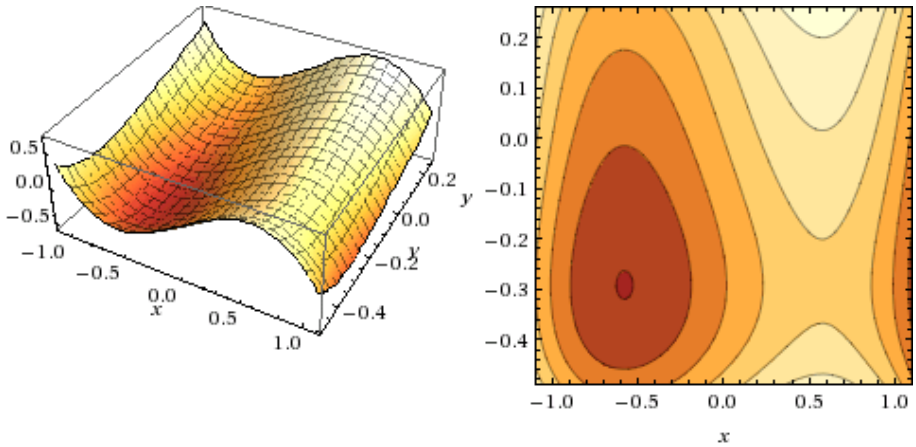
1. $x^2 + y^2$
2. $x + 10x^3 - x^5 + y + 10y^3 - y^5$
3. $x - x^3 + y + 10y^4$



Rysunek 3: Wykres i rzut funkcji $x^2 + y^2$.
Źródło: WolframAlpha



Rysunek 4: Wykres i rzut funkcji $x + 10x^3 - x^5 + y + 10y^3 - y^5$.
Źródło: WolframAlpha



Rysunek 5: Wykres i rzut funkcji $x - x^3 + y + 10y^4$.
Źródło: WolframAlpha

Uzyskane wyniki dla poszczególnych funkcji stosując różne metody zawarto w tabeli:

$f(x,y)$	Kryterium	x_o	a_0	rho	c	k_{max}	Wynik
1	Armijo	(100,100)	0.41	0.5	0.5	5	(0.0189,0.0189)
1	słabe Wolfe	(100,100)	0.41	0.5	(0.6;0.1)	5	(0.0189,0.0189)
1	silne Wolfe	(100,100)	0.41	0.5	(0.6;0.1)	5	(0.0189,0.0189)
1	Goldstein	(100,100)	0.41	0.5	0.3	5	(0.0189,0.0189)
2	Armijo	$(-1,-1)$	0.2	0.5	0.5	5	(-2.4562,-2.4562)
2	słabe Wolfe	(-1,-1)	0.2	0.5	(0.8;0.5)	5	(-2.4562,-2.4562)
2	silne Wolfe	(-1,-1)	0.2	0.5	(0.8;0.5)	5	(-2.4562,-2.4562)
2	Goldstein	(-1,-1)	0.2	0.5	0.4	5	(-2.4562,-2.4562)
3	Armijo	(-2,-2)	0.3	0.5	0.4	5	(-0.5578,-0.3006)
3	słabe Wolfe	(-2,-2)	0.3	0.5	(0.6;0.4)	5	(-0.5578,-0.3006)
3	silne Wolfe	(-2,-2)	0.3	0.5	(0.6;0.4)	5	(-0.5578,-0.3006)
3	Goldstein	(-2,-2)	0.3	0.5	0.39	5	(-0.5578,-0.3006)

Tablica 1: Zestawienie wyników z badań metod.

$f(x,y)$	Wynik
x^2+y^2	(0,0)
$x+10x^3-x^5+y+10y^3-y^5$	(-2.4563,-2.4563)
$x-x^3+y+10y^4$	(-0.5774,-0.2924)

Tablica 2: Zestawienie wyników uzyskanych przy pomocy WolframAlpha

.

5. Wnioski

Znaczący wpływ przy doborze parametrów dla wszystkich funkcji mają parametry λ oraz c. Zależność wartości parametru c dla trzech kryteriów można przedstawić jako nierówność :

$$C_{Wolfe} >= C_{Armijo} >= C_{Goldstein} \tag{7}$$

Oba te parametry mają duży wpływ na szybkość zbieżności do wyniku.
Metody Wolfe oraz Goldsteina są metodami bardziej restrykcyjnymi od metody Armijo pod względem doboru wielkości kroku, nie pozwalają na zbyt małe kroki w poszczególnych iteracjach algorytmu, co jednak z drugiej strony może powodować rozbieżność metody poszukiwania optimum. Pod tym względem metoda Armijo wydaje się być wygodniejszą, ponieważ nie jest aż tak mocno zależna od dobranych parametrów, innymi słowy trudno w tym przypadku aby nie dała wyniku. Co praktycznie pozbawia ją pesymistycznych przypadków, jakie możemy napotkać używając metod Wolfe oraz Goldsteina. W przypadku metody Wolfe przy złym doborze parametrów, utkniemy

w nieskończonej pętli algorytmu backtrakingu, z powodu niemożliwości zajścia kryterium. Natomiast w przypadku kryterium Goldsteina pomimo spełnienia kryterium, wynik może okazać się rozbieżny z oczekiwaniami, dotyczy to głównie wartości $C_{Goldstein} \notin (0, 0.5)$.

Literatura

[1] Michał Lewandowski, *Metody optymalizacji - teoria i wybrane algorytmy*. 2012.