Sprawozdanie z laboratorium Teorii Optymalizacji

Imię i nazwisko	Jacek Gołda
Temat ćwiczenia	Metody gradientowe
Data i godzina wykonania ćwiczenia	23 marca 2016, 14:00

1 Wstęp

Celem laboratorium było zbadanie metod gradientowych, w szczególności porównanie efektywności i wyników ich działania.

2 Ćwiczenie 1

Treść zadania:

Obserwacja zjawiska zygzakowania
Należy przeanalizować wpływ wydłużenia zbiorów poziomicowych funkcjonału kwadratowego i punktu startowego na tempo zbieżności metody najszybszego spadku. Badania przeprowadzić dla funkcjonału:

$$Q(x_1, x_2) = x_1^2 + a \cdot x_2^2$$

Wybierając kolejno $a = 1$, $a = 2n$, $a = 0.01n$

Za parametr n przyjęto wartość 4.

Rozwiązanie:

2.1 Rozwiązanie analityczne

Zauważono, że funkcjonał celu jest sumą dwóch wyrażeń nieujemnych, zerujących się w zerze. Wynika stąd, że minimum funkcjonału będzie osiągane w punkcie $\widehat{x} = [0, 0]^T$ i wartość minimalna funkcji będzie wynosiła 0.

2.2 Rozwiązanie numeryczne

W celu zrealizowania rozwiązania numerycznego, konieczne było zmodyfikowanie funkcji obliczających wartość funkcjonału celu i gradientu funkcjonału celu. Gradient obliczono za pomocą toolboxu do obliczeń symbolicznych. Zmodyfikowano również pliki, w których te funkcje były wywoływane (dodano argument `a` do wywołań) — nie zamieszczono tych plików w sprawozdaniu, ponieważ niewiele by do niego wnosiły)

Listing 1: Wartość funkcjonału celu

```
function [q,x]=koszt(a, x,z,d)

if nargin==3, x=x+z;
elseif nargin==4, x=x+z*d;
end
q=x(1)^2+a*x(2)^2;
```

Listing 2: Gradient funkcjonału celu

```
function g=gradie(a, x)

g=[2*x(1)
   2 * a * x(2)];
```

W celu łatwiejszego rysowania wyników napisano dwie funkcje:

Listing 3: Funkcja rysująca mapę poziomic

```
function figureHandleOut=rysujMape(figureHandle, x1, x2, q, wektor_poziomic)
    figure(figureHandle);
    hold on;
    % rysuj mapĆ poziomic
    contour(x1, x2, q, wektor_poziomic, 'ShowText', 'on');

    figureHandleOut = figureHandle;
end
```

Listing 4: Funkcja rysująca pojedyncze rozwiązanie

```
function figureHandleOut = rysujRozwiazanie(figureHandle, x_rozw)
    % rysuj kolejne linie laczace kolejne przyblizenia
    figure(figureHandle);
    hold on;
    plot(x_rozw(1, :), x_rozw(2, :), 'r');
    plot(x_rozw(1, :), x_rozw(2, :), 'rd');

    % rysuj pierwszy i ostatni punkt
    plot(x_rozw(1, 1), x_rozw(2, 1), 'b*');
    plot(x_rozw(1, size(x_rozw, 2)), x_rozw(2, size(x_rozw, 2)), 'bo');

    figureHandleOut = figureHandle;
end
```

Następnie napisano skrypt minimalizujący funkcjonał i rysujący rozwiązania:

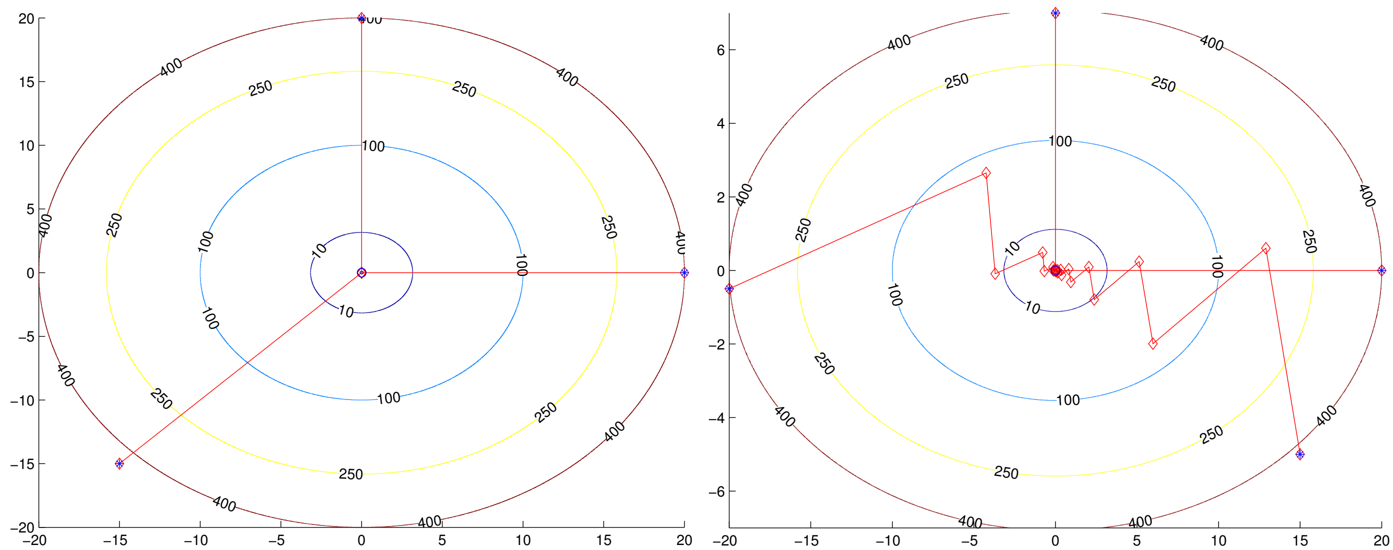
```
clear all;
close all;
clc

% wspolczynnik wystepujacy w zadaniu
par = 0; % rodzaj metody - najszybszego spadku
n = 4;

% DLA WSPOLCZYNNIKA a = 1
a = 1;

% rysowanie poziomic
[x1, x2] = meshgrid(-20:0.01:20, -20:0.01:20);
q = x1.^2 + a * x2 .^ 2;
poziomice = [0.01, 10, 100, 250, 400];

w_x0 = [20, 0;
        -15, -15;
        0, 20];

figureHandle = figure;
figureHandle = rysujMape(figureHandle, x1, x2, q, poziomice);
for i=1:size(w_x0, 1)
    x0 = w_x0(i, :)';
```

```
    granad
    figureHandle = rysujRozwiazanie(figureHandle, x_rozw);
    display(sprintf('ilosc iteracji (lacznie z x0) to: %d', size(x_rozw, 2)));
end
print -depsc2 'wykres_a_1.eps'




% DLA WSPOLCZYNNIKA a = 8 (2n)
a = 2*n;

% rysowanie poziomic
[x1, x2] = meshgrid(-20:0.01:20, -7:0.01:7);
q = x1.^2 + a * x2 .^ 2;
poziomice = [0.01, 10, 100, 250, 400];

w_x0 = [20, 0;
        15, -5;
        0, 7;
        -20, -0.5];

figureHandle = figure;
figureHandle = rysujMape(figureHandle, x1, x2, q, poziomice);
for i=1:size(w_x0, 1)
    x0 = w_x0(i, :)';
    granad
    figureHandle = rysujRozwiazanie(figureHandle, x_rozw);
    display(sprintf('ilosc iteracji (lacznie z x0) to: %d', size(x_rozw, 2)));
end
print -depsc2 'wykres_a_8.eps'




% DLA WSPOLCZYNNIKA a = 0.04;
a = 0.01*n;

% rysowanie poziomic
[x1, x2] = meshgrid(-10:0.01:10, -40:0.01:40);
q = x1.^2 + a * x2 .^ 2;
poziomice = [0.01, 3, 10,  30, 70, 100];

w_x0 = [10, 0;
        -8, 30;
        -10, -5;
        0, 40];

figureHandle = figure;
figureHandle = rysujMape(figureHandle, x1, x2, q, poziomice);
for i=1:size(w_x0, 1)
    x0 = w_x0(i, :)';
    granad
    figureHandle = rysujRozwiazanie(figureHandle, x_rozw);
    display(sprintf('ilosc iteracji (lacznie z x0) to: %d', size(x_rozw, 2)));
end
print -depsc2 'wykres_a_1.eps'
```

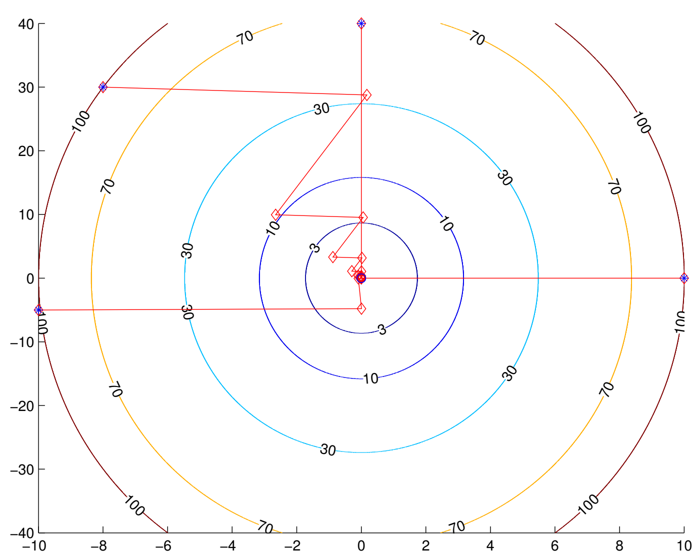
Dla kolejnych wartości wspólczynników a badano następujące punkty początkowe:

Współczynnik a	x_1	x_2
1	20	0
	-15	-15
	0	20
8	20	0
	15	-5
	0	7
	-20	-0.5
0.04	10	0
	-8	30
	- 10	-5
	0	40

Uzyskano następujące wyniki:



(a) $a = 1$ (b) $a = 8$



Rysunek 2: $a = 0.04$

Z wykresów można wyciągnąć następujące wnioski:

1. Gdy poziomice są okręgami, algorytm znajduje minimum w pierwszej iteracji bez względu na dobór punktu początkowego, ponieważ dla dowolnego punktu początkowego pierwsza wyznaczona prosta przechodzi przez punkt w którym znajduje się minimum i w efekcie algorytm kończy pracę po pierwszej iteracji.
2. W przypadku kiedy zbiory poziomicowe są elipsami, dobór punktu początkowego jest bardzo istotny. W najgorszym przypadku algorytm przebiega po zygzakowatej ścieżce, co powoduje wzrost liczby wykonanych iteracji. Odpowiedni dobór warunku początkowego ponownie umożliwia znalezienie minimum w jednej iteracji, jednak taki dobór nie jest oczywisty (w tym celu trzeba w zasadzie znać położenie minimum, co wyklucza praktyczne zastosowanie takiego podejścia). Aby uzyskać taki efekt, punkt początkowy musi znajdować się na osiach elips będących poziomicami funkcjonału celu.
3. W przypadku, kiedy poziomice nie są okręgami, algorytm szybciej znajduje minimum, gdy punkt początkowy jest blisko osi elipsy.

3 Ćwiczenie 2

Treść zadania:

Badanie porównawcze gradientowych metod optymalizacji
Badanie porównawcze gradientowych metod optymalizacji przeprowadzić dla funkcjonału:

$$Q(x_1, x_2) = 6x_1^2 + 6x_1x_2 + x_2^2 + 4.5\big(e^{x_1} - x_1 - 1\big) + 1.5\big(e^{x_2} - x_2 - 1\big)$$

Przyjąć, że norma gradientu w punkcie zatrzymania algorytmu ma być nie większa niż 0.001. Pole obserwacji graficznej: $|x_1| \leqslant 3$, $|x_2| \leqslant 3$. Punkt startowy: $(-3, 3)$ (dla metody największego spadku wypróbować też punktu $(-3, 1)$). Maksymalna liczba iteracji: 20. Maksymalny czas obserwacji: 7 minut.

Wszystkie wartości x_1, x_2 przesunięte są o $3n$

Za parametr n przyjęto wartość 4.

3.1 Rozwiązanie numeryczne

Ponownie zmodyfikowano funkcje obliczające wartość funkcjonału celu i jego gradientu (ponownie skorzystano z toolboxu do obliczeń symbolicznych). Korzystano z plików zmodyfikowanych przy okazji poprzedniego ćwiczenia, stąd nadal przekazywany jest parametr `a`, jednak jest on ignorowany. Skorzystano również z funkcji rysujących poziomice funkcjonału celu i kolejne przybliżenia rozwiązania optymalnego wykorzystywane w poprzednim zadaniu.

Listing 5: Wartość funkcjonału celu

```
function [q,x]=koszt(a, x,z,d)

if nargin==3, x=x+z;
elseif nargin==4, x=x+z*d;
end

n = 4;
x1 = x(1) - 3 * n;
x2 = x(2) - 3 * n;

q=6*x1 ^2 + 6*x1*x2 + x2 ^ 2 + ...
    4.5*(exp(x1)    - x1 - 1) + ...
    1.5 * (exp(x2) - x2 - 1);
```

Listing 6: Gradient funkcjonału celu

```
function g=gradie(a, x)

n = 4;
x = x - 3 * n;
x1 = x(1);
x2 = x(2);

g=[12*x1 + 6*x2 + (9*exp(x1))/2 - 9/2
  6*x1 + 2*x2 + (3*exp(x2))/2 - 3/2];
```

Napisano również funkcję opracowującą wyniki — obliczającą moduły różnic argumentów i różnice funkcji

Listing 7: Funkcja opracowująca wyniki

```
function [argumenty, wartosci] = opracuj(arg, wart)
    ilosc_iteracji = size(arg, 2);
    ostatni_argument = arg(:, ilosc_iteracji);
    modul_x = arg - ostatni_argument * ones(1, ilosc_iteracji);
    argumenty = sqrt(modul_x(1, :) .^ 2 + modul_x(2, :) .^ 2 );
    argumenty = argumenty';
    ostatnia_wartosc = wart(ilosc_iteracji);
    wartosci = wart - ostatnia_wartosc * ones(1, ilosc_iteracji);
    wartosci = wartosci';
end
```

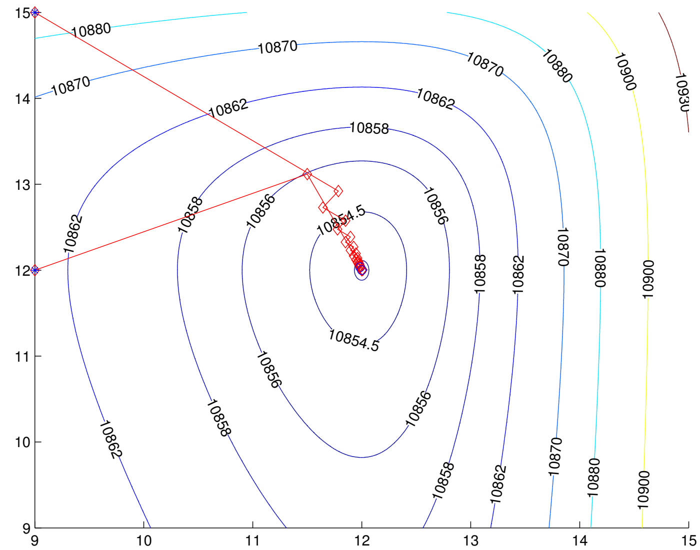
Porównywano metodę najszybszego spadku, metodę Fletchera-Reeves'a, metodę Polaka-Ribier'a i metodę z pełną formułą na współczynnik β. Napisano skrypt służący do rozwiązywania zadania.

```
clear all;
close all;
clc

% wspolczynnik wystepujacy w zadaniu
n = 4;
a = 0;
wartosci_poziomic = [
    10851,
    10854.07,
    10854.5,
    10856,
    10858,
    10862,
    10870,
    10880,
    10900,
    10930];

[z1, z2] = meshgrid(-1:0.01:1, -1:0.01:1);
q=30 * n * (z2 - z1 .^ 2 ) .^ 2 + (1 - z1 ).^ 2;




% METODA NAJSZYBSZEGO SPADKU

% rysowanie poziomic
figureHandle = figure;
figureHandle = rysujMape(figureHandle, z1, z2, q, wartosci_poziomic);

% wywolanie metody
```

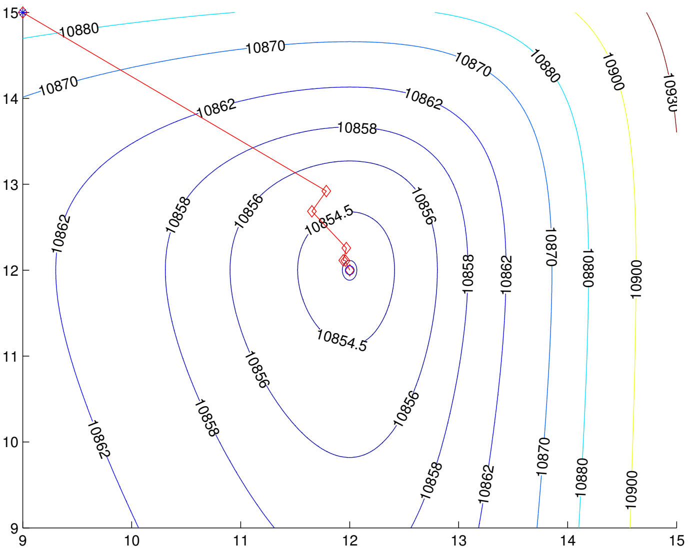
```
x0 = [9; 15];
par = 0;
granad

% opracowanie wynikow
[najszybszy_modul_x, najszybszy_roznica_q] = opracuj(x_rozw, q_rozw);

% rysuj punkty i linie
figureHandle = rysujRozwiazanie(figureHandle, x_rozw);
display(sprintf('ilosc iteracji to: %d', size(x_rozw, 2) - 1));
print -depsc2 'wykres_najszybszy_spadek_5.eps'




% METODA FLETCHERA-REEVESA

% rysowanie poziomic
figureHandle = figure;
figureHandle = rysujMape(figureHandle, z1, z2, q, wartosci_poziomic);

% wywolanie metody
x0 = [9; 15];
par = 1;
granad

% opracowanie wynikow
[fletcher_modul_x, fletcher_roznica_q] = opracuj(x_rozw, q_rozw);
figureHandle = rysujRozwiazanie(figureHandle, x_rozw);
display(sprintf('ilosc iteracji to: %d', size(x_rozw, 2) - 1));
print -depsc2 'wykres_fletcher_5.eps'


% METODA POLAKA-RIBIERA

% rysowanie poziomic
figureHandle = figure;
figureHandle = rysujMape(figureHandle, z1, z2, q, wartosci_poziomic);

% wywolanie metody
x0 = [9; 15];
par = 2;
granad

% opracowanie wynikow
[polak_modul_x, polak_roznica_q] = opracuj(x_rozw, q_rozw);

% rysuj punkty i linie
figureHandle = rysujRozwiazanie(figureHandle, x_rozw);
display(sprintf('ilosc iteracji to: %d', size(x_rozw, 2) - 1));
print -depsc2 'wykres_polak_5.eps'


% METODA Z PELNA FORMULA

% rysowanie poziomic
figureHandle = figure;
figureHandle = rysujMape(figureHandle, z1, z2, q, wartosci_poziomic);

% wywolanie metody
x0 = [9; 15];
par = 3;
```

```
granad

% opracowanie wynikow
[pelna_modul_x, pelna_roznica_q] = opracuj(x_rozw, q_rozw);

% rysuj punkty i linie
figureHandle = rysujRozwiazanie(figureHandle, x_rozw);
display(sprintf('ilosc iteracji to: %d', size(x_rozw, 2) - 1));
print -depsc2 'wykres_pelna_5.eps'
```

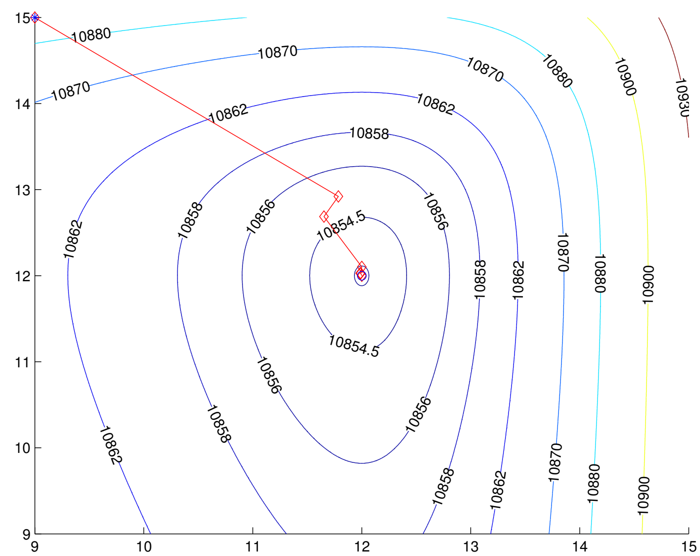
Uzyskano następujące wyniki:



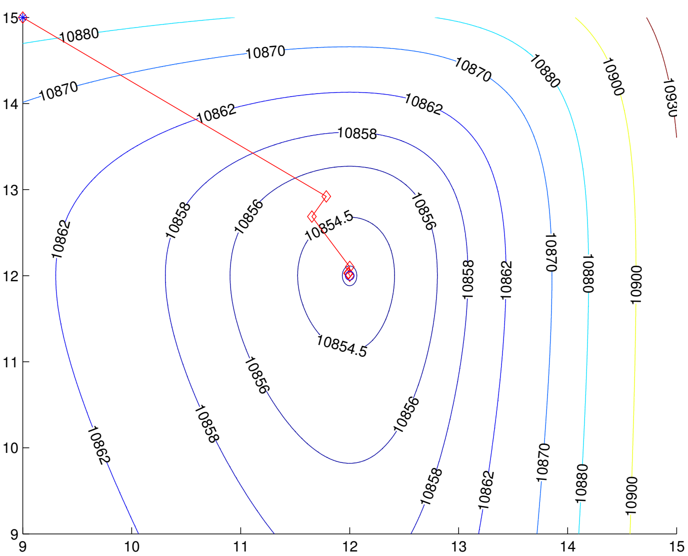
(a) Metoda najszybszego spadku



(b) Metoda Fletchera-Reevesa



(a) Metoda Polaka-Ribiera



(b) Metoda z pełną formułą na współczynnik β

Ilości wykonywanych iteracji zestawiono w tabeli:

	Metoda			
	Najszybszego spadku	Fletchera-Reeves'a	Polaka-Ribier'a	Z pełną formułą na β
Ilość iteracji	20 (4)	9	7	7

Wartość w nawiasie przy metodzie najszybszego spadku to ilość iteracji wykonana przy przesuniętym punkcie początkowym. Widać znaczą poprawę.

Kolejne wartości modułów różnic wartości bieżącego przybliżenia i znalezionego najlepszego przybliżenia, a także różnic wartości funkcji celu bieżącego przybliżenia i najlepszego przybliżenia przedstawiono w tabelkach:

Metoda najszybszego spadku

Wartość modułu różnicy argumentów	Różnica wartości funkcji celu
4,20660182789998	42,3514522107387
0,907425515151682	0,919971635443480
0,772284844692517	0,503997239651566
0,555704538046889	0,300434552137844
0,485427188345306	0,190382609169855
0,363007390036942	0,124651990288822
0,319882017387590	0,0839219550345457
0,242597842856503	0,0574118536849281
0,214034335879526	0,0398455688256577
0,162413896293847	0,0278674454859224
0,142638752684569	0,0196175701931699
0,107030656645602	0,0138311817355089
0,0929198452644613	0,00974779352005343
0,0679063593723401	0,00683092079103766
0,0575787336595769	0,00473909681331831
0,0398994009075436	0,00322628806707079
0,0321086286539421	0,00212932970792056
0,0198340942331221	0,00132917218048788
0,0135761215732276	0,000744446759856584
0,00675915147154743	0,000315327205169747
0	0

Dla tej metody widać bardzo duży spadek w pierwszej iteracji, następnie algorytm zwalnia i dalej postępuje dość niemrawo. Algorytm kończy pracę po 20 iteracjach — może to oznaczać, że zadziałało kryterium stopu związane z liczbą iteracji, a nie z osiągnięciem celu, czyli minimum (z żądaną dokładnością).

Metoda Fletcher'a-Reeves'a

Wartość modułu różnicy argumentów	Różnica wartości funkcji celu
4,24265658098509	42,3523471922627
0,945473072150500	0,920866616967461
0,767550960757315	0,455133334661053
0,259639990951523	0,0813358554522547
0,128477878406227	0,0113407057282352
0,116475012421919	0,00830271357868925
0,000230842365139998	3,13321293871806e-07
0,000141176770018257	1,43443922420049e-08
8,83592448964526e-06	7,38384575822758e-10
0	0

Tutaj również widać bardzo duży skok w pierwszej iteracji, jednak w tym wypadku algorytm zmierza z początku znacznie szybciej do minimum, w dalszych krokach jednak postępuje wolniej ale ostatecznie kończy pracę w wyniku kryterium stopu związanego z osiągnięciem celu, a nie z liczbą iteracji. Oznacza to, że algorytm lepiej sprawdza się dla tej funkcji niż algorytm największego spadku.

Metoda Polaka-Ribier'a

Wartość modułu różnicy argumentów	Różnica wartości funkcji celu
4,24264503100344	42,3523471924241
0,945460464434991	0,920866617128817
0,769709273630262	0,457528357632888
0,101553161910448	0,0187102626113725
0,0316554269687062	0,000664010384927144
0,0288253232505325	0,000495144577678723
1,25610697275486e-05	1,15209183956170e-09
0	0

Ten algorytm również kończy pracę w wyniku osiągnięcia celu, a nie wyczerpania możliwej liczby iteracji. Algorytm kończy pracę już w siódmej iteracji, czyli o dwie iteracje wcześniej niż algorytm Fletcher'a-Reeves'a.

Metoda z pełną formułą na współczynnik β

Wartość modułu różnicy argumentów	Różnica wartości funkcji celu
4,24264509941824	42,3523471924236
0,945460536769351	0,920866617128408
0,769676522067688	0,457492078880393
0,102230052130064	0,0189115977508487
0,0318457892110010	0,000672466662304072
0,0290013005912142	0,000501174550702466
1,28457034367026e-05	1,20606214550418e-09
0	0

Metoda kończy pracę w siódmej iteracji, podobnie jak metoda Polaka-Ribier'a. Różnice wartości osiągane w kolejnych iteracjach są bardzo do siebie zbliżone w tych dwóch metodach.

Fakt, że kolejne zmiany osiągane w trzech ostatnich metodach są do siebie zbliżone widać również bardzo dobrze na umieszczonych powyżej wykresach.

4 Ćwiczenie 5

Treść zadania:

„Dolina bananowa” Rossenbrocka. Wyznaczyć minimum funkcji:

$$Q(x_1, x_2) = 30n(x_2 - x_1^2)^2 + (1 - x_1)^2$$

na mapę poziomic doliny nanieść punkty pośrednie poszczególnych kroków.

Za parametr n przyjęto wartość 4.

Rozwiązanie:

4.1 Rozwiązanie analityczne

Zauważono, że funkcja składa się z dwóch nieujemnych wyrażeń. Po przyrównaniu ich do zera, uzyskano, że minimum będzie znajdować się w punkcie $\widehat{x} = [1, 1]^T$

4.2 Rozwiązanie numeryczne

Zmodyfikowano funkcje obliczające wartość funkcji celu i gradientu:

Listing 8: Wartość funkcjonału celu

```
function [q,x]=koszt(a, x,z,d)

if nargin==3, x=x+z;
elseif nargin==4, x=x+z*d;
end

n = 4;
x1 = x(1);
x2 = x(2);

q=30 * n * (x2 - x1 .^ 2 ) .^ 2 + (1 - x1 ).^ 2;
```

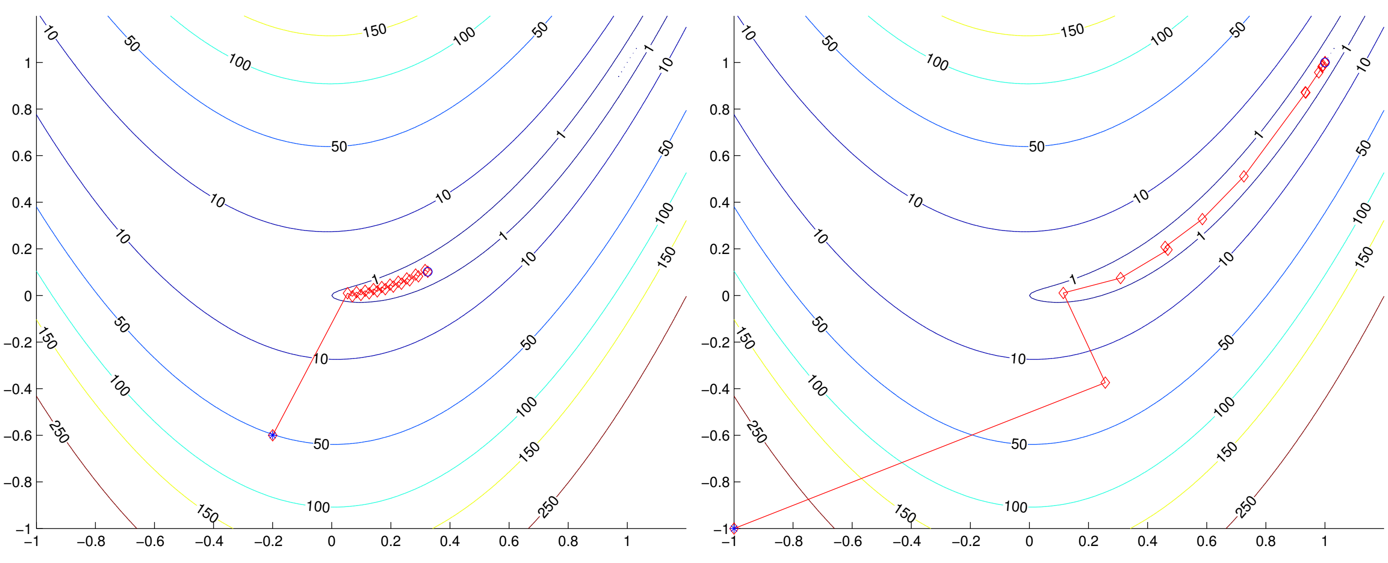
Listing 9: Gradient funkcjonału celu

```
function g=gradie(a, x)

n = 4;
x1 = x(1);
x2 = x(2);

g=[2*x1 - 120*n*x1*(- x1^2 + x2) - 2
            30*n*(- 2*x1^2 + 2*x2)];
```

Zadanie rozwiązano bardzo zbliżonym skryptem, jak zadanie poprzednie. Zmodyfikowano jedynie wartości, dla których rysowane są poziomice i punkty początkowe. Nie zamieszczono skryptu, ponieważ nie wnosiłby dużo do sprawozdania.

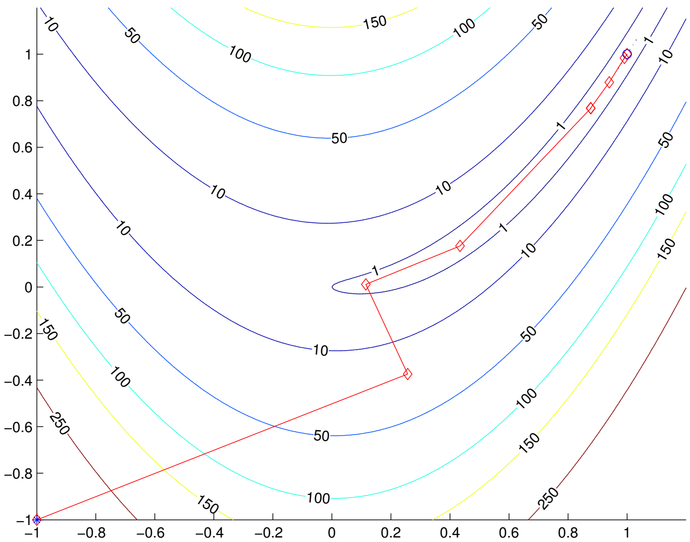
W przypadku wszystkich metod jako punkt początkowy obrano punkt $x_0 = [-1, -1]^T$

Uzyskano następujące wyniki:

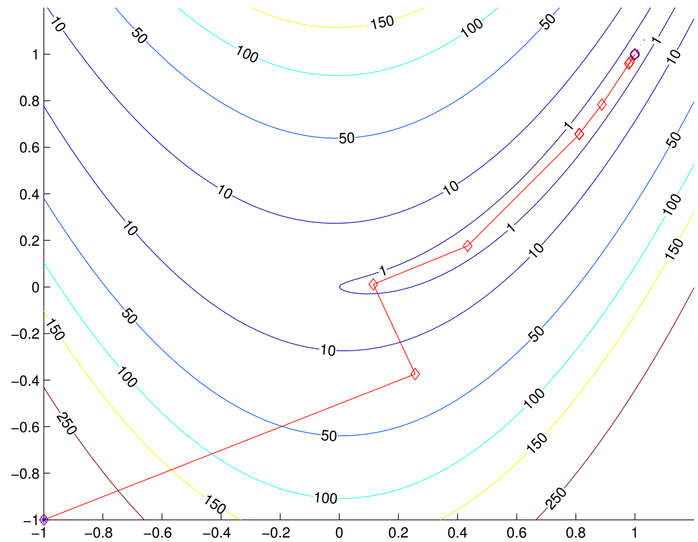


(a) Najszybszy spadek

(b) Fletcher-Reeves



(a) Metoda Polaka-Ribiera



(b) Metoda z pełną formułą na współczynnik β

Ilości wykonywanych iteracji zestawiono w tabeli:

	Metoda			
	Najszybszego spadku	Fletchera-Reeves'a	Polaka-Ribier'a	Z pełną formułą na β
Ilość iteracji	20	20	9	13

Najlepiej w tym wypadku sprawdziła się metoda Polaka-Ribiera, druga w kolejności była metoda z pełną formułą na współczynnik β.

Porównano odległość ostatnich przybliżeń wszystkich metod od punktu minimum:

	Metoda			
	Najszybszego spadku	Fletchera-Reeves'a	Polaka-Ribier'a	Z pełną formułą na β
Ilość iteracji	1.0128	7.0244e-05	2.2250e-04	5.7565e-05

Najlepiej sprawdziła się metoda z pełną formułą na współczynnik β. Wykonała ona jednak więcej iteracji niż metoda Polaka-Ribiera.

W poniższych tabelach zestawiono wartości modułu odległości pomiędzy bieżącym przybliżeniem rozwiązania optymalnego, a końcowym przybliżeniem rozwiązania optymalnego i różnice w wartości funkcji celu bieżącego przybliżenia rozwiązania optymalnego i końcowego przybliżenia rozwiązania optymalnego dla wszystkich metod.

Metoda najszybszego spadku

Wartość modułu różnicy argumentów	Różnica wartości funkcji celu
1,83852003189109	483,659139387625
0,569373267273947	23,4597107861150
0,386930949099052	0,527388645700204
0,193079259859436	0,246729227514120
0,186101486974391	0,203348298622324
0,146833389426176	0,171305718933823
0,141430406172236	0,147244707939301
0,114314636701691	0,127115033855929
0,109663556588865	0,110349937789960
0,0886292419545115	0,0955819541799182
0,0844311637876468	0,0827105494218088
0,0671515158524491	0,0710455899896468
0,0632477306788125	0,0605877441648762
0,0485475535490169	0,0509397782336029
0,0448253394836582	0,0421165790451281
0,0320904915939761	0,0338777845705954
0,0284251542634359	0,0262533232332861
0,0174706844146538	0,0190704731593820
0,0135880866396097	0,0123573514007027
0,00608679953770010	0,00599040318972943
0	0

Metoda najszybszego spadku zachowuje się podobnie, jak w przypadku poprzedniego zadania — w pierwszych dwóch iteracjach widać duży skok, ale następnie (po dojściu do dna doliny) metoda się praktycznie zatrzymuje.

Metoda Fletchera-Reevesa

Wartość modułu różnicy argumentów	Różnica wartości funkcji celu
2,82849379837262	483,999999999000
1,56242598712484	23,8005713974895
1,32766421704627	0,784816341898838
1,15506815858957	0,527239179312133
0,963837712184530	0,351152710190084
0,958534332241746	0,293069277757297
0,790023025804675	0,198494000845296
0,560704682151631	0,104563146869486
0,144613682246346	0,00580215789679020
0,144524669434651	0,00439684804668850
0,0478801555452936	0,000921834868901155
0,0205834972319863	0,000105921772135426
0,0196724283193037	7,77496206784744e-05
0,0196657594884365	7,72345348335120e-05
0,0195661939191351	7,68403244939771e-05
0,0141737829646834	5,81545930085135e-05
0,00129259463465338	1,74289088448833e-06
0,00129125623063633	3,69076296567069e-07
0,00128677389835699	3,67645405018473e-07
0,000614015343641122	1,74950051026418e-07
0	0

W przypadku tej metody obliczenia mogły zostać również przerwane przez kryterium ilości iteracji, jednak przybliżenie rozwiązania optymalnego jest znacznie lepsze — odległość od tego rozwiązania jest rzędu 10^{-5}

Metoda Polaka-Ribiera

Wartość modułu różnicy argumentów	Różnica wartości funkcji celu
2,82863816158929	483,999999990113
1,56257815607472	23,8005713886024
1,32740874222641	0,783975409472853
0,999338630777511	0,337889301869439
0,263130577266813	0,0156276189279908
0,262976923889518	0,0152469469071568
0,135220915875320	0,00562422814806446
0,0186850676820223	0,000327572202407123
3,52688117709584e-06	7,44210078255724e-09
0	0

Metoda z pełną formułą na współczynnik β

Wartość modułu różnicy argumentów	Różnica wartości funkcji celu
2,82837252290073	483,999999999338
1,56229824191117	23,8005713978277
1,32714348931158	0,783985630207405
0,999173131596218	0,338016406281195
0,391344589823799	0,0369841731022180
0,391029812230715	0,0356774507491730
0,243408540714493	0,0168960645077511
0,0381679774917571	0,000854052900241859
0,0447878189221007	0,000417032352606708
0,0447718551855543	0,000407954231696333
0,00361457895903599	1,79219322900388e-05
0,00181309113267527	7,04162781589411e-07
4,47365742288053e-06	1,19711755570825e-08
0	0

Obydwie metody zatrzymały się w wyniku osiągnięcia celu, a nie w wyniku wykonania maksymalnej ilości iteracji. Uzyskane przybliżenia rozwiązań są bardzo dobre w zestawieniu do metody najszybszego spadku. Widać, że w końcowych krokach zmiana rozwiązania (w odniesieniu do znalezionego rozwiązania suboptymalnego) jest bardzo niewielka.

5 Wnioski

W trakcie laboratorium zbadano metody gradientowe: metodę najszybszego spadku i metody gradientów sprzężonych: metodę Fletchera-Reevesa, metodę Polaka-Ribiera i metodę z pełną formułą na współczynnik β.

W pierwszym zadaniu zaobserwowano problem występujący przy stosowaniu metody najszybszego spadku: kolejne punkty są wyszukiwane po liniach zygzakowatych. Powoduje to, że metoda dość wolno zbiega do rozwiązania optymalnego. Zaobserwowano również, że efektywność metody silnie zależy od doboru punktu początkowego — w zależności od jego wartości metoda wykonywała albo jedną, albo wiele iteracji.

W drugim i piątym zadaniu porównano efektywność algorytmów. W obydwu przypadkach powtórzył się kiepski wynik metody największego spadku. Metoda wykorzystywała maksymalną dopuszczalną ilości iteracji. Metody gradientów sprzężonych spisywały się znacznie lepiej — były w stanie znaleźć lepsze rozwiązania lub nawet zakończyć pracę przed wyczerpaniem limitu ilości iteracji. W obydwu zadaniach metody Polaka-Ribiera i z pełną formułą na współczynnik β potrzebowały najmniejszej liczby iteracji do zakończenia obliczeń. Pomiędzy końcowymi przybliżeniami poszczególnych metod gradientów sprzężonych występowały niewielkie różnice.