Sprawozdanie z laboratorium Teorii Optymalizacji

Imię i nazwisko	Jacek Gołda
Temat ćwiczenia	Zewnętrzna funkcja kary
Data i godzina wykonania ćwiczenia	20 kwietnia 2016, 14:00

1 Wstęp

Celem laboratorium było zbadanie metody zewnętrznej funkcji kary.

2 Ćwiczenie 1

Treść zadania:

$$\min_{x_2=1} \left\{x_1^2 + x_2^2\right\}$$

Rozwiązać zadanie minimalizacji korzystając z metody zewnętrznej funkcji kary. Następnie porównać z rozwiązaniem teoretycznym. Jak ilość iteracji wpływa na dokładność rozwiązania? Podać interpretację graficzną rozwiązania. Sprawdzić, czy przyjęta funkcja kary spełnia założenia odnoszące się do tego typu funkcji.

Rozwiązanie:

2.1 Rozwiązanie analityczne

Można zauważyć, że funkcja celu składa się z dwóch wyrażeń nieujemnych. Minimum globalne funkcji bez ograniczeń byłoby osiągane dla punktu $\widehat{x} = \begin{bmatrix}0 & 0\end{bmatrix}$

Jednak ze względu na ograniczenie $x_2 = 1$, minimum będzie osiągane dla $\widehat{x} = \begin{bmatrix}0 & 1\end{bmatrix}$. Dla takiej wartości pierwsza część funkcji celu osiąga wartość minimalną, a druga część osiąga jedyną możliwą (ze względu na narzucone ograniczenie). Stąd, będzie to rozwiązanie optymalne.

2.2 Rozwiązanie numeryczne

Wykorzystano funkcję kary zaimplementowaną w pliku `zkara.m` (nie modyfikowano go). Funkcja ta jest opisana w instrukcji:

$$\Phi = \sum_{i=1}^{r} k^j \left[\max\left(0, g_i(x)\right)\right]^2$$

gdzie k — pewien współczynnik, j — numer iteracji.

Dzięki zastosowaniu funkcji max, uzyskuje się, że funkcja kary jest większa od zera dla punktów poza zbiorem dopuszczalnym, a dla punktów należących do zbioru dopuszczalnego osiąga wartość 0.

Pojedyncze ograniczenie równościowe należało zapisać w postaci ograniczeń nierównościowych. Skorzystano w związku z tym z dwóch ograniczeń:

$$\begin{cases} x_2 \leqslant 1 \\ x_2 \geqslant 1 \end{cases}$$

Po przekształceniu do postaci standardowej uzyskano:

$$\begin{cases} x_2 - 1 \leqslant 0 \\ 1 - x_2 \leqslant 0 \end{cases}$$

Napisano skrypt służący do rozwiązywania zadania i rysowania rozwiązania:

Listing 1: Skrypt rozwiązujący zadanie

```
clear all;
close all;
clc

x0 = [2, 1]
clear param
param = [];
param(2) = 1e-5;
param(4) = 100;
[X,F,H,ITER,K]=zkara('funkcja_zadanie_1','ograniczenie_zadanie_1', ...
    'gradient_funkcji_zadanie_1','gradient_ograniczen_zadanie_1',x0,param)

figure
x1_range = linspace(-2, 2, 60);
x2_range = linspace(0, 1.2, 30);

[x1, x2] = meshgrid(x1_range, x2_range);
val = x1 .^ 2 + x2 .^ 2;
contour(x1, x2, val, [0.01, 0.3, 0.6, 1, 2, 3], 'ShowText', 'on');

hold on;
plot(H(:,1),H(:,2),'k*');
```

Napisano funkcje obliczające wartości funkcji, gradientu funkcji, ograniczeń i gradientu ograniczeń, konieczne do pracy algorytmu:

Listing 2: Wartość funkcji

```
function Q = funkcja_zadanie_1(x)
    Q = x(1) .^ 2 + x(2) .^ 2;
end
```

Listing 3: Gradient funkcji

```
function [g] = gradient_funkcji_zadanie_1(x)
    g = [2 * x(1); 2 * x(2)];
end
```

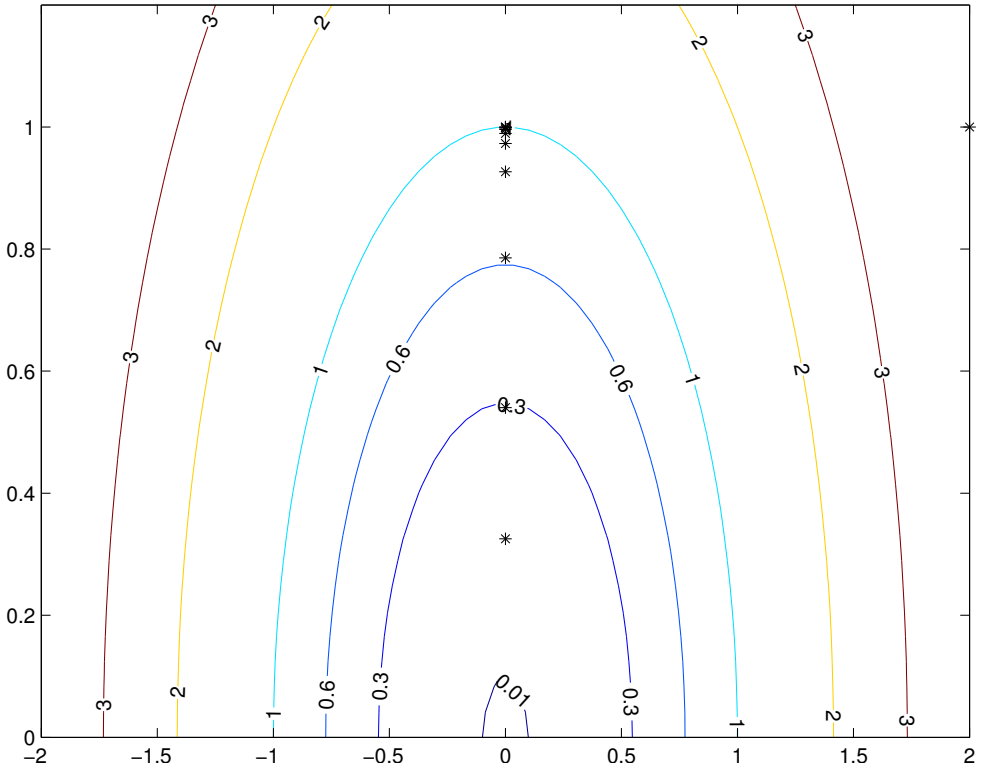
Listing 4: Ograniczenia

```
function [g] = ograniczenie_zadanie_1(x)
    g = [1 -  x(2);
         x(2) - 1;];
end
```

Listing 5: Gradient ograniczeń

```
function [g] = gradient_ograniczen_zadanie_1(x)
    g = [0, -1;
         0, 1]';
end
```

Po uruchomieniu skryptu uzyskano następujące wyniki:



Widać, że kolejne rozwiązania znajdują się w pobliżu prostej $x_1 = 0$. W kolejnych iteracjach funkcja kary zaczyna przesuwać rozwiązanie coraz bliżej rozwiązania optymalnego.

Zebrano w tabelce kolejne przybliżenia rozwiązania optymalnego.

Numer iteracji	Współrzędna x	Współrzędna y
1	2.0000	1.0000
2	-0.0000	0.3254
3	-0.0000	0.5403
4	-0.0000	0.7857
5	-0.0000	0.9268
6	-0.0000	0.9730
7	-0.0000	0.9895
8	-0.0000	0.9959
9	-0.0000	0.9984
10	-0.0000	0.9993
11	-0.0000	0.9997
12	-0.0000	0.9999
13	-0.0000	1.0000
14	-0.0000	1.0000
15	-0.0000	1.0000
16	-0.0000	1.0000

Tutaj również widać, że w pierwszej iteracji algorytm wykonał duży skok, jednak znalazł się poza zbiorem dopuszczalnym. W kolejnych iteracjach algorytm zbliża się coraz bardziej do rozwiązania dopuszczalnego. Każdy kolejny krok przynosi poprawę. W każdej kolejnej iteracji jest ona coraz mniejsza, jednak już w 16 iteracji zadziałało kryterium stopu i algorytm skończył pracę, znajdując rozwiązanie optymalne.

3 Ćwiczenie 2

Treść zadania:

Zminimalizować odległość od punktu $(0, 0)$ przy ograniczeniach:

$$\begin{cases} g_1(x) = x_1 + x_2 - 1 = 0 \\ g_2(x) = x_1 + x_2 - 2 = 0 \end{cases}$$

Sprawdzić poprawność postawionego zadania.

Rozwiązanie:

3.1 Rozwiązanie analityczne

Warto rozpocząć od zauważenia, że zbiór rozwiązań dopuszczalnych jest zbiorem pustym, gdyż ograniczenia są sprzeczne. Po odjęciu ograniczeń stronami uzyskuje się bowiem:

$$1 = 0$$

Stąd nie za bardzo można mówić o jakimkolwiek rozwiązaniu optymalnym.

3.2 Rozwiązanie numeryczne

W zasadzie nic jednak nie stoi na przeszkodzie w uruchomieniu algorytmu dla tak postawionego problemu. Metoda powinna znaleźć rozwiązanie, jednak będzie ono błędne, ponieważ nie istnieje żadne rozwiązanie dopuszczalne, a co za tym idzie, nie istnieje też rozwiązanie optymalne.

Do minimalizacji odległości od początku układu współrzędnych wykorzystano funkcjonał wykorzystywany w poprzednim zadaniu.

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

Ma to sens, ponieważ jest on równy kwadratowi odległości od początku układu współrzędnych.

Ponownie zapisano obydwa ograniczenia jako ograniczenia nierównościowe:

$$\begin{cases} x_1 + x_2 - 1 \leqslant 0 \\ -x_1 - x_2 + 1 \leqslant 0 \\ x_1 + x_2 - 2 \leqslant 0 \\ -x_1 - x_2 + 2 \leqslant 0 \end{cases}$$

Wykorzystano identyczny skrypt do uruchamiania procedury optymalizacji, obliczania funkcji i jej gradientu (ponieważ korzystano z tego samego funkcjonału), zmodyfikowano jedynie funkcje obliczające ograniczenia i ich gradienty.

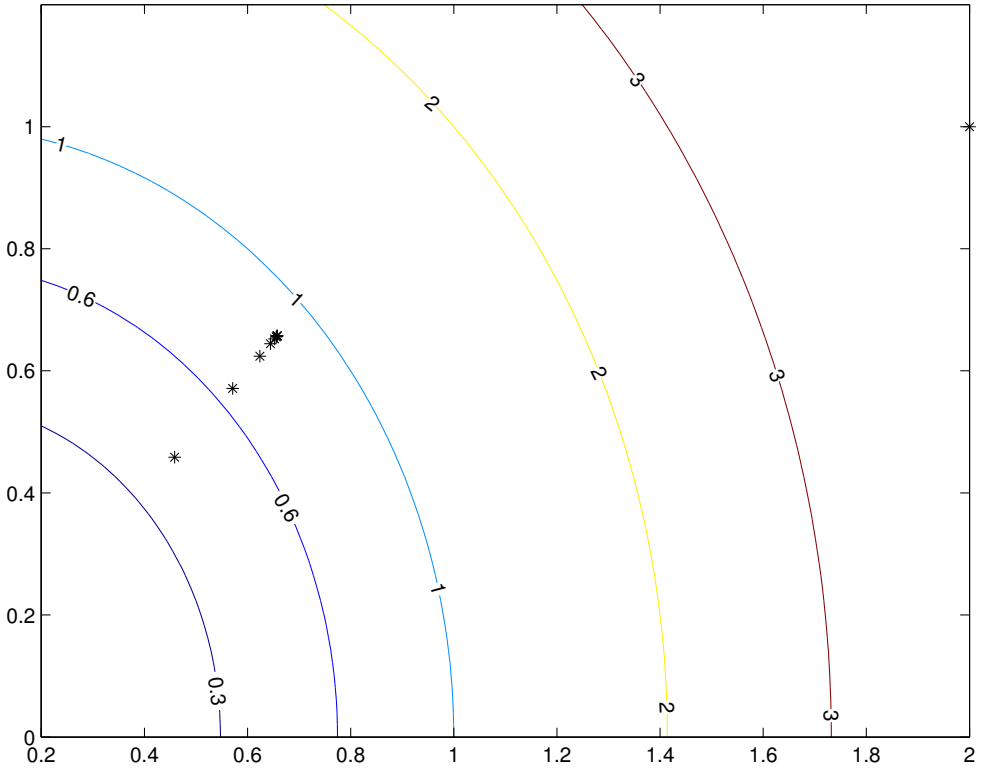
Listing 6: Ograniczenia

```
function [g] = ograniczenie_zadanie_2(x)
    g = [x(1) + x(2) - 1;
         -x(1) - x(2) + 1;
         x(1) + x(2) - 2;
         -x(1) - x(2) + 2];
end
```

Listing 7: Gradient ograniczeń

```
function [g] = gradient_ograniczen_zadanie_2(x)
    g = [1, 1;
         -1, -1;
         1, 1;
         -1, -1]';
end
```

Po uruchomieniu uzyskano następujące wyniki:



Można by odnieść wrażenie, że algorytm świetnie działa, ponieważ zbiega do rozwiązania (zapewne tego optymalnego). Jeżeli by tak było, to wynosiło by ono $\widehat{x} = \begin{bmatrix} 0.6575 & 0.6575 \end{bmatrix}$

Jednak po wstawieniu do ograniczeń okazuje się, że jest to nieprawda, ponieważ rozwiązanie nie spełnia ograniczeń:

$$\begin{cases} 0.6575 + 0.6575 - 1 = 0 \\ 0.6575 + 0.6575 - 2 = 0 \end{cases}$$

$$\begin{cases} 0.25 \neq 0 \\ -0.75 \neq 0 \end{cases}$$

Jest to oczywistą sprzecznością. Wynika z tego, że w sytuacji, gdy stosuje się funkcję kary, koniecznie trzeba sprawdzić, czy zbiór rozwiązań nie jest pusty. W przeciwnym wypadku metoda zwróci nieprawdziwe rozwiązania. Co gorsza, nie będą widoczne żadne ewidentne przesłanki nieprawdziwości rozwiązania.

4 Wnioski

W trakcie laboratorium zbadano metodę zewnętrznej funkcji kary. Przy realizacji pierwszego zadania zapoznano się z działaniem metody. Zaobserwowano, jak kolejne rozwiązania zbliżają się do granicy zbioru rozwiązań dopuszczalnych. Ostatecznie bieżące rozwiązanie trafiło do zbioru rozwiązań dopuszczalnych — było to rozwiązanie optymalne.

W drugim zadaniu zaobserwowano problem związany z metodą zewnętrznej funkcji kary. Polega on na tym, że metoda (algorytm) zwróci pewne rozwiązanie, nawet jeżeli zbiór rozwiązań dopuszczalnych dla tego zadania jest zbiorem pustym. To znalezione rozwiązanie będzie oczywiście niepoprawne. Algorytm nie daje żadnego sygnału, że znalezione rozwiązanie jest niepoprawne. Konieczne w związku z tym jest sprawdzenie, czy rozwiązanie znalezione przez algorytm jest poprawne, lub czy zbiór rozwiązań dopuszczalnych nie jest pusty.

Metoda zewnętrznej funkcji kary jest bardzo poręcznym sposobem sprowadzenia zadania z ograniczeniami do zadania bez ograniczeń. Nie można jednak zapominać o tej drobnej wadzie — konieczności sprawdzenia poprawności rozwiązania, lub sprawdzenia, czy zbiór rozwiązań dopuszczalnych jest niepusty.