Sprawozdanie z laboratorium Teorii Optymalizacji

Imię i nazwisko	Jacek Gołda
Temat ćwiczenia	Wewnętrzna funkcja kary
Data i godzina wykonania ćwiczenia	27 kwietnia 2016, 14:00

1 Wstęp

Celem laboratorium było zbadanie metody wewnętrznej funkcji kary.

Metoda wewnętrznej funkcji kary powoduje stopniowe zbliżanie się rozwiązań do rozwiązania optymalnego z wnętrza zbioru rozwiązań dopuszczalnych. Istotne jest, aby punkt początkowy znajdował się wewnątrz zbioru rozwiązań dopuszczalnych.

Metoda na pewną zaletę w stosunku do metody zewnętrznej funkcji kary, mianowicie wszystkie przybliżenia rozwiązania optymalnego są rozwiązaniami dopuszczalnymi (w sensie warunków ograniczających).

2 Ćwiczenie 1

Treść zadania:

Zminimalizować funkcjonał

$$f(x) = \frac{1}{3}(x_1 + n + 1)^3 + x_2 + n$$

przy ograniczeniach:

$$\begin{cases} x_1 - 1 + n \geqslant 0 \\ x_2 + n \geqslant 0 \end{cases}$$

Za n przyjęto wartość 4.

2.1 Rozwiązanie numeryczne

Wykorzystano funkcję kary zaimplementowaną w pliku `wkara.m` (nie modyfikowano go).

Rozpoczęto od zapisania ograniczeń w postaci standardowej:

$$\begin{cases} -x_1 - 3 \leqslant 0 \\ -x_2 - 4 \leqslant 0 \end{cases}$$

Na podstawie ograniczeń dobrano punkt startowy: $x_0 = \begin{bmatrix} -0.5 & -0.5 \end{bmatrix}$ Istotne jest, aby znajdował się on wewnątrz zbioru rozwiązań dopuszczalnych.

Napisano następujący skrypt służący do rozwiązywania zadania i rysowania rozwiązania. Napisano również pliki konieczne do uruchomienia i poprawnego działania metody.

Listing 1: Skrypt rozwiązujący zadanie

```
clear all;
close all;
```

```
clc

x0 = [-0.5, -0.5]
clear param
param = [];
param(2) = 1e-5;
param(4) = 100;
[X,F,H,ITER,K]=wkara('funkcja_zadanie_1','ograniczenie_zadanie_1','gradient_funkcji_zadanie_1','gradi

figure
x1_range = linspace(-4, 0, 60);
x2_range = linspace(-5, 0, 30);

n = 4;
[x1, x2] = meshgrid(x1_range, x2_range);
val = 1/3*(x1 + n + 1) .^ 3 + x2 + n;
contour(x1, x2, val, 'ShowText', 'on');

hold on;
plot(H(:,1),H(:,2),'k*');
x1_ogr = -3 * ones(size(x2));
x2_ogr = -4 * ones(size(x1));

plot(x1_ogr, x2, '.');
plot(x1, x2_ogr, '.');
```

Listing 2: Wartość funkcji

```
function Q = funkcja_zadanie_1(x)
    n = 4;
    x1 = x(1);
    x2 = x(2);
    Q = 1/3*(x1 + n + 1) .^ 3 + x2 + n;
end
```

Listing 3: Gradient funkcji

```
function [g] = gradient_funkcji_zadanie_1(x)
    n = 4;
    g = [(x(1) + n + 1) .^ 2;
          1];
end
```

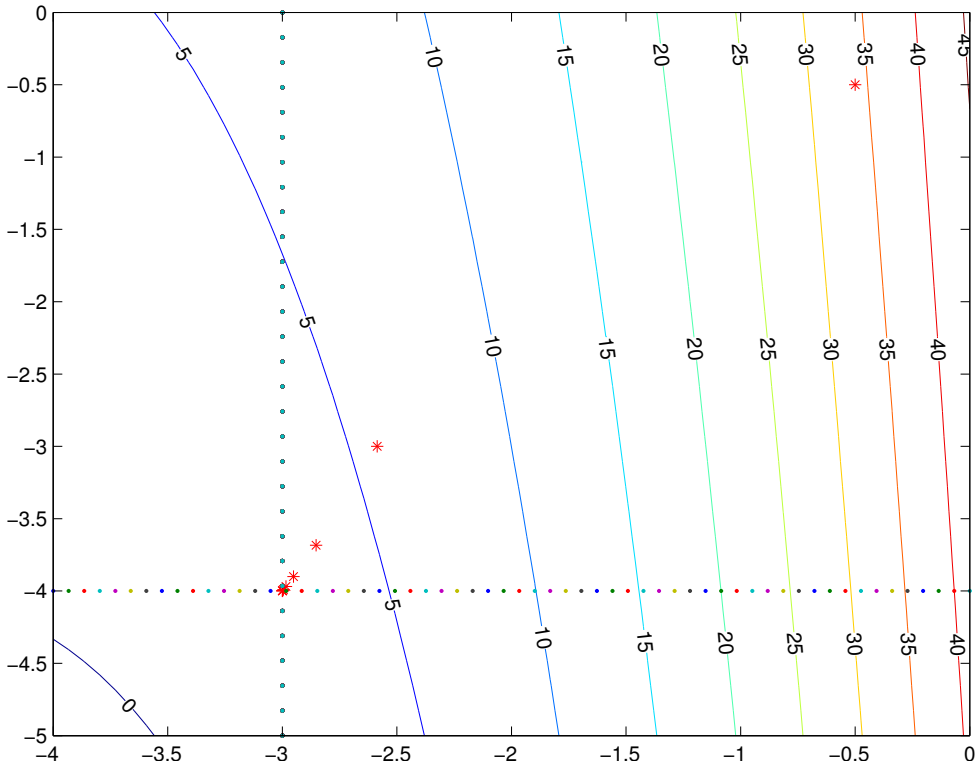
Listing 4: Ograniczenia

```
function [g] = ograniczenie_zadanie_1(x)
    g = [- x(1) - 3;
         - x(2) - 4];
end
```

Listing 5: Gradient ograniczeń

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

Po uruchomieniu skryptu uzyskano następujące wyniki:



Prawa górna część wykresu (oddzielona prostymi) stanowi zbiór rozwiązań dopuszczalnych. W pierwszej iteracji algorytm wykonał duży skok, w kolejnych iteracjach poprawa jest coraz mniejsza. Algorytm znajduje jednak rozwiązanie optymalne.

Zebrano w tabelce kolejne przybliżenia rozwiązania optymalnego.

Numer iteracji	Współrzędna x	Współrzędna y
1	-0.5000	-0.5000
2	-2.5858	-3.0000
3	-2.8527	-3.6837
4	-2.9512	-3.9000
5	-2.9843	-3.9684
6	-2.9950	-3.9900
7	-2.9984	-3.9968
8	-2.9995	-3.9990
9	-2.9998	-3.9997
10	-3.0000	-3.9999
11	-3.0000	-4.0000
12	-3.0000	-4.0000
13	-3.0000	-4.0000

Tutaj również widać, że w pierwszej iteracji algorytm wykonał duży skok. W kolejnych iteracjach algorytm zbliża się coraz bardziej do rozwiązania optymalnego. W każdej kolejnej iteracji poprawa jest coraz mniejsza, jednak już w 13 iteracji zadziałało kryterium stopu i algorytm skończył pracę, znajdując rozwiązanie optymalne.

3 Ćwiczenie 2

Treść zadania:

Zminimalizować funkcjonał:

$$f(x) = 2x_1 + 4x_2 - 3n$$

przy ograniczeniach:

$$\begin{cases} \frac{1}{2}x_1 - \frac{1}{4}n + 1 \geqslant 0 \\ -4x_1^2 - 2n^2 + 2nx_1 + 2x_2 \geqslant 0 \end{cases}$$

3.1 Rozwiązanie numeryczne

Ponownie wykorzystano plik `wkara.m`. Zapisano ograniczenia w postaci standardowej:

$$\begin{cases} -\frac{1}{2}x_1 + \frac{1}{4}n - 1 \leqslant 0 \\ 4x_1^2 + 2n^2 - 2nx_1 - 2x_2 \leqslant 0 \end{cases}$$

Ponownie napisano pliki obliczające wartość i gradient funkcji i ograniczeń. Wykorzystano praktycznie identyczny skrypt do rozwiązywania zadania. Zmodyfikowano jedynie wartość punktu startowego na punkt $x_0 = \begin{bmatrix} 0.5 & 29 \end{bmatrix}$

Listing 6: Wartość funkcji

```
function Q = funkcja_zadanie_1(x)
    x1 = x(1);
    x2 = x(2);
    n = 4;
    Q = 2 * x1 + 4 * x2 - 3 * n;
end
```

Listing 7: Gradient funkcji

```
function [g] = gradient_funkcji_zadanie_1(x)
    g = [2;
         4];
end
```

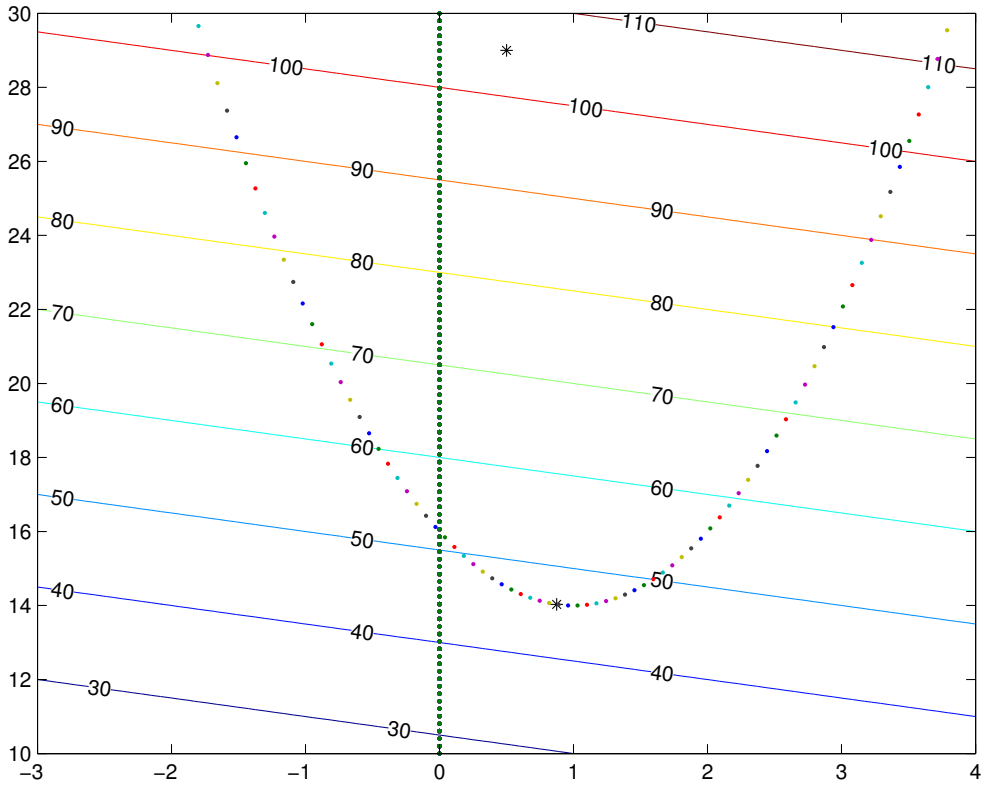
Listing 8: Ograniczenia

```
function [g] = ograniczenie_zadanie_1(x)
    x1 = x(1);
    x2 = x(2);
    n = 4;
    g = [- 0.5 * x1 + 0.25 * n - 1;
          4 * x1 .^ 2 + 2 * n ^ 2 - 2 * n * x1 - 2 * x2];
end
```

Listing 9: Gradient ograniczeń

```
function [g] = gradient_ograniczen_zadanie_1(x)
    x1 = x(1);
    x2 = x(2);
    n = 4;
    g = [-0.5,               0;
          8 * x1 - 2 * n,   -2]';
end
```

Po uruchomieniu uzyskano następujące wyniki:



Zbiór rozwiązań dopuszczalnych znajduje się ponad parabolą i po prawej stronie pionowej prostej. Algorytm w pierwszym kroku wykonał duży skok, następnie zmiany były niewielkie.

Przedstawiono kolejne przybliżenia w postaci tabeli.

Numer iteracji	Współrzędna x	Współrzędna y
1	0.5000	29.0000
2	0.8750	14.0308
3	0.8750	14.0311
4	0.8750	14.0312
5	0.8751	14.0312
6	0.8750	14.0312
7	0.8750	14.0312

Tutaj również dobrze widać, że algorytm w pierwszym kroku wykonał skok bardzo blisko znalezionego ostatecznie rozwiązania suboptymalnego.

4 Wnioski

W trakcie laboratorium zbadano metodę wewnętrznej funkcji kary. Metoda różni się tym od metody zewnętrznej funkcji kary, że przybliżenie rozwiązania optymalnego zbliża się do rozwiązania suboptymalnego niejako „od strony” zbioru rozwiązań dopuszczalnych — wszystkie kolejne przybliżenia są rozwiązaniami dopuszczalnymi. Jest to istotna zaleta, jeżeli spodziewa się, że obliczenia zostaną przerwane wcześniej, na przykład wskutek konieczności dostarczenia pewnych parametrów (wyników optymalizacji) po ściśle ustalonym czasie. Jest to istotne, gdyż nastawy te będą musiały spełniać warunki ograniczające. Wadą jest jednak fakt, że rozwiązanie początkowe musi również należeć do zbioru rozwiązań dopuszczalnych. Konieczny jest odpowiedni dobór punktu początkowego. Co więcej, metoda nie nadaje się do rozwiązywania zadań, w których występują ograniczenia równościowe.