Informatyka, studia dzienne, mgr II st. semestr I

Metody obliczeniowe optymalizacji **2011/2012**

Prowadzący: Prowadzący dzień, godzina

Data oddania: _____ Ocena: _____

Paweł Musiał nr albumu 1
Łukasz Michalski nr albumu 2

1: Optymalizacja jednowymiarowa*

1. Cel

Zaimplementować 3 metody optymalizacji jednowymiarowej spośród następujących:
— metoda dychotomii
— **metoda bisekcji**
— metoda złotego podziału
— **metoda Fibonacci**
— metoda Newtona
— **metoda Quasi-Newtona**
Przedstawiany jako rozwiązanie program powinien pozwolić wprowadzić funkcję oraz przedział, w którym poszukiwane będzie rozwiązanie. Jeśli przedział nie jest unimodalny należy wyznaczyć przedział unimodalności. Skonfrontować wyniki z wyznaczonymi za pomocą pakietu Matlab. Porównać efektywność zaimplementowanych metod.

2. Rozwiązanie zadania

2.1. Wyznaczanie podprzedziału unimodalności

Wyznaczanie podprzedziału unimodalności jest wykonywane poprzez sprawdzanie kolejnych podprzedziałów o określonej szerokości. W każdym takim

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

podprzedziale sprawdzana jest lokalna unimodalność funkcji. Gdy znajdziemy podprzedział który spełnia te kryteria zwracamy go jako nowy podprzedział do poszukiwania rozwiązania. Każdorazowe wyszukiwanie, podprzedziału unimodalności nie tylko sprawdza czy funkcja jest lokalnie unimodalna, lecz również zmniejsza zakres poszukiwań rozwiązania.

2.2. Metoda bisekcji [?]

Metoda ta polega na wybraniu trzech punktów jednakowo odległych od siebie oraz od krańców przedziału oraz wyliczeniu wartości funkcji w tych punktach, w wyniku czego można wyeliminować połowę przedziału.
Algorytm

1. Wybierz dolne i górne ograniczenie przedziału a i b oraz małą liczbę ϵ. Niech $x_m = (a+b)/2$, $L_0 = L = b - a$. Wylicz $f(x_m)$.
2. Ustal $x_1 = a + L/4$, $x_2 = b - L/4$. Wylicz $f(x_1)$ oraz $f(x_2)$.
3. Jeśli $f(x_1) < f(x_m)$, ustal $b = x_m$; $x_m = x_1$; Przejdź do 5);
   W przeciwnym wypadku przejdź do kroku 4).
4. Jeśli $f(x_2) < f(x_m)$, ustal $a = x_m$; $x_m = x_2$;
   W przeciwnym przypadku ustal $a = x_1$, $b = x_2$;
5. Wylicz $L = b - a$. Jeśli $|L| < \epsilon$, Zakończ;
   W przeciwnym przypadku przejdź do 2).

W każdej nowej iteracji algorytmu, potrzebne jest wyliczenie dwóch wartości funkcji, a przedział zmniejsza się o połowę. Po n-krotnym wyliczeniu wartości funkcji, przedział zmniejsza się do około $0.5^{n/2}L_0$. Czyli ilość razy n, ile trzeba policzyć wartości funkcji, aby osiągnąć daną dokładność ϵ można policzyć z następującego wzoru :

$$(0.5)^{n/2}(b-a) = \epsilon$$

2.3. Metoda Fibonacci [?]

W metodzie liczb Fibonacciego, w każdej iteracji algorytmu proporcja zmniejszania się przedziału z iteracji na iteracje zmienia się tak, aby przedział zmniejszał się w sposób optymalny (tzn. jak najbardziej). Jeśli p_k oznacza proporcje, o jaką zmniejsza się przedział w k-tej iteracji, to w metodzie Fibonacciego zachodzi następujący związek:

$$\frac{1-p_{k-1}}{1} = \frac{p_k}{1-p_k}$$

Okazuje się, że wartościami $p_k \in (0, 1/2]$, gdzie k = 1,..., N, które minimalizują wyrażenie $(1-p_1)(1-p_2)...(1-p_N)$ i które spełniają powyższy związek, są następujące liczby:

$$p_k = 1 - \frac{F_{N-k-1}}{F_{N-k+2}}$$

gdzie F_k oznaczają liczby Fibonacciego. Liczby Fibonacciego mają następującą charakterystykę: $F_1 = 1, F_2 = 1$ i $F_k = F_{k1} + F_{k2}$ gdzie k = 3, 4, ...

Algorytm

1. Znajdź n takie, że $\frac{(b-a)}{F_n} \geq 2\epsilon$
2. Oblicz $x_1 = \frac{F_{n-1}}{F_n}(b-a)$ i $x_2 = \frac{F_{n-1}}{F_n}(b-a)$
3. Jeżeli $f(x_1) < f(x_2)$ to podstaw :
   — $b = x_2$
   — $x_2 = x_1$
   — $n = n - 1$
   — $x_1 = b - \frac{F_{n-1}}{F_n}(b-a)$
   W przeciwnym wypadku :
   — $a = x_1$
   — $x_1 = x_2$
   — $n = n - 1$
   — $x_2 = a + \frac{F_{n-1}}{F_n}(b-a)$
4. Jeśli $|x_2 - x_1| > \epsilon$ i $n \geq 2$ przejdź do 3.

W tym algorytmie przedział redukuje się do $\frac{2}{F_{N+1}}L$ do czego potrzebnych jest n wyliczeń wartości funkcji. Zatem ilość potrzebnych wyliczeń wartości funkcji przy danej dokładności ϵ można wyliczyć z następującego wzoru:

$$\frac{2}{F_{n+1}} = \epsilon$$

Jednym z minusów metody liczb Fibonacciego jest fakt, iż trzeba wyliczać kolejne liczby Fibonacciego w każdej iteracji.

2.4. Metoda Quasi-Newtona

Metody quasi-Newtonowskie bazują na metodzie Newtona znajdowania punktów stacjonarnych funkcji. Metoda Newtona zakłada, że funkcja może być lokalnie aproksymowana funkcją kwadratową w otoczeniu optimum, oraz używają pierwszych i drugich pochodnych (gradient i hesjan) w celu znalezienia punktów stacjonarnych.
W metodzie Quasi-Newtona hesjan(macierz drugich pochodnych) minimalizowanej funkcji nie musi być obliczany. Hesjan jest przybliżany przez analizowanie kolejnych wektorów gradientu. Metody Quasi-Newtona są uogólnieniem metody siecznych znajdowania pierwiastków pierwszej pochodnej na problem wielowymiarowy.
Kolejne przybliżenia są wyznaczane na podstawie :

$$x_{k+1} = x_k - \frac{x^k - x^{k-1}}{f'(x^k) - f'(x_{k-1})} f'(x_k) \tag{1}$$

3. Opis programu

Program składa się z implementacji 3 wyznaczonych metod, oraz funkcji pomocniczych do określania podprzedziału unimodalności i rysowania wykresu funkcji z nałożonymi punktami kolejnych przybliżeń rozwiązania.

3.1. Wyznaczanie podprzedziału unimodalności

```
function [a, b] = unimodality_check(f,a,b, step)

unimodal=0;
a=a+step;

while unimodal==0 && a<b
    %found max || min
    if (feval(f,a-step) < feval(f,a) &&
     feval(f,a+step) < feval(f,a)) ||
      (feval(f,a-step) > feval(f,a)
       && feval(f,a+step) > feval(f,a))
         unimodal=1;
    else
        a=a+step;
    end
end

if unimodal == 0
    disp('przy danym kroku probkowania,
     nie znaleziono podprzedzialu unimodalnosci');
    a=-1;b=-1;
elseif unimodal == 1
    b=a+step;
end
```

3.2. Metoda bisekcji

```
function [c,yc,iter] = bisection (f,a,b,eps)
f=inline(f);
xL=a;xR=b;
maxIterations=5+round((log(a-b)-log(eps))/log(2));

while ( abs(xR - xL) > eps && i< maxIterations)
    c=(xL+xR)/2.0;
    yc=feval(f,c);
    l=xR-xL;
    x1=xL+(l/4.0);
    x2=xR-(l/4.0);
    if feval(f,x1)<feval(f,c)
        xR=c;
        c=x1;
    elseif feval(f,x2)<feval(f,c)
        xL=c;
        c=x2;
    else
        xL=x1;
        xR=x2;
    end
    i=i+1
end
```

4. Metoda Fibonacciego

```
1
function x = fib(n)
if n<=0
    x=0;
elseif n==1
    x=1;
else x=fib(n-1)+fib(n-2);
end


function [c,yc,iter] = fibonaccio (f,a,b,eps)
f=inline(f);
n=0;
while 1.0/fib(n) > eps
    n=n+1;
end

x1=b-(fib(n-1)/fib(n))*(b-a);
x2=a+(fib(n-1)/fib(n))*(b-a);

while ( abs(x2 - x1) > eps && n >= 2)
    c=(a+b)/2;
    yc=feval(f,c);
    if feval(f,x1)<feval(f,x2)
        b=x2;
        x2=x1;
        n=n-1;
        x1=b-(fib(n-1)/fib(n))*(b-a);
    else
        a=x1;
        x1=x2;
        n=n-1;
        x2=a+(fib(n-1)/fib(n))*(b-a);
    end
end

c=(a+b)/2;
yc=feval(f,c);
```

4.1. Metoda Quasi-Newtona

```
function [b,by,k]=quasi_newton(f,a,b,eps)
f=inline(f);

maxIterations=5+round((log(a-b)-log(eps))/log(2));

k=1;
while  k < maxIterations
    p2=b-diffp(f,b)*(b-a)/(diffp(f,b)-diffp(f,a));
    err=abs(p2-b);
    a=b;
```

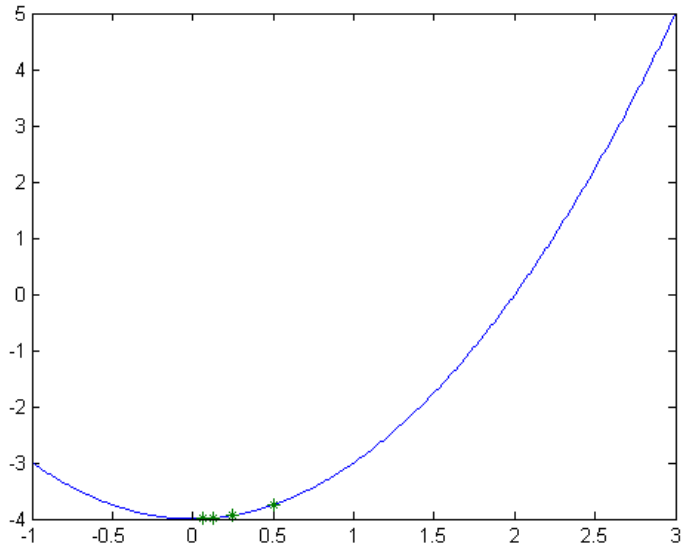
```
12      b=p2;
        by=feval(f,b);
14      if (err<eps)
            break
16      end
        k=k+1;
18  end

20  function var = diffp (f,p)
    var=feval(inline(diff(sym(f))),p);
```

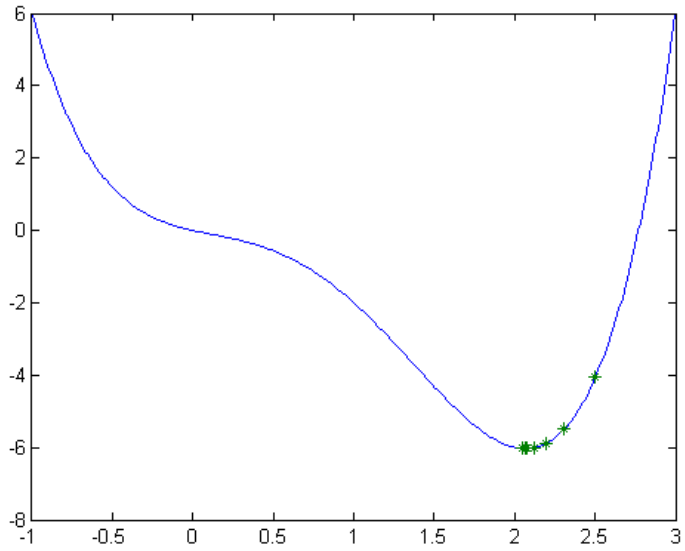
5. Wyniki

funkcja	metoda	xs	xk	eps	xmin	ymin	iter
x^2-4	bisekcja	-1	3	0.1	0.0313	-3.9961	4
x^2-4	bisekcja	-1	3	0.01	0.0039	-3.9999	7
x^2-4	bisekcja	-1	3	0.001	4.8828e-004	-4.0000	10
x^2-4	fibonacci	-1	3	0.01	0.0174	-3.9997	7
x^2-4	fibonacci	-1	3	0.001	0.0016	-4.0000	12
x^2-4	quasi-newton	-1	3	0.001	0	-4	2
$-x+x^2-3*x^3+x^4$	bisekcja	-1	3	0.1	2.0625	-6.0339	4
$-x+x^2-3*x^3+x^4$	bisekcja	-1	3	0.01	2.0664	-6.0339	7
$-x+x^2-3*x^3+x^4$	bisekcja	-1	3	0.001	2.0664	-6.0341	10
$-x+x^2-3*x^3+x^4$	fibonacci	-1	3	0.01	2.0729	-6.0337	7
$-x+x^2-3*x^3+x^4$	fibonacci	-1	3	0.001	2.0673	-6.0341	12
$-x+x^2-3*x^3+x^4$	quasi-newton	-1	3	0.001	2.0666	-6.0341	4
$sin(x)+x-x^3$	bisekcja	-15	15	0.01	-0.7578	-1.0100	7
$sin(x)+x-x^3$	bisekcja	-15	15	0.001	-0.7583	-1.0100	10
$sin(x)+x-x^3$	fibonacci	-15	15	0.01	-0.7604	-1.0099	7
$sin(x)+x-x^3$	fibonacci	-15	15	0.001	-0.7592	-1.0099	12
$sin(x)+x-x^3$	quasi-newton	-15	15	0.001	-0.7585	-1.0100	6
$x^3*arctan(x)-exp(x)$	bisekcja	-1	3	0.01	1.0039	-1.9317	7
$x^3*arctan(x)-exp(x)$	bisekcja	-1	3	0.001	1.0005	-1.9327	10
$x^3*arctan(x)-exp(x)$	fibonacci	-1	3	0.001	1.0016	-1.9327	12
$x^3*arctan(x)-exp(x)$	quasi-newton	-1	3	0.001	0.9685	-1.9350	4

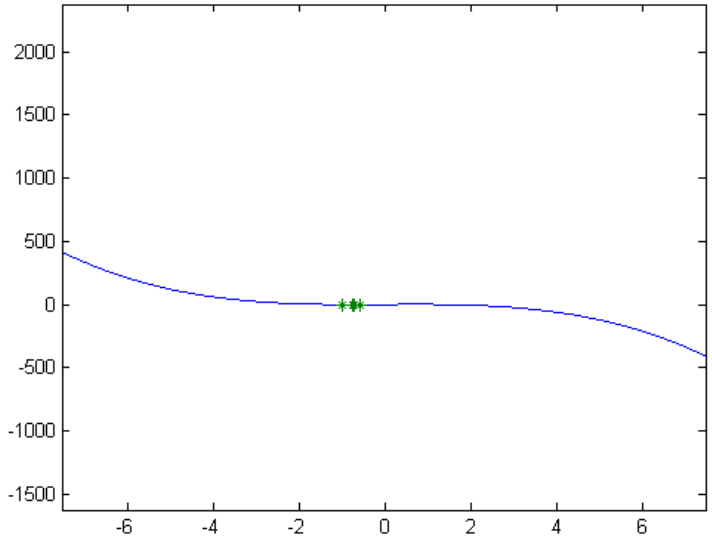
Tabela 1. Zestawienie wyników z badań metod.



Rysunek 1. Przykład graficzny kolejnych przybliżeń w metodzie bisekcji.



Rysunek 2. Przykład graficzny kolejnych przybliżeń w metodzie liczb Fibonacciego.



Rysunek 3. Przykład graficzny kolejnych przybliżeń w metodzie Quasi-newtona.

funkcja	x	y
$x^2 - 4$	-4	0
$-x + x^2 - 3 * x^3 + x^4$ [1]	2.06659	-6.03405
$sin(x) + x - x^3$ [2]	-0.758481	-1.00995
$x^3 * arctan(x) - exp(x)$ [3]	0.968539	-1.93503

Tabela 2. Zestawienie wyników uzyskanych przy pomocy WolframAlpha[4]

.

6. Wnioski

Na podstawie wykonanych badań, można stwierdzić, że metoda Quasi-newtona, osiąga najlepsze wyniki odnosząc się do wyników zestawionych w tabeli 5. Równocześnie nie tylko najdokładniejsza ale również najszybsza w działaniu. W każdym przypadku zadaną dokładność osiągała po około połowie iteracji których potrzebowały metody Fibonacciego i bisekcji.

Literatura

[1] Michał Lewandowski, *Metody optymalizacji - teoria i wybrane algorytmy*. 2012.
[2] `http://pl.wikipedia.org/wiki/Metoda_quasi-Newtona`