Labb 3: Ekvationslösning med Matlab (v2)

Envariabelanalys
2007-03-05

Björn Andersson (IT-06), bjoa@kth.se
Johannes Nordkvist (IT-06), nordkv@kth.se

Det finns flera sätt att lösa ekvationer. För enklare uttryck, som polynom av låg grad, kan man snabbt och utan större ansträngning hitta exakta lösningar. Andra uttryck kräver att man tar fram approximationer med hjälp av numeriska metoder. Syftet med denna laboration är att visa styrkorna - men också svagheterna med användandet av numeriska metoder för att lösa ekvationer i matematikbearbetningsverktyget Matlab.

Uppgift 21

Uppgiften gick ut på att pröva funktionen ”roots” på olika givna polynom.

”roots” är en numerisk funktion som letar reda på nollställen för den funktion vars koefficienter som man skickar som parameter. Den fungerar enbart för polynom.

Koefficienterna till de olika termerna i polynomen matas in i gradordning i funktionen ”roots”

Om man t.ex. matar in polynomet $1+3x-2x^3+x^5$ så blir det: roots([1 0 2 0 3 1])

Resultat av beräkningarna i Matlab:

$1+3x-2x^3+x^5$ ger rötterna $1{,}24 \pm 0{,}66i$ och $-1{,}06 \pm 0{,}53i$ och -0,36

x^2 ger dubbelroten 0

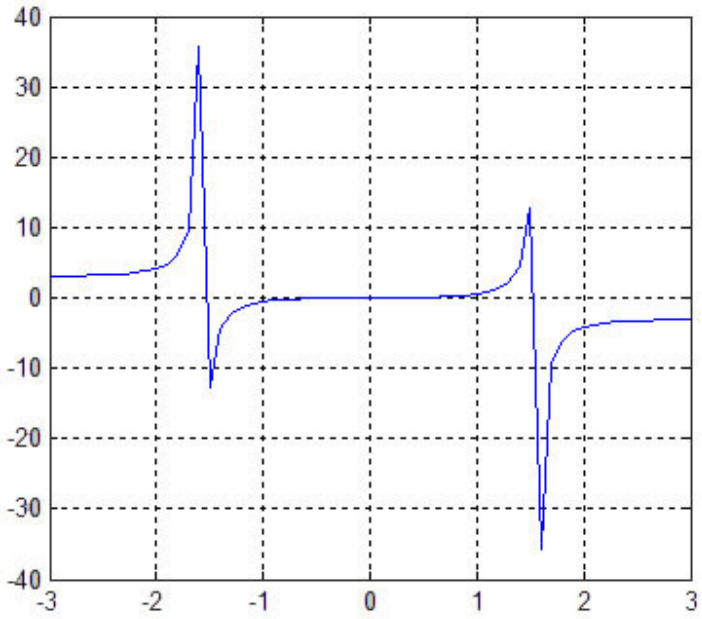
x^2+1 ger rötterna $\pm i$

Uppgift 22

Uppgiften var att testa den numeriska matlab-funktionen fzero för några givna funktioner (ej polynom). Till skillnad från ”roots” så kräver den här funktionen inget polynom men hittar endast en rot åt gången. Fzero klarar inte heller imaginära rötter.

Vi börjar att plotta funktionerna för att se var de kan tänkas ha ett nollställe. För funktionen tan(x) – x ser vi (figur 1) att första nollstället kommer strax efter x=1,2. Eftersom vi enbart ska testa funktionen så undersökta vi bara första nollstället efter x=0 för de olika funktionerna. fzero tar två parametrar, en funktion och ett startvärde. Som exempel för funktionen tan(x)-x satte vi funktionen enligt

fzero(’tan(x)-x’,1.2).



Figur 1: f(x) = tan(x)-x

$\tan(x)-x$	ger roten 1,57
x^2+1	ger inget nollställe eftersom fzero inte klarar imaginära tal.
$1-\cos(x)$	Fzero klarar inte denna funktion (säger att roten är imaginär). Vid x=0 finns dock ett nollställe (f(0)=0). Den numeriska metoden bakom fzero tycks därför vara beroende av att uttrycket ska skära och passera x-axeln för att kunna ge ett svar.

Uppgift 23

Uppgiften gick ut på att konstruera en snurra som finner gränsvärdet för den rekursivt definierade talföljden $x_{n+2}=\sqrt{x_{n+1}+x_n+1} \quad n\in \mathrm{N}, \quad x_0=x_1=1$. Vi vill med andra ord se var (och om) funktionen konvergerar.

I snurran utnyttjar vi egenskapen hos konvergerande uttryck att differensen mellan funktionsvärdena går mot noll när n går mot oändligheten. I vår snurra anser vi att vi fått ett tillräckligt noggrant värde på x gränsvärde när differensen mellan nuvarande och föregående x är en $1/10^6$ (se koden i figur 2). Om så är fallet kommer vi inte fortsätta att räkna ut nästkommande värde utan skriver ut gränsvärdet.

```
%Vi letar efter gränsvärdet, alltså när uttrycket konvergerar
bigdif = 1;
k = 0;
newn = 0;
nplus = 1;
n = 1;
while(bigdif & (k < 1000))
    newn = sqrt(nplus + n + 1);
    %Skillnaden ska vara väldigt liten
    bigdif = (abs(nplus - n)>0.000001);
    k = k + 1;
end
if(k ~= 1000)
    newn
else
    display('Konvergerar inte')
end
```

Figur 2: Matlab-kod

Vi har även infört en variabel k. Denna variabel ökar med ett för varje loop. Om talföljden inte konvergerat efter 1000 beräkningar (k = 1000 → n = 1000 + 2) så antar vi att uttrycket inte kommer att konvergera och skriver ut ett felmeddelande.

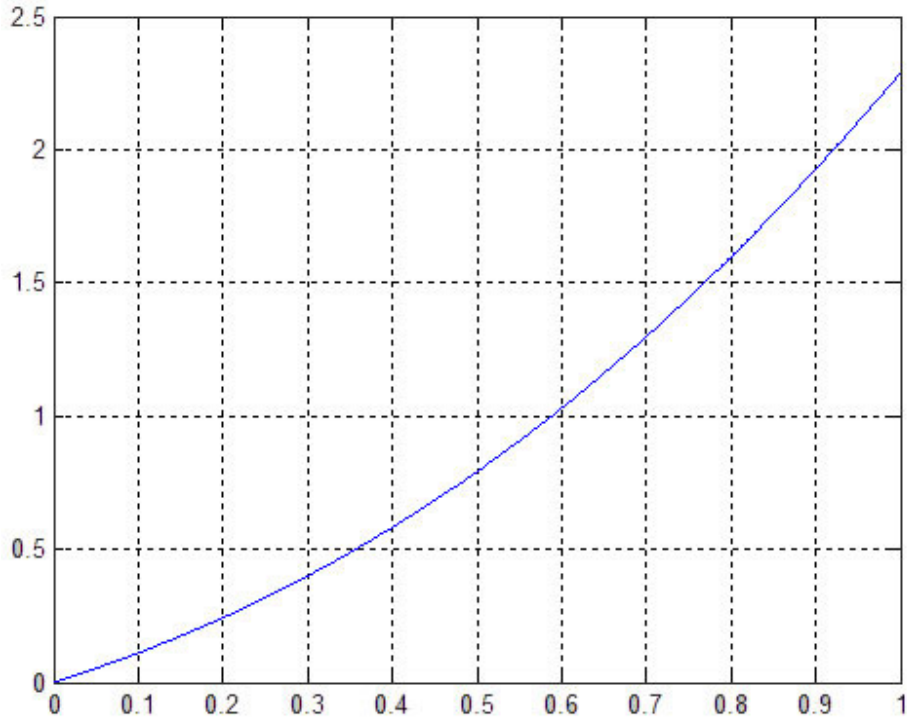
Den rekursivt definierade talföljden $x_{n+2} = \sqrt{x_{n+1} + x_n + 1} \quad n \in \mathrm{N}, \quad x_0 = x_1 = 1$ har gränsvärdet:

1,7321

Uppgift 24

Uppgiften syftar till att med olika metoder bestämma det största värdet till funktionen $f(x) = e^x \cdot \sin(x)$ i intervallet $[0,1]$ $(0 \le x \le 1)$.

Lokala extrempunkter för funktionen kommer att finns i ändpunkterna och eventuellt också där derivatan är noll. För att få en uppfattning av derivatan plottar vi funktionen i Matlab (se figur 3).



Figur 3: $f(x) = e^x \cdot \sin(x)$

Vi ser i grafen att derivatan ser ut att vara konsekvent ökande och sakna nollställen i intervallet $(0 \le x \le 1)$. I uppgiften står det dock att vi ska använda en numerisk metod för att försöka bestämma extrempunkter. Då vi inte har något polynom kan vi inte använda roots utan använder fzero. Vi använder fzero med derivatan av $f(x) = e^x \cdot \sin(x)$ som funktion-parameter och startvärde x=0 då detta inleder intervallet.

Vi börjar med att derivera funktionen på papper enligt produktregeln.

$$f(x) = e^x \cdot \sin(x) \Rightarrow f'(x) = e^x \cdot \sin(x) + e^x \cdot \cos(x) = e^x(\sin(x) + \cos(x))$$

Och exekverar därefter följande i matlab:

fzero('e^x*(sin(x)+cos(x))', 0)

Detta ger en rot då x = -0,7854, vilket inte är en del av intervallet. Misstankarna från figur 3 visade sig därför stämma. Vidare kan vi i figur 3 se att funktionens lokala max-värde kommer att ligga i ändpunkten (tillhör intervallet) med störst x-värde, då derivatan i intervallet är konsekvent positiv.

Vi beräknar med Matlab följande funktion:

fmax = e*sin(1)
fmax = 2,2874

Vidare skulle vi beräkna **exakt** värde för den lokala max-punkten. Vi sätter därför in x=1 i det ursprungliga uttrycket.

$$f(x) = e^x \cdot \sin(x)\ ;\ x_{änd} = 1 \Rightarrow f_{\max}(1) = e^1 \cdot \sin(1) = e\sin(1)$$

Slutligen skulle vi använda funktionen fmin för att ta reda på max-värdet. Då fmin är ”deprecated” och inte fungerar i Matlab R2006a använde vi fminbnd som fungerar på samma sätt. Den använder sig av intervallhalveringsteknik. För att kunna använda fmin till att ta ut maxvärden så sätter man ett minustecken framför funktionen som man använder som parameter. Övriga parametrar var start och slutvärde på det givna intervallet [0,1].

fminbnd(’-(2.71828^x*sin(x))’,0,1)

Vi erhöll med denna metod f_{max}= 2,2871

Vad vi kan se är att resultatet från fminbnd (2,2871) skiljer sig från värdet vi erhöll då vi satte in ändpunkten (x=1) av funktionen $f(x) = e^x \cdot \sin(x)$. Detta beror på att fminbnd är en numerisk funktion som använder sig av intervallhalveringsteknik. Precis som Arne Persson och Lars-Christer Böiers skriver i boken Analys i en variabel så kommer denna numeriska metod (likt andra numeriska metoder) endast fungera som en grov approximation (som bör kompletteras med andra metoder). För att intervallhanteringsteknik ska ge noggranna resultat krävs många exekveringssteg, vilket skulle kräva mycket resurser - även för verktyg som Matlab.

Resultat från uppgift 24:
Max-värde m.h.a. insättning av ändpunkt i funktion: 2,2874
Max-värde m.h.a. fminbnd: 2,2871 (se anledning ovan)

Uppgift 25

Uppgiften var att göra en Matlab-funktion som använder sig av Newton-Rhapsons algoritm. Den skulle ta tre parametrar: en matematisk funktion, ett startvärde och ett toleransvärde.

Den matematiska funktionen definierades som matematisk funktion m.h.a. Matlab-funktionen fcnchk som även används av matlab-funktioner som fzero.

För att beräkna derivatan till funktionen så tog vi två olika värden för funktionen och jämförde skillnaden i y- och x-led. De två värden vi tog var: startvärde-0,0001 och startvärde+0,0001. Det ena kallade vi för y1 och det andra för y2. Skillnaden mellan dessa två y-värden kallade vi för dy. dx var alltid 0,0002, oavsett värden på y. På så vis fick vi ut lutningen.

Vi kunde därefter sätta in de erhållna värdena enligt Newton-Raphsons metod

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

x_{n+1} blev nya startvärdet när loopen började på nytt.

Uttrycket i Newton-Raphsons metod kommer att konvergera. Vi använder därför en snurra som liknar den vi konstruerade i uppgift 24. Snurran tar fram nya värden till dess att differensen mellan talet x_{n+1} och x_n är mindre än den av användaren givna toleransen.

Nedan visas Matlab-funktionen:

```
function rot = NRrot(f,x0,tolerans)

func = fcnchk(f);
previousRoot = x0 + 1000;
format long
while(tolerans<abs(previousRoot-x0))
    fvalue = func(x0);
    y2 = func(x0+0.0001);
    y1 = func(x0-0.0001);
    dx = 0.0002;
    dy = y2 - y1;
    deriv = dy/dx;
    previousRoot = x0;
    x0 = x0 - fvalue/deriv;
end

rot = previousRoot;
end
```

Vi visar nu att ovanstående funktion ger korrekta rötter (en åt gången) för olika matematiska funktioner. Vi börjar med att söka rötterna för olika polynom. I figur 4 visas hur NRrot används för att hitta roten (närmast 2) för polynomet x^2-1(=0). Därefter finner vi roten enligt $x^2 = 2 \rightarrow x = 2^{1/2}$. Som vi ser i figuren ger de båda inmatningarna i Matlab samma resultat.

Vi ger därefter rot närmast x=2 för x^3-8(=0) och x^4-16(=0) enligt ovan nämnda tillvägagångssätt.

```
>> NRrot('x^2-2',2,0.0001)

ans =

   1.41421568627451

>> 2^(1/2)

ans =

   1.41421356237310

>> NRrot('x^3-8',2,0.0001)

ans =

     2

>> 8^(1/3)

ans =

     2

>> NRrot('x^4-16',2,0.0001)

ans =

     2

>> 16^(1/4)

ans =

     2
```

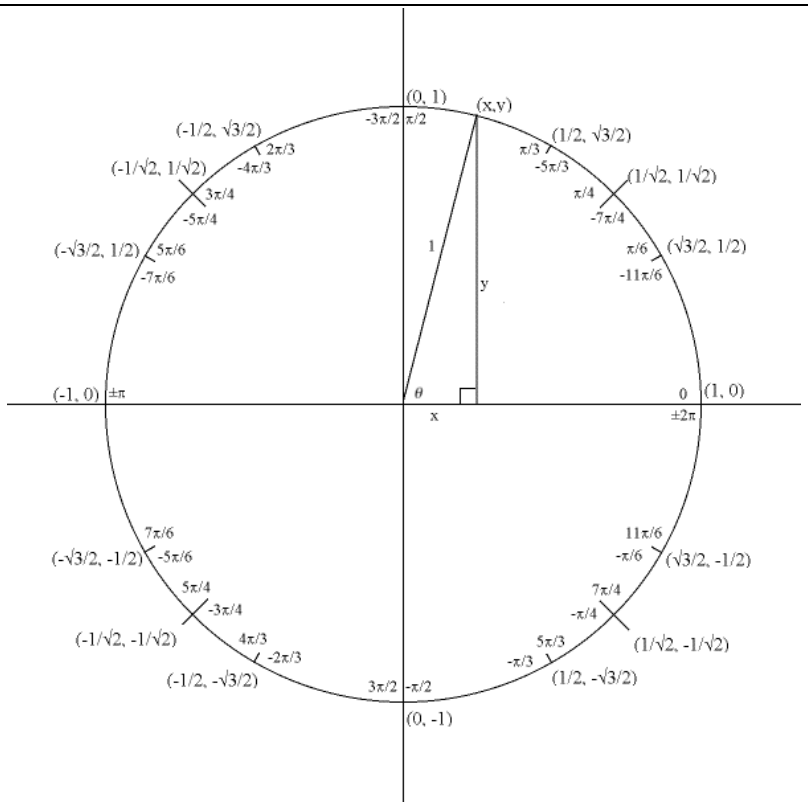
Figur 4: Test av NRrot i Matlab

I figur 5 testar vi funktionen NRrot för matematiska funktioner som inte är polynom. Vi finner roten närmast 2 för funktionerna f(x) = cos(x) och f(x) = sin(x).

```
>> NRrot('cos(x)',2,0.0001)

ans =

   1.57080400830615

>> NRrot('sin(x)',2,0.0001)

ans =

   3.14159265368188
```

Figur 5: Test av NRrot för funktioner (icke-polynom)

I figur 5 ser vi att NRrot ger roten $1{,}5708 = \frac{\pi}{2}$ för f(x)=cos(x). I figur 6 ser vi att enhetscirkeln visar $\cos\left(\frac{\pi}{2}\right) = 0$. Vidare ger NRrot roten 3,1416=π för f(x)=sin(x) nära x=2. Vi ser att även detta stämmer med enhetscirkeln i figur 6.



Figur 6: Enhetscirkeln (bildkälla: http://sv.wikipedia.org/wiki/Cos)

Resultat

Uppgift 21

$1+3x-2x^3+x^5$ ger rötterna $1{,}24 \pm 0{,}66i$ och $-1{,}06 \pm 0{,}53i$ och -0,36

x^2 ger dubbelroten 0

x^2+1 ger rötterna $\pm i$

Uppgift 22

$\tan(x)-x$ ger nollstället 1,57

x^2+1 ger inget nollställe eftersom fzero inte klarar imaginära tal.

$1-\cos(x)$ Fzero klarar inte denna funktion (säger att roten är imaginär). Vid x=0 finns dock ett nollställe (f(0)=0).

Uppgift 23

Den rekursivt definierade talföljden $x_{n+2} = \sqrt{x_{n+1} + x_n + 1} \quad n \in \mathrm{N}, \quad x_0 = x_1 = 1$ har gränsvärdet:

1,7321

Uppgift 24

Max-värde m.h.a. insättning av ändpunkt i funktion: 2,2874
Max-värde m.h.a. fminbnd: 2,2871 (se uppg 24 för anledning)

Uppgift 25

```
function rot = NRrot(f,x0,tolerans)

func = fcnchk(f);
previousRoot = x0 + 1000;
format long
while(tolerans<abs(previousRoot-x0))
    fvalue = func(x0);
    y2 = func(x0+0.0001);
    y1 = func(x0-0.0001);
    dx = 0.0002;
    dy = y2 - y1;
    deriv = dy/dx;
    previousRoot = x0;
    x0 = x0 - fvalue/deriv;
```

```
end

rot = previousRoot;
end
```

Källor

Persson, Böiers. Analys i en variabel. Andra upplagan. Tryckt 2006, utgiven av Studentlitteratur.

Kompendium: Envariabelanalys i Matlab. Läsåret 2001. 2007-03-02: http://www.math.kth.se/math/student/courses/5B1147/ME/200607/matlabenvar/5B1147matlab.pdf

Bilaga: Matlab-kod

```
clc, clear
display('21: roots')
display('..............................................................
')
display('1+3x-2x^3+x^5')
roots([1 0 -2 0 3 1])

display('x^2')
roots([1 0 0])

display('x^2+1')
roots([1 0 1])
display('..............................................................
')

display('22: fzero')
display('..............................................................
')
display('tan(x)-x')
x=-3:0.1:3;
plot(x,tan(x) -x),grid
fzero('tan(x)-x', 1.2)

display('1-cos(x)')
x=-5:0.1:5;
figure(2)
plot(x,1-cos(x)),grid
display('Vi ser i grafen att f(x)=1-cos(x) aldrig skär x-axeln')
display('')

display('x^2+1')
x=-5:0.1:5;
figure(3)
plot(x,x.^2+1),grid
display('Vi ser i grafen att f(x)=x^2+1 aldrig skär x-axeln')
display('..............................................................
')

display('23: Finn gränsvärdet för newn = sqrt(nplus + n + 1)')
display('..............................................................
')
%Vi letar efter gränsvärdet, alltså när uttrycket konvergerar
bigdif = 1;
k = 0;
newn = 0;
nplus = 1;
n = 1;
while(bigdif & (k < 1000))
    newn = sqrt(nplus + n + 1);
    %Skillnaden ska vara väldigt liten
    bigdif = (abs(nplus - n)>0.000001);
    k = k + 1;
end
```

```
if(k ~= 1000)
    newn
else
    display('Konvergerar inte')
end
display('.....................................................................
')
display('24: Finn maxvärde för funktionen f(x)=e^x*sin(x)')
display('.....................................................................
')
display('x nära noll som ger max f(x)')

%Vi itererar fram funktionsvärden
index = 0;
for x = 0:0.001:1
   index = index +1;
   y(index) = 2.71828^x*sin(x);
end

figure()
x = 0:0.001:1;
plot(x,y),grid
fzero('2.71828^x*(cos(x)+sin(x))', 0)

display('x-värdet tillhör inte intervallet, störst i ändpunkten (då
x=1)')
x=1;
2.71828^x*sin(x)
display('Det exakta max-värdet är e(sin(1)+cos(1)')
display('')
x = fminbnd('-(2.71828^x*sin(x))',0,1);
max_med_fminbnd = 2.71828^x*sin(x)


function rot = NRrot(f,x0,tolerans)

func = fcnchk(f);
previousRoot = x0 + 1000;
format long
while(tolerans<abs(previousRoot-x0))
    fvalue = func(x0);
    y2 = func(x0+0.0001);
    y1 = func(x0-0.0001);
    dx = 0.0002;
    dy = y2 - y1;
    deriv = dy/dx;
    previousRoot = x0;
    x0 = x0 - fvalue/deriv;
end

rot = previousRoot;
end
```