TANA17 Matematiska beräkningar med MATLAB för M, DPU

Fredrik Berntsson, Linköpings Universitet

Föreläsning 5

- Funktioner. Programstruktur. Rekursiva funktioner.
- Exempel: Skalärprodukt. Ekvationslösning.
- Funktionshandtag. Integraler och Ekvationer.
- Tillämpning: Vattenlinjen för en båt.

Funktioner i MATLAB

Definition En *funktion* har ett antal *inagrument* och beräknar ett antal *utargument*.

Exempel Funktionen `zeros` har som inparameter två heltal N och M. Utparameter är en matris av dimension $N \times M$ med nollor.

```
>> Z = zeros( N , M );
```

I MATLAB skapas funktioner genom att man samlar kommandon på en fil, exemelvis `min_funktion.m`. Dessutom skall man skriva ett *funktionshuvud*.

```
%
% Inledande kommentar
%
function [ut1,ut2]=min_funktion( in1,in2,in3 )

   <beräkna ut1,ut2 givet in1,in2,in3>

end
```

En funktion kan ha godtyckligt antal in-, respektive ut-parametrar. Den inledande kommentaren skrivs ut om man skriver

```
>> help min_funktion
```

Exempel På filen `funk.m` har vi skrivt

```
function [f]=funk(x,y)
   y=2*x+y;
   f=x^2+2*y;
end
```

Vad händer om vi skriver följande kommandon

```
>> x=1;y=2;
>> x=funk( x , y );
>> disp(x),disp(y)
```

Exempel På filen `serie.m` har vi skrivt

```
function [S]=serie(N)
   S=0;
   for k=1:N,S=S+1/k^2;,end
end
```

Skriver vi följande kommandon

```
>> S=serie( 100 );
>> disp(S)
     1.6350
>> disp(k)
     Undefined function or variable 'k'.
```

Variabler är *lokala*. Skapas, eller ändras, en variabel i en funktion är det en lokal kopia som ändras. Då funktionen avslutas är det endast utparametrar som sparas.

Exempel Skalärprodukten mellan två vektorer x och y kan beräknas med formeln

$$x \cdot y = \sum_{i=1}^{n} x_i y_i.$$

Skriv en funktion som beräknar skalärprodukten. Funktionen skall användas enligt

```
>> S = ScalarProd( x , y );
```

På filen `ScalarProd.m` skriver vi:

```
% ScalarProd: Beräkna skalärprodukt mellan två
% vektorer x och y. Anropas enligt:
%
%   >> S = ScalarProd( x , y );
%
function [S]=ScalarProd( x , y )
  n=length(x);
  S=0;
  for i=1:n
       S=S+x(i)*y(i);
  end;
```

Kommentar Då $x \cdot y = x^T y$ kan vi skriva funktionen enklare. Det finns även en fördefinierad funktion `dot()`.

Exempel Då man testar sorteringsmetoder vill man ofta ha en vektor som nästa är sorterad. Skriv en funktion som tar en sorterad vektor som inparameter och gör ett antal slumpmässiga platsbyten.

Vi skriver en funktion `Swap`

```
function [L]=Swap(L,i,j)
    tmp=L(i);L(i)=L(j);L(j)=tmp;
end
```

och vill utnyttja den för att skriva en `Scramble` funktion som utför ett antal slumpmässiga byten.

På filen `Scramble.m` skriver vi

```
function L=Scramble( L,N )
   n=length(L);
   for i=1:N
      i=randi( [1 N],1);
      j=randi( [1 N],1);
      L=Swap( L , i , j );
    end
end
```

Vi kan använda funktionen genom att skriva

```
>> L = (0:9);
>> L = Scramble( L , 4 )
  L =
   0   8   2   3   1   5   6   7   9   4
```

Exempel I ett spel försöker vi slå en tärning högst 10 gånger i följd. Målet är att få samma resultat två gånger i rad.

Skriv ett Matlab funktion som simulerar en sådan spelomgång. För varje lyckad omgång skriver vi ut antalet kast som krävdes. Vi skall kunna anropa funktionen som

```
>>[ AntalKast,Vinst]=SpelOmgang();
```

I filen `SpelOmgang.m` skriver vi

```
function [AntalKast,Vinst]=SpelOmgang()
  TidigareKast=randi([1 6],1);
  Vinst=0;
  for k=2:10
     NyttKast=randi([1 6],1); % Nästa tärning
     if NyttKast == TidigareKast;
       Vinst = 1;
       break
     end
     TidigareKast=NyttKast;
  end
  AntalKast=k;
end
```

Nu kan vi använda funktionen för att simulera flera spel omgångar!

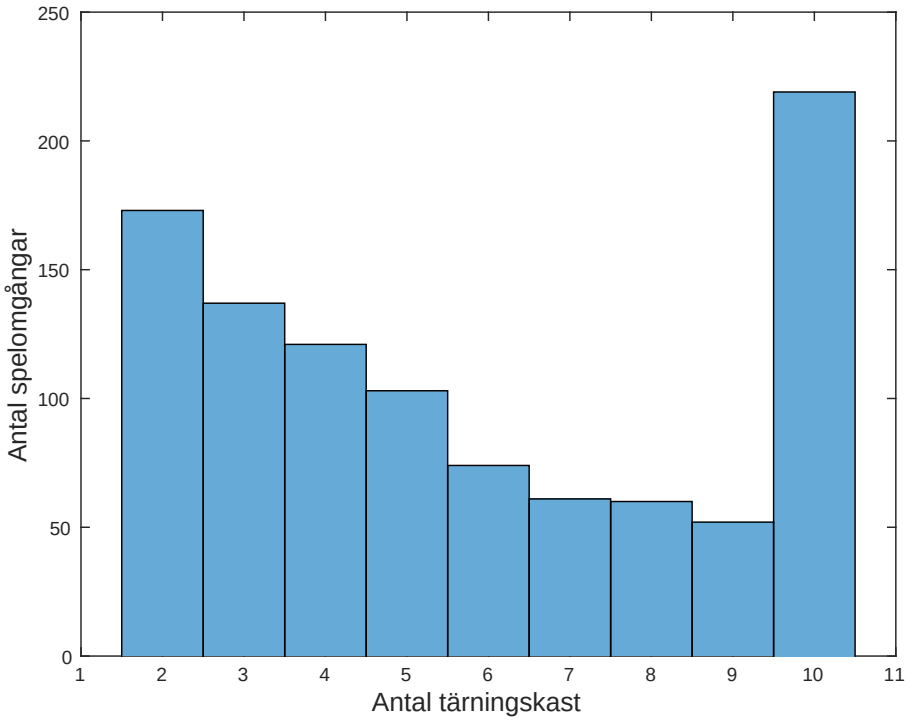
Funktionen `histogram` används för att illustrera en fördelning på olika utfall. Om de möjliga utfallen består av heltal används

```
histogram( x , 'BinMethod','integers')
```

för att skapa ett histogram som visar vektorn x.

Exempel Vi vill använda vårt program för att studera hur tärningsspelet fungerar. Simulera $N = 1000$ omgångar och hitta andelen omgångar där vi lyckas.

Rita dessutom upp ett histogram där vi tydligt ser fördelningen för antalet tärningskast i varje omgång.



I Matlab får vi resultatet

```
>> TarningSpel
 Vi vann totalt 821 av 1000 omgångar
```

Kommentar Med funktioner kan man dela upp en större uppgift i mindre delar som blir lättare att lösa.

Rekursiva funktioner

En funktion som anropar sig själv kallas *rekursiv*. Vid varje funktionsanrop skapas nya lokala kopior av variabler.

Exempel Fakulteten är definerad genom att

$$n! = n \cdot (n-1)!, \qquad 0! = 1.$$

Skriv en MATLAB funktion som beräknar $n!$ för ett givet heltal n.

Lösning På filen `Fakultet.m` skriver vi

```
function [F]=Fakultet(N)

    if N==0,
      F=1;
    else
      F=N*Fakultet(N-1);
    end;
end
```

Vad händer om vi skriver

```
>> Fakultet( 4 );
```

Funktioner som inargument

Kommandot `feval` kan användas för att anropa en funktion med givna inparametrar.

Exempel Har vi skrivit en fil `funk.m` som innehåller

```
function [z]=funk( x , y )
   z=x^2+3*y;
end
```

så kan vi anropa funktionen med inargument $x = 2$ och $y = 2.5$ genom att skriva

```
>> feval( @funk , 2 , 2.5 )
 ans =
  11.5000
```

Lösning av Icke-Linjära Ekvationer

Exempel Vi skall beräkna roten x^* till ekvationen,

$$f(x) = \cos(3x)\mathrm{e}^{\sqrt{x}}.$$

Metod Om vi beräknar funktionsvärden för $x=0$ och $x=1$ så ser vi att $f(0)=1>0$ och $f(1)=-2.69<0$. Alltså finns det en rot i intervallet $0<x<1$.

Beräkna sedan $f(0.5)=0.144>0$ och vi ser att roten finns i intervallet $\frac{1}{2}<x<1$. Detta kallas *intervallhalveringsmetoden*.

Avsluta när intervallet $[a_n, b_n]$ som innehåller roten uppfyller $b_n - a_n < \varepsilon = 10^{-6}$.

Övning Skriv en funktion `IntHalv` som beräknar roten till $f(x) = 0$ med hjälp av intervallhalveringsmetoden.

Frivilliga inargument

Kommandot `nargin` ger det antal inargument som gavs då en funktion anropades.

Exempel På filen `funk.m` skriver vi

```
function [y]=funk( x , p )

    if nargin<2 ,  p=1;  end
    y=exp(x^p);
end
```

I Matlab skriver vi sedan

```
>> y = funk( 2 );          % p=1 används.
```

Anropar vi med `y=funk(2,1.5)` används $p = 1.5$.

Enkla funktioner

Enkla funktioner kan skapas direkt i Matlab. Man skriver

```
>> f = @( <variabler> )  <uttryck>;
```

Man får då ett handtag till en funktion.

Exempel Skriv

```
>> f = @(x) exp(2*x).*cos(x.^2)+4*x
  f =
    @(x)exp(2*x).*cos(x.^2)+4*x
```

Funktionen kan beräknas med `feval` eller direkt `f(4)`.

Exempel Skapa funktionerna

$$f(x) = \sqrt{1 - x^2}\mathrm{e}^{-2x}, \quad \text{och} \quad g(x) = (x_1^2 - 1\,,\ \sqrt{1 + x_2})^T.$$

Tänk på att $g(x)$ måste ha en vektor som inagrument och returnera en vektor i $\mathbb{R}^2$.

Försök få $f(x)$ att acceptera vektor argument så att det går att skriva

```
>> x = 0:0.01:1;
>> plot( x , f(x) )
```

Funktionen `integral` beräknar integraler. Den anropas

```
>> I = integral( fun , a , b )
```

där `f` är ett funktionshandtag.

Exempel Beräkna samma integral som tidigare genom

```
>> f = @(x) exp(2*x).*cos(x.^2)+4*x;
>> I = integral( f , 0 , 1)
  I =
   4.6736
```

Det finns även `integral2` för dubbelintegraler.

Exempel Lös ekvationen $f(x) = \mathrm{e}^{-2x} - x = 0$.

Börja med att skapa en funktion

```
>> f = @(x) exp(-2*x)-x;
```

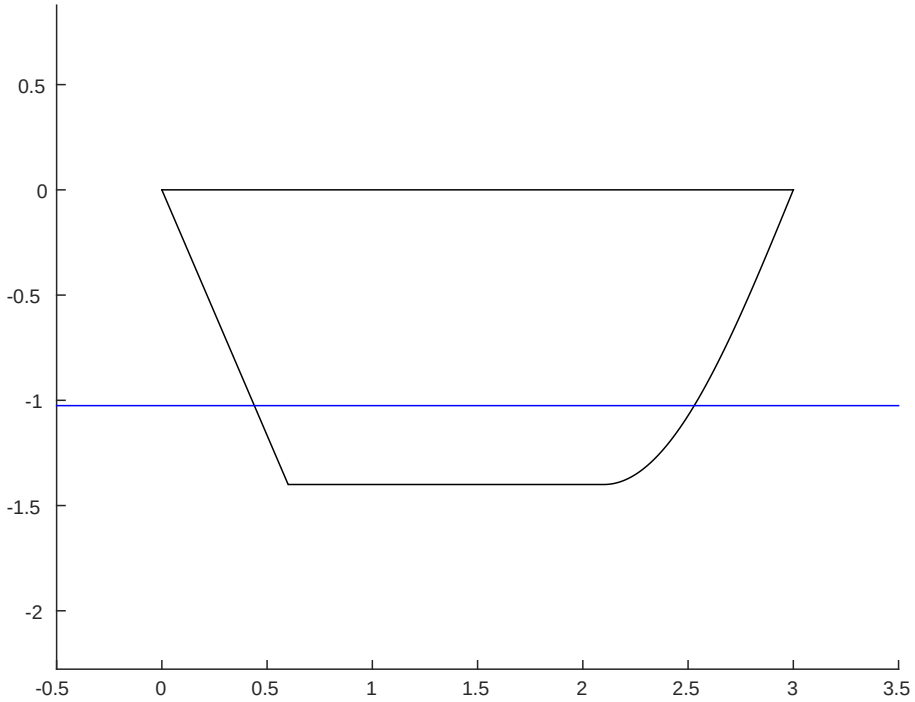
använd sedan `fzero` för att hitta roten

```
>> x = fzero( f , 1 )
 x =
  0.4263
```

Kommentar Funktionen `fzero` använder en avancerad numerisk metod. Det finns färdiga metoder för att lösa en stor mängd problem i MATLAB.

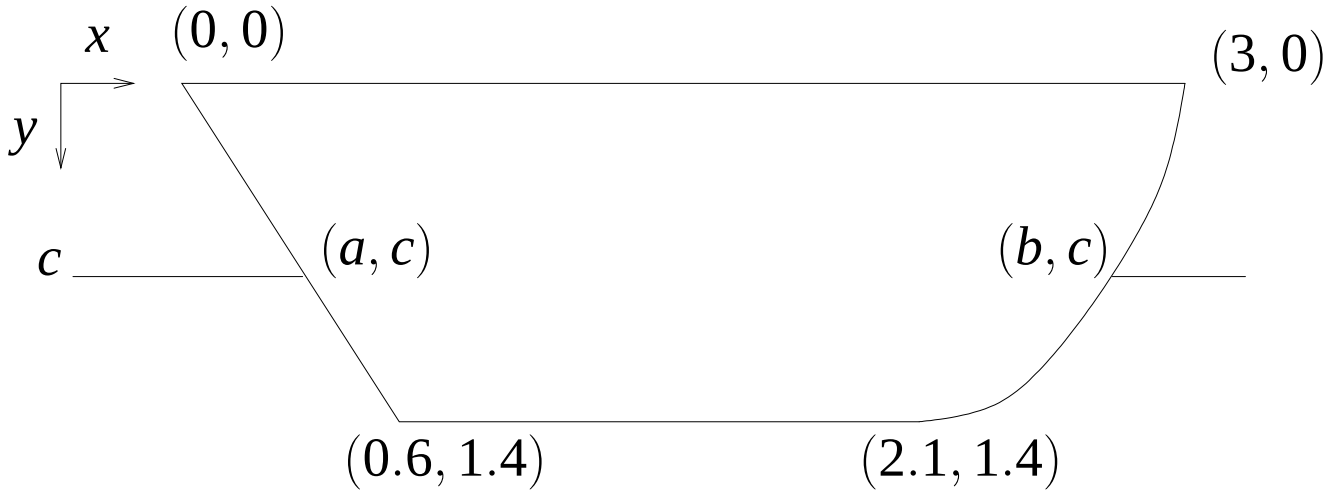
Tillämpning: Hur stor del av en båt är under vatten?

Problem En båt som lastas sjunker precis så mycket att båtens vikt blir precis motsvarar det vatten den tränger undan.



Givet en viss vikt hur skall vi beräkna hur stor del av båten som hamnar under vatten?

Problem beskrivning Inför relevanta storheter och beskriv lösningsmetoden. Båtens profil ges av en skiss



Båtens vikt inklusive last är $L = 697\,kg$. Vattnets densitet är $998\,kg/m^3$.

Funktioner `ShipShape(x)`, `DisplayShip(c)` och `DisplacedVolume(c)`.

Funktionen `ShipShape` blir

```
function F = ShipShape( x )
  F=zeros(size(x));
  for i=1:length(x)
     if x(i)<0.6
        F(i)=1.4*x(i)/0.6;
     elseif x(i)<2.1
        F(i)=1.4;
     else
        F(i)=1.4*cos( pi*(x(i)-2.1)/(3.0-2.1)/2 );
     end
  end
end
```

Kommentar Rät linje från $(0, 0)$ till $(0.6, 1.4)$ följt av en cosinus från $(2.1, 1.4)$ till $(3.0, 0)$. Funktionen måste klara en vektor x som inargument.

Funktionen `DisplayShip` blir

```
function DisplayShip( c )
   x=0:0.01:3;
   F=ShipShape( x );
   clf,hold on
   plot( x , -F , 'k-' )
   plot( [0 3],[0,0],'k-')
   plot( [-0.5 3.5],[-c -c],'b')
   axis([-0.5 3.5 -1.6 0.2 ] )
   axis equal
end
```

Kommentar y-axeln är definierad så att $y = 0$ är överst på båten. Måste alltså byta tecken innan vi ritar!

Funktionen `DisplacedVolume` blir

```
function V=DisplacedVolume( c )

  % Första roten bör ligga nära x=0.4;
  a=fzero( @(x)ShipShape(x)-c ,0.4);

  % Andra roten bör ligga närmare x=2.5;
  b=fzero(@(x)ShipShape(x)-c ,2.5);

  % Undanträngd volym är en ingetral över [a,b]

  V=integral( @(x)ShipShape(x)-c ,  a , b );
end
```

Kommentar Hitta skärningspunkterna mellan skrovet och vattenytan. Integrera sedan för att hitta totala arean under ytan.

Huvudprogrammet blir

```
L=697;  % Last i Kg
rho=998; % Vattnets densitet i kg/m3

% Ekvation Volym( c )*rho - L = 0. Start gissning är

c = fzero( @(c)DisplacedVolume( c )*rho-L , 0.9 )
DisplayShip( c )

weight=DisplacedVolume( c) * rho;
str=['Med last L=',num2str(L),'kg fås c='];
str=[str,num2str(c),' m'];disp(str)
```

Resultatet blir

```
Med last L=697kg fås c=1.0253 m
```

För Laborationer med lite större uppgifter

- Tänk igenom vad som skall göras. Inför beteckningar för obetanta och konstanter.
- Identifiera deluppgifter som kan lösas varför sig. Detta ger lämpliga funktioner att skriva.
- Normalt relativt lite tid då man faktiskt arbetar vid datorn.