MAI/Linköpings universitet Fredrik Berntsson

Tentamen TANA17 Matematiska beräkningar
Provkod: DAT1 **Godkänd:** 8p av totalt 20p **Tid:** 14:e januari klockan 8.00-12.00

Redovisning Lös först uppgifterna i Matlab. Då du har en färdig lösning skriv då ner de kommandon du använde på papper. Redovisa även eventuella resultat du fick då körde dina Matlab kommadon. Grafer behöver inte redovisas.

(2p) **1:** Använd kolon-notation för att skapa en vektor som innehåller talen $1, 2, 3, \ldots, 99, 100$. Använd sedan elementvisa operationer och kommandot `sum` för att beräkna summan

$$S = \sum_{k=1}^{100} \frac{1}{1+k}.$$

(2p) **2:** Använd `polyfit` för att anpassa en rät linje till följande tabell

x	0	1	2	3
y	1.2	1.5	1.9	2.1

Plotta även den räta linjen på intervallet $0 \leq x \leq 3$ som en blå heldragen linje. Rita även in punkterna (x_i, y_i) i samma graf med röda `x`.

(1p) **3:** Beräkna lösningen till det linjära ekvationssystemet $Ax = b$ där

$$A = \begin{pmatrix} 1 & -2 & 2 \\ 3 & 2 & -1 \\ 2 & -1 & 3 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}.$$

(2p) **4:** Följande kommandon är tänkta att kontrollera ifall en vektor är sorterad i stigande ordning:

```
for i=2:length(x)
  if x(i)>x(i-1)
    Stigande=1;
  else
    Stigande=0;
  end
end
```

Förklara vad som går fel och gör de ändringar som krävs för att programmet skall fungera som det är tänkt.

(2p) **5:** Låt A vara en matris av storlek $n \times m$. Skriv ett Matlabscript som använder `for`-loopar för att beräkna summan av de positiva talen i matrisen. Testa ditt program på matrisen

$$A = \begin{pmatrix} 1 & -1 & 0 & 3 \\ -3 & 2 & 1 & 0 \\ 2 & -1 & 3 & -1 \end{pmatrix}.$$

(2p) **6:** Ofta vill man införa olika storleksmått för vektorer. Exempelvis definieras

$$\|x\|_1 = \sum_{k=1}^{n} |x_k|.$$

Skriv en funktion `Storlek` med en vektor `x` inparametrar och med storleksmåttet $S = \|x\|_1$ som utparameter.

Utnyttja funktionen för att beräkna storleken av vektorn $x = (2\,,\, -3\,,\, 1)^T$.

OBS I Matlab finns en inbyggd funktion `norm` som kan utföra precis denna beräkning. Du kan använda den för att kontrollera att du gjort rätt men får inte utnyttja den för att lösa uppgiften. Du får heller inte använda kommandot `sum`.

(3p) **7:** I ett spel skall vi kasta tärningar i följd tills vi får en sexa. Skriv ett Matlab program som simularar en sådan spelomgång genom att slumpa fram heltal mellan 1 och 6 tills resultatet blev en sexa. Ditt program skall göra utskrifter enligt följande mönster

```
Tärning 1: 2
Tärning 2: 3
Tärning 3: 1
Tärning 4: 6
Vi fick en sexa efter 4 tärningskast.
```

(2p) **8:** Vi vill undersöka funktionen

$$f(x) = \cos(x)\mathrm{e}^{-2x} - 2x, \quad 0 \le x \le 1.$$

a) Rita en graf över funktionen $f(x)$ på intervallet $0 \le x \le 1$.

b) Vi ser att ekvationen $f(x) = 0$ har en rot nära $x = 0.3$. För att kunna hitta roten skall du skriva en Matlab funktion `funk` med en vektor x som inparameter som beräknar en vektor y med motsvarande funktionsvärden.

Din funktion skall vara skriven på ett sådant sätt att det går att beräkna roten med kommandot

```
>> x = fzero( 'funk' , 0.3 )
```

(4p) **9:** En magisk kvadrat har egenskapen att alla rader, kolumner, och bägge diagonalerna har samma summa. Exempelvis är

$$A = \begin{pmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{pmatrix}$$

en magisk kvadrat då summan av elementen på varje enskild rad är 65, och samma summa fås för varje kolumn, och även för de bägge diagonalerna.

Du skall skriva en funktion `magisk` med en kvadratisk matris `A` som inparameter och en logisk variabel som utparameter. Utparametern skall ha värdet *sant* om `A` är en magisk kvadrat och värdet *falskt* annars.

Tips I Matlab finns en funktion `magic` som kan användas för att skapa magiska kvadrater av godtycklig dimension. Dessa kan användas för att testa din funktion.

Lösningsförslag till tentan för TANA17 Januari 2016.

1: Vi beräknar summans värde med kommandona

```
>> k=1:100;
>> S=sum( 1./(1+k) )
  S =
   4.1973
```

2: Vi använder kommandona

```
>> x=[0 1 2 3];y=[ 1.2 1.5 1.9 2.1];
>> P1 = polyfit(x,y,1);
>> plot( [0 3],polyval( P1 , [0 3] ),'b-')
>> hold on
>> plot( x,y,'rx')
>> hold off
```

Vi ser att linjen ligger nära punkterna.

3: Bilda matrisen och lös problemet med

```
>> A=[ 1 -2 2 ; 3 2 -1 ; 2 -1 3]; b=[ 2 1 0]';
>> x=A\b
  x =
   1.0769
  -1.7692
  -1.3077
```

4: Problemet är att endast sista kontrollen räknas. Enklaste lösningen är att avbryta då vi hittar två element som inte ligger i stigande ordning med `break`. Vi får

```
for i=2:length(x)
  if x(i)>x(i-1)
    Stigande=1;
  else
    Stigande=0;
    break
  end
end
```

Det finns flera alternativa lösningar som ger enklare kod. Exempelvis

```
Stigande=1;
for i=2:length(x)
  if x(i) < x(i-1) % Vänt på jämförelsen!
```

```
    Stigande=0;
    break
  end
end
```

5: Vi bildar matrisen och beräknar summan med

```
A=[ 1 -1 0 3 ; -3 2 1 0 ; 2 -1 3 -1];

[n,m]=size(A);
S=0;
for i=1:n
  for j=1:m
     if A(i,j)>0, S=S+A(i,j); end
   end
end
disp(S)
```

vilket ger värdet $S = 12$.

6: Funktionen blir

```
function [E]=Storlek(x)
      n=length(x);
      E=0;
      for i=1:n
         E=E+abs( x(i) );
      end
end
```

vilket get

```
>> Storlek( [2 -3 1]' )
  ans =
      6
```

7: Vi simulerar spelomgången med

```
SistaKast=-1;  % Bara så att inte while-satsen avbryts direkt
AntalKast=0;

while SistaKast ~= 6
       AntalKast=AntalKast+1;
       SistaKast=randi([1 6],1);
       fprintf(1,'Tärning %i: %i\n',AntalKast,SistaKast);
end
fprintf(1,'Vi fick en sexa efter %i tärningskast.\n',AntalKast);
```

Det går lika bra att göra utskeifterna med

```
disp(['Tärning ',num2str(AntalKast),': ',num2str(SistaKast)]);
```

men `fprintf` är enklare.

8: Foljande kommandon plottar grafen

```
>> x=0:0.01:1;
>> f=cos(x).*exp(-2*x)-2*x;
>> plot( x , f ;
```

På filen `funk.m` skriver vi sedan

```
function y=funk(x)
   y=cos(x).*exp(-2*x)-2*x;
end
```

I Matlab kan vi sedan beräkna roten med

```
>> x=fzero('funk',0.3)
 x =
  0.2766
```

9: På filen `Magisk.m` skriver vi

```
function [Test]=Magisk( A )
   [n,m]=size(A)
   Test=1;   % Antag sant och motbevisa
   D1=0;D2=0;
   for i=1:n   % Diagonalsummor först
      D1=D1+A(i,i);
      D2=D2+A(i,n-i+1);
   end
   if D1 ~= D2
      Test=0; % Motbevisat!
   else

     for i=1:n, % kolla en rad eller kolumn i taget
        R=0; C=0;
        for j=1:n
          R=R+A(i,j); C=C+A(j,i);
        end;
        if ( R ~= D1 ) | ( C ~= D1 )
          Test=0; break
         end
     end
   end
end
```