TANA17 Matematiska beräkningar med Matlab

Datorlektion 4. Funktioner

1 Egna Funktioner

Uppgift 1.1 En funktion $f(x)$ ges av uttrycket

$$f(x) = \begin{cases} 0, & x \leq 0, \\ \sin(x), & 0 < x \leq \frac{\pi}{2}, \\ 1, & x > \frac{\pi}{2} \end{cases}$$

a) Skriv en Matlab funktion `funk.m` som implementarar uttrycket ovan. Din funktion skall ha ett reellt tal x som inparameter, och returnera motsvarande funktionsvärde som utparameter.

b) Skriv ett program som ritar en graf över $f(x)$, på intervallet $-1 < x < 2$. Ditt program skall använda funktionen `funk.m` ifrån **a)**. □

Uppgift 1.2 Skalärprodukten mellan två vektorer x och y definieras som

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

Skriv en funktion `ScalarProd`, med två vektorer x och y som inparametrar och som beräknar skalär produkten med hjälp av en `for`-loop.

Använd din funktion för att beräkna skalärprodukten mellan vektorerna $x = (-1\,,\, 4\,,\, -2)^T$ och $y = (3\,,\, -2\,,\, 1)^T$. □

Tips Det finns en standard funktion `dot` som beräknar skalärprodukten. Du kan använda den för att kontrollera att din funktion ger rätt svar. □

Uppgift 1.3 Den Euklidiska längden av en vektor x ges av uttrycket

$$\|x\|_2 = \left(\sum_{k=1}^{n} x_k^2 \right)^{1/2}.$$

Skriv en funktion `VektorLangd` med x som inparameter och vektorns längd som utparameter. □

Tips I Matlab finns en funktion `norm` som beräknar samma Euklidiska längd. Du kan använda den för att verifiera att din funktion fungerar.

Uppgift 1.4 Ett polynom, av grad $\leq n$, kan skrivas

$$P_n(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n,$$

Skriv en funktion `Polynom` som har en vektor $c = (\, c_0 \quad c_1 \quad \ldots \quad c_n \,)$ och en vektor x, med ett antal x-värden, som inparametrar, och som beräknar polynomets värden för dessa x–värden. Då din funktion funktion är färdig skall du alltså kunna skriva

```
>>x=-1:0.01:1;
>>plot( x , Polynom([-1 0 2],x));
```

så skall polynomet $P_2(x) = -1 + 2x^2$ plottas på intervallet $-1 < x < 1$. □

Uppgift 1.5 Fibonaccitalen är en talföljd som definieras av följande rekursionsformel

$$\begin{cases} F_1 = 1, \ F_2 = 1, \\ F_{n+2} = F_{n+1} + F_n, \quad n = 1, 2, 3, \ldots \end{cases}$$

a) Skriv en funktion `Fibonacci`, med ett heltal N som inparameter, som returnerar en vektor med de första N Fibonaccitalen som utparameter. Exempelvis skall vektorn

$$F = \begin{pmatrix} 1 & 1 & 2 & 3 & 5 \end{pmatrix},$$

ges som utparameter om $N = 5$.

b) Skriv ett program som hittar det minsta N för vilket summan av de första N Fibonacci talen är större än 10^4. Ditt program skall använda funktionen ifrån (a) för att beräkna successivt längre följder av Fibonaccital. □

Uppgift 1.6 Kvadratroten av ett positivt tal a kan beräknas genom att gränsvärdet till talföljden

$$\begin{cases} x_1 = 1, \\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right), \quad n = 1, 2, \ldots \end{cases}$$

bestäms. Tillräckligt noggrannhet anses ha uppnåtts då $|x_{n+1} - x_n| < 10^{-9}$. Skriv en funktion `Kvadratrot` med ett tal a som inparameter och en approximation av $\sqrt{a}$ som utparameter. Använd din funktion för att beräkna $\sqrt{3}$. □

Uppgift 1.7 En matris sägs vara diagonaldominant om

$$\sum_{j=1, j\neq i}^{n} |a_{ij}| \leq |a_{ii}|, \qquad i = 1, 2, \ldots, n,$$

med *sträng* olikhet för åtminstone ett i. Skriv en funktion `DiagonalDominant` med en matris A som inparameter som retunerar *sant* (dvs 1) om A är diagonaldominant, och *falskt* (dvs 0) annars. □

Uppgift 1.8 (Svår) Maximumnormen för en matris A definieras som:

$$\|A\|_\infty = \max_{i=1,\ldots,n} \left(\sum_{j=1}^{n} |a_{i,j}| \right),$$

dvs genom att man summerar beloppet av elementen i varje rad och därefter väljer den största rad summan. Skriv en funktion `MaxNorm` som beräknar maximumnormen för en godtycklig matris. □

Tips Du kan använda Matlabs inbyggda funktion `norm(A,Inf)` för att kontrollera att din funktion fungerar som den skall. Du får dock inte använda Matlabs inbyggda funktion i ditt program.

Uppgift 1.9 (Svår) 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. □

Uppgift 1.10 (Svår) Ett lokalt maximum för en vektor `x` definieras som ett element `x(i)` för vilket villkoret,

$$\texttt{x(i-1)} \leq \texttt{x(i)} \geq \texttt{x(i+1)},$$

är uppfyllt. För det första och sista elementet i vektorn skall vara ett lokalt maximum gäller ett liknande villkor.

Skriv en funktion `loc_max` med en vektor `x` som inparameter, och som returnerar en vektor `ind` innehållande samtliga index `i` sådana att `x(i)` är lokala maxima. Exempelvis skall du kunna skriva

```
>>x=[3 4 5 4 3 2 1 7 3];
>>[ind]=loc_max(x);
```

och `ind` skall ges värdet `ind=[3 8]` eftersom lokala maxima finns på platserna 3 och 8 i `x`. □

2 Enklare funktioner, Ekvationer och Integraler

Enklare funktioner skapas direkt utan att man skriver en funktionsfil. Exempelvis ger

```
>> f = @(x) x.*sin(x)
```

en funktion $f(x) = x\sin(x)$. Notera att vi använder `.*` så att uttrycket fungerar för vektorer x. Vi använder funktionen genom att skriva exempelvis

```
>> x=[1 2 4];
>> f(x)
```

På liknande sätt kan vi skapa funktioner med flera inargument. Exempelvis

```
>> g = @(x,y) cos(y)*sqrt(1+x)
```

Uppgift 2.1 Skapa ett funktionshandtag som motsvarar funktionen $f(x) = 3\sin(x^2)$. Beräkna sedan $f(2)$. □

Uppgift 2.2 Skapa ett funktionshandtag som motsvarar funktionen $f(x) = \sqrt{1+x}\cos(3x)$. Din funktion skall klara att x är en vektor. Använd funktionen för att plotta en graf över $f(x)$ på intervallet $[0, 2]$. □

Uppgift 2.3 Funktionen $f(x) = x - \mathrm{e}^{-x}$ har en rot $x^* \approx 0.6$. Hitta en bra approximation av roten med `fzero`. □

Uppgift 2.4 Beräkna integralen

$$I = \int_{-1}^{1} \mathrm{e}^{-2x^2} dx. \qquad \square$$

Uppgift 2.5 (Svår) Låt α vara en konstant. Integralen

$$I(\alpha) = \int_{-1}^{1} \mathrm{e}^{-\alpha x^2}\sqrt{1+x^2},$$

kan beräknas för varje konkret värde på konstanten α. Hitta det värde α som ger $I(\alpha) = 0.77$. □

TANA17 Matematiska beräkningar med Matlab

Facit till Datorlektion 4.

1 Egna Funktioner

Uppgift 1.1 På filen `funk.m` skriver vi

```
function [f]=funk(x)
   if x <= 0
      f=0;
   elseif x <= pi/2
      f=sin(x);
   else
      f=1;
   end
end
```

Då funktionen inte klara vektorargument måste vi beräkna $f(x)$ med en `for`-loop.

```
>> x = -1:0.05:2;  f=zeros(size(x));
>> for i=1:length(x), f(i)=funk(x(i)); , end
>> plot( x , f )
```

Uppgift 1.2 På filen `ScalarProd.m` skriver vi

```
function [S]=ScalarProd( x , y )
    S=0;
    for i=1:length(x)
      S=S+x(i)*y(i);
    end
end
```

I Matlab terminalen kan vi sedan skriva

```
>> ScalarProd( [-1 4 -2 ] , [3 -2 1] )
 ans =
    -13
```

Uppgift 1.3 På filen `VektorLangd.m` skriver vi

```
function [E]=VektorLangd( x )
  E=0;
  for i=1:length(x)
     E=E+x(i)^2;
  end
  E=sqrt(E);
end
```

Uppgift 1.4 På filen `Polynom.m` skriver vi

```
function [P]=Polynom( c , x );
  P=zeros(size(x));
  for k=1:length(c)
    P=P+c(k)*x.^(k-1);
  end
end
```

Uppgift 1.5 På filen `Fibonacci.m` skriver vi

```
function [F]=Fibonacci(N)
    F=zeros(1,N);
    F(1)=1;
    if N>1,F(2)=1;,end       % Specialfallet N=1 skall bara ett tal
    for i=3:length(F)
        F(i)=F(i-1)+F(i-2);
    end
end
```

Programmet blir sedan

```
N=1;S=1;   % S är summan av Fibonacci följden av längd N
while S<10^4
  N=N+1;
  F=Fibonacci(N);
  S=sum(F);
end
disp( N ),disp( S )
```

vilket ger $N = 19$ och summan $S = 10945$.

Uppgift 1.6 På filen `kvadratrot.m` skriver vi

```
function [x1]=kvadratrot( a )

  x0=0; x1=1; n=1;
  while abs(x0-x1)>10^-9
    x0=x1;            % Tidigare tal i följden
    n=n+1;
    x1=(x1+a/x1)/2;
  end
end
```

I teminalen skriver vi sedan

```
>> kvadratrot( 3 )
 ans =
     1.7321
```

Uppgift 1.7 På filen `DiagonalDominant.m` skriver vi

```
function [D]=DiagonalDominant( A )
  [n,m]=size(A);
  EnStrikt=0;   % Sann om bi hittat en strikt olikhet
  AllaSanna=1;  % Sätt till falsk ifall vi hittar en ogiltig olikhet
  for i=1:n
     % beräkna summan över raden
     S=0;
     for j=1:m, S=S+abs(A(i,j)); , end
     if S>2*abs(A(i,i)), % diagonal elementet ingår i S
       AllaSanna=0;
       break
     end
    if S<2*abs(A(i,i))   % hittat en strikt olikhet
      EnStrikt=1;
    end
  end
  if EnStrikt & AllaSanna  % Alt D = EnStrikt & AllaSanna;
     D=1;
  else
     D=0;
  end
end
```

Uppgift 1.8 På filen `MaxNorm.m` skriver vi

```
function [N]=MaxNorm( A )
   [n,m]=size(A);
   N=0;  % Hittils största radsumman
   for i=1:n
     S=0; % beräkna nästa radsumma
     for j=1:m
       S=S+abs(A(i,j));
     end
     if N<S, N=S; end  % uppdatera om hittat större radsumma
   end
end
```

Uppgift 1.9 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
```

Uppgift 1.10 På filen `loc_max.m` skriver vi

```
function [ind]=loc_max(x)
   n=length(x);
   ind=[]; % Tom vektor initialt
   if x(1) >= x(2) % x(1) är lokalt max
     ind=[ind,1];
   end
   for i=2:n-1
      if ( x(i-1) <= x(i) ) & ( x(i) >= x(i+1) )
         ind=[ind,i];
      end
   end
   if x(n) >= x(n-1)
     ind=[ind,n];
   end
end
```

2 Enklare funktioner, Ekvationer och Integraler

Uppgift 2.1 I terminalen skriver vi

```
>> f = @(x) 3*sin(x.^2)
>> f(2)
   ans =
      -2.2704
```

Uppgift 2.2 I terminalen skriver vi

```
>> f = @(x) sqrt(1+x).*cos(3*x);
>> x=0:0.01:2;
>> plot( x , f(x) )
```

Uppgift 2.3 I terminalen skriver vi

```
>> f = @(x) x-exp(-x);
>> x=fzero( f , 0.6 )
```

vilket ger en rot $x = 0.5671$.

Uppgift 2.4 I terminalen skriver vi

```
>> f = @(x) exp(-2 *x.^2 )
>> I=integral( f , -1 , 1 )
```

vilket ger $I = 1.1963$.

Uppgift 2.5 Enklast är att skriva en vanlig funktion först. På filen `IntAlpha.m` skriver vi

```
function I=IntAlpha( alpha )
    f=@(x) exp(-alpha*x.^2).*sqrt(1+x.^2);
    I=integral( f , -1,1);
end
```

I terminalen kan vi sedan lösa ekvationen med

```
>> g = @(alpha) IntAlpha(alpha)-0.77;
>> fzero( g , 1 )
```

vilket ger $I(\alpha) = 0.77$ för $\alpha = 5.7330$.