TANA17 Matematiska beräkningar med Matlab

Datorlektion 3. Repetitionssatser och Programmering

1 Introduktion

Denna övning syftar till att träna programmering med repetitionssatser och villkorssatser. Undvik därför att använda vektor beräkningar i största möjliga mån. Skriv alltså inte `D=sum(x.*y)` om du vill räkna ut en skalärprodukt utan använd en `for`-loop.

2 Repetionssatsen `for`

I MATLAB finns två konstruktioner för upprepad exekvering av satser: `for` och `while`.

Kommandot `for` används då en grupp satser skall exekveras ett fixt antal gånger. Den generella formen på en `for`–sats är:

```
for <variabel>=<uttryck>
  <satsgrupp>
end
```

Här tilldelas *<variabel>*, den så kallade loopvariabeln, ett startvärde. Varje gång kommandona i *<satsgrupp>* har utförts så ökas värdet på loopvariabeln. Detta upprepas tills ett bestämt slutvärde uppnås.

Uppgift 2.1 Skriv ett program som beräknar summan $S = \sum_{k=1}^{100} k$. □

Uppgift 2.2 Vad blir `x(5)` då följande program exekveras?

```
x=zeros(5,1);
for k=2:1:5
  x(k)=x(k-1)+k
end
```

Pröva om du är osäker! □

Uppgift 2.3 Exponentialfunktionen kan approximeras bra genom att använda serien

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \ldots \approx \sum_{k=0}^{n} \frac{x^k}{k!}.$$

där vi alltså tar med de första $n+1$ termerna i summan. Välj $n = 10$ och skriv en `for` loop som beräknar summan ovan. Testa ditt program genom att beräkna e^2 genom dels standard funktionen `exp` och dels ditt program. □

Tips Då du beräknar termerna i summan behöver du k fakultet. Utnyttja att $k! = (k-1)!k$ och inför en extra variabel där $k!$ lagras och som uppdateras i varje steg i `for`–loopen.

Uppgift 2.4 Skriv ett program som bildar den så kallade Hilbert matrisen, dvs en $n \times n$ matris H, vars element ges av $H(i,j) = 1/(i+j-1)$. Du kan tilldela parametern n ett värde överst i ditt program. □

Tips Det finns en inbyggd standard funktion, `hilb`, i MATLAB som du kan använda för att kontrollera att ditt program fungerar.

Uppgift 2.5 Antag att vi har en vektor x som innehåller n element. Använd en `for`-loop för att hitta det största elementet i vektorn. □

Tips Skapa en test vektor `x=rand(5,1)` och testa så att ditt program fungerar.

Uppgift 2.6 Låt $c = (1\,,\, -2\,,\, 1.5)^T$. Vi vill beräkna polynomet $p(x)$, som ges av

$$p(x) = \sum_{k=0}^{n-1} c_k x^k,$$

för ett antal x-värden sparade i en vektor `x`. Använd en `for`-loop för att beräkna polynomets värden. Använd `length(c)` för att bestämma hur många koefficienter som finns (och därmet polynomets gradtal).

Då du är färdigt skall du använda ditt program för att rita en graf över polynomet $p(x) = 1 - 2x + 1.5x^2$ på intervallet $0 \leq x \leq 2$. □

Uppgift 2.7 Låt `a` och `b` vara två vektorer med n element, och antag att alla element i vektorn `b` är positiva. Skriv ett program som beräknar värdet,

$$M = \max_{1 \leq i \leq n} |\frac{a_i}{b_i}|. \qquad \square$$

Uppgift 2.8 Låt A vara en $n \times m$ matris som innehåller både positiva och negativa element. Beräkna summan av de positiva elementen i matrisen. □

Tips Som exempel kan du bilda en testmatris med `A=rand(5,4)-0.5`. Använd `[n,m]=size(A);` i ditt program för att ta reda på hur stor matrisen är. □.

Uppgift 2.9 (**Svår**) 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 en rad i. Skriv ett program som kontrollerar om en matris A är *diagonal dominant*. Ifåfall skall en utskrift göras till skärmen. □

Uppgift 2.10 (Svår) Multiplikation mellan två matriser A och B definieras som att $C = AB$, där

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Skriv ett program som beräknar matrisen C. Produkten AB är endast definierad om A och B har dimensioner som passar ihop. Ditt program skall innehålla en villkorssats som testar detta. □

3 Repetionssatsen `while`

Kommandot `while` exekverar en satsgrupp så länge som ett logiskt villkor är *sant*. Den generella formen på en `while`–sats är

```
while <logiskt villkor>
   <satsgrupp>
end
```

Det är viktigt att en `while`–sats kan avbrytas.

Uppgift 3.1 Skriv ett program som skriver ut det minsta tal av formen 3^n som är större än 5000. Talet n skall vara ett heltal. □

Uppgift 3.2 En reell rot till ekvationen

$$p(x) = x^3 - 3ax + 1 = 0, \qquad \text{där } a \geq 1,$$

kan bestämmas som gränsvärde av talföljden $\{x_k\}_{k=0}^{\infty}$ definierad av

$$\begin{cases} x_0 = 0 \\ x_{k+1} = \frac{x_k^3+1}{3a}, \qquad k = 0, 1, 2, \ldots \end{cases}$$

Skriv ett program som bestämmer detta gränsvärde. Tilldela först konstanten a ett värde och beräkna sedan $x_1, x_2, \ldots$. Avbryt då villkoret $|x_{k+1} - x_k| < 10^{-6}$. Skriv ut värdet på a, x_{k+1}, $k+1$, och $p(x_{k+1})$. Vilket resultat fås då $a = 3$? □

Uppgift 3.3 Skriv ett program som beräknar en approximation till summan

$$S = \sum_{k=1}^{\infty} \frac{\sin(k)}{k^2}.$$

Avbryt summeringen då nästa term är till beloppet mindre än en given tolerans. Vad blir summans beräknade värde om toleransen $\epsilon = 10^{-8}$ används? □

Uppgift 3.4 En talföld bildas enligt följande regler:

$$\begin{aligned} &x_0 = 1 \text{ och } x_1 = 2 \\ &x_n = 5x_{n-1} - 4x_{n-2} \quad n = 2, 3, \ldots \end{aligned}$$

Skriv ett program som bildar följden $\{x_n\}_{n=0}^{\infty}$ enligt ovanstående regler. Programmet skall beräkna nya termer i talföljden ända tills nästa term x_n är större än 10^4. Programmet skall då göra en utskrift liknande:

```
n=9 ger xn=29983 vilket ar storre an 10000
```

Uppgift 3.5 Fibonaccitalen är en talföljd som definieras av följande rekursionsformel:

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

De första 5 talen i följden är alltså

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

Skriv ett program som hittar det största Fibonacci talet som fortfarande är mindre än 1000. □

Uppgift 3.6 (Svår) Ett kast med en tärning kan simuleras med hjälp av funktionen `randi`. Vi vill ta reda på hur många tärningskast som krävs i genomsnitt för att få en sexa. Gör följande:

- Skriv ett program som simulerar tärningskast. Ränka hur många kast som krävs innan du får en sexa.
- Bygg ut ditt program så att $N = 100$ omgångar simuleras. Beräkna genomsnitttliga antalet kast som krävdes för att få en sexa. □

Tips Skapa först en $N \times 1$ vektor `AntalKast` som innehåller nollor. Efter att du genomfört en omgång tärningskast så sparar du antalet kast som krävdes på en plats i vektorn `AntalKast`. I slutet kan du använda `mean` för att beräkna genomsnittliga antalet kast.

TANA17 Matematiska beräkningar med Matlab

Facit till Datorövning 3.

2 Repetionssatsen `for`

Uppgift 2.1 Följande MATLAB kommandon beräknar summan

```
S=0;                % <- används för att lagra en partialsumma
for k=1:100
  S=S+k;
end;
disp(S)
```

Uppgift 2.2 Programmet skapar en kolumnvektor $x = (\,0\ 2\ 5\ 9\ 14\,)^T$. `x(5)` blir alltså 14.

Uppgift 2.3 Vi sparar en variabel `kfak` för att spara $k!$. Programmet blir

```
x=0.7;     % Eller det x-värde man är intresserad av
kfak=1;S=1;n=10;
for k=1:n
    kfak=kfak*k;            % Blir 1*1 första gången!
    S=S+x^k/kfak;
end
disp(S)
```

Uppgift 2.4 För att skapa Hilbert matrisen H kan följande program användas.

```
N=5;
H=zeros(N,N);
for i=1:N
  for j=1:N
    H(i,j)=1/(i+j-1);
  end;
end;
```

Uppgift 2.5 Vi sparar en variabel `x_min` som innehåller det hittils minsta elementet vi hittat. Programmet blir då

```
x_min=x(1); n = length(x);
for k=2:n
   if x(k)<x_min
      x_min=x(k); % Hittat mindre! Uppdatera x_min.
   end
end
disp(x_min)
```

Uppgift 2.6 Vi skapar vektorn c, en vektor med x-värden och beräknar polynomet med

```
c=[1 -2 1.5]; x=0:0.01:2;

p=c(1)*x.^0;          % ger första termen. elementvis op. ty x är vektor.
for k=2:length(c)
    p=p+c(k)*x.^k;
end
plot(x,p)
```

Uppgift 2.7 Vi skriver

```
M = abs(a(1)/b(1));
for k=2:length(a)
   if abs(a(k)/b(k))>M
      M = abs(a(k)/b(k));
   end
end
disp(M)
```

Uppgift 2.8 Vi använder `size` för att hitta matrisens storlek. Summan beräknas med

```
[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)
```

Uppgift 2.9 Vi behöver två logiska variabler: Den första `AllaMindre` skall bli falsk om någon rad bryter mot olikheten. Den andra `EnStrikt` skall bli sann så snart vi hittar en strikt olikhet på någon rad. Programmet blir

```
[n,m]=size(A);
AllaMindre = 1; % Alla rader vi undersökt hittils uppfyller olikheten.
```

```
EnStrikt = 0;   % Vi har ännu inte hittat någon strikt olikhet
for i=1:n
  % Undersök rad i.
  S=0;
  for j=1:m
     S=S+abs(A(i,j));  % S blir radsumman
  end;
  if S<2*abs(A(i,i))   % Vi har |A(i,i)| i S också.
     EnStrikt = 1;
  end
  if S > 2*abs(A(i,i))
     AllaMindre = 0;  % Hittat en ogiltig olikhet!
     break;           % Vi behöver inte fortsätta. Vet nu att falskt!
  end
end

% Nu lägger vi ihop resultatet i en ny logisk variabel.
DiagDom = AllaMindre & EnStrikt ;
if DiagDom, disp('Matrisen är diagonal dominant'),end
```

Uppgift 2.10 Givet två matriser A och B skriver vi

```
[n1,m1]=size(A);[n2,m2]=size(B);

if m1 ~= n2   % I detta fallet fungerar det inte
  disp('dimensioner passar ej');
else
  C=zeros(n1,m2);  % Skapa C med rätt storlek
  for i=1:n1
     for j=1:m2
        % beräkna C(i,j) med formeln.
        for k=1:m1
            C(i,j)=C(i,j)+A(i,k)*B(k,j);
        end
     end
  end
end
```

3 Repetionssatsen `while`

Uppgift 3.1

```
n=1;
while (3^n <= 5000 )
   n=n+1;
end
```

```
disp('villkoret uppfyllt for n=')
n
```

Uppgift 3.2 Följande program bestämmer gränsvärdet:

```
a=3; k=-1; Xk=1;
Xkp1=0         % Garanterar att avbrottsvillkoret är sant första gången.
while ( abs(Xk-Xkp1) >= 1e-6 )
  Xk=Xkp1;
  k=k+1;
  Xkp1=(Xk^3+1)/3/a;
end;
disp('Polynomets varde:')
 p=Xkp1^3-3*a*Xkp1+1      % Övriga uttskrifter på samma sätt.
```

Resultatet blir att $x_3 = 0.111264\ldots$ och $p(x_3) = 2.332 \cdot 10^{-8}$.

Uppgift 3.3 Summan beräknas med programmet:

```
S=0;k=0;term=1;
while ( abs(term)>1e-8 )
   k=k+1;
   term=sin(k)/k^2;
   S=S+term;
end
disp('Summan blir:'), S
```

Summans värde blir $S = 1.01396\ldots$.

Uppgift 3.4 Programmet blir

```
x(1)=1;x(2)=2;k=2;
while x(k)<10^4
  k=k+1;
  x(k)=5*x(k-1)-4*x(k-2);
end;
disp(['k=',num2str(k),' ger xk=',num2str(x(k)),' vilket är större än 1000'])
```

Ger talet 21846 för $k = 9$.

Uppgift 3.5 Programmet blir

```
F(1)=1;F(2)=1;k=2;
while F(k)<1000
  k=k+1;
```

```
    F(k)=F(k-1)+F(k-2);
  end;
  disp(F(k-1))
```

Ger talet 987.

Uppgift 3.6 Det färdiga programmet blir

```
  N=100;
  AntalKast=zeros(N,1);
  for i=1:N
     % Simulera en omgång
     k=1; D6 = randi([1,6],1);
     while D6 ~= 6
       % Ej fått sexa. Kasta en gång till
       k=k+1; D6 = randi([1,6],1);
     end;
     AntalKast(i)=k;
 end;
 M =  mean(AntalKast)
```

Ger medelvärdet $M = 5.52$ (eller däromkring).