TANA17 Matematiska beräkningar med Matlab

Datorlektion 2. Linjär Algebra, Villkor och Logik

1 Linjär Algebra

Programsystemet Matlab utvecklades ursprungligen för att underlätta beräkningar från linjär algebra.

Addition, subtraktion och multiplikation av matriser utförs med de vanliga symbolerna för skalära räkneoperationer. Dessutom finns funktioner som utför viktiga operationer. Exempelvis beräknas inversen till en matris med hjälp av funktionen `inv`:

```
>> A= [ 2 3 -1 ; 3 1 0 ; 0 2 -1];
>> inv(A)
 ans =
  -1.0000    1.0000    1.0000
   3.0000   -2.0000   -3.0000
   6.0000   -4.0000   -7.0000
```

Det finns även funktioner för att beräkna determinanter, skalärprodukter, egenvärden, och mycket mer. Läs hjälptexten för funktionerna `det`, `dot` och `eig`. Transponat av en matris eller vektor fås genom att skriva

```
>> B = A'
```

där alltså apostrofen `'` betyder transponat.

Division mellan matriser är normalt inte definierat i Linjär algebra men i Matlab betyder

```
>> C = A \ B;
```

att $C = A^{-1}B$. Skriver vi istället `C=A/B` så beräknas $C = AB^{-1}$. Det är alltså viktigt åt vilket håll divisions–tecknet lutar.

Uppgift 1.1 Skapa två vektorer $x = (1\,,\,-2\,,\,3)^T$ och $y = (2\,,\,0\,,\,-1)^T$. Beräkna skalärprodukten genom att använda elementvis multiplikation och en summa, dvs `S=sum(x.*y)`. Jämför resultatet med funktionen `dot`. □

Uppgift 1.2 Skriv in matriserna

$$A = \begin{pmatrix} 2 & -1 & 3 \\ 2 & 2 & -3 \\ 4 & 3 & 9 \end{pmatrix}, \text{ och } B = \begin{pmatrix} -1 & 2 & 3 \\ 2 & -3 & 2 \\ 1 & 3 & 2 \end{pmatrix}.$$

Undersök om AB är samma som BA? Kontrollera även att $(AB)^{-1}$ är samma sak som $B^{-1}A^{-1}$. □

Uppgift 1.3 Skalärprodukten mellan två kolumnvektorer x och y kan beräknas som $z = x^T y$. Skapa två vektorer $x = (2\,,\,-1\,,\,1)^T$ och $y = (1\,,\,2\,,\,0)^T$. Vektorerna kallas *ortogonala* om skalärprodukten mellan dem är noll. Undersök om vektorerna är ortogonala. □

Uppgift 1.4 Determinanten av en matris kan beräknas med hjälp av radoperationer och radbyten. I Matlab finns funktionen `det` som beräknar determinanten av en matris. Här skall vi utnyttja Matlab för att verifiera determinanträkneregler. Skriv först in matriserna

$$A = \begin{pmatrix} 4 & -2 & 0 \\ 2 & 0 & -1 \\ 2 & 1 & -1 \end{pmatrix}, \text{ och } B = \begin{pmatrix} -1 & 1 & 2 \\ -2 & 3 & 1 \\ 1 & 0 & 2 \end{pmatrix}.$$

Beräkna först determinanterna för matriserna A och B med hjälp av funktionen `det`. Gör sedan följande

a) Verifiera att $\det(AB) = \det(A)\det(B)$.

b) Byt ordningen på två rader i matrisen A genom att skriva `A=A([1 3 2],:)`. Verifiera att ett radbyte gör att determinanten växlar tecken.

c) Genomför två radoperationer på B-matrisen för att skapa nollor på plats b_{11} och b_{21}. För att skapa nollan på plats b_{11} adderar vi sista raden till den första. Skriv in Matlabkommandot `B(1,:)=B(1,:)+B(3,:)`. Eliminera på liknande sätt elementet b_{21}. Verifiera sedan att radoperationer inte ändrar determinantens värde.

d) Inversen av matrisen A kan beräknas med funktionen `inv`. Utnyttja detta för att verifiera $\det(A^{-1}) = 1/\det(A)$. □

Uppgift 1.5 Skriv in följande matris och vektor

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 5 & 7 \end{pmatrix}, \text{ och } b = \begin{pmatrix} 6 \\ 9 \\ 16 \end{pmatrix}.$$

Vi vill lösa det linjära ekvationssystemet $Ax = b$. Skriv in de kommandon som krävs för att beräkna x. □

Tips Det går att använda `inv` men normalt är det bättre att använda \-divisionen.

Uppgift 1.6 Skriv in följande matris och vektor

$$A = \begin{pmatrix} 3 & -2 & 0 \\ -1 & 2 & 2 \\ 1 & 2 & -1 \end{pmatrix}, \text{ och } b = \begin{pmatrix} 6 \\ 2 \\ -1 \end{pmatrix}.$$

Lös det linjära ekvationssystemet $Ax = b$. □

Uppgift 1.7 (Svår) En kvadrat ritas upp med hjälp av kommandona

```
>> P=[ 0 0 1 1 0 ; 0 1 1 0 0 ];
>> plot( P(1,:),P(2,:) ,'b-');
```

där varje kolumn `P(:,i)` representerar en av kvadratens hörn. Låter vi A vara en 2×2-matris så kommer uttrycket `V=A*P` multiplicera varje kolumn med A, alltså `V(:,i)=A*P(:,i)`. De gamla hörnpunkterna avbildas alltså på nya med hjälp av matrisen A.

En *skuvning* beskrivs av $(x_1, y_1)^T$ avbildas på $(x_1+\alpha y_1, y_1)^T$, där α är en konstant som beskriver hur kraftig skuvningen är. Låt $\alpha = 0.2$. Bilda sedan en matris A som beskriver skuvningsavbildningen enligt ovan. Beräkna sedan de nya hörnpunkterna V och plotta kvadratens utseende, med röda streckada linjer, efter skuvningen i samma grafikfönster som ursprungskvadraten. □

2 Logiska uttryck

Logiska variabler och tester är en nödvändig del av de flesta programmeringsspråk. I Matlab finns sex olika relationsoperatorer för att göra jämförelser mellan matriser och skalärer.

De olika relationsoperationerna är `<`, `<=`, `>`, `>=`, `==` och `~=`.

Relationsoperatorerna returnerar ett resultat som skall tolkas som *sant* eller *falskt*. I MATLAB gäller att värdet 0 tolkas som *falskt* och att värdet 1 tolkas som *sant*.

Relationsoperatorerna kan användas för att jämföra såväl skalärer som matriser. För matriser jämförs motsvarande element och en matris innehållande enbart ettor och nollor returneras.

Förutom relationsoperatorerna finns ett antal logiska operatorer. I MATLAB finns `&` (*och*), `|` (*eller*) och `~` (*negation*). Dessa kan kombineras med relationsoperatorerna ovan för att skapa mer komplicerade logiska uttryck.

De logiska operatorerna har lägst prioritet i MATLAB. Relationsoperationer och aritmetiska operationer beräknas innan de logiska operationerna i ett uttryck. Ofta är det lämpligt att införa parenteser i logiska uttryck för att göra utttrycken tydligare.

Uppgift 2.1 Låt `a=3` och `b=6` Vad blir resultatet av testerna `a<=b`, `a==b` och `a>b`? □

Uppgift 2.2 Låt a, b, och c vara reella tal. Ange ett uttryck som ger resultatet *sant* då alla talen är *lika*, och falskt annars. Pröva om ditt uttryck är korrekt genom att testa några olika värden på parametrarna a, b, och c.

Uppgift 2.3 Låt a, b, och c vara reella tal. Ange ett uttryck som ger resultatet *sant* då alla talen är *olika*, och falskt annars. Pröva om ditt uttryck är korrekt genom att testa några olika värden på parametrarna a, b, och c.

Uppgift 2.4 Låt x vara ett reelt tal. Skriv ett logiskt uttryck som ger värdet sant om $0 < x \leq 10$.

Uppgift 2.5 Skriv in följande data:

$$A = \begin{pmatrix} 2 & -1 & 3 \\ -2 & 1 & 1 \\ 1 & 0 & 3 \end{pmatrix}, \quad \text{och,} \quad B = \begin{pmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 2 & -1 & 3 \end{pmatrix}.$$

Vad blir resultatet av testerna `A==B`, `A~=B`, och `A<=B`?

Vad blir resultatet av uttrycket `A>B | A<B`? Kan samma uttryck skrivas på ett enklare sätt? □

Som vi sett blir resultatet av jämförelser mellan matriser och vektorer en ny vektor eller matris med nollor eller ettor beroende på om jämförelsen är sann eller falsk för respektive element. Exempelvis så visar kommandot

```
>> [ 1 , 0 , -1] > [ 0 , 2 , 3]
 ans =
     1     0     0
```

att jämförelsen är sann endast för det första elementet. Detta kan utnyttjas för beräkningar.

Uppgift 2.6 Låt x vara en vektor med $n = 10$ element. Använd en jämförelse operation och kommandot `sum` för att hitta antalet positiva element i vektorn x. □

Tips För att testa att ditt uttryck fungerar kan du skapa en slump vektor med `x=rand(10,1)-0.5`.

Uppgift 2.7 Antag att y är en vektor som innehåller n stycken tal. Vi har tidigare använt `min` för att hitta det minsta elementet i vektorn. Hitta ett sätt att räkna hur många gånger det minsta talet förekommer i vektorn y. I exemplet

```
>> y = [ 1 , 3 , -1 , 4 , -1 , 5 , 3 , -1]
```

så skall vi få svaret 3 då talet -1 förekommer på tre platser. □

Uppgift 2.8 Antag att vi har en vektor x och vill kontrollera om den är sorterad i stigande ordning. Hur skall `sum` användas tillsammans med en jämförelseoperation för att testa detta? □

Tips Med kolon-notation så ger `x(2:end)` en vektor som innehåller allt utom första elementet i vektorn x.

3 Villkorssatser

I mera avancerade program finns ofta satser som enbart skall utföras om något villkor är uppfyllt, eller några satser som skall hoppas över för vissa fall. I Matlab finns två konstruktioner som kan användas för att detta syfte, `if`–satsen och `switch`–satsen.

Den enklaste formen av en `if`–sats är

```
if <logiskt villkor>
  <satsgrupp>
end
```

Med *<satsgrupp>* menas en eller flera satser, dvs Matlabkommandon. Det finns även mera komplicerade varianter.

```
if <logiskt villkor 1>
  <satsgrupp 1>
elseif <logiskt villkor 2>
  <satsgrupp 2>
else
  <satsgrupp 3>
end
```

I fall *<logiskt villkor 1>* är sant kommer MATLAB kommandona i *<satsgrupp 1>* att utföras. Om *<logiskt villkor 1>* är falskt och *<logiskt villkor 2>* är sant kommer *<satsgrupp 2>* att utföras. I övriga fall kommer kommandona i *<satsgrupp 3>* att utföras.

Uppgift 3.1 Antag att vi har beräknat en konstant C. Vi vet att konstanten måste vara positiv. För att vara säker på att beräkningen blev korrekt vill vi testa detta genom att skriva en villkorssats som testar om C verkligen är positiv. Om så inte är fallet skall en varning skrivas ut. □

`Tips` Använd `disp('Varning: C ej positiv')`. Utskrift av text återkommer senare i kursen.

Uppgift 3.2 Funktionen $f(x)$ ges av uttrycket

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

Skriv en villkorssats som beräknar $y = f(x)$ för ett givet x–värde. Det är lämpligt att skriva din villkorssats i en M–fil. I MATLAB finns en fördefinierad funktion `pi` som returnerar värdet på π. Testa att ditt program fungerar! □

Uppgift 3.3 Antag att $a = 5$ och att variabeln `flagga` har värdet falskt (dvs `flagga=0` i MATLAB). Vad händer i följande fall?

```
if flagga                if flagga
   if  a<10                   if a<10
      a=a+1                      a=a+1
   else                       end
      a=a-1                 else
   end                          a=a-1
end                         end
```

Pröva om du är osäker! □

Uppgift 3.4 Vad skulle ha hänt i uppgiften ovan om $a=5$, men variabeln `flagga` istället hade haft värdet sant? □

4 Simulering och Sannolikhetslära

I Matlab finns flera funktioner för att skapa slumptal. Ett exempel är `rand` som skapar matriser där elementen är slumptal. Skriver vi

```
>> P = rand(10,3);
```

så skapas en 10×3-matris där elementen är likformigt fördelade på intervallet $[0, 1]$. Det finns även en funktion `randi` som skapar slumpmässiga heltalsvärden.

Uppgift 4.1 Antag att vi kastar 5st tärningar. Hur många sexor kan vi förvänta oss att få? Svara på frågan genom att skapa en 5×1000-matris P där varje element är ett slumpmässigt heltal mellan 1 och 6. Varje kolumn `P(:,i)` representerar då ett kast med fem tärningar. Använd `sum` och en jämförelse för att skapa en 1×1000 vektor där varje element är summan av antalet sexor för respektive omgång med fem tärningskast. Använd sedan `mean` för att beräkna det genomsnittliga antalet sexor man får vid kast med fem tärningar. □

Uppgift 4.2 (Svår) Antag att en hink innehåller totalt 8 bollar, varav fem är svarta och tre är vita. Vi drar två bollar från hinken och undersöker hur stor sannolikhet det är att vi drar två vita bollar. För att simulera en omgång då vi drar två bollar skall vi göra följande:

- Då första bollen dras är det 5/8 chans att få en svart och 3/8 chans att få en vit. Använd `rand` eller `randi` för att simulera första bollen vi drar.
- Då vi skall plocka den andra bollen från hinken så beror sannolikheterna på vilken boll vi redan plockat upp. Använd en `if`-sats för skilja på de två fallen (dvs första bollen blev antingen vit eller svart) och simulera vad som händer när den andra bollen plockas upp. □

Tips Använd utskrifter av typ `disp('Två vita bollar')` för att visa vad som händer. Enklast är att använda nästlade `if`-satser.

Uppgift 4.3 (svår) En integral,

$$I = \int_a^b f(x)dx,$$

kan beräknas med följande metod: Hitta konstanter sådana att $c_1 \leq f(x) \leq c_2$. Bilda sedan ett stort antal slumpmässigt punkter $(x_i, y_i)^T$ i rektangeln $[a, b] \times [c_1, c_2]$. För varje sådan punkt undersöker vi om den ligger över eller under funktionskurvan, dvs om $y_i \leq f(x_i)$ eller inte. Andelen punkter som ligger under kurvan ger en god approximation till integralens värde.

Vi skall använda denna metod för att beräkna en approximation till

$$I = \int_0^1 \sqrt{1+x}\cos(x).$$

Gör följande

- Plotta funktionen $f(x) = \sqrt{1+x}\cos(x)$ på intervallet $[0, 1]$.
- Välj konstanter c_1 och c_2 så bra som möjligt.
- Sätt $n = 100$ och skapa en $2 \times n$-matris P med likformigt fördelade slumptal i intervallet $[0, 1]$. Använd en vektor-operation för att transformera raden `P(2,:)` till att istället innehålla slumptal i intervallet $[c_1, c_2]$.
- Vi har nu två vektorer `x=P(1,:)` och `y=P(2,:)`. Använd en vektor-jämförelse av typen $y \leq f(x)$ och `sum` för att undersöka hur stor andel av punkterna som ligger under kurvan.
- Om andelen α av punkterna ligger under kurvan kan integralen approximeras

$$I \approx (c_2 - c_1)\alpha + c_1.$$

Genomför ovanstående beräkningar och hitta en approximation av I. □

TANA17 Matematiska beräkningar med Matlab

Facit till Datorlektion 2.

1 Linjär Algebra

Uppgift 1.1 Skapa vektorerna med

```
>> x=[ 1 -2 3]'; y=[ 2 0 -1]';
```

Både `sum(x.*y)` och `dot(x,y)` ger svaret `-1`.

Uppgift 1.2 Skriv in matriserna med

```
>> A = [2 -1 3 ; 2 2 -3 ; 4 3 9]; B = [ -1 2 3 ; 2 -3 2 ; 1 3 2];
```

Beräkningarna `A*B` och `B*A` ger olika resultat medans `inv(A*B)` och `inv(B)*inv(A)` ger samma resultat.

Uppgift 1.3 Skapa vektorerna med

```
>> x=[ 2 -1 1]'; y=[ 1 2 0]';
```

Skalärprodukten beräknas med `(x'*y` vilket ger 0 så vektorerna är ortogonala.

Uppgift 1.4 Skriv in matriserna med

```
>> A = [4 -2 0 ; 2 0 -1 ; 2 1 -1]; B = [ -1 1 2 ; -2 3 1 ; 1 0 2];
```

Beräkningarna `det(A*B)` och `det(A)*det(B)` ger bägge -28.

Vi byter sedan ordningen på raderna i matrisen A med

```
>> A = A([1 3 2],:)

  A =
   4    -2     0
   2     1    -1
   2     0    -1
>>det(A)
ans =
    -4
```

vilket är samma som innan radbytet men med ett minustecken. För att kontrollera en radoperation utför vi kommandona

```
>> B(1,:)=B(1,:)+B(3,:);
>> B(2,:)=B(2,:)+2*B(3,:)
  B =
     0     1     4
     0     3     5
     1     0     2
>>det(B)
ans =
    -7
```

vilket även det är samma som tidigare. Slutligen beräknar vi `det(inv(A))=-0.25` vilket är `1/det(A)`.

Uppgift 1.5 Följande kommandon löser uppgiften:

```
>> A=[1 2 3 ; 2 3 4 ; 4 5 7];b=[ 6 9 16 ]';
>> x = A\b
```

Vi får $x = (1\,,\, 1\,,\, 1)^T$.

Uppgift 1.6 Ekvationssystemet löses med

```
>> A=[3 -2 0 ; -1 2 2 ; 1 2 -1];b=[ 6 2 -1 ]';
>> x = A\b
```

Vi får $x = (1.8\,,\, -0.3\,,\, 2.2)^T$.

Uppgift 1.7 Skuvningsmatrisen blir $A = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$.

Följande Matlab program löser uppgiften

```
P=[ 0 0 1 1 0 ; 0 1 1 0 0 ];
plot( P(1,:),P(2,:) , 'b-');

alpha=0.2;
A=[1 alpha ; 0 1];
V=A*P;      % Applicera skuvningen på punkterna.
hold on
plot( V(1,:),V(2,:),'r--');
axis([-0.1 1.3 -0.1 1.1]); % Ändra axlarna så att hela figuren
hold off                   % syns.
```

2 Logiska uttryck

Uppgift 2.1 Uttrycket `a<=b` ger resultatet 1, dvs *sant*, `a==b` ger 0, dvs *falskt*. Uttrycket `a>b` ger *falskt*.

Uppgift 2.2 Exempelvis `( a==b ) & ( a==c ) & ( b==c )`.

Uppgift 2.3 Exempelvis `( a~=b ) & ( a~=c ) & ( b~=c )`.

Uppgift 2.4 Uttrycket blir `( 0<x )&( x<=10 )`.

Uppgift 2.5 De logiska uttrycken ger matriser som svar. De är:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \quad \text{och} \quad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Uttrycket kan skrivas som `A~=B`. Resultatet blir

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Uppgift 2.6 Använd

```
>> AntalPositiva = sum( x > 0 );
```

Uppgift 2.7 Raderna

```
>> y=[1 3 -1 4 -1 5 3 -1]
>> n = sum( y == min(y) )
```

ger svaret $n = 3$.

Uppgift 2.8 Skriv

```
>> sum( x(2:end)-x(1:end-1) < 0 )
```

då vektorn `x(2:end)-x(1:end-1)` är positiv om $x_k - x_{k-1} > 0$ för alla k, dvs om vektorn är sorterad i stigande ordning. Testa med

```
>> x = 0:10
```

så fås svaret 0 vilket beryder att inget element ligger i fel ordning.

3 Villkorssatser

Uppgift 3.1 Använd

```
if C<0
   disp('Varning: C ej positiv')
end
```

Uppgift 3.2 Exempelvis kan filen `funktion.m` innehålla raderna:

```
if ( x < 0 )
  y=x^2;
elseif ( x < pi/2 )
  y=sin(x);
else
  y=1;
end
disp(y)
```

Uppgift 3.3 I första fallet fås `a=5`. I andra fallet fås `a=4`.

Uppgift 3.4 Nu fås istället `a=6` i bägge fallen.

4 Simulering och Sannolikhetslära

Uppgift 4.1 Följande steg löser uppgiften.

```
N=1000;
P=randi( [1,6],5,N);    % 5xN matris med heltal 1,..,6
V=sum( P == 6 , 1 );    % summera över första dimensionen. Dvs raderna.
mean(V)
```

Genomsnittet blir ungefär 0.83 sexor.

Uppgift 4.2 Programmet

```
I = randi([1 8]);
if I <= 5
    % Första bollen blev svart. 7 bollar kvar. 4 svarta.
    I =  randi([1 7]);
    if I <= 4
      disp('Två svarta bollar.')
    else
      disp('En vit och en svart boll.');
    end
else
    % Första bollen blev vit. 7 bollar kvar. 5 svarta
    I =  randi([1 7]);
    if I <= 5
      disp('En vit och en svart boll.')
    else
      disp('Två vita bollar');
    end
end
```

Uppgift 4.3 Plotta först funktionen

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

Vi ser att det går att välja $c_1 = 0.7$ och $c_2 = 1.1$. Fortsätt sedan med kommandon

```
c1=0.7;c2=1.1;
n=100; P=rand(2,n);
P(2,:)=(c2-c1)*P(2,:)+c1;
x = P(1,:); y=P(2,:);
test =  y <= sqrt(1+x).*cos(x);   % test blir en vektor med
                                  % noll eller ett.
alpha = sum(test)/n
I = (c2-c1)*alpha + c1
```

Integralens värde är $I \approx 1.0098$ och approximationen bör bli ungefär samma. Fler punkter, dvs större n, ger bättre resultat.