Matriser och vektorer i MATLAB

1 Inledning

I denna laboration ser vi på hantering och uppbyggnad av matriser samt operationer på matriser.

En matris av storleken $m \times n$ är ett rektangulärt talschema med m rader och n kolonner,

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Ett matriselement a_{ij} (rad nr i, kolonn nr j) skrivs `A(i,j)` i MATLAB.

Med `[m,n]=size(A)` får vi matrisens storlek, medan `m=size(A,1)` ger endast antal rader och `n=size(A,2)` ger endast antal kolonner.

En matris av storleken $m \times 1$ kallas kolonnvektor och en matris av storleken $1 \times n$ kallas radvektor,

$$\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_n \end{bmatrix}$$

Element nr i ges i MATLAB av `b(i)` och antalet element ges av `m=length(b)`. Motsvarande gäller för radvektorn $\mathbf{c}$.

Som exempel tar vi

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 0 & 2 & 4 & 6 & 8 \end{bmatrix}$$

Vi beskriver dessa i MATLAB enligt

```
>> A=[1 4 7 10; 2 5 8 11; 3 6 9 12]
```

och som svar får vi i Command Window utskriften

```
A =
     1     4     7    10
     2     5     8    11
     3     6     9    12
```

Man använder hakparanteser (`[ ]`) för att bygga upp matriserna. Semikolon (`;`) innanför hakparanteserna betyder radbyte.

Så här beskriver vi kolonnvektorn

```
>> b=[1; 3; 5]
b =
     1
     3
     5
```

och så här radvektorn

```
>> c=[0 2 4 6 8]
c =
     0     2     4     6     8
```

Ett matriselement a_{ij}, dvs. elementet på rad nr i, kolonn nr j, skrivs i MATLAB med `A(i,j)` och ett vektorelement b_i skrivs med `b(i)`.

Indexeringen i matriser och vektorer i MATLAB börjar alltid på `1`, vi kan inte påverka detta. Sista index för en rad eller kolonn ges av `end`. T.ex. `b(1)` är första elementet i vektorn `b` och `b(end)` är sista elementet.

2 Hantering av matriser

En matris kan betraktas som en samling av kolonner:

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & \ldots & \vdots & \ldots & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_j & \cdots & \mathbf{a}_n \end{bmatrix}$$

med kolonnerna

$$\mathbf{a}_1 = \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix}, \quad \mathbf{a}_j = \begin{bmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{bmatrix}, \quad \mathbf{a}_n = \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix}$$

På motsvarande sätt kan man även betrakta den som en samling av rader.

I MATLAB plockar man ut kolonn nr j med `A(:,j)`. Här är j kolonnindex medan radindex $i = 1, \ldots, m$ representeras av tecknet kolon (`:`). Man kan även skriva `A(1:m,j)`. På motsvarande sätt ges rad nr i av `A(i,:)` eller `A(i,1:n)`.

Vi kan ta ut ett block ur en matris med `A(iv,jv)` där `iv` är en vektor med radindex och `jv` är en vektor med kolonnindex. Resultatet blir en matris med `length(iv)` rader och `length(jv)` kolonner.

```
>> A=[1 3 5; 7 9 11; 2 4 6]
A =
     1     3     5
     7     9    11
     2     4     6
```

```
>> B=A([2 3],[1 3])
B =
     7    11
     2     6
```

Transponatet $\mathbf{A}^\mathsf{T}$ av en matris $\mathbf{A}$ ges av apostrof (`'`).

```
>> A=[1 3 5; 7 9 11]
A =
     1     3     5
     7     9    11
```

```
>> B=A'
B =
     1     7
     3     9
     5    11
```

Uppgift 1. Skriv in följande matriser i MATLAB.

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 4 & 5 & 6 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 0 & 2 & 4 \end{bmatrix}$$

(a). Sätt in kolonnvektorn $\mathbf{c}$ som 3:e kolonn i $\mathbf{A}$ och sätt in radvektorn $\mathbf{d}$ som 2:a rad i $\mathbf{B}$.

(b). Låt 1:a och 3:e raden i $\mathbf{A}$ byta plats och låt därefter den 2:a och 4:e kolonnen byta plats.

3 Bygga upp matriser

Med funktionerna `zeros` och `ones` kan man i MATLAB bilda matriser med nollor och ettor. Exempelvis `zeros(m,n)` ger en matris av storleken $m \times n$ fylld med nollor. Med `zeros(size(A))` får vi en matris fylld med nollor av samma storlek som `A`. Motsvarande gäller för `ones`.

Enhetsmatriser bildas med funktionen `eye`, med `eye(n)` får vi enhetsmatrisen av storleken $n \times n$. Man kan också använda `eye` för att bilda rektangulära matriser med ettor på huvuddiagonalen och nollor för övrigt. Med `eye(m,n)` får vi en sådan matris av storleken $m \times n$ och med `eye(size(A))` får vi en av samma storlek som `A`.

Med `diag` bildas diagonalmatriser. En vektor kan läggas in på en viss diagonal enligt

```
>> d=[2 3 7];
>> A=diag(d,0) % eller A=diag(d)
A =
     2     0     0
     0     3     0
     0     0     7
```

```
>> d=[2 3 7];
>> A=diag(d,1)
A =
     0     2     0     0
     0     0     3     0
     0     0     0     7
     0     0     0     0
```

Huvuddiagonalen markeras med `0`, diagonalen ovanför till höger med `1`, diagonalen nedanför till vänster med `-1`, osv. Matrisen blir så stor att vektorns alla element får plats längs angiven diagonal.

Man kan bygga upp matriser blockvis av andra matriser med hakparanterer (`[ ]`). Vi har gjort detta i samband med att vi bildade den utökade matrisen $\mathbf{E} = [\mathbf{A}\ \mathbf{b}]$. Då satte vi i MATLAB ihop $n \times n$-matrisen `A` med $n \times 1$-matrisen (kolonnvektor) `b` till $n \times (n+1)$-matrisen `E` enligt `E=[A b]`.

Vi kan sätta ihop två matriser `A` och `B` horisontellt med `[A B]` om antal rader i de två matriserna är lika. Två matriser `A` och `B` kan sättas ihop vertikalt med `[A; B]` om antal kolonner i de två matriserna är lika.

Uppgift 2. Radera matrisen $\mathbf{B}$ (`clear B`) och skriv in den igen genom att först bilda kolonnerna

$$\mathbf{b}_1 = \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix}, \quad \mathbf{b}_3 = \begin{bmatrix} 6 \\ 1 \\ 1 \end{bmatrix}$$

och sedan sätta in dem i matrisen $\mathbf{B} = [\mathbf{b}_1\ \mathbf{b}_2\ \mathbf{b}_3]$.

Uppgift 3. Bilda matrisen

$$\mathbf{A} = \begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix}$$

Bilda delmatriserna $\mathbf{A}_{ij}$, av storlek 2×2, i partitioneringen

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

genom att ta ut delar av matrisen $\mathbf{A}$. (Se förra avsnittet hur man gör i MATLAB.)

Hur kan man bilda $\mathbf{A}$ i MATLAB, då delmatriserna $\mathbf{A}_{ij}$ är givna? Försök återskapa $\mathbf{A}$ med delmatriserna $\mathbf{A}_{11}, \mathbf{A}_{12}, \mathbf{A}_{21}$ och $\mathbf{A}_{22}$

4 Matris- och vektorfunktioner

Vi ser nu på några användbara inbyggda funktioner som tar matriser eller vektorer som argument. För exemplen använder vi följande matris och vektorer.

$$\mathbf{A} = \begin{bmatrix} 11 & 4 & 3 & 7 \\ 2 & 6 & 8 & 5 \\ 9 & 12 & 1 & 10 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 1 \\ 5 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 4 & 2 & 8 & 0 & 6 \end{bmatrix}$$

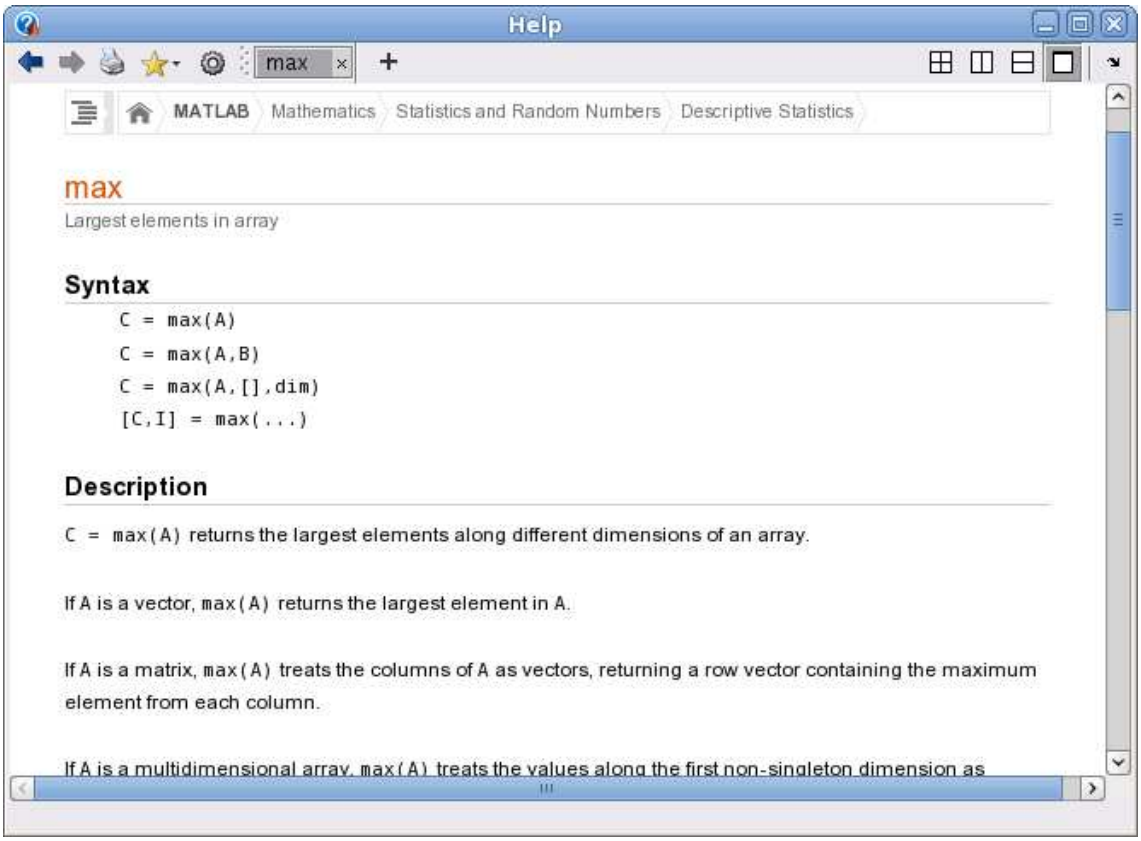
Antalet rader och kolonner i $\mathbf{A}$ får vi med

```
>> [m,n]=size(A)
m =
     3
n =
     4
```

och antal element i vektorn $\mathbf{c}$ ges av

```
>> l=length(c)
l =
    5
```

Största eller minsta elementet i en vektor eller en matris får man med funktionerna `max` och `min`. Här är hjälptexten till `max`.



Vi ser hur vi får största elementet i en vektor och hur vi får de största elementen i varje kolonn för en matris.

```
>> v=max(c)
v =
    8

>> v=max(A)
v =
    11    12     8    10
```

Vi ser också att vi med `[v,i]=max(c)` kan få reda på var det maximala värdet finns någonstans.

Uppgift 4. Skriv in matrisen **A** samt vektorerna **b** och **c** vi använt som exempel. Pröva `size` på vektorerna **b** och **c**. Hur ser man att den ena är en kolonnvektor och att den andre är en radvektor? Bestäm största och minsta elementet i matrisen **A** med hjälp av funktionerna `max` och `min`. Vad har dessa element för rad- respektive kolonnindex?

Summan och produkten av elementen i vektorn fås med `sum` och `prod`. För en matris blir det summan eller produkten av varje kolonn.

```
>> s=sum(b)
s =
     9

>> s=sum(A)
s =
    22    22    12    22
```

I förra laborationen beräknade vi en summa $s = 3 + 4 + 5 + \cdots + 52$ med en `for`-sats, vi skulle även kunna beräkna den med `sum` enligt

```
>> s=sum(3:52)
s =
        1375
```

Detta kallas att vektorisera beräkningen.

Vi kan också dela upp i två satser

```
>> t=3:52;      % Bildar en vektor t med 3, 4, ..., 52
>> s=sum(t)     % Summerar alla element i vektorn t
s =
        1375
```

Först bildar vi vektorn `t` med elementen $3, 4, \cdots, 52$ och sedan summerar vi elementen i vektorn.

Uppgift 5. Beräkna summan $s = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$ med `sum` och komponentvis kvadrering.

Vill vi sortera en vektor i stigande ordning gör vi det med funktionen `sort`. För en matris blir det varje kolonn som sorteras. För att sortera i avtagande ordning se hjälptexten för `sort`.

5 Operationer på matriser

Matris-vektorprodukten $\mathbf{y} = \mathbf{Ax}$ av en $m \times n$-matris och en n-kolonnvektor är en m-kolonnvektor som ges av

$$\underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}}_{\mathbf{Ax}}$$

eller elementvis

$$y_i = \sum_{j=1}^{n} a_{ij}x_j = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n$$

Matris-vektorprodukten $\mathbf{y} = \mathbf{Ax}$ kan beräknas i MATLAB med den inbyggda matrismultiplikationen (`*`) enligt `y=A*x` eller med lite egen programmering (som bygger upp `y` elementvis)

```
>> y=zeros(m,1);
>> for i=1:m
       s=0;
       for j=1:n
           s=s+A(i,j)*x(j);
       end
       y(i)=s;
   end
```

Här måste `A` och `x` redan skrivits in i MATLAB. Även `m` och `n`, som ger matrisens typ, måste ha värden eller ges värden med `[m,n]=size(A)`.

Ett alternativt sätt att introducera matris-vektorprodukt är att definiera $\mathbf{Ax}$ som en linjärkombination av kolonnerna i $\mathbf{A}$, (se Lemurell avsnitt 4.2, definition 4.17.)

$$\mathbf{y} = \mathbf{Ax} = \begin{bmatrix} \mathbf{a_1} & \cdots & \mathbf{a_n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = x_1\mathbf{a_1} + x_2\mathbf{a_2} + \cdots + x_n\mathbf{a_n} =$$

$$= \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix} x_2 + \cdots + \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix} x_n$$

I MATLAB skulle vi, för t.ex. $n = 3$, skriva

```
>> y=A(:,1)*x(1)+A(:,2)*x(2)+A(:,3)*x(3)
```

och för ett större värde på n skulle vi kunna bilda linjärkombinationen enligt

```
>> y=zeros(m,1);
>> for j=1:n
       y=y+A(:,j)*x(j);
   end
```

Matris-matrisprodukten $\mathbf{C} = \mathbf{AB}$ av en $m \times n$-matris $\mathbf{A}$ och en $n \times p$-matris $\mathbf{B}$, med kolonner $\mathbf{b}_1, \cdots, \mathbf{b}_p$, är en $m \times p$-matris som ges av

$$\mathbf{C} = \mathbf{AB} = \mathbf{A}[\mathbf{b}_1, \cdots, \mathbf{b}_p] = [\mathbf{Ab}_1, \cdots, \mathbf{Ab}_p]$$

eller elementvis

$$c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}, \quad i = 1, \cdots, m, \ j = 1, \cdots, n$$

Matrismultiplikationen $\mathbf{C} = \mathbf{AB}$ kan beräknas i MATLAB med den inbyggda matrismultiplikationen (`*`) enligt `C=A*B` eller med lite egen programmering (som bygger upp `C` elementvis)

```
>> C=zeros(m,p);
>> for i=1:m
       for j=1:p
           cij=0;
           for k=1:n
               cij=cij+A(i,k)*B(k,j);
           end
           C(i,j)=cij;
       end
   end
```

Alternativt bygger vi upp kolonnvis enligt

```
>> C=zeros(m,p);
>> for j=1:p
       C(:,j)=A*B(:,j);
   end
```

Här använder vi den inbyggda matris-vektormultiplikationen för att bilda kolonnerna.

Uppgift 6. Skriv in följande matriser i MATLAB.

$$\mathbf{A} = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 4 & 5 & 6 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Beräkna följande produkter, både för hand, dvs. med penna och papper, och med MATLAB, dvs. med inbyggda matrismultiplikationen ($*$),

$$\mathbf{Ax}, \quad \mathbf{Bx}, \quad \mathbf{AB}, \quad \mathbf{ax}, \quad \mathbf{xa}, \quad \mathbf{aB}.$$

Beräkna produkten $\mathbf{Ax}$ även genom att ni skriver en egen programkod i MATLAB. Skriv snyggt och tydligt.

Uppgift 7. Bilda

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 2 & 1 & 1 \\ 4 & 1 & 0 \\ -2 & 2 & 1 \end{bmatrix}$$

(a). Kontrollera att associativa och distributiva lagarna gäller för dessa matriser.

Du skall alltså se att $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$ respektive $\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB}+\mathbf{AC}$ och $(\mathbf{B}+\mathbf{C})\mathbf{A} = \mathbf{BA}+\mathbf{CA}$.

(b). Vanligtvis är matrismultiplikation inte kommutativ. T.ex är $\mathbf{AC} \neq \mathbf{CA}$ och $\mathbf{BC} \neq \mathbf{CB}$ (kontrollera gärna), men vad gäller för $\mathbf{AB}$ och $\mathbf{BA}$?

1 Målsättning

Avsikten med denna laborationen är att se på uppbyggnad och hantering av matriser. Vi skall kunna modifiera en matris eller vektor samt ta ut delar av eller enskilda element. Vidare skall vi bekanta oss med inbyggda matris- och vektorfunktioner. Avslutningsvis skall vi se på operationer på matriser.

2 Kommentarer och förklaringar

I avsnittet "Matris- och vektorfunktioner" vi hur man tar reda på storleken på en matris. Ibland låter man matriser eller vektorer växa upp medan man löser ett problem och det är då bra att efteråt kunna bestämma storleken (`size`, `length`).

Att enkelt kunna summera alla element i en matris eller vektor behövs då och då, precis som att ta produkt, minsta eller största värde samt sortera element (`sum`, `prod`, `min`, `max`, `sort`). Funktionerna `zeros` och `ones` kommer vi använda ofta, och ibland vill vi ha slumptal med `rand`.

Matris-vektorprodukt samt matris-matrisprodukt gör man givetvis normalt med den i MATLAB inbyggda matrisprodukten (`*`). Den egna programmeringen vi gjorde i "Operationer på matriser" var enbart en övning i just programmering samt i att hantera indexering och liknande i MATLAB.

3 Lärandemål

Efter denna laboration skall du i MATLAB kunna

- bygga upp och modifiera matriser och vektorer
- bestämma storlek av en matris eller vektor med `size` och `length`
- bestämma största och minsta värde i en matris eller vektor samt bilda summa och produkt med `min`, `max`, `sum` och `prod`
- utföra de vanliga matrisoperationerna