

Matriser och linjära ekvationssystem

Matriser

En matris är som ni redan vet ett rektangulärt talschema:

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

Matrisen ovan har m rader och n kolonner, vi säger att den är av typ $m \times n$. Ett matriselement i rad nr i , kolonn nr j tecknas a_{ij} , där i är radindex och j är kolonnindex. I MATLAB skrivs detta $\mathbf{A}(i,j)$ och $[\mathbf{m},\mathbf{n}]=\text{size}(\mathbf{A})$ ger matrisens typ.

Indexeringen av en matris i MATLAB är alltid som ovan, dvs. rad- och kolonnindex börjar alltid på ett och vi kan inte ändra på det.

En matris av typ $m \times 1$ kallas kolonnmatris (kolonnvektor) och en matris av typ $1 \times n$ kallas radmatris (radvektor):

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix}, \quad \mathbf{b} = [b_1 \quad \cdots \quad b_n]$$

Du kommer att se att vi använder oftast kolonnvektorer för att representera kvantiteter som vi beräknar.

Element nr i ges i MATLAB av $\mathbf{a}(i)$ och antalet element ges av $\mathbf{m}=\text{length}(\mathbf{a})$. Motsvarande gäller för radvektorn $\mathbf{b}$. Även för vektorer gäller att indexeringen alltid börjar på ett.

Som exempel tar vi

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

Vi skriver in detta i MATLAB enligt

```
>> A=[1 4 7 10; 2 5 8 11; 3 6 9 12]
>> a=[1; 3; 5]
>> b=[0 2 4]
```

Uppgift 1. Skriv in följande matriser i MATLAB.

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

Skriv ut matriselementen a_{23} , b_{23} , x_2 . Prova `size` och `length`. Ändra b_{23} genom att skriva `B(2,3)=5`.

En matris kan betraktas som en kollektion av kolonner:

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

med kolonnerna

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

Man kan även betrakta den som en kollektion av rader, men vi använder oftast kolonnrepresentationen. I MATLAB plockar man ut kolonn nr j med $\mathbf{A}(:, j)$. Här är j kolonnindex medan radindex $i = 1, \dots, m$ representeras av tecknet kolon $(:)$. På liknande vis ges rad nr i av $\mathbf{A}(i, :)$.

Uppgift 2. Skriv ut kolonn nr 1, 2 och 3 ur matrisen $\mathbf{A}$ i uppgift 1. Sätt in kolonnvektorn $\mathbf{x}$ som första kolonn i $\mathbf{B}$ genom att skriva $\mathbf{B}(:, 1) = \mathbf{x}$.

Uppgift 3. 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 $\mathbf{B} = [\mathbf{b}_1 \quad \mathbf{b}_2 \quad \mathbf{b}_3]$.

Matris-vektor-multiplikation

Vi definierar produkten av en radmatris och en kolonnmatris (med samma antal element) enligt:

$$\mathbf{a}\mathbf{x} = [a_1 \quad \cdots \quad a_n] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = a_1x_1 + a_2x_2 + \cdots + a_nx_n.$$

Observera att detta är samma formel som för skalärprodukt. Produkten $\mathbf{y} = \mathbf{A}\mathbf{x}$ av en matris av typen $m \times n$ och en kolonnvektor av typen $n \times 1$ definieras på liknande vis genom att vi multiplicerar matrisens rader i tur och ordning med kolonnvektorn. Vi får en kolonnvektor av typen $m \times 1$ och den ges av

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{1}$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \cdots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}, \tag{2}$$

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

Observera att typerna måste stämma överens för att produkten ska vara definierad enligt regeln $m \times 1 = (m \times n)(n \times 1)$.

Ett alternativt sätt att introducera matris-vektor-multiplikation är att definiera $\mathbf{Ax}$ som en linjärkombination av kolonnerna i $\mathbf{A}$, (se Lay avsnitt 1.4)

$$\mathbf{Ax} = [\mathbf{a}_1 \quad \cdots \quad \mathbf{a}_n] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots + x_n \mathbf{a}_n.$$

Detta leder förstås till samma uttryck som i (2),

$$x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots + x_n \mathbf{a}_n = \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix} x_2 + \cdots + \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix} x_n = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}.$$

I MATLAB skriver man helt enkelt $\mathbf{y}=\mathbf{A}*\mathbf{x}$.

Uppgift 4. Skriv in följande matriser i MATLAB.

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

Beräkna följande produkter, först för hand sedan med MATLAB.

$$\mathbf{Ax}, \quad \mathbf{Bx}, \quad \mathbf{ax}, \quad \mathbf{Aa}.$$

Observera att alla dessa produkter inte nödvändigtvis är definierade. Varför, och hur märks det i MATLAB?

Linjärt ekvationssystem

Matriser används bland annat för att skriva ned linjära ekvationssystem. Exempel: ekvationssystemet

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 14 \\ 3x_1 + 2x_2 + x_3 = 10 \\ 7x_1 + 8x_2 = 23 \end{cases}$$

kan skrivas på matrisform

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 7 & 8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 14 \\ 10 \\ 23 \end{bmatrix},$$

dvs.

$$\mathbf{Ax} = \mathbf{b}, \quad \text{med } \mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 7 & 8 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 14 \\ 10 \\ 23 \end{bmatrix}.$$

Vi ska lära oss hur man löser sådana ekvationssystem. I MATLAB finns backslash-kommandot ($\backslash$) eller alternativt kommandot `rref` (row-reduced-echelon form) som löser systemet, $\mathbf{Ax} = \mathbf{b}$:

```
>> x=A\b
>> rref([A b])
```

I det första fallet fungerar det bra om lösningen är entydig men sämre om det finns fria variabler eller inga lösningar alls. I det andra fallet reducerar MATLAB den utökade matrisen $[\mathbf{A} \ \mathbf{b}]$ till reducerad trappstegsform.

Uppgift 5. Skriv följande ekvationssystem på matrisform (utökad matris!) och lös dom sedan med `\` respektive `rref`.

$$\begin{cases} x_1 + 5x_2 + 9x_3 = 29 \\ 2x_1 + 5x_3 = 26 \\ 3x_1 + 7x_2 + 11x_3 = 39 \end{cases} \quad \begin{cases} x_1 + x_2 + 3x_3 + 4x_4 = 2 \\ -2x_1 + 2x_2 + 2x_3 = -4 \\ x_1 + x_2 + 2x_3 + 3x_4 = 1 \\ x_1 - x_2 - 2x_3 - x_4 = 1 \end{cases}$$

Ibland blir det något enklare att tolka svaret om vi ger kommandot `format rat` före beräkningarna, då skriver MATLAB svaret med rationella tal. Med `format short` får vi tillbaka standard formatet.

Skapa matris kolonnvis

Vi bygger ofta upp matriser med hjälp någon beräkningsalgoritm, ibland görs beräkningen kolonnvis. Vi ger ett exempel. Matrisen

$$\mathbf{C} = \begin{bmatrix} 1 & 2 & 3 & 4 & \cdots & 10 \\ 1 & 2 & 3 & 4 & \cdots & 10 \\ 1 & 2 & 3 & 4 & \cdots & 10 \\ 1 & 2 & 3 & 4 & \cdots & 10 \end{bmatrix}$$

skapas av `for`-loopen

```
clear C
ett=ones(4,1);
for i=1:10
    C(:,i)=i*ett;
end
```

Uppgift 6. Skriv en skriptfil som gör detta. Tag bort semikolon så att du ser vad som händer i varje steg.

Du kan också prova att bilda $\mathbf{C}$ genom en s.k. yttreprodukt enligt `C=ett*[1:10]`. Lägg märke till att `ett` är en kolonnvektor medan `[1:10]` är en radvektor.

Vi kan använda `tic` och `toc` för att mäta den tid beräkningarna tar. Det blir tydligare om vi tar en större matris. Prova följande som en skriptfil.

```
clear C
ett=ones(2000,1);
tic
for i=1:100
    C(:,i)=i*ett;
end
toc
```

Tänk på att sätta semikolon på alla beräkningar, så inte skärmen fylls med ointressanta utskrifter.

Matrisen **C** kommer succesivt byggas upp och bli av typen 2000×100 . Om vi nu istället för `clear C` skriver `C=zeros(2000,100)`; så får matrisen **C** sin slutliga storlek på en gång. Vi börjar med en stor matris fylld med nollor, där vi i loopen succesivt sätter in rätt kolonner. Kör nu den nya skriptfilen och se om beräkningstiden blev kortare.

Matrisiterationer

Avslutningsvis skall vi kort se på några matrisiterationer.

Uppgift 7. Låt

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Skriv en `for`-loop som skapar matrisen $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_5]$ där

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k.$$

Uppgift 8. Genom att rita ut punkter $\mathbf{x}_k$ i planet som genereras av iterationsformeln

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{b}$$

där matrisen **A** och vektorn **b** väljs slumpvis enligt vissa regler, kan vi producera en bild som liknar t.ex. ett ormbunksblad. (Detta är en s.k. fraktal.)

I varje iteration låter vi $\mathbf{A} = \mathbf{A}_i$ och $\mathbf{b} = \mathbf{b}_i$ med sannolikheten P_i , där

$$\mathbf{A}_1 = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 0.2 & -0.26 \\ 0.23 & 0.22 \end{bmatrix}, \quad \mathbf{A}_3 = \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix}, \quad \mathbf{A}_4 = \begin{bmatrix} 0 & 0 \\ 0 & 0.16 \end{bmatrix}$$

och

$$\mathbf{b}_1 = \mathbf{b}_2 = \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}, \quad \mathbf{b}_3 = \begin{bmatrix} 0 \\ 0.44 \end{bmatrix}, \quad \mathbf{b}_4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

samt

$$P_1 = 0.85, \quad P_2 = P_3 = 0.07, \quad P_4 = 0.01$$

Starta med $\mathbf{x}_0 = (0.5, 0.5)$ och rita ut punkterna $\mathbf{x}_k$ för $k = 0, 1, \dots, 100$.

När man snabbt skall rita en snygg bild på skärmen (då krävs det ca 20000 punkter) använder man lämpligen ett grafikobjekt vars data uppdateras i varje iteration.

Lite hjälp på vägen om det behövs ...

```
figure('Name','Ormbunksblad','Menu','none','NumberTitle','off',...
      'Position',[256 412 320 512])
set(gcf,'Color',[0.1 0.1 0.1])
kmax=20000; x=[0.5;0.5];
h=plot(x(1),x(2),'.g','markersize',1);
set(h,'erasemode','none');
axis([-3 3 0 10]), axis off, shg
p=[0.85 0.92 0.99 1.00];
A1=[ 0.85 0.04;-0.04 0.85]; b1=[0;1.6 ];
A2=[ 0.20 -0.26; 0.23 0.22]; b2=[0;1.6 ];
A3=[-0.15 0.28; 0.26 0.24]; b3=[0;0.44];
A4=[ 0      0      ; 0      0.16]; b4=[0;0   ];
for k=1:kmax
    r=rand;
    if r<p(1)
        x=A1*x+b1;
    elseif r<p(2)
        x=A2*x+b2;
    elseif r<p(3)
        x=A3*x+b3;
    else
        x=A4*x+b4;
    end
    set(h,'xdata',x(1),'ydata',x(2));
    drawnow
end
```