Linjära ekvationssystem

1 Inledning

Denna laboration börjar med att vi ser på matriser i MATLAB och deras uppbyggnad. Sedan ser vi på martismultiplikation. Avslutningsvis ser vi på linjära ekvationssystem, där vi också ser lite på system med stora och glesa koefficientmatriser. I teknik- och naturvetenskap är det vanligt med stora och glesa ekvationssystem. Redan nästa läsperiod kommer ni lösa sådana i mekanikkursen.

2 Matriser

En matris är som ni 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 `A(i,j)` och `[m,n]=size(A)` ger matrisens typ, medan `m=size(A,1)` ger endast antal rader och `n=size(A,2)` ger endast antal kolonner.

Indexeringen i är alltid som i matrisen 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):

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

Du kommer att se att vi använder oftast kolonnvektorer för att representera kvantiteter som vi beräknar. Element nr i ges i MATLAB av `b(i)` och antalet element ges av `m=length(b)`. Motsvarande gäller för radvektorn $\mathbf{c}$. Indexeringen börjar alltid på ett även för vektorer.

Som exempel tar vi

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

Vi skriver in detta i MATLAB enligt

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

Vi ser av vilken typ matrisen är och några element med

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

>> A(2,3)
ans =
     8
```

Prova gärna `length` och `size` på `b` och `c`. Någon skillnad? Skriv ut något element också.

En matris kan betraktas som en kollektion av kolonner:

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

med kolonnerna

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

Man kan även betrakta den som en kollektion av rader, men vi använder oftast kolonnrepresentationen. 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 `:`. På liknande vis ges rad nr i av `A(i,:)`.

```
>> a1=A(:,1)
a1 =
     1
     2
     3

>> A2=A(2,:)
A2 =
     2     5     8    11
```

Uppgift 1. Skriv in följande matriser i MATLAB.

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

(a). Skriv ut matriselementen a_{23}, b_{23}, c_2 och d_3. Prova `size` och `length` på $\mathbf{A}$, $\mathbf{B}$, $\mathbf{c}$ och $\mathbf{d}$. Ändra b_{23} genom att skriva `B(2,3)=5`.

(b). Skriv ut kolonn nr 1, 2 och 3 ur matrisen $\mathbf{A}$. Sätt in kolonnvektorn $\mathbf{c}$ som 2:a kolonn i $\mathbf{A}$ genom att skriva `A(:,2)=c`.

(c). 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 $\mathbf{B} = [\mathbf{b}_1\ \mathbf{b}_2\ \mathbf{b}_3]$.

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 2. Låt som i uppgift 1

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

(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 4:e raden i $\mathbf{A}$ byta plats och låt därefter den 2:a och 3: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 typen $m \times n$ fylld med nollor. Med `zeros(size(A))` får vi en matris fylld med nollor av samma typ som `A`. Motsvarande gäller för `ones`.

Enhetsmatriser bildas med funktionen `eye`, med `eye(n)` får vi enhetsmatrisen av typen $n \times n$. 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 typen $m \times n$ och med `eye(size(A))` får vi en av samma typ 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.

4 Matris-vektorprodukt

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

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

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
```

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

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

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

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
```

Uppgift 3. Skriv in följande matriser i MATLAB.

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

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

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

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

5 Matris-matrisprodukt

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
```

Uppgift 4. Skriv in följande matriser i MATLAB.

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

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

Du skall alltså se att

$$\begin{aligned} &\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C} \\ &\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB}+\mathbf{AC} \\ &(\mathbf{B}+\mathbf{C})\mathbf{A} = \mathbf{BA}+\mathbf{CA} \end{aligned}$$

(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 $\mathbf{BA}$?

6 Linjärt ekvationssystem

Linjära ekvationssystem med små koefficientmatriser, löser vi med backslash-kommandot `x=A\b` eller alternativt med kommandot `rref` enligt `rref([A b])`.

Backslash-kommandot (\) används när $\mathbf{A}$ är en kvadratisk matris och vi vet att vi har en entydig lösning. Har vi fria variabler så använder vi `rref` som reducerar den utökade matrisen $[\mathbf{A}\ \mathbf{b}]$ till reducerad trappstegsform.

Vid numeriska beräkningar skall man *inte* bilda $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, dvs. man skall inte bilda inversen och multiplicera med den. Det blir mindre effektivt och mindre noggrant, speciellt för lite större matriser.

Som exempel på ett linjärt ekvationssystem tar vi

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

som 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 bildar koefficientmatrisen $\mathbf{A}$ och högerledsvektorn $\mathbf{b}$ med

```
>> A=[1 2 3;3 2 1;7 8 0]
>> b=[14;10;23]
```

Med kommandot `rref` kommer vi till radreducerad trappstegsform (row-reduced-echelon form) så att vi kan läsa av lösningen till $\mathbf{Ax} = \mathbf{b}$.

Först bildar vi den utökade matrisen $\mathbf{E} = [\,\mathbf{A}\ \mathbf{b}\,]$ med

```
>> E=[A b]
E =
     1     2     3    14
     3     2     1    10
     7     8     0    23
```

och sedan får vi den reducerade matrisen med

```
>> R=rref(E)
R =
     1     0     0     1
     0     1     0     2
     0     0     1     3
```

Lösningen ser vi i sista kolonnen i `R` och vi har

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Som ytterligare ett exempel ser vi på följande ekvationssystem med oändligt många lösningar

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 10 \\ 3x_1 + 2x_2 + x_3 = 14 \\ 7x_1 + 8x_2 + 9x_3 = 46 \end{cases}$$

eller på matrisform

$$\mathbf{Ax} = \mathbf{b} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 14 \\ 46 \end{bmatrix}$$

```
>> A=[1 2 3;3 2 1;7 8 9]
>> b=[10;14;46]
```

Vi reducerar utökande matrisen med

```
>> R=rref([A b])
R =
     1     0    -1     2
     0     1     2     4
     0     0     0     0
```

Vi har en fri variabel. Om vi sätter $x_3 = t$ får vi

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2+t \\ 4-2t \\ t \end{bmatrix}$$

där t är ett godtyckligt reellt tal.

Uppgift 5. Skriv följande ekvationssystem på matrisform och lös dem sedan med `rref`.

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

Skulle det finnas oändligt många lösningar skriv upp en formel för samtliga lösningar.

Uppgift 6. Läs "Balancing Chemical Equations" i Lay avsnitt 1.6. Lös sedan stökiometriuppgiften Lay 1.6: 10. Be MATLAB räkna med rationella tal med kommandot `format rat` så blir det något enklare att tolka svaret. Med `format short` får vi sedan tillbaka standard formatet. Skriv ned den balanserade formeln på papper.

7 Stora och glesa linjära ekvationssystem

En del problem har väldigt många obekanta och ger stora matriser. Finns det få kopplingar blir det många nollor i matrisen, vi får en s.k. gles matris. Vi skall se hur MATLAB hanterar detta. När väl matrisen är lagrad känner MATLAB av att det är en gles matris när vi t.ex. vill beräkna lösningen till ett linjärt ekvationssystem.

Allmän gleshetsstruktur

Som exempel tar vi matrisen

$$\mathbf{A} = \begin{bmatrix} 7 & 0 & 0 & 5 & 0 \\ 0 & 2 & -8 & 0 & 0 \\ 3 & 0 & 0 & 0 & 11 \\ 1 & 0 & 0 & 4 & 0 \\ 0 & -5 & 9 & 0 & 6 \end{bmatrix}$$

Detta är en gles matris och en lagringsmetod är att lagra tripplar (i, j, a_{ij}) för de element som är skilda från noll. Vi bildar en tabell över nollskilda element och deras rad- respektive kolonnindex.

i	1	1	2	2	3	3	4	4	5	5	5
j	1	4	2	3	1	5	1	4	2	3	5
a_{ij}	7	5	2	−8	3	11	1	4	−5	9	6

I MATLAB bildar man tre vektorer av rad- och kolonnindex samt matriselement och sedan den glesa matrisen med funktionen `sparse` enligt

```
>> ivec=[1 1 2 2 3 3 4 4 5 5 5];
>> jvec=[1 4 2 3 1 5 1 4 2 3 5];
>> aijvec=[7 5 2 -8 3 11 1 4 -5 9 6];
```

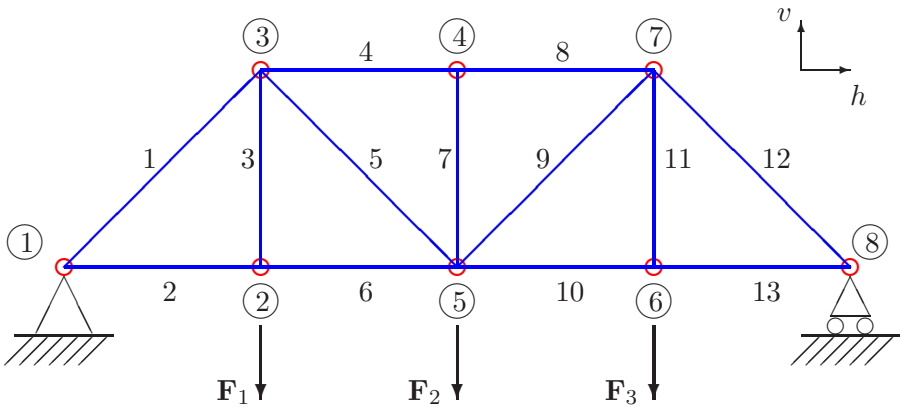
```
>> A=sparse(ivec,jvec,aijvec)
A =
   (1,1)         7
   (3,1)         3
   (4,1)         1
   (2,2)         2
   (5,2)        -5
   (2,3)        -8
   (5,3)         9
   (1,4)         5
   (4,4)         4
   (3,5)        11
   (5,5)         6
```

Utskriften visar alla nollskilda element och deras plats i matrisen. Med funktionen `full` kan vi omvandla en gles matris till en vanlig fylld matris enligt

```
>> FA=full(A)
FA =
     7     0     0     5     0
     0     2    -8     0     0
     3     0     0     0    11
     1     0     0     4     0
     0    -5     9     0     6
```

Här får vi en kontroll att vi bildat rätt matris.

Som exempel på en lite större gles matris tar vi: *Fackverk* – Krafterna i de olika grenarna av det *statiskt bestämda* fackverket i figuren nedan skall bestämmas då angivna yttre krafter är anbringade.



Genom att ansätta kraftjämvikt i horisontal- och vertikalled i knutpunkterna får vi ett linjärt ekvationssystem $\mathbf{Ax} = \mathbf{b}$ för de sökta krafterna i fackverkets grenar.

Beroende på om vi använder s.k. friläggning eller inte blir matrisen $\mathbf{A}$ av typen 13×13 eller 16×16, vi har alltså 13 eller 16 obekanta. (Fackverksproblem kommer ni stöta på i vårens kurs i mekanik.) Nu väljer vi att sätta upp ekvationssystemet utan friläggning.

Låt $u = \sin(\frac{\pi}{4}) = \frac{1}{\sqrt{2}}$ och låt x vara vektorn med de sökta krafterna. Vidare tar vi de pålagda krafternas storlek som 8, 10 resp. 12. Vi får ekvationerna

$$
\begin{array}{rlrl}
-x_2 + x_6 = 0 & \text{h knut 2} & -ux_1 + x_4 + ux_5 = 0 & \text{h knut 3} \\
x_3 - 8 = 0 & \text{v knut 2} & -ux_1 - x_3 - ux_5 = 0 & \text{v knut 3} \\
-x_4 + x_8 = 0 & \text{h knut 4} & -ux_5 - x_6 + ux_9 + x_{10} = 0 & \text{h knut 5} \\
-x_7 = 0 & \text{v knut 4} & ux_5 + x_7 + ux_9 - 10 = 0 & \text{v knut 5} \\
-x_{10} + x_{13} = 0 & \text{h knut 6} & -x_8 - ux_9 + ux_{12} = 0 & \text{h knut 7} \\
x_{11} - 12 = 0 & \text{v knut 6} & -ux_9 - x_{11} - ux_{12} = 0 & \text{v knut 7} \\
-ux_{12} - x_{13} = 0 & \text{h knut 8} & &
\end{array}
$$

Detta blir med matrisbeteckningar $\mathbf{Ax} = \mathbf{b}$ med

$$
\mathbf{A} = \begin{bmatrix}
0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u & 0 & 0 & 1 & u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-u & 0 & -1 & 0 & -u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -u & -1 & 0 & 0 & u & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & u & 0 & 1 & 0 & u & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -u & 0 & 0 & u & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -u & 0 & -1 & -u & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -u & -1
\end{bmatrix}, \quad
\mathbf{b} = \begin{bmatrix}
0 \\ 8 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 10 \\ 0 \\ 12 \\ 0 \\ 0 \\ 0
\end{bmatrix}
$$

Vi matar in de nollskilda elementen med motsvarande rad- och kolonnindex

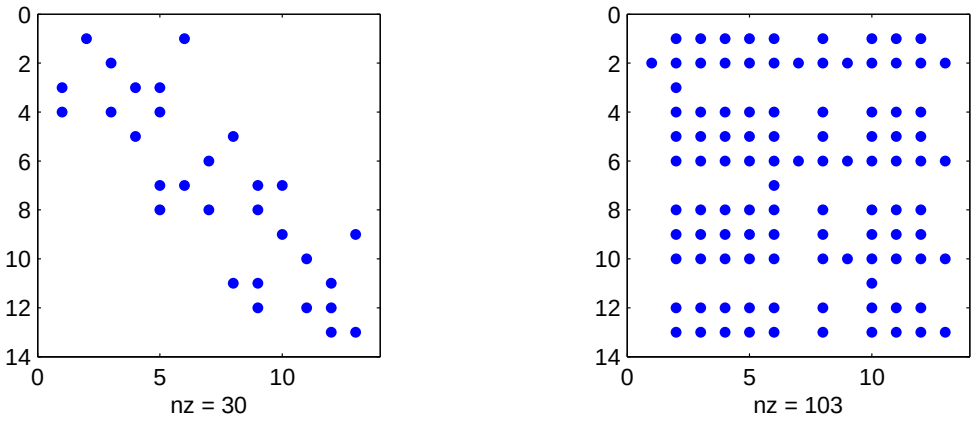
```
>> u=1/sqrt(2);
>> rad=[ 1 1 2  3 3 3  4  4  4  5 5  6  7  7 7  7 8 8 8  9  9 10 11 11 11 12 12 12 13 13 ];
>> kol=[ 2 6 3  1 4 5  1  3  5  4 8  7  5  6 9 10 5 7 9 10 13 11  8  9 12  9 11 12 12 13 ];
>> ele=[-1 1 1 -u 1 u -u -1 -u -1 1 -1 -u -1 u  1 u 1 u -1  1  1 -1 -u  u -u -1 -u -u -1 ];
```

Vi bildar den glesa matrisen $\mathbf{A}$, högerledsvektorn $\mathbf{b}$ och beräknar lösningen enligt

```
>> A=sparse(rad,kol,ele);
>> b=[0 8 0 0 0 0 0 10 0 12 0 0 0]';
>> x=A\b
x =
  -19.7990
   14.0000
    8.0000
  -20.0000
    8.4853
   14.0000
         0
  -20.0000
    5.6569
   16.0000
   12.0000
  -22.6274
   16.0000
```

Vi löser ekvationssystemet med kommandot `x=A\b` eftersom matrisen $\mathbf{A}$ är kvadratisk och vi vet att vi har en entydig lösning. Då matrisen är lagrad som en gles matris kommer MATLAB beräkna lösningen till systemet med metoder som utnyttjar gleshetsstrukturen på ett mycket effektivt sätt.

Med `spy(A)` ser vi att bara 30 av de 169 elementen i matrisen är skilda från noll.



Vi gör `spy(inv(A))` och ser att inversen $\mathbf{A}^{-1}$ är en nästan fylld matris, detta är typiskt för inversen till glesa matriser. Detta är ett av skälen att man inte skall bilda inverser och multiplicera med dem, utan istället direkt lösa med lämplig metod.

Vid lite mer komplicerade tekniska tillämpningar är 1000-tals eller 10000-tals obekanta inget ovanligt.

Bandmatriser

Linjära ekvationssystem från tekniska tillämpningar har ofta en matris med diagonaler av noll-skilda element, en s.k. *bandmatris*. En sådan matris kan man bilda med `sparse` men enklare och mer effektivt är att använda funktionen `spdiags`.

Som exempel på en tillämpning där `spdiags` är lämplig att använda tar vi: *Stationär värmeledning* i en isolerad stav med en värmekälla

Temperaturen $u(x)$ beskrivs av ett randvärdesproblem för andra ordningens differentialekvation

$$\begin{cases} -cu''(x) = f(x),\ 0 \le x \le L \\ u(0) = g_0,\ u(L) = g_L \end{cases}$$

där g_0 och g_L är temperaturen i stavens ändpunkter, c är stavens värmeledningsförmåga och $f(x)$ är värmekällan.

Inför en indelning $x_i = ih$, $i = 0, 1, \cdots, n+1$ av intervallet $0 \le x \le L$, med $h = \frac{L}{n+1}$.

Om vi i differentialekvationen ersätter $u''(x_i)$ med differenskvoten

$$\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}$$

i de inre punkterna $x_i, i = 1, \cdots, n$, får vi

$$-u(x_{i+1}) + 2u(x_i) - u(x_{i-1}) = \tfrac{h^2}{c} f(x_i) \quad i = 1, 2, \cdots, n$$

Dessa linjära ekvationer ger sambandet mellan temperaturer på olika ställen på staven.

Låter vi u_i beteckna approximationer av $u(x_i)$ och utnyttjar randvillkoren $u(0) = g_0,\ u(L) = g_L$ kan vi med matriser skriva detta $\mathbf{Au} = \mathbf{b}$ där

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix},\ \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix},\ \mathbf{b} = \begin{bmatrix} \frac{h^2}{c} f(x_1) + g_0 \\ \frac{h^2}{c} f(x_2) \\ \vdots \\ \frac{h^2}{c} f(x_{n-1}) \\ \frac{h^2}{c} f(x_n) + g_L \end{bmatrix}$$

Matrisen $\mathbf{A}$ är en *bandmatris* (den kallas också en tridiagonal matris) och vi bildar den med `spdiags`.

Vi bildar först en vektorer fylld med 1:or, därefter placerar vi in denna vektor (multiplicerad med 2 respektive -1) som diagonaler med `spdiags` enligt

```
>> n=30; ett=ones(n,1);
>> A=spdiags([-ett 2*ett -ett],[-1 0 1],n,n);
```

Diagonalerna sätts ihop som kolonner i en matris med `[-ett 2*ett -ett]`, de måste därför vara lika långa. Kolonnerna placeras som diagonaler i ordningen som ges av vektorn `[-1 0 1]` och det hela skall bli en $n \times n$-matris, därav `n,n` allra sist.

När vi har bildat högerledsvektorn $\mathbf{b}$ löser vi med backslash (\). När matrisen är lagrad som gles matris känner MATLAB av att det och lösningen av ekvationssystemet görs effektivt med metoder som tar vara på gleshetsstrukturen.

Uppgift 7. Låt $c = 2,\ L = 1,\ g_0 = 40,\ g_L = 60$ samt $f(x) = 200\exp(-(x - \frac{L}{2})^2)$. Lös värmeledningsproblemet och rita upp lösningen. Tag $n = 30$.

Vår värmeledningsekvation är ett specialfall av *linjära* randvärdesproblem, dvs. problem som kan skrivas

$$\begin{cases} u''(x) + a(x)u'(x) + b(x)u(x) = f(x),\ x_a \le x \le x_b \\ u(x_a) = u_a,\ u(x_b) = u_b \end{cases}$$

Är $a(x)$ och $b(x)$ konstanta, så såg vi förra läsperioden hur vi kan försöka räkna fram en formel för lösningen. Om $a(x)$ och $b(x)$ inte konstanta så går det vanligtvis inte. Metoden att ersätta derivator med differenskvoter för att beräkna en numerisk lösning fungerar däremot alltid för dessa allmännare problem.