Matriser och linjära ekvationssystem.

1 Matriser

En matris är som ni vet ett rektangulärt talschema:

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

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.

Indexeringen i MATLAB är alltid som i (1), 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):

$$\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_n \end{bmatrix} \tag{2}$$

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)`. Även för vektorer gäller att indexeringen alltid börjar på ett. Motsvarande gäller för radvektorn $\mathbf{c}$.

Som exempel tar vi

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

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]
```

och ser på typerna 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.

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

(a). Skriv ut matriselementen a_{23}, b_{23}, x_2. Prova `size` och `length`. 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 $\mathbf{x}$ som första kolonn i $\mathbf{B}$ genom att skriva `B(:,1)=x`.

(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 $\mathbf{A}^T$ 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

$$\mathbf{B} = \begin{bmatrix} 4 & 5 & 6 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

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

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

2 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.

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.

3 Speciella matriser

MATLAB har funktionerna `zeros, ones, eye` för att bilda speciella matriser med nollor, ettor och enhetsmatris.

Med `zeros(2,5)` får vi en matris av typen 2×5 fylld med nollor och med `ones(2,5)` får vi en matris av samma typ, men fylld med ettor. För att få en enhetsmatris av typen 5×5 ger vi `eye(5,5)`. Med `ones(size(A))` får vi en matris fylld med ettor av samma typ som `A`. Motsvarande gäller för `zeros` och `eye`.

Det finns funktioner `rand` och `randn` för att bilda matriser fyllda med slumptal.

4 Matris-vektor-multiplikation

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

$$\mathbf{ax} = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = a_1x_1 + a_2x_2 + \cdots + a_nx_n.$$

Observera att detta är samma formel som för skalärprodukt. Produkten $\mathbf{y} = \mathbf{Ax}$ 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

$$\mathbf{y} = \mathbf{Ax} \tag{3}$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}, \tag{4}$$

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

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 $\mathbf{A}$,

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

Detta leder förstås till samma uttryck som i (4),

$$x_1\mathbf{a_1}+x_2\mathbf{a_2}+\cdots+x_n\mathbf{a_n} = \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix} x_2 + \cdots + \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix} x_n = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}.$$

I MATLAB skriver man helt enkelt `y=A*x`.

Uppgift 3. Skriv in följande matriser i MATLAB.

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

Beräkna följande produkter, först för hand sedan med MATLAB.

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

5 Operationer på matriser

Matris-vektorprodukten $\mathbf{y} = \mathbf{Ax}$ kan beräknas i MATLAB med den inbyggda matrismultiplikationen (`*`) enligt `y=A*x` (som i förra avsnittet) 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
```

Om man istället betraktar matris-vektorprodukt som en linjärkombination av kolonnerna i $\mathbf{A}$ får man som i förra avsnittet

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

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
```

Uppgift 4. Bilda

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

(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 $\mathbf{BA}$?

6 Linjära 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 (\) eller alternativt kommandot `rref` (row-reduced-echelon form) som löser systemet, $\mathbf{Ax} = \mathbf{b}$:

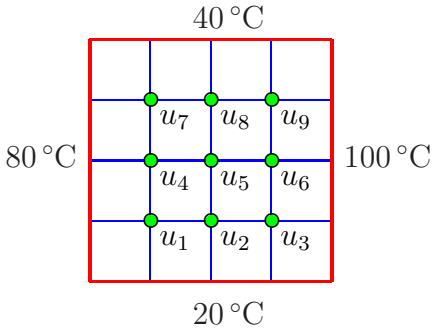
```
>> x=A\b
>> rref([A b])
```

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 [**A b**] till reducerad trappstegsform.

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

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

Uppgift 6. Vi skall beräkna temperaturen på en stålplatta där plattans kanter hålls vid temperaturer enligt figuren.



Antag att temperaturen i en nodpunkt är medelvärdet av temperaturena i de närmsta nodpunkterna i *väster*, *öster*, *söder* och *norr*. Låt $u_1, u_2, \cdots, u_9$ beteckna temperaturerna i de olika nodpunkterna. Sätt upp de ekvationer som ger temperaturen i de olika nodpunkterna. Skriv det linjära ekvationssystemet på matrisform $\mathbf{Au} = \mathbf{b}$ och lös detta i MATLAB.

7 Stora och glesa linjära ekvationssystem

En del problem har väldigt många obekanta och ger stora matriser. Finns det få kopplingar mellan ekvationerna 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.

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.

```
>> 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];
```

och bildar den glesa matrisen med funktionen `sparse` enligt

```
>> 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
```

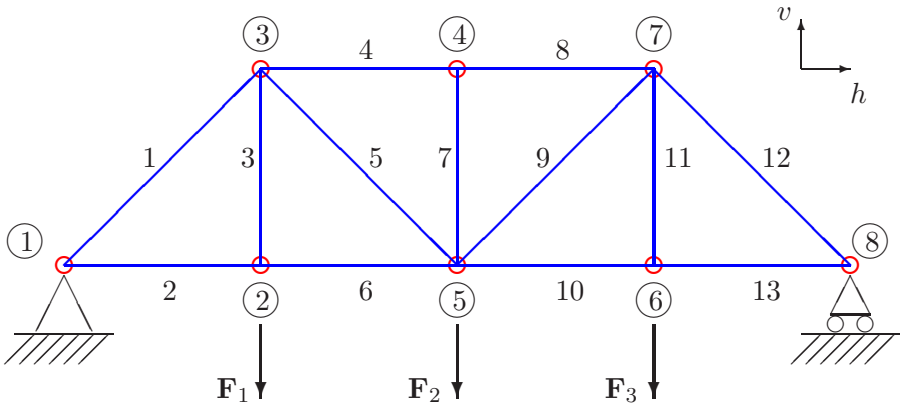
Utskriften visar alla nollskilda element och deras plats i matrisen. Med funktionen `full` kan vi omvandla en gles matris till en vanlig fylld matris enligt

```
>> 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.

Detta var ingen stor matris, vi ville visa principerna för hur man bildar en gles matris med `sparse`.

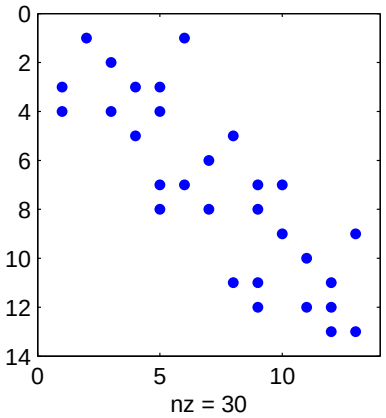
Som exempel på en lite större gles matris tar vi: *Fackverk* – Krafterna i de olika grenarna av det *statiskt bestämda* fackverket i figuren nedan skall bestämmas då angivna yttre krafter är anbringade.

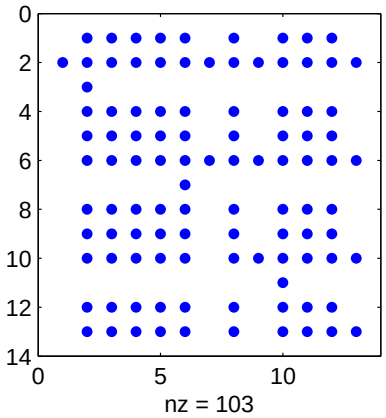


Genom att ansätta kraftjämvikt i horisontal- och vertikalled i knutpunkterna får vi ett linjärt ekvationssystem $\mathbf{Ax} = \mathbf{b}$ för de sökta krafterna i fackverkets grenar. Beroende på om vi använder friläggning eller inte blir matrisen $\mathbf{A}$ av storleken 13×13 eller 16×16, vi har alltså 13 eller 16 obekanta.

Efter att i MATLAB ha bildat den glesa matrisen $\mathbf{A}$, högerledsvektorn $\mathbf{b}$ så beräknar vi lösningen $\mathbf{x}$ med `x=A\b`. Eftersom matrisen är lagrad som en gles matris kommer MATLAB lösa ekvationssystemet med metoder som utnyttjar gleshetsstrukturen för att mycket effektivt och noggrant beräkna lösningen.

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 riktiga tillämpningar är 1000-tals eller 10000-tals obekanta inget ovanligt, även 100000-tals obekanta förekommer titt som tätt. Då kan vi prata om stort.

I många linjära ekvationssystem från tekniska tillämpningar är matrisen en s.k. *bandmatris*, dvs. matrisen har diagonaler av nollskilda element oftast samlade nära huvuddiagonalen. En sådan matris kan man bilda med `sparse` men enklare och mer effektivt är att använda funktionen `spdiags`.

Som ett första litet exempel tar vi matrisen

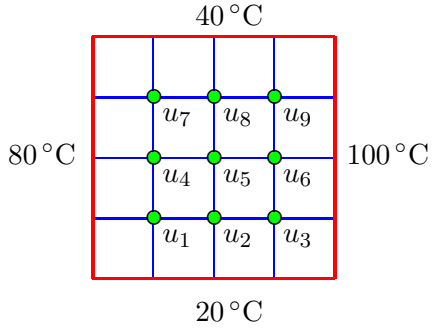
$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

Detta är en bandmatris med tre diagonaler (ofta kallad en tridiagonal matris). I MATLAB bildar vi en vektor fylld med 1:or, därefter placerar vi in denna vektor (multiplicerad med 2 respektive -1) som diagonaler med `spdiags` enligt

```
>> n=5; 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.

Som exempel på ett lite större linjärt ekvationssystem med en gles matris där `spdiags` är lämplig att använda tar vi: *Värmeledning* – Vi skall beräkna temperaturen på en stålplatta där plattans kanter hålls vid temperaturer enligt figuren.



Vi lägger ett gitter (grid) över området med $n = 3$ punkter horisontellt respektive vertikalt och betecknar temperaturerna i de sammanlagt $n^2 = 9$ olika gitterpunkterna med $u_1, u_2, \cdots, u_9$. Antar vi att temperaturen i en gitterpunkt är medelvärdet av temperaturena i de närmsta gitterpunkterna i *väster*, *öster*, *söder* och *norr* så kan vi skriva upp de ekvationer som ger temperaturerna.

$$\begin{cases} u_1 = \frac{1}{4}(80 + u_2 + 20 + u_4) \\ u_2 = \frac{1}{4}(u_1 + u_3 + 20 + u_5) \\ u_3 = \frac{1}{4}(u_2 + 100 + 20 + u_6) \\ \vdots \end{cases} \quad \Leftrightarrow \quad \begin{cases} 4\,u_1 - u_2 - u_4 = 20 + 80 \\ 4\,u_2 - u_1 - u_3 - u_5 = 20 \\ 4\,u_3 - u_2 - u_6 = 20 + 100 \\ \vdots \end{cases}$$

Vi skriver på matrisform $\mathbf{A}\mathbf{u} = \mathbf{b}$ enligt

$$\begin{bmatrix} 4 & -1 & 0 & \cdots \\ -1 & 4 & -1 & \cdots \\ 0 & -1 & 4 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} 20 + 80 \\ 20 + 0 \\ 20 + 100 \\ \vdots \end{bmatrix}$$

Uppgift 7. Skriv ned alla ekvationer på papper och gör färdigt matrisformen. Bilda matrisen med `spdiags`, ungefär på sammma sätt som i exemplet högst upp på sidan. Tänk på att ni inte har -1:or överallt på diagonalerna precis nedanför och ovanför huvuddiagonalen. Ni måste justera matrisen ni bildat. Bilda sedan högerledsvektorn och lös ekvationssystemet i MATLAB. Får ni rimliga temperaturvärden?