# Mer om linjära ekvationssystem

## 1 Inledning

Denna laboration fortsätter med linjära ekvationssystem och matriser. Vi ser på hantering och uppbyggnad av matriser samt operationer på matriser. Därefter ser vi på linjära ekvationssystem, dels med små koefficientmatriser men också med stora och glesa koefficientmatriser som ofta uppstår i tekniska tillämpningar. Men allra först lite repetition från första läsperioden.

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 $\mathbf{c}$.

## 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 $2 \times 2$, i partitioneringen (se Lay avsnitt 2.4)

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

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 Operationer på matriser

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

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

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 $\mathbf{y} = \mathbf{A}\mathbf{x}$ 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
```

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.

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

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 5.** 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 $\mathbf{BA}$?

# 5 Små linjära ekvationssystem

Matriser används bland annat för att skriva ned linjära ekvationssystem. I MATLAB finns backslash-kommandot (\) eller alternativt kommandot `rref` (row-reduced-echelon form) som löser systemet, $\mathbf{Ax} = \mathbf{b}$ enligt

```
>> x=A\b
>> 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.

# 6 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.

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
```

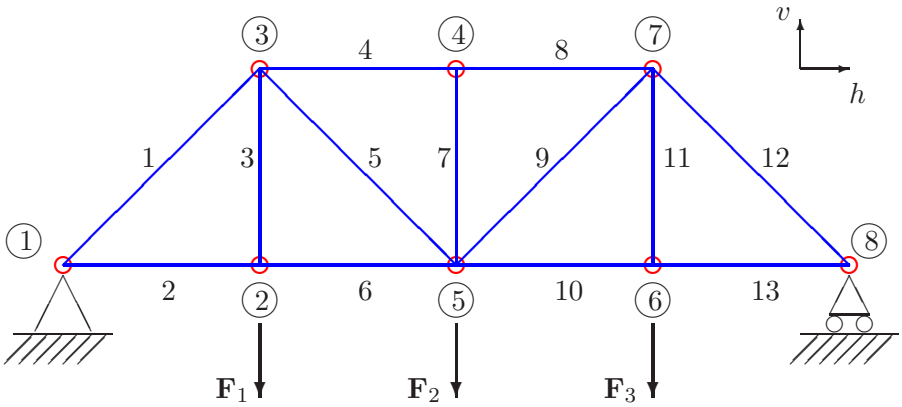
Utskriften visar alla nollskilda element och deras plats i matrisen. Med funktionen `full` kan vi omvandla en gles matris till en vanlig fylld matris enligt

```
>> 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`.

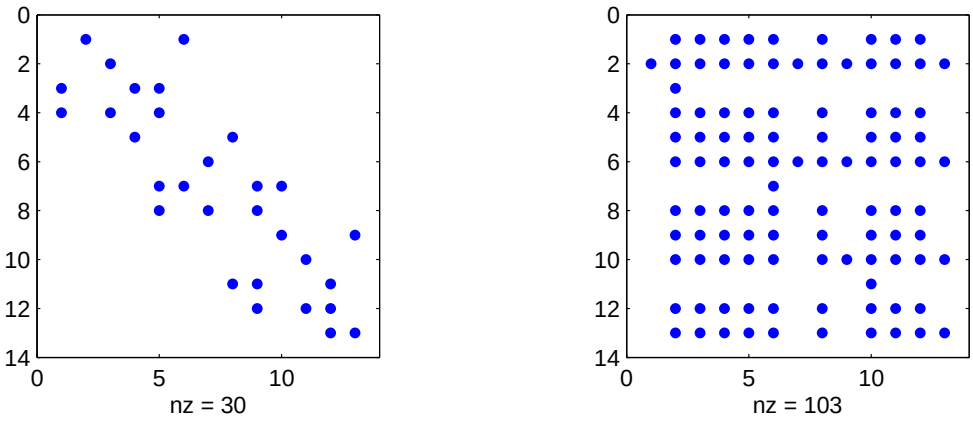
Som exempel på en lite större gles matris tar vi: *Fackverk* – Krafterna i de olika grenarna av det *statiskt bestämda* fackverket i figuren nedan skall bestämmas då angivna yttre krafter är anbringade.




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 $13 \times 13$ eller $16 \times 16$, vi har alltså 13 eller 16 obekanta. Gör det gärna om ni hinner.

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

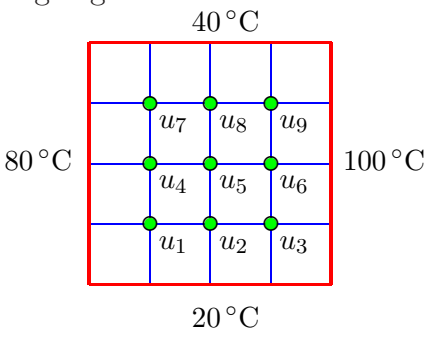
$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

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.

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

Vi skriver på matrisform $\mathbf{Au} = \mathbf{b}$ 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 6.** 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?

Resten av laborationen är *frivillig*!

När ni är klara med sista uppgiften har ni förhoppningsvis kommit fram till följande resultat:

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} 20 \\ 20 \\ 20 \\ 0 \\ 0 \\ 0 \\ 40 \\ 40 \\ 40 \end{bmatrix} + \begin{bmatrix} 80 \\ 0 \\ 100 \\ 80 \\ 0 \\ 100 \\ 80 \\ 0 \\ 100 \end{bmatrix}$$

Vi har markerat en blockstruktur, block som beskriver samband horisontellt respektive vertikalt. Med $n$ inre punkter i gittrets båda led får vi $n \times n$ stycken block av storleken $n \times n$. Matrisen $\mathbf{A}$ blir av storleken $n^2 \times n^2$. Högerledet $\mathbf{b}$ blir en kolonnvektor med $n^2$ element.

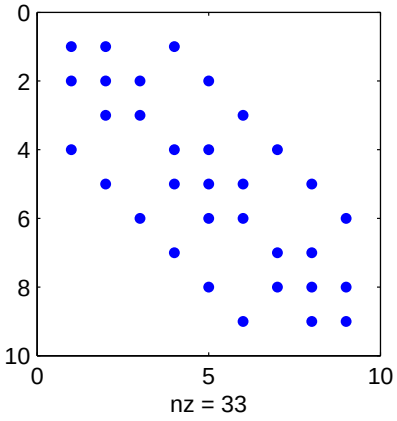
Om vi väljer $n$ större så får vi ett tätare gitter och därmed en bättre approximation av temperaturen på plattan.

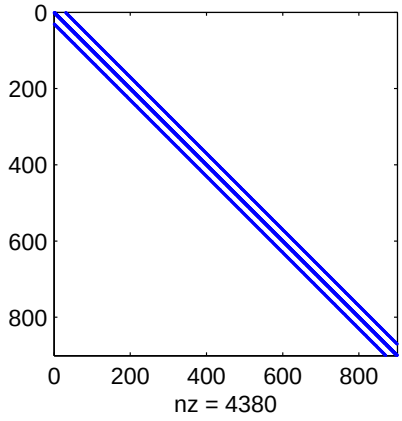
Utgående från strukturen hos matrisen $\mathbf{A}$ och högerledet $\mathbf{b}$ för $n = 3$ kan vi sluta oss till hur $\mathbf{A}$ och $\mathbf{b}$ ser ut för allmännt $n$.

Här är ett skript som fungerar för godtyckligt $n$. Testa genom att sätta $n = 3$ och använda `full` för att se efter att resultatet blir samma som förut.

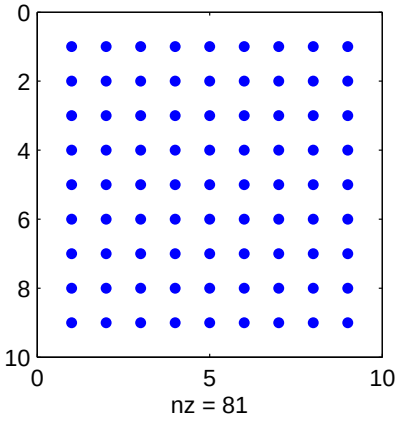
```
n=30; n2=n^2; e=ones(n2,1);
% Bilda A
A=spdiags([-e -e 4*e -e -e],[-n -1 0 1 n],n2,n2);  % Bra början, men vi har fått
for k=n:n:(n-1)*n   % några -1:or där det skall vara 0:or (röda i matrisen ovan)
    A(k,k+1)=0;     % justera diagonal +1
    A(k+1,k)=0;     % justera diagonal -1
end
% Temperaturen på kanterna (nedre, övre, vänstra, högra)
g1=20*ones(n,1); g3=40*ones(n,1); g2=80*ones(n,1); g4=100*ones(n,1);
% Bilda b
b=[g1                % Block med känd temp från nedre kant ovanför
   zeros((n-2)*n,1)  % n-2 block med 0:or och
   g3];              % underst block med känd temp från övre kant
for j=1:n
    j1=(j-1)*n+1;    % Första index för block j och
    jn=j*n;          % sista index för block j
    b(j1)=b(j1)+g2(j);  % Addera i början av varje block och
    b(jn)=b(jn)+g4(j);  % i slutet av varje block
end
% Beräkna u
u=A\b;
```

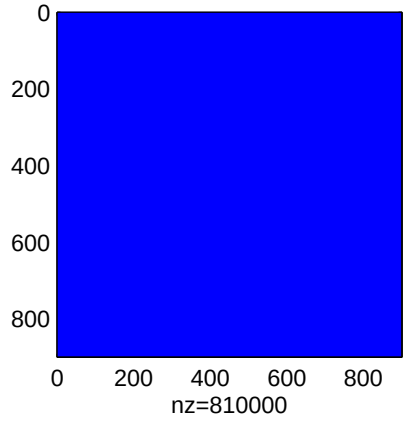
Här ser vi gleshetsstrukturen för $\mathbf{A}$ då $n = 3$, dvs. då $\mathbf{A}$ är en $9 \times 9$-matris, och då $n = 30$, dvs. då $\mathbf{A}$ är en $900 \times 900$-matris. Vi ser att vi har glesa matriser och förstår varför man kallar dem bandmatriser. Utskriften av `nz` ger antal element skilda från noll.




Motsvarande inverser är fyllda matriser. Bara att multiplicera med dem är kostsamt (då $n$ är lite större), för att inte tala om att beräkna dem. Även lagra så mycket data i onödan är dumt.




Här ser vi beräkningstider (sekunder på typisk labdator) för olika $n$.

| Gittertäthet $n$ | 3 | 10 | 20 | 30 | 60 | 100 |
|---|---|---|---|---|---|---|
| Antal obekanta $n^2$ | 9 | 100 | 400 | 900 | 3600 | 10000 |
| Gles `x=A\b` | 0.001 | 0.002 | 0.002 | 0.003 | 0.01 | 0.03 |
| Fylld `x=A\b` | 0.00008 | 0.0004 | 0.008 | 0.03 | 1.2 | 26.3 |
| Invers `x=inv(A)*b` | 0.00008 | 0.0009 | 0.01 | 0.2 | 14.0 | 290 |

Vi ser att när matriserna blir lite större så är det viktigt att använda lämplig teknik för glesa matriser.

Nu skall vi rita upp temperaturen, både som en färgad yta och med s.k. nivåkurvor, som visar lika temperatur (isotermer).

Vi skall se temperaturen som en funktion $U(x, y)$ är två variabler $x$ (läget horisontellt) och $y$ (läget vertikalt), se figuren nedan. Motsvarigheten till en funktionskurva, som vi ritar med `plot`, blir en funktionsyta, som vi ritar med `surf`. Ytans nivåer (tänk höjd över havet) ritar vi med `contour`. (I kursen flervariabelanalys kommer vi i laborationerna använda `surf` och `contourf` en hel del.)

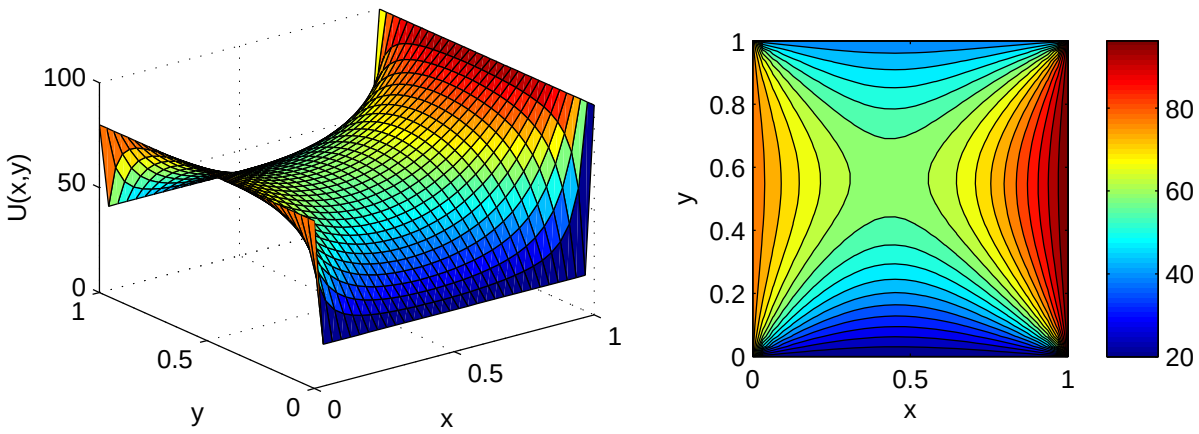
För att rita plattans temperaturer måste vi göra en rekonstruktion. Vi har placerat temperaturen i de olika gitterpunkterna i en kolonnvektor `u` och vi skall bilda en matris `U` för att efterlikna plattan enligt

$$\begin{bmatrix} 80 & 40 & 40 & 40 & 100 \\ 80 & u_7 & u_8 & u_9 & 100 \\ 80 & u_4 & u_5 & u_6 & 100 \\ 80 & u_1 & u_2 & u_3 & 100 \\ 80 & 20 & 20 & 20 & 100 \end{bmatrix}$$

Vi har satt in plattans kanttemperaturer som ram i matrisen och i mitten fyllt på med temperaturen i de olika gitterpunkterna. Så här gör vi i MATLAB

```
% rekonstruktion  -- bilda matris U med u-värden på plattan
U=reshape(u,n,n)';   % Här styckar vi upp u
U=[g2(1) g1' g4(1)
   g2    U   g4
   g2(n) g3' g4(n)];
% plottning -- rita upp matrisen U
L=1; h=L/(n+1); x=0:h:L; y=0:h:L;
figure(1)
surf(x,y,U)
xlabel('x'), ylabel('y'), zlabel('U(x,y)')
figure(2)
contourf(x,y,U,20)
axis equal, axis tight, colorbar
xlabel('x'), ylabel('y')
```

Här ser vi resultatet av den beräkning där vi tog $n = 30$.




Vi ser hög temperaturen som rött och låg temperatur som blått. Isotermerna ser vi till höger. Stapeln ger en översättning mellan färg och temperatur.

# 1 Målsättning

Avsikten med denna laborationen är att fortsätta med linjära ekvationssystem och matriser. Vi ser på hantering och uppbyggnad av matriser samt operationer på matriser. Därefter ser vi på linjära ekvationssystem, dels med små koefficientmatriser men också med stora och glesa koefficientmatriser som ofta uppstår i tekniska tillämpningar.

# 2 Kommentarer och förklaringar

Matris-vektorprodukt samt matris-matrisprodukt gör man givetvis normalt med den i MATLAB inbyggda matrisprodukten (*). Den egna programmeringen vi gjorde 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 hantera matriser
- utföra de vanliga matrisoperationerna
- bygga stora, glesa matriser med `sparse` och `spdiags`