Mer om linjära ekvationssystem

1 Inledning

Denna studioövning fortsätter med linjära ekvationssystem och matriser, som vi först tittade på i studioövning 7 i förra läsperioden. 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 förra 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 2×2, i partitioneringen (mer om detta finns i Lay kapitel 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

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

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.

Uppgift 6. En liten stökiometriuppgift. Vid förbränning av propan (C_3H_8) gäller följande kemiska reaktionsformel (obalanserad)

$$(x_1)\,C_3H_8 + (x_2)\,O_2 \to (x_3)\,CO_2 + (x_4)\,H_2O$$

För att balansera formeln representerar vi C_3H_8 med vektorn $(3, 8, 0)$, O_2 med vektorn $(0, 0, 2)$, osv. På detta sätt kan den obalanserade ekvationen skrivas

$$\begin{bmatrix} 3 \\ 8 \\ 0 \end{bmatrix} x_1 + \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} x_2 - \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} x_3 - \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} x_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Lösningen $\mathbf{x} = (1, 5, 3, 4)$ (den minimala heltalslösningen) ger den balanserade reaktionsformeln

$$C_3H_8 + 5\,O_2 \to 3\,CO_2 + 4\,H_2O$$

(a). Balansera nu den kemiska formeln

$$MnS + As_2Cr_{10}O_{35} + H_2SO_4 \to HMnO_2 + AsH_3 + CrS_3O_{12} + H_2O$$

som är av intresse bl.a. vid framställning av arsin (AsH_3).

Med `format rat` skrivs beräkningsresultat med rationella tal och det blir enklare att tolka svaret. Standardformatet får vi sedan tillbaka med `format short`.

(b). Skriv ned den balanserade formeln på papper.

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. Nu var detta ingen stor matris, vi villebara visa principerna för hur man bildar en gles matris med `sparse`.

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 på ett mycket effektivt sätt beräkna lösningen.

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. I nästa avsnitt skall vi använda matrisen ovan.

7 Randvärdesproblem för differentialekvationer

Vi skall se på *randvärdesproblem* för differentialekvation av typen

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

Oftast går det inte att ge en användbar formel för lösningen, utan vi måste använda beräkningsmetoder för att bestämma en numerisk lösning.

Det finns olika beräkningsmetoder, gemensamt för dem är att de leder till lösning av ett icke-linjärt ekvationssystem eller ett linjärt ekvationssystem i det fall f är linjär i u och u'.

Vi skall i nästa läsperiod se på hur man löser system av icke-linjära ekvationer. För tillfället kommer vi nöja oss med att se på *linjära* randvärdesproblem, dvs. sådana problem som kan skrivas

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

Vanligtvis måste vi beräkna en numerisk lösning, men om a och b är konstanter, dvs. inte beror av x, så finns det som ni vet formler för lösningarna.

Vi använder differensmetoden för att lösa problemet

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

gör vi en likformig indelning av intervallet $a \le x \le b$ i ett nät

$$a = x_0 < x_1 < \cdots < x_{i-1} < x_i < x_{i+1} < \cdots < x_n < x_{n+1} = b$$

där $x_i = a + ih, i = 0, 1, \cdots, n+1$ med $h = \frac{b-a}{n+1}$.

Vi ersätter $u''(x_i)$ och $u'(x_i)$ med s.k. centraldifferenskvoter

$$u''(x_i) \approx D_+D_-u(x_i) = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}, \qquad u'(x_i) \approx D_0u(x_i) = \frac{u_{i+1} - u_{i-1}}{2h}$$

i de inre nätpunkterna $x_i, i = 1, \cdots, n$, och får ekvationssystemet

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + a(x_i)\frac{u_{i+1} - u_{i-1}}{2h} + b(x_i)u_i = f(x_i),\ i = 1, 2, \cdots, n$$

Låter vi u_i beteckna en approximation av $u(x_i)$ så resulterar detta i ett linjärt ekvationssystem.

Som exempel tar vi: *Stationär värmeledning* i en isolerad stav med en värmekälla. Temperaturen $u(x)$ beskrivs av randvärdesproblemet

$$\begin{cases} -\kappa u''(x) = f(x),\ 0 \le x \le L \\ u(0) = u_0,\ u(L) = u_L \end{cases}$$

där u_0 och u_L är temperaturen i stavens ändpunkter, κ är stavens värmeledningsförmåga och $f(x)$ är värmekällan.

Inför nätet $x_i = ih$, $i = 0, 1, \cdots, n+1$ med $h = \frac{L}{n+1}$ på $0 \leq x \leq L$ och låt u_i beteckna approximationer av $u(x_i)$, $i = 0, 1, \cdots, n+1$.

Ersätt $u''(x_i)$ med centraldifferenskvoter i de inre nätpunkterna $x_i, i = 1, \cdots, n$, och vi får

$$-u_{i+1} + 2u_i - u_{i-1} = \frac{h^2}{\kappa} f(x_i) \quad i = 1, 2, \cdots, n$$

Med matriser kan detta skrivas

$$\mathbf{Au} = \mathbf{b}$$

där

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{h^2}{\kappa} f(x_1) + u_0 \\ \frac{h^2}{\kappa} f(x_2) \\ \vdots \\ \frac{h^2}{\kappa} f(x_{n-1}) \\ \frac{h^2}{\kappa} f(x_n) + u_L \end{bmatrix}$$

Matrisen $\mathbf{A}$ är precis den bandmatris vi såg på i förra avsnittet, När vi har bildat högerledsvektorn $\mathbf{b}$ löser vi med backslash (\). Eftersom 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 $\kappa = 2$, $L = 1$, $u_0 = 40$, $u_L = 60$ samt $f(x) = 200\mathrm{e}^{-(x-\frac{L}{2})^2}$. Lös värmeledningsproblemet och rita upp lösningen. Tag $n = 30$.