1.1 MATLABs kommandon för matriser

Det finns en mängd kommandon för att hantera vektorer, matriser och linjära ekvationssystem. Vi ger här en kort sammanfattning av dessa kommandon. För en mera detaljerad diskussion se Eva Pärt-Enander kapitel 3 och 4.

1. Generation av en radvektor $\mathbf{x} = (1, 2, 3)$

   ```
   x=[1 2 3]
   ```

2. Generation av en radvektor $\mathbf{x} = (0, 0, 0, ..., 0)$ med n element

   ```
   x=zeros(1,n)
   ```

3. Generation av en radvektor $\mathbf{x} = (1, 1, 1, ..., 1)$ med n element

   ```
   x=ones(1,n)
   ```

4. Generation av en kolonnvektor $\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

   ```
   x=[1; 2; 3]
   ```

5. Generation av en kolonnvektor $\mathbf{x} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ med n element

   ```
   x=zeros(100,1)
   ```

6. Generation av en kolonnvektor $\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$ med n element

   ```
   x=ones(100,1)
   ```

7. Generation av en matris $\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 5 \\ 6 & 3 & 3 \end{pmatrix}$

   ```
   A=[1 2 3; 4 1 5; 6 3 3]
   ```

8. Generation av en $m \times n$ matris $\mathbf{A}$ med bara nollor.

   ```
   A=zeros(m,n);
   ```

9. Generation av en $m \times n$ matris $\mathbf{A}$ med bara ettor.

   ```
   A=ones(m,n);
   ```

10. Transponat $\mathbf{A}^T$ (rader och kolonner byter plats)

    ```
    A'
    ```

11. Matrisinvers $\mathbf{A}^{-1}$

```
inv(A)
```

12. Lösning till $\mathbf{Ax} = \mathbf{b}$ med hjälp av Gauss-elimination.

```
x=A\b
```

Observera att snedsträcket lutar åt vänster!
Vi illustrerar användandet av dessa kommandon med två exempel.

Exempel 1. Bestäm transponatet till

$$\mathbf{A} = \begin{pmatrix} 1 & 3 & 5 \\ 4 & 1 & 5 \end{pmatrix}$$

Vi börjar med att definiera matrisen

```
A = [1 3 5; 4 1 5];
```

Då vi skriver

```
A'
```

svarar MATLAB med

```
ans =

     1     4
     3     1
     5     5
```

Exempel 2. För att lösa ekvationssystemet

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 1 \\ x_1 + 2x_2 + 3x_3 &= 2 \\ 2x_1 - x_2 + 4x_3 &= 3 \end{aligned}$$

börjar vi med att definiera koefficientmatrisen och högerledet

```
A = [1 2 -1; 1 2 3; 2 -1 4];
b = [1; 2; 3];
```

Vi skriver sedan

```
x=A\b
```

varvid MATLAB svarar

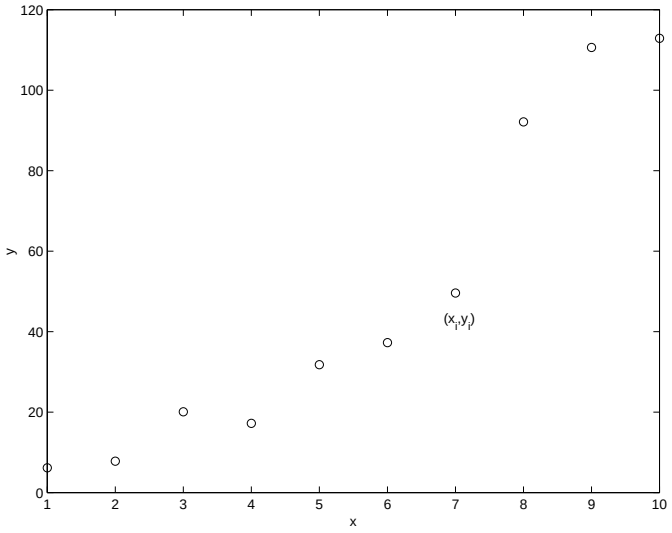
```
x =

   1.05000000000000
   0.10000000000000
   0.25000000000000
```

Om man använder snedsträck åt höger och skriver

```
x=A/b
```

får man felmeddelandet

```
??? Error using ==> /
Matrix dimensions must agree.
```



Figur 1: Mätvärde till vilka en funktion skall anpassas.

2 Funktionsanpassningar

I tillämpningar inom naturvetenskap och teknik behöver man ofta anpassa en funktion $f(x)$ till ett antal mätvärden (x_1, y_1), (x_2, y_2),..., (x_m, y_m). Allmänt kan vi låta anpassningsfunktionen ha formen

$$f(x) = c_1\varphi_1(x) + c_2\varphi_2(x) + ... + c_n\varphi_n(x)$$

där $\varphi_1(x), \varphi_2(x), ..., \varphi_n(x)$ är givna så kallade *basfunktioner*. Problemet blir då att bestämma koefficienterna $c_1, c_2, ..., c_n$ så att anpassningen i någon mening blir så bra som möjligt. Kravet att anpassningsfunktionen $f(x)$ skall gå genom alla mätvärdena leder till ett ekvationssystem av formen

$$\begin{cases} \varphi_1(x_1)c_1 + \varphi_2(x_1)c_2 + \ldots + \varphi_n(x_1)c_n = y_1 \\ \varphi_1(x_2)c_1 + \varphi_2(x_2)c_2 + \ldots + \varphi_n(x_2)c_n = y_2 \\ \qquad\qquad \ldots \\ \varphi_1(x_m)c_1 + \varphi_2(x_m)c_2 + \ldots + \varphi_n(x_m)c_n = y_m \end{cases} \tag{1}$$

Systemet ovan skrivs ofta på matrisform och vi har

$$\mathbf{Ac} = \mathbf{y},$$

där $\mathbf{A}$ är en $m \times n$ koefficientmatris med element $a_{ij} = \varphi_j(x_i)$, $\mathbf{c}$ en $n \times 1$ kolonnvektor och $\mathbf{y}$ en $m \times 1$ kolonnvektor

$$\mathbf{A} = \begin{pmatrix} \varphi_1(x_1) & \varphi_2(x_1) & ... & \varphi_n(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & ... & \varphi_n(x_2) \\ \vdots & & & \\ \varphi_1(x_m) & \varphi_2(x_m) & ... & \varphi_n(x_m) \end{pmatrix} \qquad \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_m \end{pmatrix}.$$

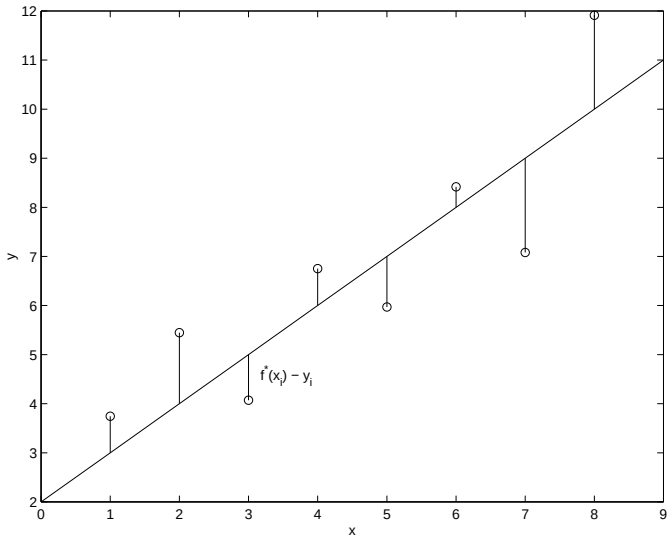
Normalt gäller att $m > n$, det vill säga fler ekvationer än obekanta, och vi har ett *överbestämt* system som oftast saknar lösning.

2.1 Minsta-kvadratanpassningar

Vid minsta-kvadratanpassningar frångår man kravet att anpassningsfunktionen skall gå genom alla datapunkterna. Istället bestämmer man koefficienterna för basfunktionerna genom att minimera det kvadratiska avståndet mellan datapunkterna och anpassningsfunktionen $f(x)$, dvs vi ska bestämma c_1, c_2,...,c_n så att den kvadratiska summan

$$D = \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

blir så liten som möjligt. Minsta kvadrat-anpassning av en rät linje till mätdata är illustreras i figur 2.



Figur 2: Minsta-kvadratanpassning till en rät linje.

Man kan visa (se Gunnar Sparr Linjär algebra sidan 156) att de koefficienter $c_1, c_2, ..., c_n$ som minimerar den kvadratiska summan ovan fås som lösning till den så kallade *normalekvationen*

$$\mathbf{A}^T\mathbf{A}\mathbf{c} = \mathbf{A}^T\mathbf{y}.$$

Normalekvationen har alltid en entydig lösning.

2.2 MATLABs kommando för minsta-kvadratanpassningar

Vid minsta-kvadratanpassningar med MATLAB startar man med att mata in koefficientmatrisen $\mathbf{A}$ och kolonnvektorn $\mathbf{y}$ som hör till det överbestämda ekvationssystemet (1). Då man ger kommandot

```
c = A\y
```

känner MATLAB själv av att systemet är överbestämt och löser automatiskt normalekvationen

$$\mathbf{A}^T\mathbf{A}\mathbf{c} = \mathbf{A}^T\mathbf{y}.$$

Syntaxen för att lösa ett ekvationssystem i vanlig mening och i minsta-kvadratmening är alltså densamma. Det som styr är dimensionen på matrisen $\mathbf{A}$. Om matrisen är kvadratisk så löses ekvationsystemet med Gausselimination om matrisen har fler rader än kolonner konstrueras och löses normalekvationen.

2.3 Polynomanpassningar

Utifrån problemets natur och den bakomliggande teorin vet man ofta vilka basfunktioner man skall anpassa till mätdata. Vid *polynomanpassningar* sätter man $\varphi_1(x) = 1$, $\varphi_2(x) = x$, $\varphi_3(x) = x^2$,...,$\varphi_n(x) = x^{n-1}$, varvid anpassningsfunktionen får formen

$$f(x) = c_1 + c_2x + c_3x^2 + ... + c_nx^{n-1}.$$

Motsvarande koefficientmatris blir då (jämför ekvation (1) på sidan 4)

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ & & \dots & & \\ & & \dots & & \\ 1 & x_m & x_m^2 & \dots & x_m^{n-1} \end{pmatrix}$$

Vi illustrerar polynomanpassningar med ett exempel.

Exempel 1. Vi ska anpassa ett andragradspolynom

$$f(x) = c_1 + c_2x + c_3x^2$$

till mätvärdena

x_k	0.0	1.0	2.0	3.0
y_k	0.01	0.91	4.00	8.12

Det överbestämda ekvationssystemet har formen

$$\mathbf{Ac} = \mathbf{y}$$

där koefficientmatrisen $\mathbf{A}$ och högerledet $\mathbf{y}$ ges av

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Vi börjar med definiera x_k och y_k som kolonnvektorer

```
x = [0; 1; 2; 3];
y = [0.01; 0.91; 4.00; 8.12];
```

Koefficientmatrisen $\mathbf{A}$ genereras sedan bekvämt med kommandot

```
A = [x.^0 x.^1 x.^2];
```

Då vi ger kommandot

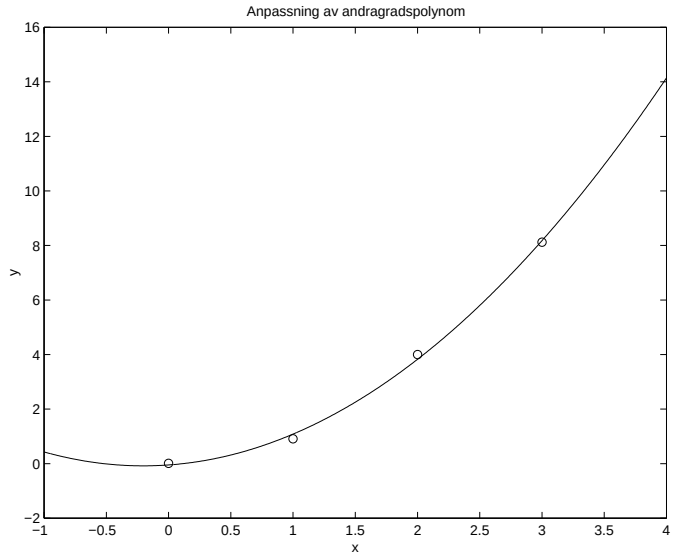
```
c = A\y
```

svarar MATLAB med

```
c =

   -0.0480
    0.3270
    0.8050
```

Anpassningsfunktionen har alltså formen

$$f(x) = -0.0480 + 0.3270\,x + 0.8050\,x^2.$$



Figur 3: Anpassning av ett andragradspolynom

För att plotta mätvärdena och den anpassade funktionen skriver vi

```
plot(x,y,'o')              % Plottar de mätvärdena med cirklar
hold on                    % Lås grafikfönstret
x=0:0.01:3;                % Generera en vektor x med element från 0 till 3
y=c(1)+c(2)*x+c(3)*x.^2;   % Beräkna funktionsvärdena för anpassningsfunktionen
plot(x,y)                  % Plotta anpassningsfunktionen
title('Anpassning av andragradspolynom')  % Skriv rubrik
xlabel('x')                % Skriv text på x-axeln
ylabel('y')                % Skriv text på y-axeln
```

varvid vi får figur 3.

2.4 Allmänna funktionsanpassningar

För att få ökad insikt om minsta-kvadratanpassningar och deras möjligheter så tar vi nu ett exempel på en anpassning där basfunktionerna $\varphi_1(x)$, $\varphi_2(x), ..., \varphi_n(x)$ inte är polynom.

Exempel 1. Vi ska anpassa en kombination (superposition) av sinus och cosinus funktioner

$$u(t) = c_1 \sin t + c_2 \cos t$$

till de 13 mätpunkterna

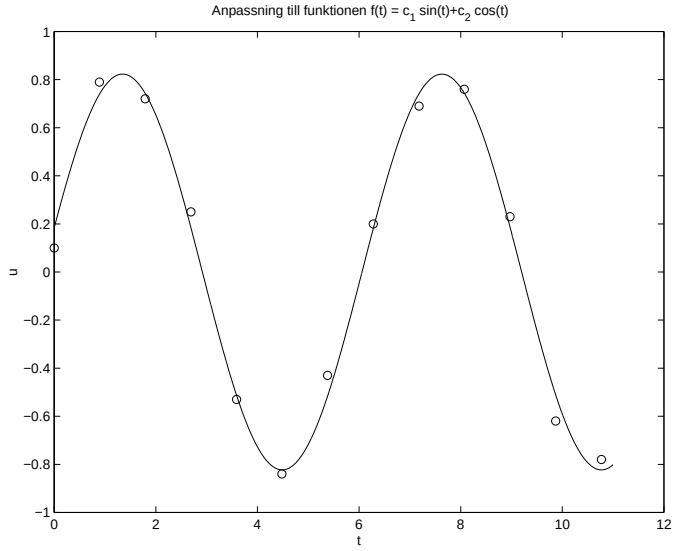
t_k	0.00	0.89	1.79	2.69	3.59	4.48	5.38	6.28	7.18	8.07	8.97	9.87	10.77
u_k	0.10	0.79	0.72	0.25	−0.53	−0.84	−0.43	0.20	0.69	0.76	0.23	−0.62	−0.78

Det överbestämda ekvationssystemet har formen

$$\mathbf{Ac} = \mathbf{u}$$

där koefficientmatrisen $\mathbf{A}$ och högerledet $\mathbf{u}$ ges av

$$\mathbf{A} = \begin{pmatrix} \sin(t_1) & \cos(t_1) \\ \sin(t_2) & \cos(t_2) \\ \cdots & \\ \sin(t_{13}) & \cos(t_{13}) \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \cdots \\ u_{13} \end{pmatrix}$$



Figur 4: Anpassning till funktionen $u(t) = c_1 \sin(t) + c_2 \cos(t)$

Vi börjar med definiera t_k och u_k som kolonnvektorer

```
t=[0.00; 0.89; 1.79; 2.69; 3.59; 4.48; 5.38; 6.28; 7.18; 8.07; 8.97; 9.87; 10.77];
u=[0.10; 0.79; 0.72; 0.25; -0.53; -0.84; -0.43; 0.20; 0.69; 0.76; 0.23; -0.62; -0.78];
```

Koefficientmatrisen **A** genereras sedan med kommandot

```
A = [sin(t) cos(t)];
```

Då vi ger kommandot

```
c = A\u
```

svarar MATLAB med

```
c =

    0.8023
    0.1838
```

Anpassningsfunktionen har alltså formen

$u(t) = 0.8023 \sin(t) + 0.1838 \cos(t)$.

För att plotta mätvärdena och den anpassade funktionen skriver vi

```
plot(t,u,'o')               % Plotta mätvärdena med cirklar
hold on                     % Lås grafikfönstret
t=0:0.01:11;                % Generera en vektor med element från 0 till 11
u=c(1)*sin(t)+c(2)*cos(t);  % Beräkna funktionsvärdena för anpassningsfunktionen
plot(t,u)                   % Plotta anpassningsfunktionen
title('Anpassning till funktionen f(t) = c_1sin(t)+c_2cos(t)')  % Skriv rubrik
xlabel('t')                 % Skriv text på x-axeln
ylabel('u')                 % Skriv text på y-axeln
```

varvid vi får figur 4.

2.5 Anpassningar till exponentialuttryck

I naturen träffar man ofta på kvantiteter som är exponentiellt avtagande (eller växande). Ett exempel är strålningsintensiteten hos ett radioaktivt preparat, vilken avtar med tiden enligt formeln

$$I(t) = I(0)e^{-\lambda t},$$

där λ är sönderfallskonstanten och $I(0)$ den ursprungliga intensiteten. För att kunna anpassa exponentialfunktionen till de experimentella data, dvs bestämma $I(0)$ och λ, måste man logaritmera exponentialuttrycket

$$\ln I(t) = \ln I(0)e^{-\lambda t} = \ln I(0) + \ln e^{-\lambda t} = \ln I(0) - \lambda t = c_1 + c_2 t.$$

Koefficienterna $c_1 = \ln I(0)$ och $c_2 = -\lambda$ bestämmes genom att minsta-kvadratanpassa

$$\ln I(t) = c_1 + c_2 t$$

till de logaritmerade datapunkterna $\ln I_k$.

Exempel Anpassa en exponentialfunktion

$$I(t) = I(0)e^{-\lambda t}$$

till följande sju datapunkter

t_k	0.2	0.3	0.4	0.5	0.6	0.7	0.8
I_k	3.16	2.38	1.75	1.34	1.00	0.74	0.56

Vi omformulerar problemet och anpassar

$$\ln I(t) = \ln I(0) - \lambda t = c_1 + c_2 t$$

till de logaritmerade datapunkterna $b_k = \ln I_k$. Motsvarande överbestämda ekvationssystem har formen

$$\mathbf{Ac} = \mathbf{b}$$

där koefficientmatrisen $\mathbf{A}$ och högerledet $\mathbf{b}$ ges av

$$\mathbf{A} = \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \dots & \\ 1 & t_7 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} \ln I_1 \\ \ln I_2 \\ \dots \\ \ln I_7 \end{pmatrix}$$

Vi börjar med att mata in kolonnvektorerna $\mathbf{t}$ och $\mathbf{I}$ och definiera $\mathbf{b}$

```
t = [0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8];
I = [3.16; 2.38; 1.75; 1.34; 1.00; 0.74; 0.56];
b = log(I);
```

Koefficientmatrisen $\mathbf{A}$ genereras sedan med kommandot

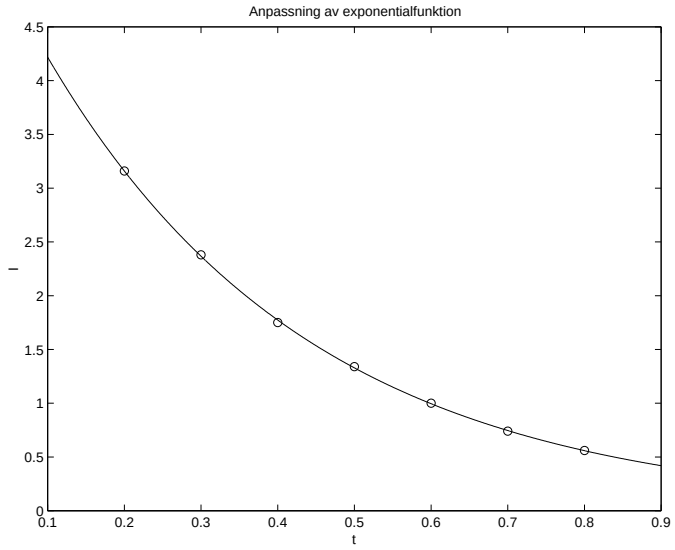
```
A = [t.^0 t];
```

Då vi ger kommandot

```
c = A\b
```

svarar MATLAB med

```
c =

    1.7283
   -2.8883
```



Figur 5: Anpassning av exponentialfunktion

Vi har alltså

$$\ln I(t) = 1.7283 - 2.8883t$$

och de ursprungliga data kommer alltså att beskrivas av funktionen

$$I(t) = e^{1.7283-2.8883t} = e^{1.7283}e^{-2.8883t} = 5.6310\, e^{-2.8883t}$$

För att plotta de ursprungliga datapunkterna och den anpassade exponentialfunktionen skriver vi

```
plot(t,I,'o')          % Plotta mätvärdena med cirklar
hold on                % Lås grafikfönstret
t = 0.1:0.001:0.9;     % Generera en vektor med element från 0.1 till 0.9
I = exp(c(1)+c(2)*t); % Beräkna funktionsvärdena för anpassningsfunktionen
plot(t,I)              % Plotta anpassningsfunktionen
title('Anpassning av exponentialfunktion')   % Skriv rubrik
xlabel('t')            % Skriv text på x-axeln
ylabel('I')            % Skriv text på y-axeln
```

varvid vi får figur 5.