Minste kvadraters metode i MATLAB og LabVIEW

Av Finn Haugen (finn@techteach.no)
TechTeach (http://techteach.no)

22.12 2002

Innhold

Forord

Dette dokumentet beskriver funksjoner for minste kvadraters metode i MATLAB [1] og LabVIEW [2]. Dokumentet utgjør tilleggsmateriale til læreboka *Dynamiske systemer – modellering, analyse og simulering* [3]. Dette dokumentet og filer som benyttes eller som det henvises til i dokumentet, kan lastes ned via hjemmesiden for læreboka på **http://techteach.no** (benytt brukernavn dynsystbruk og passord dynpass ifm. nedlastingen).

Skien, desember 2002

Finn Haugen

Kapittel 1

Minste kvadraters metode i MATLAB

Gitt det lineære likningssystemet

$$\underline{y} = \Phi \underline{\theta} \tag{1.1}$$

der *utgangsvektoren* $\underline{y}$ er en kjent vektor med m elementer og Φ, som gjerne kalles *regresjonsmatrisen*, er en kjent matrise med m rader og n kolonner. $\underline{\theta}$ er en ukjent vektor med n elementer, som skal beregnes.[1] (1.1) representerer altså et lineært likningssystem med m likninger for n ukjente θ-elementer:

$$\underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_{\underline{y}} = \underbrace{\begin{bmatrix} \varphi_{11} & \cdots & \varphi_{1n} \\ \vdots & \ddots & \vdots \\ \varphi_{m1} & \cdots & \varphi_{mn} \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}}_{\underline{\theta}} \tag{1.2}$$

Et analytisk uttrykk for minste kvadraters-løsningen av (1.1) er

$$\underline{\theta}_{\mathrm{LS}} = (\Phi^T \Phi)^{-1} \Phi^T \underline{y} \tag{1.3}$$

Det er enkelt å implementere LS-løsningen (1.3) i MATLAB. Vi kan programmere (1.3) direkte eller bruke slash-operatoren "\". Begge mulighetene demonstreres nedenfor.

[1] En annen nokså vanlig måte å skrive (1.1) på, er $\underline{b} = A\underline{x}$.

Som konkret eksempel skal vi ut fra 3 gitte sammenhørende sett av data for z og x beregne LS-løsningen av parametrene c og d i modellen

$$z = xc + d \tag{1.4}$$

Datasettene er $z = [0,8, \quad 3,0, \quad 4,0]$ og $x = [1, \quad 2, \quad 3]$. Innsetting av dette i modellen gir

$$\begin{aligned} 0,8 &= 1c + d & (1.5)\\ 3,0 &= 2c + d & (1.6)\\ 4,0 &= 3c + d & (1.7) \end{aligned}$$

som kan skrives på formen

$$\underbrace{\begin{bmatrix} 0,8 \\ 3,0 \\ 4,0 \end{bmatrix}}_{\underline{y}} = \underbrace{\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} c \\ d \end{bmatrix}}_{\underline{\theta}} \tag{1.8}$$

$$= \underbrace{\begin{bmatrix} \underline{\varphi_1} \\ \underline{\varphi_2} \\ \underline{\varphi_3} \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} c \\ d \end{bmatrix}}_{\underline{\theta}} \tag{1.9}$$

som utgjør regresjonsmodellen. $\underline{\theta}$ beregnes fra (1.3):

$$\begin{aligned} \underline{\theta}_{\mathrm{LS}} &= (\Phi^T\Phi)^{-1}\Phi^T\underline{y} & (1.10)\\ &= \left(\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0,8 \\ 3,0 \\ 4,0 \end{bmatrix} & (1.11)\\ &= \begin{bmatrix} 1,6 \\ -0,6 \end{bmatrix} = \begin{bmatrix} c_{\mathrm{LS}} \\ d_{\mathrm{LS}} \end{bmatrix} & (1.12) \end{aligned}$$

Implementering av (1.3) direkte

MATLAB-skriptet `ls_intro.m` nedenfor viser hvordan (1.3) kan implementeres i MATLAB.

Bruk av slash-operatoren "\"

Gitt likningssystemet (1.1). Løsningen θ kan beregnes med minste kvadraters metode, men *en måte å skrive* beregningen av løsningen på, er

$$\theta \simeq \Phi^{-1}y = \Phi \backslash y \tag{1.13}$$

MATLAB-skriptet ls_intro.m nedenfor viser hvordan (1.13) kan implementeres i MATLAB.

```
%MATLAB-skript ls_intro.m
Phi=[1, 1
2, 1
3, 1];
y=[0.8
3
4];
theta_lsformel=inv(Phi'*Phi)*Phi'*y %Egenprogrammert LS
theta_slash=Phi\y %Innebygd LS-funksjon
```

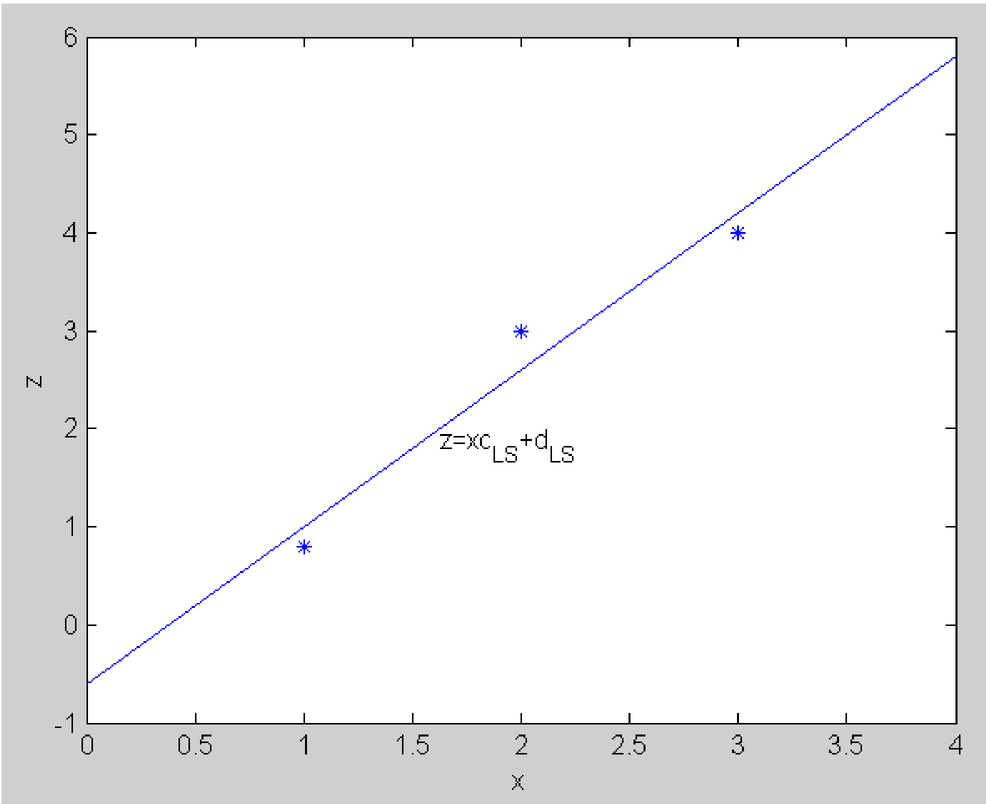
Resultatet av skriptet er

```
theta_lsformel =
1.6000
-0.6000
theta_slash =
1.6000
-0.6000
```

(De to metodene for implementering av minste kvadraters metode gir samme resultat, selvsagt.) Figur 1.1 viser de tre gitte observasjonene (punktene) og linjen

$$z = xc_{\mathrm{LS}} + d_{\mathrm{LS}} \tag{1.14}$$

der $c_{\mathrm{LS}} = \theta_{1_{\mathrm{LS}}} = 1,6$ og $d_{\mathrm{LS}} = \theta_{2_{\mathrm{LS}}} = -0,6$.



Figur 1.1: De tre observasjonene og modellen $z = xc_{\mathrm{LS}} + d_{\mathrm{LS}}$ der $c_{\mathrm{LS}} = \theta_{1_{\mathrm{LS}}} = 1,6$ og $d_{\mathrm{LS}} = \theta_{2_{\mathrm{LS}}} = -0,6$.

Kapittel 2

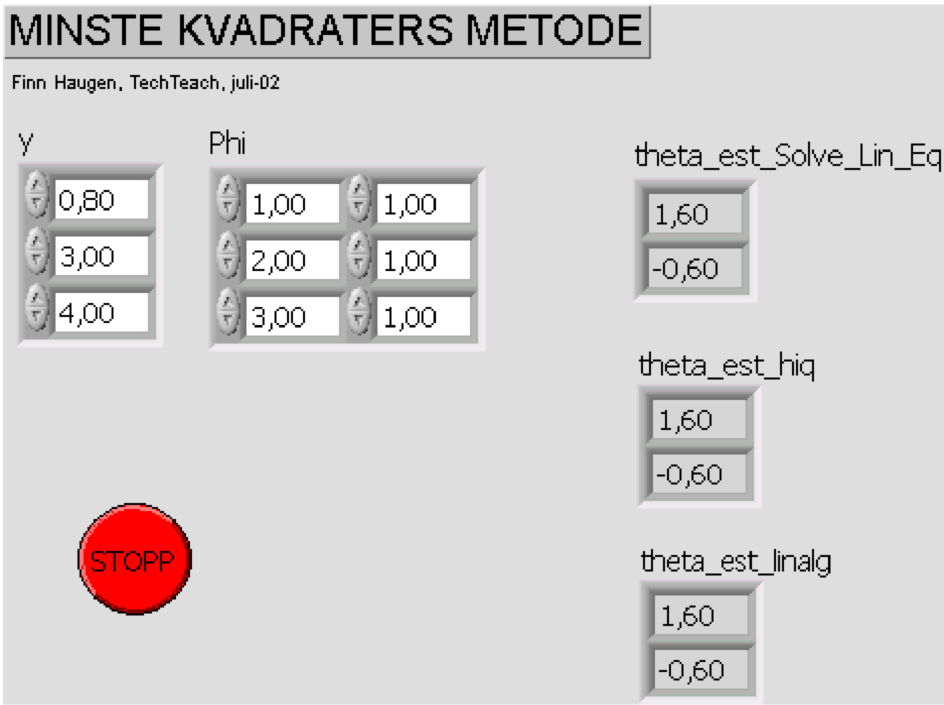
Minste kvadraters metode i LabVIEW

Vi kan beregne minste kvadraters løsningen av (1.1) i LabVIEW på minst 3 forskjellige måter:

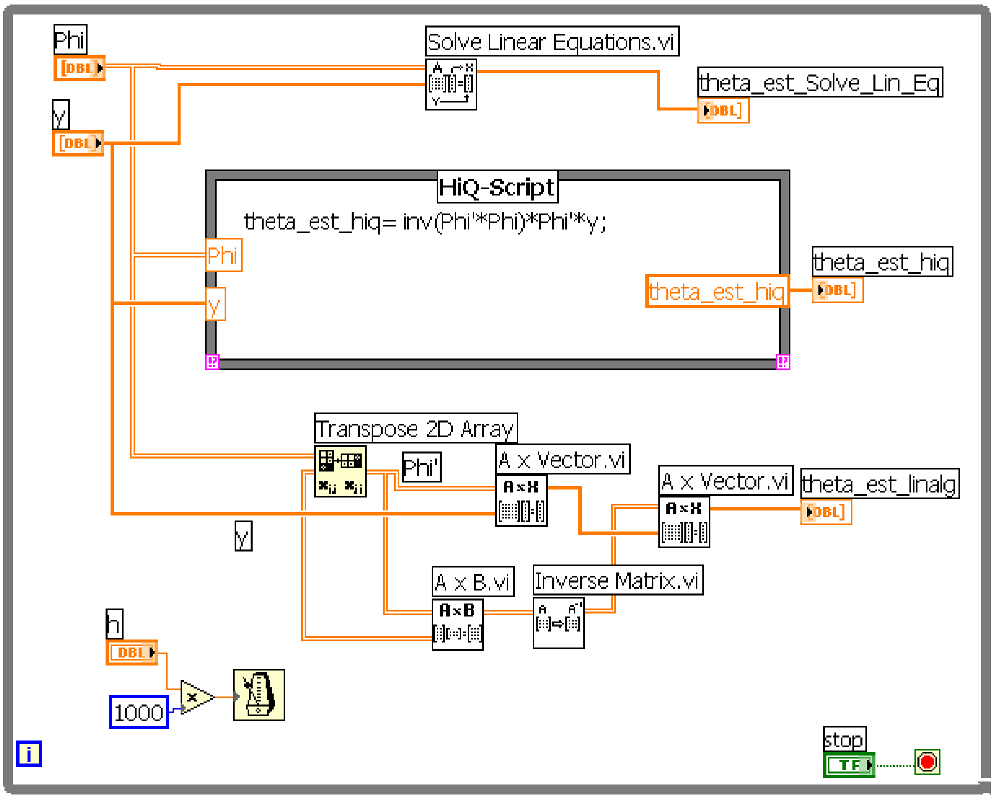
- Med funksjonen Solve Linear Equations
- Vi kan programmere (1.3) i HiQ
- Vi kan programmere (1.3) i vha. LabVIEWs innebygde blokkbaserte lineær algebra-funksjoner for matrise-vektor-beregninger, men programmeringen blir litt tungvinn og diagrammet uoversiktlig.

(Vi kan tenke oss å bruke Formula Node også, men det er neppe hensiktsmessig siden Formula Node ikke støtter matrise-vektor-uttrykk.)

De tre metodene beskrevet ovenfor er implementert i et LabVIEW-program, hvis frontpanel er vist i figur 2.1 og diagram er vist i figur 2.2. LabVIEW-programmet beregner LS-løsningen for eksempelet beskrevet på s. 7.



Figur 2.1: Frontpanelet for et LabVIEW-program som implementerer minste k-vadraters metode på 3 måter. (Fil: ls_intro.vi)



Figur 2.2: Diagrammet for et LabVIEW-program som implementerer minste k-vadraters metode på 3 måter. (Fil: ls_intro.vi)

Bibliografi

[1] Finn Haugen: **Lær MATLAB trinn for trinn**, Tapir Akademisk Forlag, 2003

[2] Finn Haugen: **Lær LabVIEW trinn for trinn**, Tapir Akademisk Forlag, 2003

[3] Finn Haugen: **Dynamiske systemer - modellering, analyse og simulering**, Tapir Akademisk Forlag, 2003