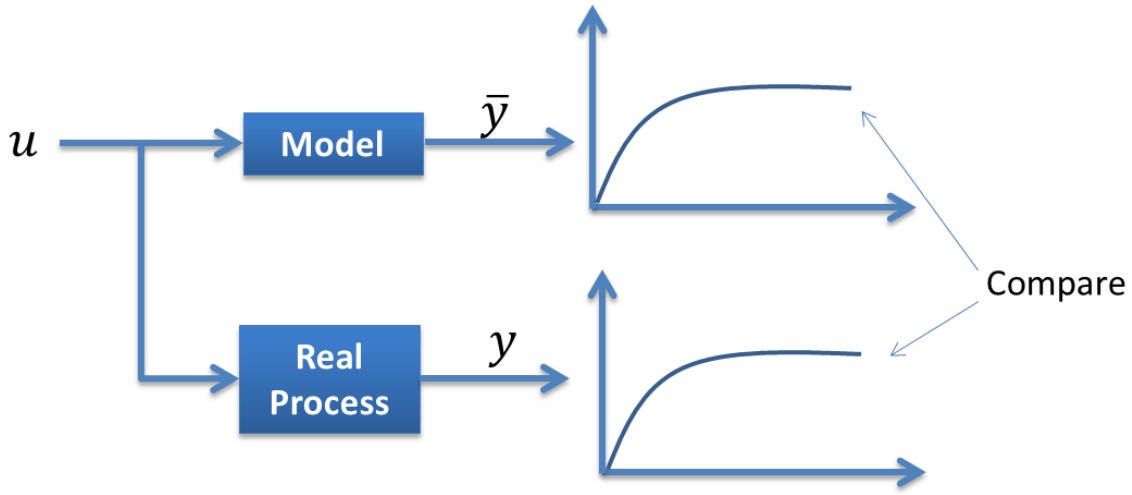
University College of Southeast Norway

Lecture Notes

Systemidentifikasjon

HANS-PETTER HALVORSEN



http://home.hit.no/~hansha

Forord

Dette dokumentet brukes som forelesningsnotater i modellbasert regulering over temaet systemidentifikasjon. Noen forenklinger er gjort underveis da det legges vekt på praktisk forståelse og implementering. Notatet tar for seg praktisk bruk av systemidentifikasjon og hvordan dette kan implementeres i LabVIEW (og MathScript).

Hva er systemidentifikasjon?

Systemidentifikasjon gjør det mulig å lage matematiske modeller av et dynamisk system basert på målinger av inngang- og utgangssignal. Dynamisk betyr at innganger, tilstander og utganger for signalet endrer seg over tid.

Hva kan systemidentifikasjon brukes til?

Disse modellene kan så brukes til regulering av systemet som vi har funnet en modell av eller til prediksjon og tilstandsestimering.

Hvordan kan man identifisere et system?

Enkelt sagt gjør vi dette ved å velge en eller annen form for modell av systemet, og så tilpasser man parametrene i modellen slik at denne passer best mulig til det fysiske systemet. Modellen kan være en matematisk modell basert på en fysisk beskrivelse av prosessen eller en såkalt black-box modell.

Innholdsfortegnelse

1 Innledning

1.1 Hva er Systemidentifikasjon?

Metoder innen Systemidentifikasjon brukes til finne matematiske modeller av dynamiske systemer basert på observasjoner av målte inngangssignaler (u) og utgangssignaler fra systemet (y).

Vi vil gå gjennom 2 hovedtyper av metoder ifm systemidentifikasjon:

- Parameterestimering basert på at man har utviklet en matematisk modell vha fysikkens lover og man ønsker å finne de ukjente modellparametrene. Her vil vi bruke Minste kvadraters metode som et eksempel.
- Blackbox/ Subspace-metoder: Systemidentifikasjon basert på at man ikke har en matematisk modell tilgjengelig.

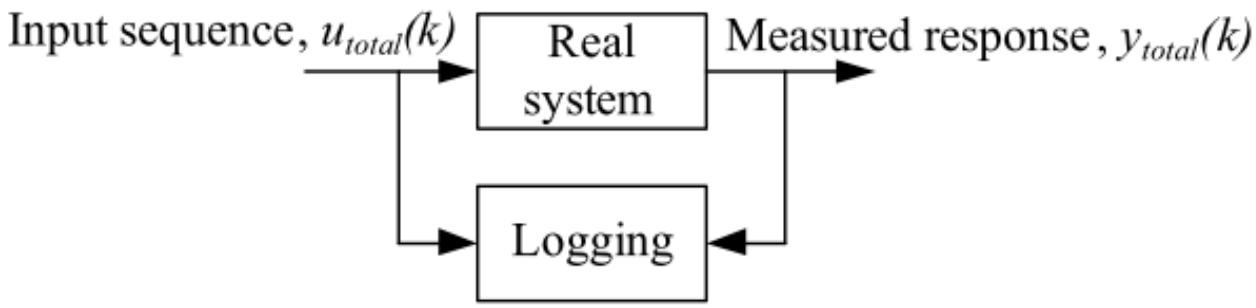
Vi vil bruke LabVIEW og MathScript som eksempler på praktisk implementering av systemidentifikasjon.

"**LabVIEW Control Design and Simulation Module**" har funksjonalitet for å implementere blant annet Kalmanfilter og Observers, mens "**LabVIEW System Identification Toolkit**" har funksjonalitet knyttet opp mot Systemidentifikasjon.

I dette notatet vil vi se på følgende metoder:

- **"Prøv og feil" metoden**
- **Sprangrespons**
- **Minste kvadraters metode**
- **Blackbox/Subspace-metoder**

→ Alle metodene går i prinsippet ut på å "logge" input og output data fra det virkelige systemet.



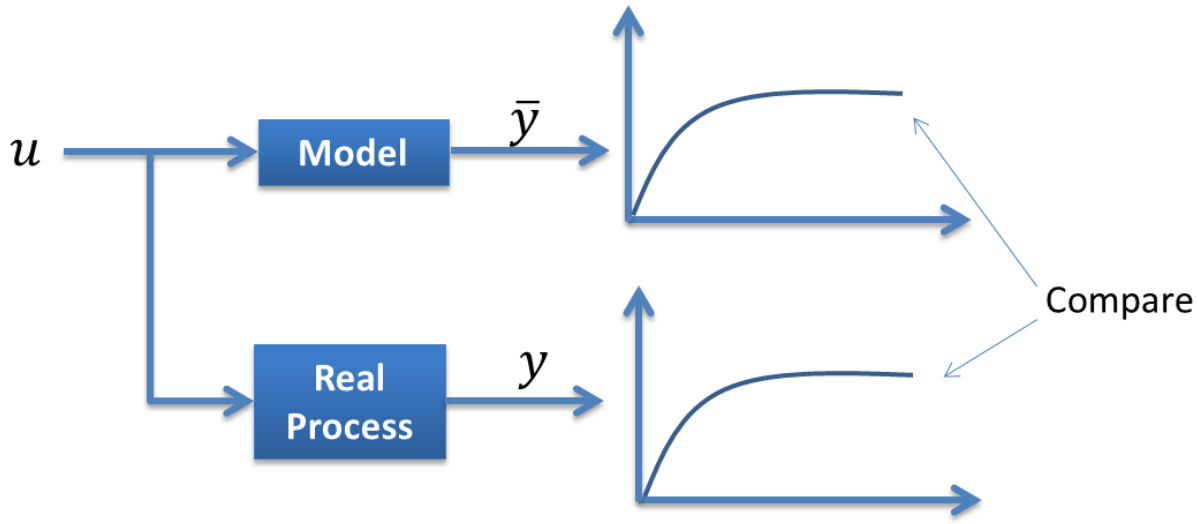
[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

1.2 Prøv og feil metoden

Vi må lage en matematisk modell av prosessen og påtrykker samme pådrag på modellen og den virkelige prosessen. Vi plotter utgangen for modellen ($\hat{y}$) og prosessen (y) i samme plot og sammenligner disse.

Kort fortalt går "Prøv og feil" metoden ut på følgende:

1. Vi sammenligner modellrespons med virkelig respons
2. Vi justerer verdiene på modellparametrene til vi oppnår ønsket resultat.



Så dette vi være en iterativ prosedyre:

1. Gjett på noen fornuftige verdier på prosessparametrene
2. Sammenlign modellrespons med virkelig respons etter f.eks. et sprang i pådraget
3. Gjenta Pkt. 1 og 2 helt til du får tilfredsstillende resultater.

1.3 Sprangrespons

Dette er en litt mer systematisk metode en den helt enkle «Prøv og feil» metoden. Sprangresponsmetoden er best egnet på en 1.ordens prosess (med eller uten dødtid).

Man kan finne modellparametrene ved å utføre en enkel sprangrespons på systemet. Sprangresponsen for et 1.ordens system er gitt ved følgende transferfunksjon:

$$H(s) = \frac{K}{Ts+1} e^{-\tau s}$$

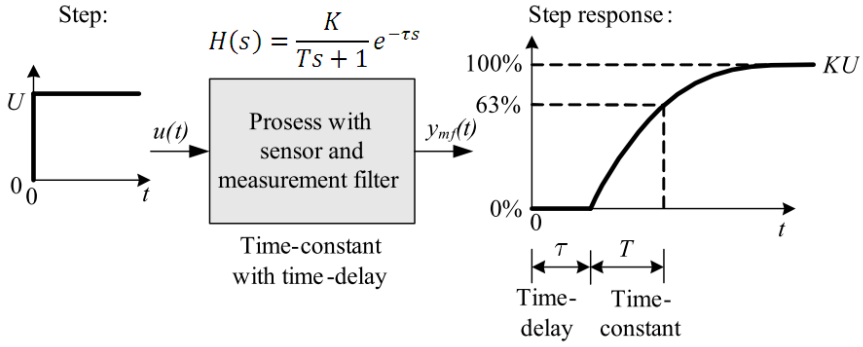
der:

K er prosessens forsterkning

T er prosessens tidskonstant

τ er prosessens dødtid

Vi kan dermed lese av prosessparametrene direkte fra sprangresponsen som illustrert nedenfor:



[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

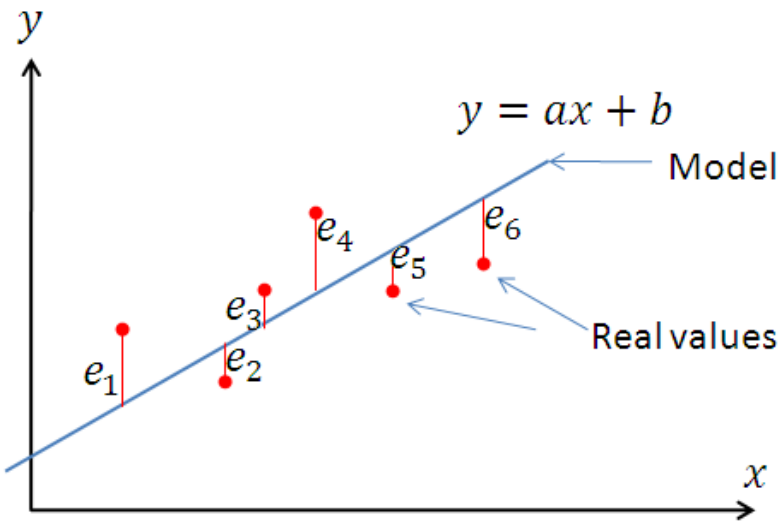
1.4 Minste kvadraters metode

Minste kvadraters metode krever at modellen må settes opp på følgende form basert på input-output data:

$$Y = \Phi\theta$$

Fra matematikken har vi tilsvarende $b = Ax$ der vi ønsker å løse et lineært likningssystem. I prinsippet er det samme, men som du ser så vi bruker litt annen notasjon innenfor systemidentifikasjon.

Vi kan illustrere det med en enkel skisse:



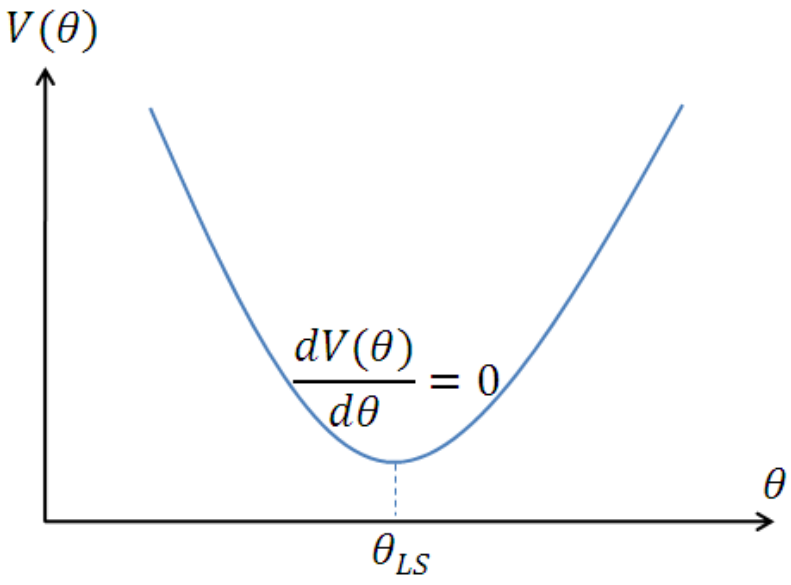
→ Vi Ønsker å minimalisere summen av avvikene $e_1, e_2, e_3, \cdots, e_m$

Dvs. vi kan definere dette som følgende kriteriefunksjon:

$$V(\theta) = e_1^2 + e_2^2 + e_3^2 + \cdots + e_m^2$$

Vi finner minimum ved å sette den deriverte lik 0:

$$\frac{dV(\theta)}{d\theta} = 0$$



Ut fra dette kan vi utlede følgende formel:

$$\boxed{\theta_{LS} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y}}$$

(utledningen blir vist senere)

1.5 Blackbox, Subspace-metoder

I en Blackbox modell kjenner vi ikke til modellen, men vi bruker algoritmer til å finne en matematisk modell basert på input og output data.

Kort fortalt går Subspace-metoder ut på følgende:

- Finne en matematisk modell som ikke er basert på fysiske lover (Newtons lover, m.fl.)
- Finner modell basert på input (pådrag) – output (målinger) data

Teorien og matematikken bak slike Subspace-metoder er kompleks, så vi vil ikke gå inn på det her. I dette notatet vil vi fokusere på praktisk bruk av en slik metode. I et senere kapitel viser vi hvordan vi kan bruke en av de innebygde funksjonene i LabVIEW (”**LabVIEW System Identification Toolkit**”).

En subspace-metode kan typisk finne en diskret tilstandsrommodell av typen:

$$x_{k+1} = Ax_k + Bu_k$$

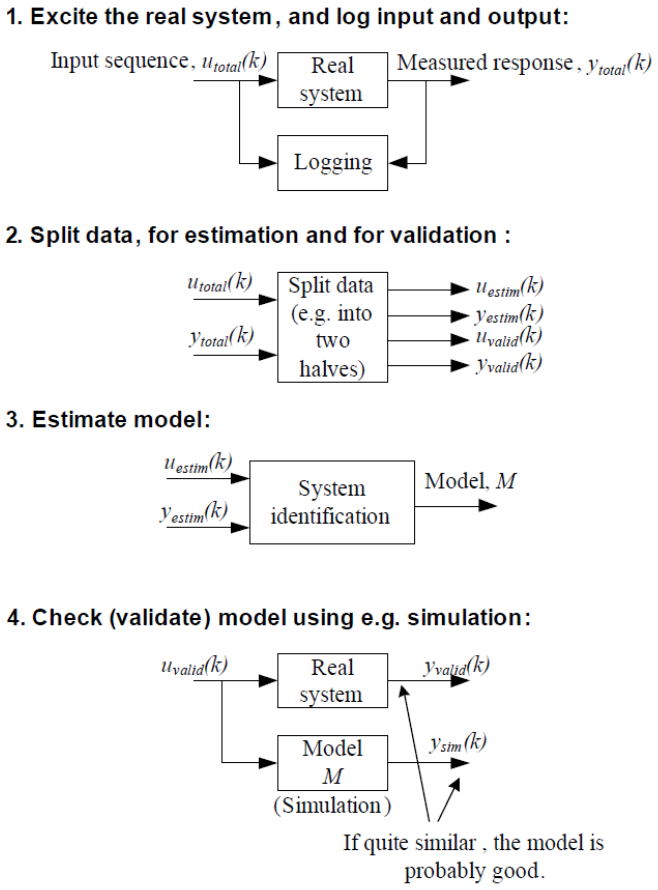
$$y_k = Cx_k + Du_k$$

2 Systemidentifikasjon – Trinn for Trinn

I systemidentifikasjon inngår ulike steg, i hovedsak har vi følgende:

1. **Eksperimenter/Logging** – Vi setter opp et eksperiment og samler inn et sett med data fra inngangssignaler og utgangssignaler fra systemet som vi ønsker å identifisere. Det er viktig at systemet blir tilstrekkelig eksitert. Vi logger dataene til fil eller liknende.
2. **Oppsplitting** - Dele opp dataene i ulike datasett; ett datasett brukes til å finne modellen, mens det andre datasettet brukes til å validere modellen.
3. **Modellestimering** – Vi finner modellen eller modellparametrene vha. en eller flere systemidentifikasjonsmetoder (Minste kvadraters metode, Black-box, osv.).
4. **Validering** – Sjekk om modellen er bra nok vha f.eks simuleringer. Her er det viktig at vi bruker valideringsdataene – og ikke de samme dataene som vi brukte når vi fant modellen i trinn 3.

Nedenfor ser vi en figur som illustrerer disse stegene:

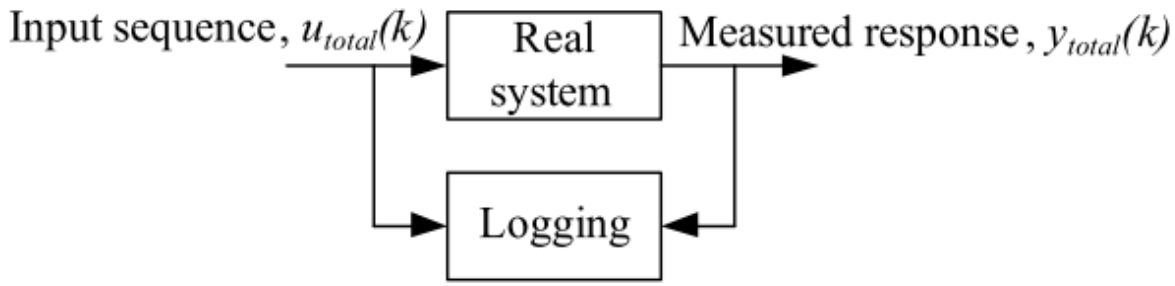


[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

2.1 Eksperimenter/Logging

Vi setter opp et eksperiment og samler inn et sett med data fra inngangssignaler og utgangssignaler fra systemet som vi ønsker å identifisere. Det er viktig at systemet blir tilstrekkelig eksitert.

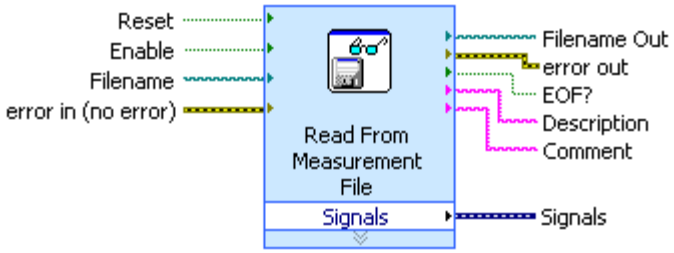
Dette kan illustreres slik:



[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

LabVIEW:

I LabVIEW kan vi bruke "**Write to Measurement File**" for å logge input og output dataene til en fil.

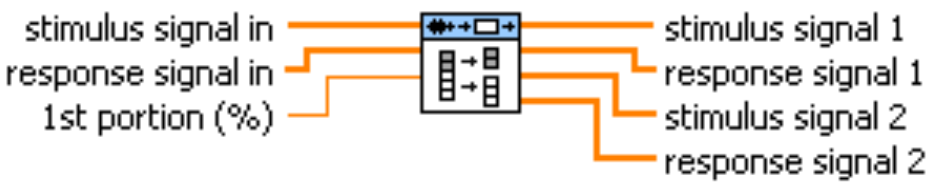


2.2 Oppsplitting

Vi deler opp dataene i ulike datasett; ett datasett brukes til å finne modellen, mens det andre datasettet brukes til å validere modellen.

LabVIEW:

I LabVIEW kan vi bruke funksjonen "**SI Split Signal.vi"** til å dele opp datasettet.



F.eks. kan man bruke de første 50% til å finne modellen, mens de resterende 50% kan brukes til å validere modellen.

2.3 Modellestimering

Vi finner modellen eller modellparametrene vha. en eller flere systemidentifikasjonsmetoder (Minste kvadraters metode, Black-box, osv.).

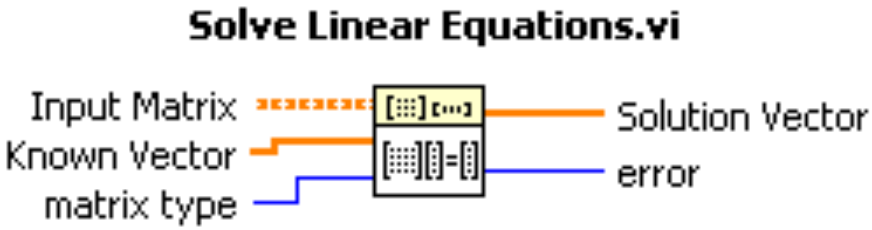
Det første vi må bestemme er om vi ønsker å lage en matematisk modell av systemet basert på fysiske lover eller om dette ikke er ønskelig.

I begge tilfellene vi vi bruke loggedataene fra forrige trinn.

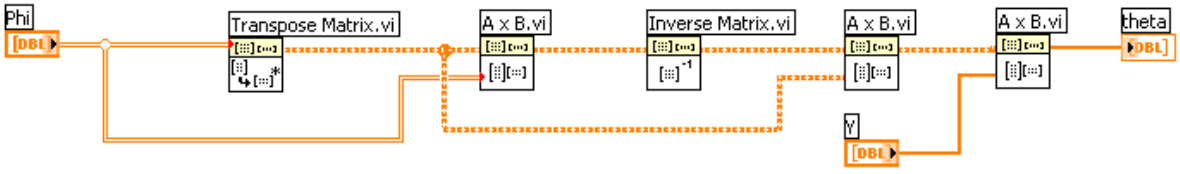
LabVIEW:

I LabVIEW har vi innebygde funksjoner får å bruke Minste kvadraters metode, samt Subspace-metoder, m.m.

Ifm Minste kvadraters metode kan vi f.eks bruke ”Solve Linear Equations.vi”:



Vi kan også bruke de innebygde matrise-funksjonene for å løse $\theta_{LS} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y}$:

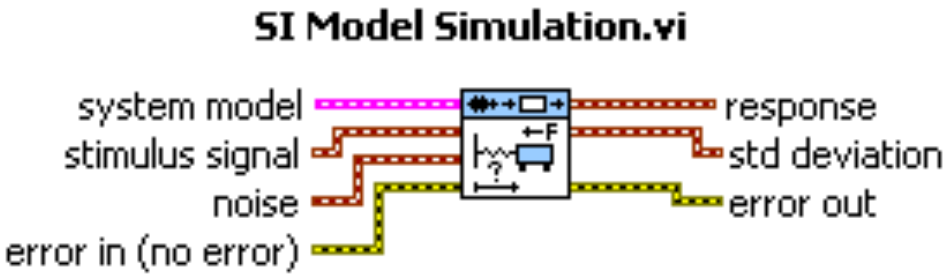


2.4 Validering

Vi sjekker om modellen er bra nok vha f.eks simuleringer. Her er det viktig at vi bruker valideringsdataene – og ikke de samme dataene som vi brukte når vi fant modellen.

LabVIEW:

I LabVIEW finnes det flere funksjoner vi kan bruke til å validere modellen, f.eks. ”**SI Model Simulation.vi**”:



3 Minste kvadraters metode

3.1 Innledning

I forbindelse med systemidentifikasjon og parameter estimering blir Minste kvadraters metode brukt til å finne ukjente modellparametre i en matematisk modell.

I matematikken blir Minste kvadraters metode brukt til å løse et sett med lineære likninger som kan settes opp påfølgende form:

$$Ax = b$$

Å sette likningene opp på denne formen gjør det enklere å løse likningene, dvs finne x ved bruk av ulike metoder, f.eks Gauss' elimineringsmetode , Minste kvadraters metode, eller liknende

Eksempel:

Gitt følgende likninger:

$$x_1 + 2x_2 = 5$$

$$3x_1 + 4x_2 = 6$$

Disse kan skrives på formen ($Ax = b$):

$$\underbrace{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 5 \\ 6 \end{bmatrix}}_{b}$$

Den enkleste måten å løse likningssettet på er:

$$x = A^{-1}b$$

Dette forutsetter for eksempel at A matrisa er inverterbar. Hvis ikke, må vi bruke for eksempel Minste kvadraters metode som vi skal lære mer om i dette kapittelet.

Når vi skal løse et sett likninger i matematikken har vi som regel like mange likninger som ukjente, dv. $n = N$, der n er antall ukjente og N er antall likninger. I systemidentifikasjon derimot har vi somregel at $N \gg n$, dvs. vi logger mange flere punkter enn det er antall ukjente.

Gitt at det er n ukjente parametre og N observasjoner i datasettet.

$\boldsymbol{n = N}$: Hvis $n = N$ og at de $n = N$ likningene er lineært uavhengige, kan vi bestemme de ukjente parametrene entydig ved $x = A^{-1}b$

$n < N$: Hvis vi har at det er n ukjente parametre og N observasjoner og $n < N$ vil vi få et overbestemt likningssystem (dvs. færre variable enn det er likninger). Da må vi bruke andre metoder, siden A ikke lenger vil være kvadratisk (og kan dermed ikke inverteres). Da kan vi f.eks bruke Minste kvadrters metode.

Vi kan f.eks. bruke MathScript for å finne svaret:

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

x=inv(A)*b
```

Vi får da følgende svar:

```
x =

           -4

            4.5
```

Dvs.:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -4 \\ 4.5 \end{bmatrix}$$

[Slutt på eksempel]

3.2 Minste kvadraters metode

I forbindelse med systemidentifikasjon og parameter-estimering er det mer vanlig å bruke følgende notasjon:

$$\boldsymbol{Y = \Phi\theta}$$

(som er det samme som $b = Ax$)

der

θ er en vektor med de ukjente parametrene som vi ønsker å finne verdiene på

Y er en vektor med kjente målinger

Φ er den såkalte regresjonsmatrisen. Denne matrisen består av kjente verdier

Minste kvadraters løsning for likningen $Y = \Phi Y$ er da gitt ved følgende formel:

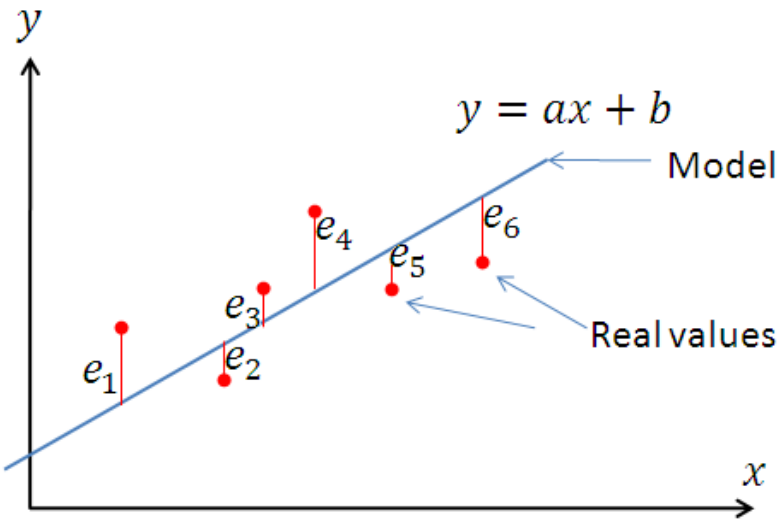
$$\boldsymbol{\theta_{LS} = (\Phi^{T}\Phi)^{-1}\Phi^{T}Y}$$

(utledning i neste avsnitt)

Denne likningen kan enkelt implementeres i ulike programmeringsspråk, for eksempel MathScript eller LabVIEW.

Ved minste kvadraters metode velger man den løsningen som gir at summen av kvadratene av avvikene fra de gitte betingelsene er et minimum. Om man for eksempel vil bestemme en rett linje ut fra en rekke punkter som er observert, velger man den linjen hvor summen av kvadratene av avstandene fra de observerte punkter til linjen er så liten som mulig.

Dette kan illustreres slik:



Dvs Minste kvadraters metode går ut på å minimalisere summen av kvadratavvikene:

$$e_1^2 + e_2^2 + \cdots + e_m^2$$

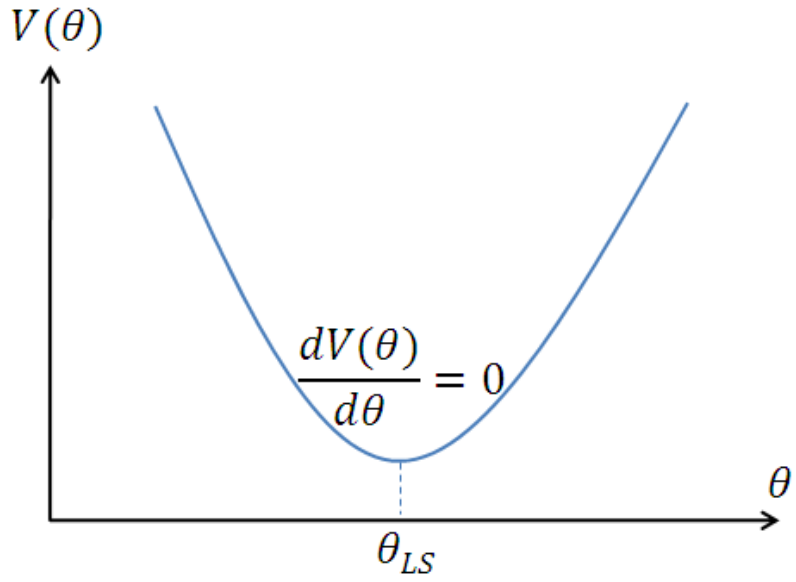
Dvs. vi kan definere dette som følgende kriteriefunksjon:

$$V(\theta) = e_1^2 + e_2^2 + e_3^2 + \cdots + e_m^2$$

Vi finner minimum ved å sette den deriverte lik 0:

$$\frac{dV(\theta)}{d\theta} = 0$$

Dette kan illustreres slik:



Minste kvadraters løsning for likningen $Y = \Phi Y$ er da gitt ved følgende formel:

$$\theta_{LS} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y}$$

Antall Observasjoner vs. Antall Parametre:

Gitt at det er n ukjente parametre og N observasjoner i datasettet.

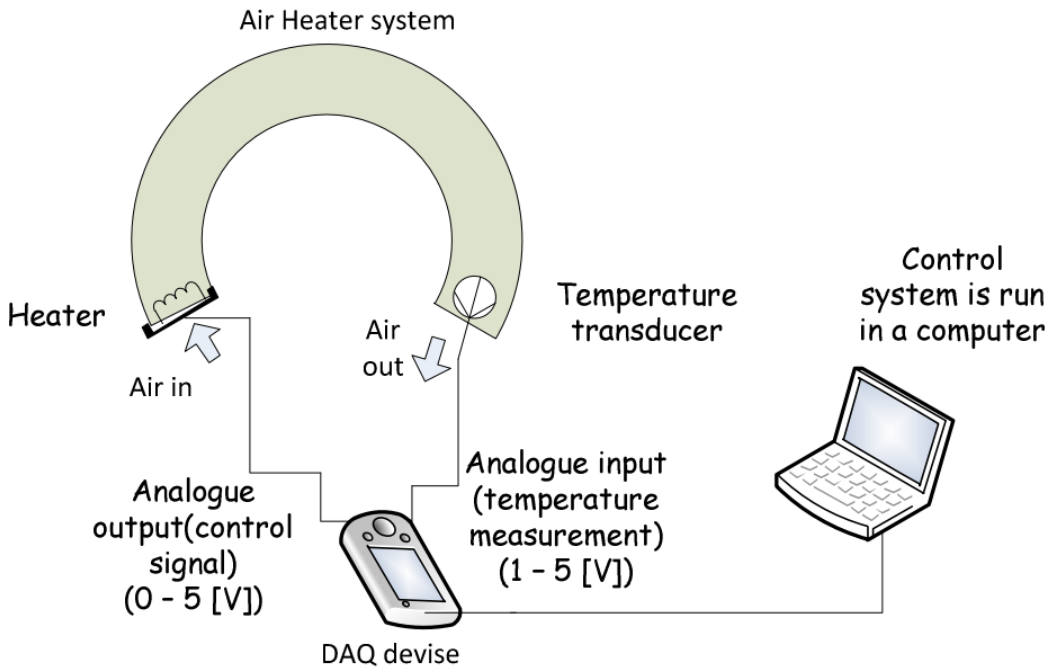
$n = N$: Hvis $n = N$ og at de $n = N$ likningene er lineært uavhengige, kan vi bestemme de ukjente parametrene entydig ved $\theta = \Phi^{-1}Y$

Men i Systemidentifikasjon er det vanlig at $n \ll N$, dvs. det er ønskelig å ha mange flere observasjoner enn det er ukjente parametre, dette pga av observasjonene (måledataene) kan inneholde støy, m.m.

$n < N$: Hvis vi har at det er n ukjente parametre og N observasjoner og $n < N$ vil vi få et overbestemt likningssystem (dvs. færre variable enn det er likninger). Da må vi bruke andre metoder, siden A ikke lenger vil være kvadratisk (og kan dermed ikke inverteres). Det er her Minste kvadraters metode kommer inn.

Eksempel:

Gitt følgende system av en varmluftsprosess:



Systemet har følgende matematiske modell:

$$\dot{T}_{out} = \frac{1}{\theta_t}\{-T_{out} + [K_h u(t-\theta_d) + T_{env}]\}$$

hvor:

- $u\ [V]$ er kontrollsignalet til varmeelementet
- $\theta_t\ [s]$ er tidskonstanten til systemet
- $K_h\ [deg\ C\ /\ V]$ er varmeelementets forsterkning
- $\theta_d\ [s]$ er tidsforsinkelsen i systemet
- T_{env} er romtemperaturen

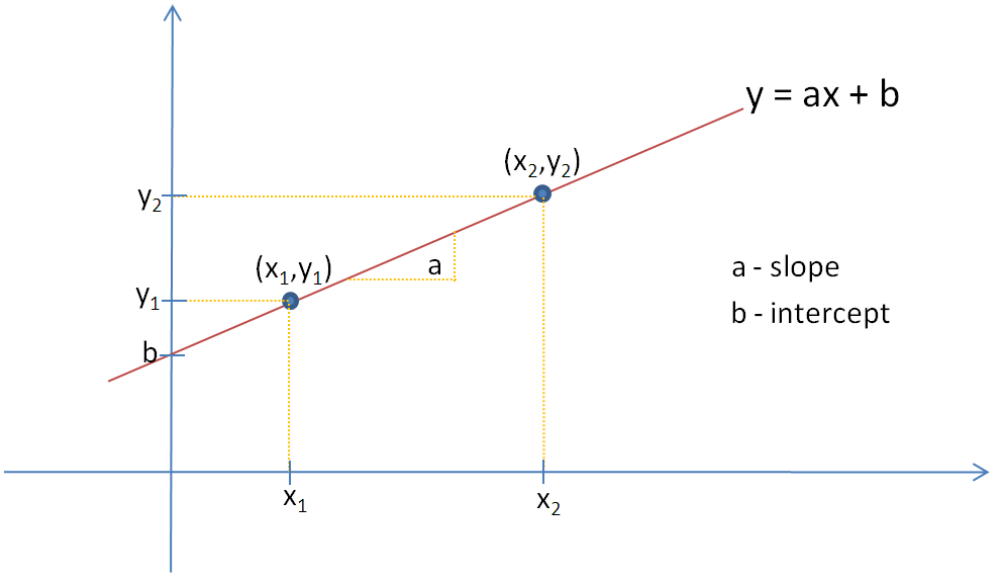
Vi ønsker å logge temperaturen vha en DAQ enhet som gir oss et spenningssignal mellom $1-5V$. Dette signalet ønsker vi å konvertere til en temperaturverdi mellom $20-50℃$, dvs. vi trenger å skalere $1-5V$ til $20-50℃$.

Siden dette vil være en lineær skalering kan vi bruke følgende:

$$y(x) = ax + b$$

dvs vi ønsker å finne parametrene a og b for denne funksjonen.

Dette kan vi illustrere på følgende måte:



Dvs vi har 2 punkter på linja:

$$(x1, y1) = (1, 20)$$

$$(x2, y2) = (5, 50)$$

eller:

$$y(1) = 20$$

$$y(5) = 50$$

Vi ønsker å sette det opp på regresjonsformen $Y = \Phi\theta$ og får følgende:

$$20 = a \cdot 1 + b$$

$$50 = a \cdot 5 + b$$

som gir:

$$\underbrace{\begin{bmatrix} 20 \\ 50 \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} 1 & 1 \\ 5 & 1 \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\theta}$$

Dette kan vi nå løse (dvs. finne a og b) på følgende enkle måte (siden Φ er kvadratisk og kan inverteres, $n = N = 2$):

$$\theta = \Phi^{-1}\mathrm{Y}$$

Dvs. vi har 2 likninger og 2 ukjente.

Innsatt verdier får vi:

$$\theta = \begin{bmatrix} 1 & 1 \\ 5 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 20 \\ 50 \end{bmatrix} = \begin{bmatrix} -\frac{1}{4} & \frac{1}{4} \\ \frac{5}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} 20 \\ 50 \end{bmatrix} = \begin{bmatrix} \frac{15}{2} \\ \frac{25}{2} \end{bmatrix} = \begin{bmatrix} 7.5 \\ 12.5 \end{bmatrix}$$

Dvs.:

$$\theta = \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 7.5 \\ 12.5 \end{bmatrix}$$

Dette gir:

$$\underline{\underline{y(x) = 7.5x + 12.5}}$$

Merk! For $2x2$ matrise A:

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

beregnes den inverse matrisen A^{-1} på følgende måte:

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

og

$$\det(A) = |A| = a_{11} \cdot a_{22} - a_{21} \cdot a_{12}$$

MathScript:

Dette kan løses i MathScript på følgende måte:

```
phi = [1, 1; 5 ,1];
Y = [20; 50];

theta = inv(phi)*Y
```

Vi kan også bruke minste kvadraters metode for å løse dette:

$$\theta_{LS} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y}$$

Dette kan gjøres på følgende måte i MathScript:

```
phi = [1, 1; 5 ,1];
Y = [20; 50];
theta_ls = inv(phi'*phi)*phi'*Y
```

evt kan vi bruke den innebygde operatoren "\":

```
theta_ls = phi\Y
```

Svaret blir det samme, dvs:

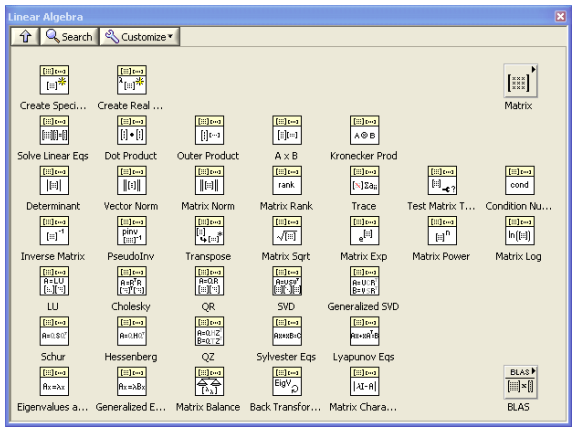
```
theta_ls =

            7.5

           12.5
```

[Slutt på eksempel]

3.3 Eksempler

Her vil vi se på noen flere eksempler hvor vi bruker Minste kvadraters metode for å finne modellparametrene. Vi vil bruke LabVIEW og MathScript som eksempler på verktøy ifm med dette.

LabVIEW har innebygde funksjoner som kan brukes ifm Minste kvadraters metode. I "**Linear Algebra**" (lokalisert i «Mathematics» paletten) paletten finnes det mange funksjoner som kan brukes for å finne Minste kvadraters løsning:



Eksempel:

Gitt følgende modell:

$$y(u) = au + b$$

Følgende verdier er funnet fra eksperimenter:

$$y(1) = 0.8$$

$$y(2) = 3.0$$

$$y(3) = 4.0$$

Vi ønsker å finne de ukjente modellparametrene a and b ved å bruke Minste kvadraters metode i **MathScript/LabVIEW**.

Vi har følgende:

$$Y = \Phi\theta$$

der

θ er en vector med de ukjente parametrene som vi ønsker å finne verdiene på

Y er en vektor med kjente målinger

Φ er den såkalte regresjonsmatrisen. Denne matrisen består av kjente verdier

Minste kvadraters løsning for likningen $Y = \Phi Y$ er da gitt ved følgende formel:

$$\boldsymbol{\theta_{LS} = (\Phi^{T}\Phi)^{-1}\Phi^{T}Y}$$

Dermed får vi:

$$0.8 = a \cdot 1 + b$$

$$3.0 = a \cdot 2 + b$$

$$4.0 = a \cdot 3 + b$$

Som blir:

$$\underbrace{\begin{bmatrix}0.8\\3.0\\4.0\end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix}1 & 1\\2 & 1\\3 & 1\end{bmatrix}}_{\Phi}\underbrace{\begin{bmatrix}a\\b\end{bmatrix}}_{\theta}$$

Her vil $n(2) < N(3)$

MathScript:

Vi definerer Y og Φ i MathScript og finner θ:

```
phi = [1 1; 2 1; 3 1];
Y = [0.8 3.0 4.0]';

theta = inv(phi'*phi)* phi'*Y

%or simply by
theta=phi\Y
```

Svaret blir:

```
theta =    1.6

          -0.6
```

Dvs.:

$$a = 1.6$$

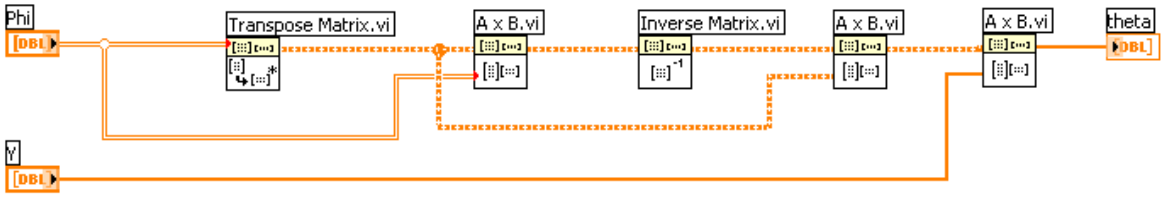
$$b = -0.6$$

Som gir følgende modell:

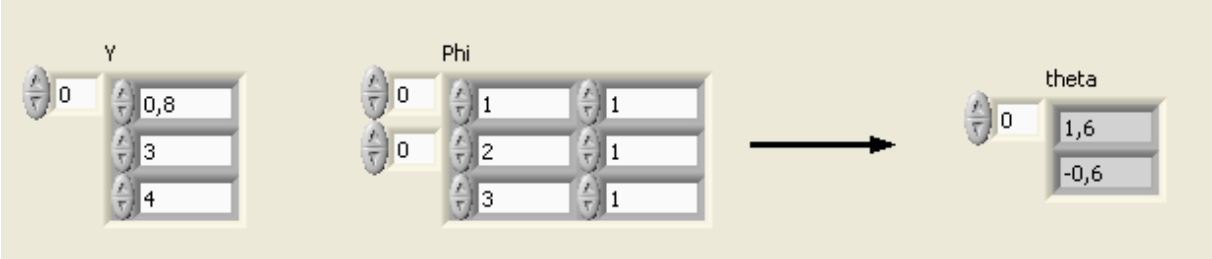
$$y(u) = 1.6u - 0.6$$

LabVIEW:

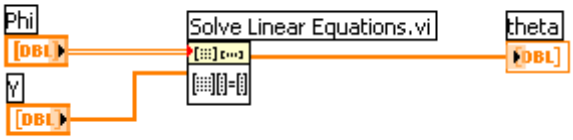
Blokkdiagrammet blir som følger:



Frontpanelet blir som følger:



Vi kan også bruke funksjonen "**Solve Linear Equations.vi**" direkte:



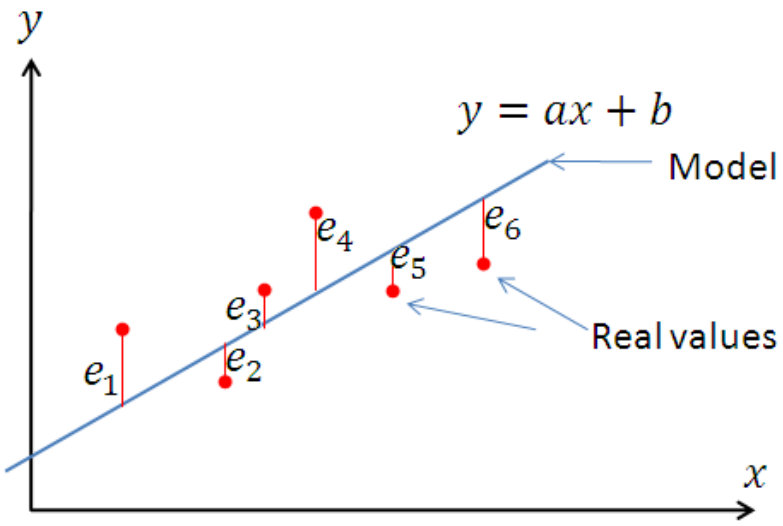
[Slutt på eksempel]

3.4 Utledning

Miste kvadraters metode forutsetter at systemet settes opp på denne formen:

$$\boldsymbol{Y = \Phi\theta}$$

Metoden går ut på at vi ønsker å tilpasse dataene våre til en gitt matematisk modell, dette er illustrert i figuren under:



Det vi ønsker er å finne parametrene i modellen slik at denne blir mest mulig tilpasset de dataene vi har.

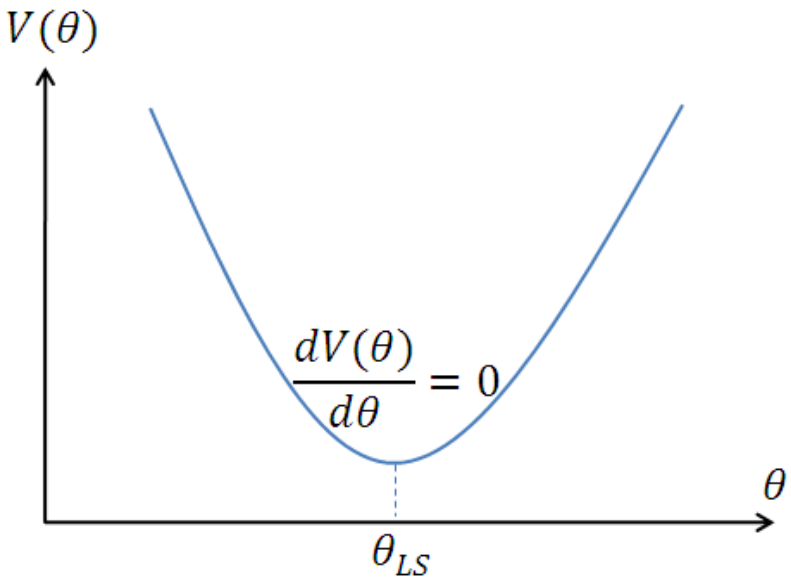
→ Ønsker å minimalisere summen av avvikene $e_1, e_2, e_3, \cdots, e_m$, dvs vi bruker følgende kriteriefunksjon:

$$V(\theta) = e_1^2 + e_2^2 + e_3^2 + \cdots + e_m^2$$

Vi finner minimum ved å sette den deriverte lik 0:

$$\frac{dV(\theta)}{d\theta} = 0$$

Dette kan illustreres slik:



Ut fra dette kan vi utlede følgende formel:

$$\boxed{\theta_{LS} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y}}$$

Hvordan vi kommer frem til denne formelen vil bli utledet under.

Utledning:

Utledningen krever kunnskap om matriser og vektorer og regler for derivasjon av matriser og vektorer.

Vi ønsker å minimalisere følgende kriteriefunksjon

$$V(\theta) = e_1^2 + e_2^2 + e_3^2 + \cdots + e_m^2$$

Vi har regresjonsmodellen:

$$Y = \Phi\theta$$

Avviket er gitt ved:

$$e = Y - \Phi\theta$$

der e vil være en vektor:

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix}$$

Vi får da:

$$V(\theta) = e^T e = (Y - \Phi\theta)^T (Y - \Phi\theta)$$

Som gir:

$$V(\theta) = \left(Y^T - \theta^T \Phi^{\mathrm{T}}\right)(Y - \Phi\theta)$$

(Merk: vi har brukt følgende: $(AB)^T = B^T A^T$)

Vi ganger ut:

$$Y^T Y - Y^T \Phi\theta - \theta^T \Phi^{\mathrm{T}} Y + \theta^T \Phi^{\mathrm{T}} \Phi\theta$$

Vi deriverer for å finne minimum:

$$\frac{dV(\theta)}{d\theta} = \frac{d}{d\theta}(Y^T Y) - \frac{d}{d\theta}(Y^T \Phi\theta) - \frac{d}{d\theta}(\theta^T \Phi^{\mathrm{T}} Y) + \frac{d}{d\theta}(\theta^T \Phi^{\mathrm{T}} \Phi\theta) = 0$$

Vi tar ledd for ledd:

$\frac{d}{d\theta}(Y^TY) = 0$, dvs den deriverte av en konstant er lik 0

$\frac{d}{d\theta}(Y^T\Phi\theta) = (Y^T\Phi)^T = \Phi^{\mathrm{T}}\mathrm{Y}$, dvs har brukt regel: $\frac{d}{dx}(Ax) = A^T$ og $(AB)^T = B^TA^T$

$\frac{d}{d\theta}(\theta^T\Phi^{\mathrm{T}}Y) = \Phi^{\mathrm{T}}Y$, dvs har brukt regel: $\frac{d}{dx}(x^TA) = A$

$\frac{d}{d\theta}(\theta^T\Phi^{\mathrm{T}}\Phi\theta) = \Phi^{\mathrm{T}}\Phi\theta + \Phi^{\mathrm{T}}\Phi\theta = 2\Phi^{\mathrm{T}}\Phi\theta$, dvs har brukt regel: $\frac{d}{dx}(x^TAx) = Ax + A^Tx$

Dette gir:

$$\frac{dV(\theta)}{d\theta} = 0 - \Phi^{\mathrm{T}}\mathrm{Y} - \Phi^{\mathrm{T}}\mathrm{Y} + 2\Phi^{\mathrm{T}}\Phi\theta = 2\Phi^{\mathrm{T}}\Phi\theta - 2\Phi^{\mathrm{T}}\mathrm{Y} = 0$$

Dvs.

$$\Phi^{\mathrm{T}}\Phi\theta = \Phi^{\mathrm{T}}\mathrm{Y}$$

(denne kalles forøverig normallikningen)

Vi løser mtp. θ, resultatet blir:

$$\boldsymbol{\theta_{LS} = \left(\Phi^{\mathrm{T}}\Phi\right)^{-1}\Phi^{\mathrm{T}}\mathrm{Y}}$$

dvs. vi har funnet formelen for minste kvadraters løsning av θ.

4 Blackbox identifikasjon og Subspace-metoder

Subspace-metoder baserer seg på å finne modell som ikke er basert på fysiske lover. Modeller funnet på denne måten kalles Blackbox modeller. Subspace-metoder baserer seg på at vi finner modeller basert på input (pådrag) – output (målinger) data fra prosessen.

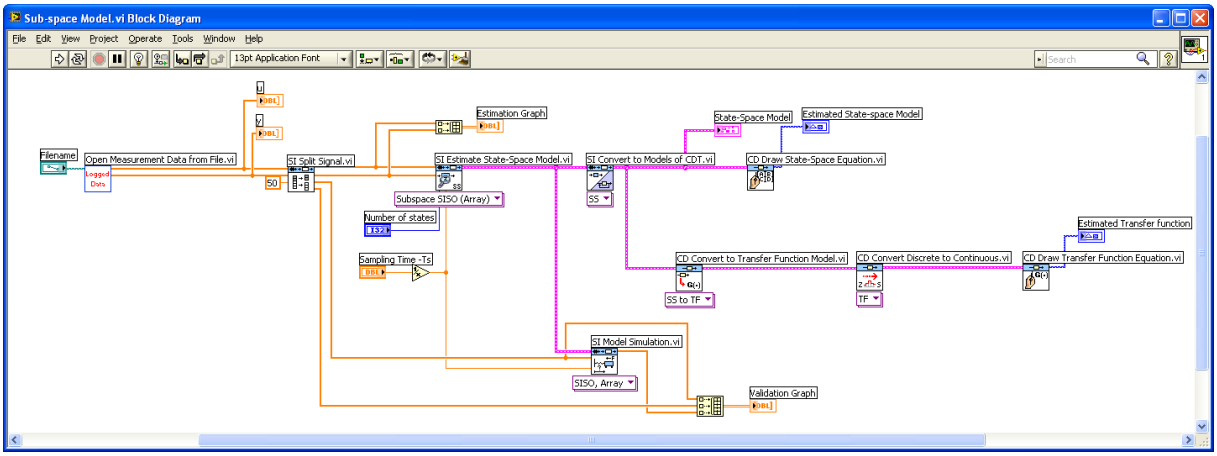
En subspace-metode kan typisk finne en diskret tilstandsrommodell av typen:

$$x_{k+1} = Ax_k + Bu_k$$

$$y_k = Cx_k + Du_k$$

LabVIEW har innebygd funksjonalitet for a finne Blackbox modeller basert på Subspace metoder, for eksempel i paletten "Parametric Model Estimation" finner vi blant annet VI'en "**Estimate State-Space**".

Nedenfor ser vi et eksempel på hvordan dette kan implementeres i LabVIEW.



Appendiks A: Matriser

Transponert

Gitt en matrise A:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$$

Den transponerte av A er gitt ved:

$$A^T = \begin{bmatrix} a_{11} & \cdots & a_{n1} \\ \vdots & \ddots & \vdots \\ a_{1m} & \cdots & a_{nm} \end{bmatrix}$$

Merk!

$$(AB)^T = B^T A^T$$

Eksempel:

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

Den transponerte av A blir:

$$A^T = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}^T = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix}$$

MathScript

```
A=[0 1; -2 -3];
A'
```

Dette gir følgende svar:

```
ans =
     0    -2

     1    -3
```

[Slutt på eksempel]

Matrisemultiplikasjon

Gitt matrisene $A \in R^{nxm}$ og $B \in R^{mxp}$, da får vi:

$$C = AB \in R^{nxp}$$

hvor

$$c_{jk} = \sum_{l=1}^{n} a_{jl} b_{lk}$$

Eksempel:

MathScript:

```
A=[0 1;-2 -3];
B=[1 0;3 -2];
A*B
```

Dette gir:

```
ans =
     3    -2

   -11     6
```

[Slutt på eksempel]

Merk!

$$n\left[\overset{m}{A}\right] m\left[\overset{p}{B}\right] = n\left[\overset{p}{C}\right]$$

Merk!

Følgende gjelder:

$$AB \neq BA$$

$$A(BC) = (AB)C$$

$$(A+B)C = AC + BC$$

$$C(A+B) = CA + CB$$

$$(AB)^T = B^T A^T$$

Matriseaddisjon

Gitt matrisene $A \in R^{nxm}$ og $B \in R^{nxm}$, da får vi:

$$C = A + B \in R^{nxm}$$

Eksempel:

MathScript:

```
A=[0 1;-2 -3];
B=[1 0;3 -2];
A+B
```

Dette gir:

```
ans =
     1     1

     1    -5
```

[Slutt på eksempel]

Determinant

Determinanten er bare definert for kvadratiske matriser. Determinanten til en kvadratisk matrise A er definert ved den skalare "tallverdien".

$$\det(A) = |A|$$

2x2 Systemer

Slik finner vi determinanten for et 2x2 system:

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

$$\boxed{\det(A) = |A| = a_{11} \cdot a_{22} - a_{21} \cdot a_{12}}$$

Eksempel:

$$A = \begin{bmatrix} 3 & -2 \\ 5 & 1 \end{bmatrix}$$

$$\det(A) = |A| = 3 \cdot 1 - 5 \cdot (-2) = 3 + 10 = \underline{\underline{13}}$$

I MathScript kan vi gjøre følgende:

```
A=[3 -2;5 1];
det(A)
```

```
rank(A)
```

Som gir følgende svar:

```
ans = 13 (determinant)

ans = 2 (rang)
```

[Slutt på eksempel]

3x3 Systemer

Det blir litt mer komplisert for systemer med større orden, men vi vil maks håndregne på 3x3 systemer. For større systemer bruker vi et dataprogram til å beregne dette, f.eks MathScript.

Slik finner vi determinanten for et 3x3 system:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Vi utvikler determinanten langs en rekke eller en kolonne.

Her utvikler vi determinanten langs første kolonne:

$$\det(A) = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{21}\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} + a_{31}\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix}$$

Vi ser at determinanten til et høyere ordens system kan uttrykkes som en sum av lavere ordens determinanter.

Eksempel:

$$A = \begin{bmatrix} -1 & 3 & 0 \\ 2 & 1 & -5 \\ 1 & 4 & -2 \end{bmatrix}$$

$$\det(A) = (-1)\begin{vmatrix} 1 & -5 \\ 4 & -2 \end{vmatrix} - 2\begin{vmatrix} 3 & 0 \\ 4 & -2 \end{vmatrix} + 1\begin{vmatrix} 3 & 0 \\ 1 & -5 \end{vmatrix}$$

Dette gir:

$$\begin{vmatrix} 1 & -5 \\ 4 & -2 \end{vmatrix} = -2 - (-20) = 18$$

$$\begin{vmatrix} 3 & 0 \\ 4 & -2 \end{vmatrix} = -6 - 0 = -6$$

$$\begin{vmatrix} 3 & 0 \\ 1 & -5 \end{vmatrix} = -15 - 0 = -15$$

Som gir:

$$\det(A) = -18 + 12 - 15 = \underline{\underline{-21}}$$

I MathScript kan vi gjøre følgende:

```
A=[-1 3 0; 2 1 -5; 1 4 -2];
det(A)
rank(A)
```

Som gir følgende svar:

```
ans = -21 (determinant)

ans = 3 (rang)
```

[Slutt på eksempel]

Merk!

$$\det(AB) = \det(A)\det(B)$$

Merk!

$$\det(A^T) = \det(A)$$

Invertering av matriser

Gitt en $2x2$ matrise:

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

Den inverse av A er gitt ved:

$$\boxed{A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}}$$

Eksempel:

Gitt følgende matrise:

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

Den inverse av A er gitt ved:

$$A^{-1} = \frac{1}{2} \begin{bmatrix} -3 & -1 \\ 2 & 0 \end{bmatrix}$$

MathScript:

```
A=[0 1; -2 -3]
inv(A)
```

Matrise og vektor derivasjon

Her er noen derivasjonsregler ifm matriser og vektorer som blir brukt ifm utledningen av Minste kvadraters metode.

$$\frac{d}{dx}(x) = I$$

$$\frac{d}{dx}(x^T x) = 2x$$

$$\frac{d}{dx}(Ax) = A^T$$

$$\frac{d}{dx}(x^T A) = A$$

$$\frac{d}{dx}(x^T Ax) = Ax + A^T x$$

Referanser

Di Ruscio, D. (2003). Matrisemetoder og lineær algebra, Forelesningsnotater.

Haugen, F. (1996). Regulering av dynamiske systemer 2, Tapir.

Haugen, F. (2010). Advanced Dynamics and Control, TechTeach.

Ljung, L. (1995). System Identification.

Skretting, K. (2010). MIK 130 Systemidentifikasjon, Forelesningsnotater.

www.wikipedia.org (2011). Black Box.

www.wikipedia.org (2011). System Identification.

Telemark University College

Faculty of Technology

Kjølnes Ring 56

N-3914 Porsgrunn, Norway

www.hit.no

Hans-Petter Halvorsen, M.Sc.

Telemark University College

Faculty of Technology

Department of Electrical Engineering, Information Technology and Cybernetics

E-mail: hans.p.halvorsen@hit.no

Blog: http://home.hit.no/~hansha/
