

Simulering og eksperimentel modelbestemmelse

Henrik Vie Christensen

`vie@es.aau.dk`

Automation & Control

Aalborg University

Denmark

Dagens program

- SENSTOOLS til parameter estimation
 - Navne konventioner
 - Procedure for parameter estimation
- Parameter følsomhed
 - Følsomhedsmål
 - Følsomhedsellipse

SENSTOOLS til parameter estimation

- **Senstools** er en samling af Matlab programmer, der implementere følsomhedsmetoden for **direkte parameter estimation**, **eksperiment design** og **model validering**.
- Programmerne kan fordelagtigt organiseres som en **Matlab Toolbox**.
- Alt hvad brugeren behøver at programmere er **simulerings programmet for den specifikke proces**.
- Programmerne er organiseret som main programmer (script filer), der kalder under programmer (funktioner) og bruger data (mat filer).

Navne konventioner

- Program og data filnavnene indeholder information om program type og navnet på den aktuelle proces. Begyndelses bogstaverne indikere typen:
 - `main` main program (script fil)
 - `sim` simulerings program for processen (funktion)
 - `meas` input/output måledata (mat fil)
 - `prog` program data (mat fil)
- Navne på filer tilknyttet en specifik proces skal indeholde proces navnet. Eksempel:
`process='motor' simmotor.m` og `measmotor.mat`
- Programnavnene indeholder også info om funktionen. Eksempel: `mainest.m` (main program for estimation)

Procedure for parameter estimation

For en proces med navn `xxx`:

1. **Opret simulerings programmet** som en Matlab funktion: `y = simxxx.m`
2. **Gem de målte data** `t`, `u` og `y`: `save measxxx t u y`
3. **Indtast nødvendige program data**, en af de tre måder:
 - a) Direkte i workspace: `process='xxx' ;`
`par0=[1 2] ;`
 - b) Indlæses fra en mat fil (`progdatabxxx.mat`). Sker automatisk hvis filen eksisterer. progdata-filen oprettes med `progprogxxx.m` fil.
 - c) Brug default værdier for main programmet.
4. **Kør** `mainest.m` for parameter estimation.

Hvordan de tre input metoder skelnes

Programkoden for mainest.m:

```
% mainest is the main program for parameter estimation
:
% 20/9-94,MK. 26/11-02,MK

% Default values:
if ~exist('process'), process='ktau'; end % Process name
if ~exist('no'), no=''; end % Measurement number
if exist(['prodata',process,no,'.mat'])==2 & ~exist('par0')
    load(['prodata',process,no]), end % prodata loades
if exist(['meas',process,no,'.mat'])==2, load(['meas',process,no]),
else
    disp(['data: meas',process,no,'.mat missing !']), break, end
if ~exist('ploty'), ploty=2; end
if ~exist('par0'), par0=[1.5 3]; end
simmod=['sim',process];
:
```

Eksempel: Par. estim. med mainest.m

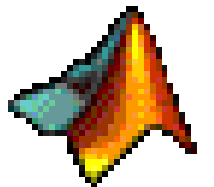
System `kutau`

Model: $\frac{Y(s)}{U(s)} = \frac{K(u)}{1 + s\tau}$ hvor $K(u) = k_0(1 + \frac{k_1}{0.5+u^2})$

`measkutau.mat` eksisterer (målinger u, y, t)

a) Manuel indlæsning af programdata

```
» process='kutau';  
» par0=[1 2 3];  
» mainest
```



Eksempel: Par. estim. med mainest.m

System **kutau**

Model: $\frac{Y(s)}{U(s)} = \frac{K(u)}{1 + s\tau}$ hvor $K(u) = k_0(1 + \frac{k_1}{0.5+u^2})$

measkutau.mat eksisterer (målinger u, y, t)

b) progprogkutau.m dannet programdata

» clear

» process='kutau';

» mainest

```
% progprogkutau.m creates program  
% data for mainest  
% with process kutau, Example 9.  
% 27/11-02,MK
```

```
clear
```

```
process='kutau';
```

```
par0=[.85 1.8 2.24];
```

```
save progdatakutau process par0
```

```
% creates progdatakutau.mat
```


Eksempel: Par. estim. med `mainest.m`

System `ktau`

Model: $\frac{Y(s)}{U(s)} = \frac{K(u)}{1 + s\tau}$ hvor $K(u) = k_0(1 + \frac{k_1}{0.5+u^2})$

`measkutau.mat` eksisterer (målinger u, y, t)

c) Default

» `clear`

» `mainest`

Bemærk: Kun to parametre, `process` → `ktau`

Eksempel: DC-motor (SIMO)

```
» help simdcml
```

```
y=[i,w]=simdcml(u,t,par) simulates a linear dc-motor with  
input u and outputs i and w.
```

```
w/u = K/R/(J*s+B+K^2/R), i/u=(J*s+B)/R/(J*s+B+K^2/R)
```

```
par=[R K J B]
```

```
27/11-02,MK
```

DC-motor modellen er en **SIMO** model (single input multiple output) så målingerne skal organiseres som en matrix:

$$y = \begin{bmatrix} i & \omega \end{bmatrix} = \begin{bmatrix} i[1] & \omega[1] \\ i[2] & \omega[2] \\ \vdots & \vdots \\ i[N] & \omega[N] \end{bmatrix}$$

Eksempel: DC-motor (SIMO)

```
» help simdcml
```

`y=[i,w]=simdcml(u,t,par)` simulates a linear dc-motor with input `u` and outputs `i` and `w`.

$w/u = K/R/(J*s+B+K^2/R)$, $i/u = (J*s+B)/R/(J*s+B+K^2/R)$

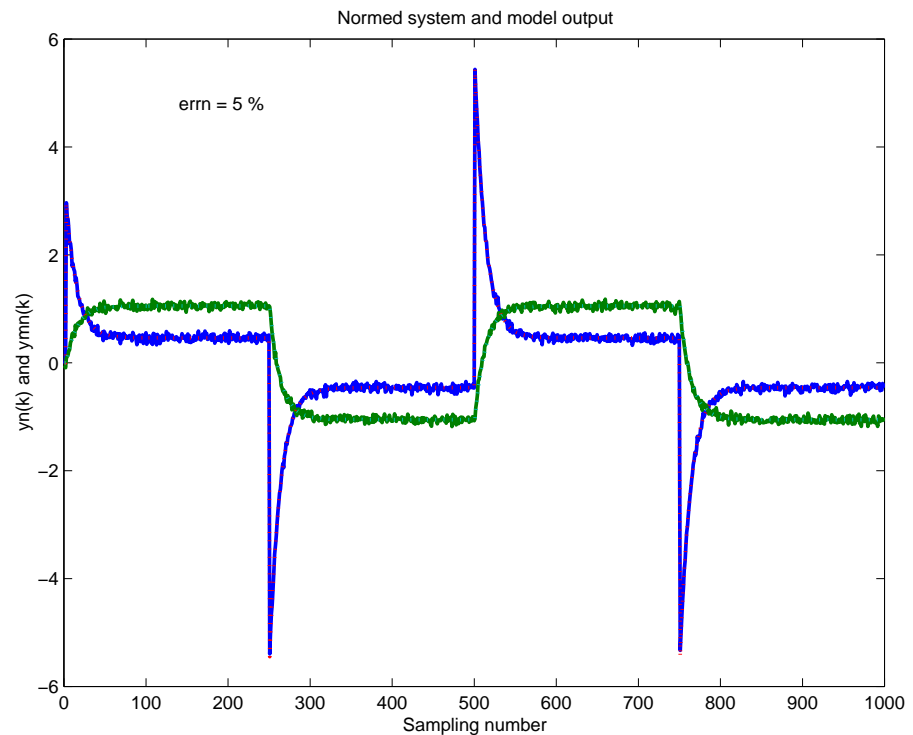
```
par=[R K J B]
```

27/11-02,MK

```
» process='dcml';
```

```
» ploty=1;
```

```
» mainest
```



Eksempel: DC-motor (SIMO)

```
» help simdcml
```

`y=[i,w]=simdcml(u,t,par)` simulates a linear dc-motor with input `u` and outputs `i` and `w`.

$w/u = K/R/(J*s+B+K^2/R)$, $i/u = (J*s+B)/R/(J*s+B+K^2/R)$

```
par=[R K J B]
```

27/11-02,MK

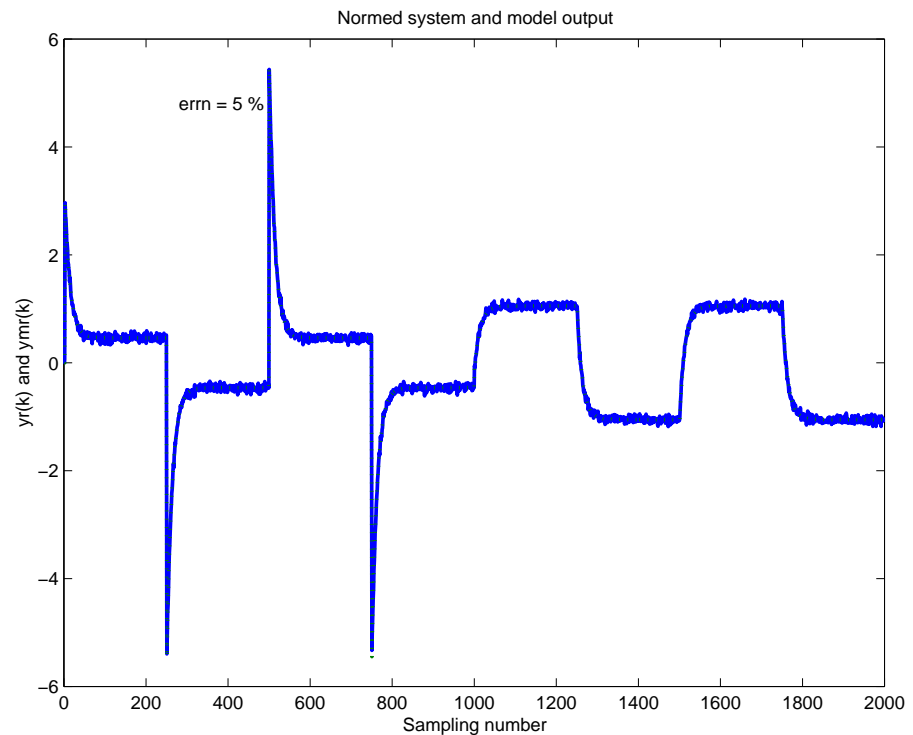
```
» process='dcml';
```

```
» ploty=1;
```

```
» mainest
```

```
» ploty=2;
```

```
» mainest
```



Evaluering af model fit

Parameter estimat: $\theta_N = \underset{\theta}{\operatorname{argmin}} P(u_N, y_N, \theta)$

Performance funktion: $P(\theta) = \frac{1}{2N} \sum_{k=1}^N \epsilon^2(k, \theta)$

Minimum værdien $P(\theta_N)$ giver **ikke** brugbar information om hvor godt fit der er opnået.

Normed root mean square output error:

$$errn = \sqrt{\frac{\sum_{k=1}^N (y(k) - y_m(k, \theta_N))^2}{\sum_{k=1}^N y^2(k)}} \cdot 100 [\%]$$

der giver god information om graden af fit.

Gode modeller og lav støj på målinger: 5–8%

Meget komplicerede systemer: 20–25%

Parameter følsomhed

Model fejlen kan splittes i to bidrag:

$$\epsilon(k, \theta) = \epsilon_0(k) + \epsilon_p(k, \theta)$$

hvor ϵ_0 er støj og undermodellering, ϵ_p er parameter afhængig bidrag:

$$\epsilon_p(k, \theta) = y_m(k, \theta) - y_m(k, \theta_N) \approx \psi^\top(k, \theta_N)(\theta - \theta_N)$$

hvor ψ er model gradienten $\psi(k, \theta) = \frac{\partial y_m(k, \theta)}{\partial \theta}$.

Parameter følsomhed

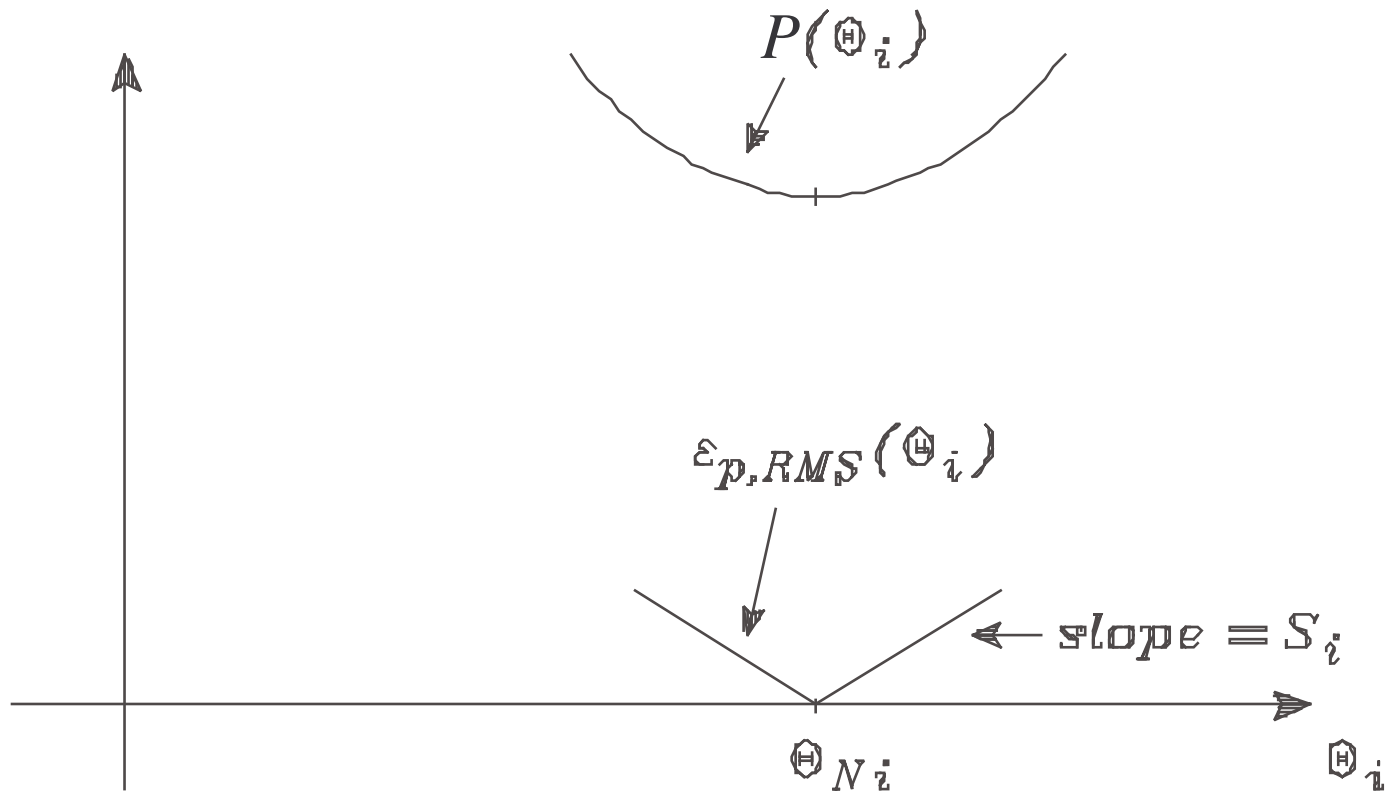
$$\begin{aligned}\epsilon_{p,RMSn}(\theta) &= \sqrt{\frac{1}{n} \sum_{k=1}^N \epsilon_{pn}^2(k, \theta)} \approx \sqrt{\frac{1}{n} \sum_{k=1}^N (\theta_r - \theta_{Nr})^\top \psi_{rn}^\top \psi_{rn} (\theta_r - \theta_{Nr})} \\ &= \sqrt{(\theta_r - 1_v)^\top \tilde{H}_{rn} (\theta_r - 1_v)}\end{aligned}$$

idet $\tilde{H} = \frac{1}{N} \psi^\top \psi$, $\theta_{ri} = \frac{\theta_i}{\theta_{Ni}}$, $\theta_{Nri} = \frac{\theta_{Ni}}{\theta_{Ni}}$ og $\theta_{Nr} = 1_v$.

Parameter følsomhed m.h.t. den i 'te parameter θ_i :

$$S_i = \frac{\partial \epsilon_{p,RMSn}}{\partial \theta_{ri}} = \sqrt{h_{rni}} \quad \left(= \{ \tilde{H}_{rn} \}_{ii} \right)$$

Parameter følsomhed



Følsomhedsellipse

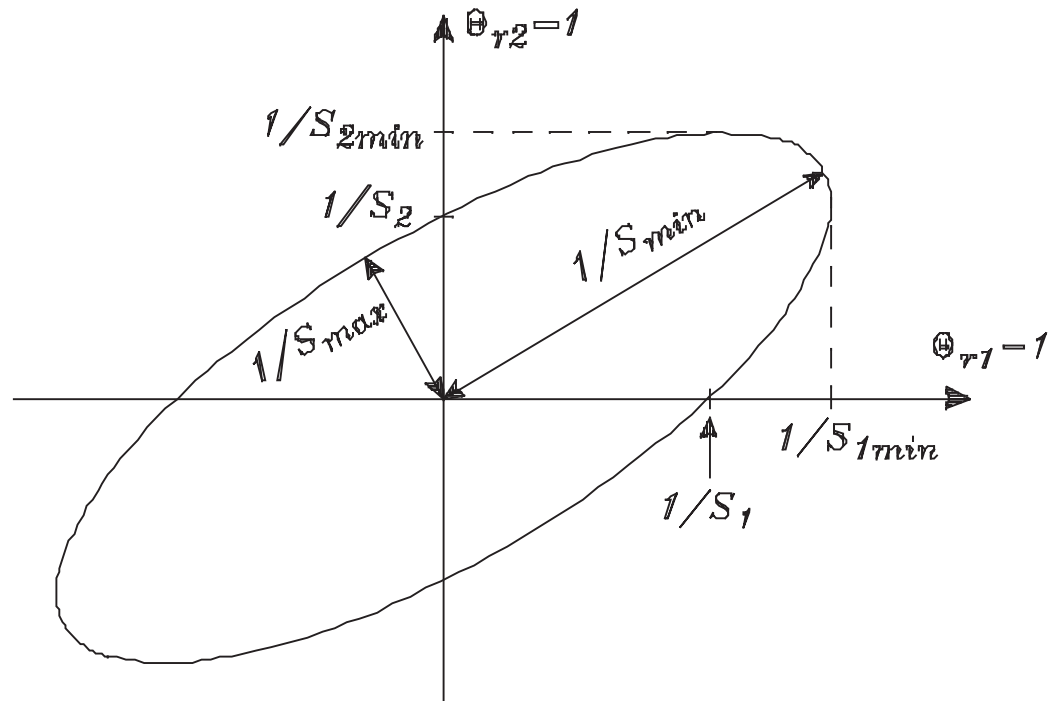
Tilfældet to parametre: $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

$\epsilon_{p,RMSn}(\theta_r) = 1$ kaldes følsomhedsellipsen

idet $x^\top H x = c$ er forskriften for en ellipse (H symmetrisk).

I sådan en ellipse er afstanden fra (0,0) til et punkt på ellipsen $(x_i, 0) = \frac{c}{\sqrt{h_{ii}}}$

Følsomhedsellipse



- S_i følsomhed af θ_i alene
- $S_{i \min}$ minimum følsomhed af θ_i
- S_{\min} minimum følsomhed i vilkårlig retning
- S_{\max} maksimum følsomhed i vilkårlig retning



Følsomhedsellipse

Disse karakteristiske mål er de mest beskrivende, specielt for flere end 2 parametre:

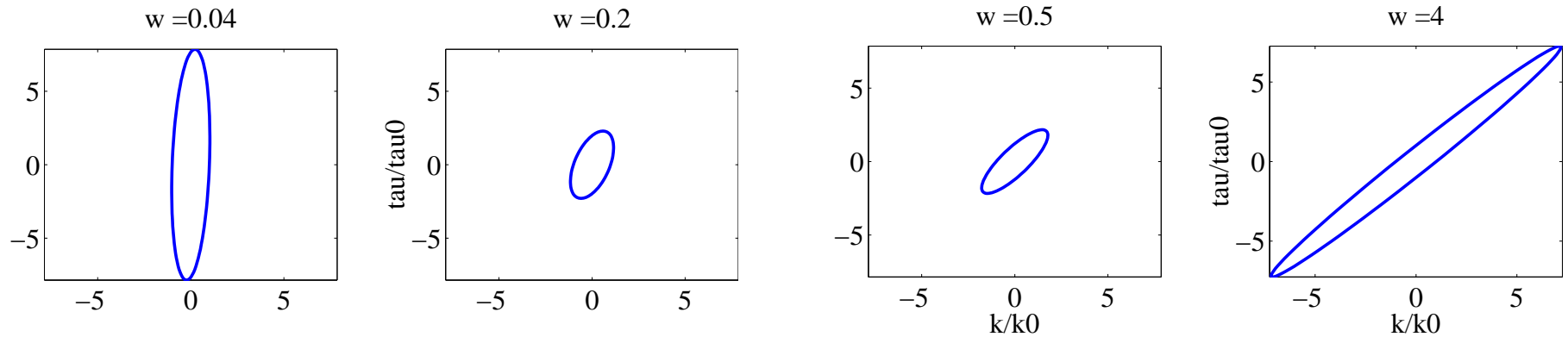
S_{\min} minimum følsomhed, reciprok af major half axis
– så stor som muligt

$S_{i \min}$ minimum følsomhed af θ_i – så stor som muligt

$R = \frac{S_{\max}}{S_{\min}}$ forhold mellem max. og min. følsomhed i
vilkårlig retning – så tæt på 1 som muligt

$R_i = \frac{S_i}{S_{i \min}}$ forhold mellem følsomhed af θ_i alene og min.
følsomhed af θ_i – så tæt på 1 som muligt.
 $R_i \gg 1$ indikerer at to eller flere parametre er
korrollerede

Følsomhedsellipse



ω	0.04	0.2	0.5	4
S_{\min}	0.13	0.42	0.37	0.10
$S_{i \min}$	0.98/0.13	0.86/0.44	0.56/0.46	0.14/0.14
R	7.9	2.5	3.3	13.8
R_i	1.03/1.03	1.2/1.2	1.8/1.8	6.9/6.9

Næste Forelæsning

Næste gang ser vi på:

- Parameter nøjagtighed
- Input-signal design