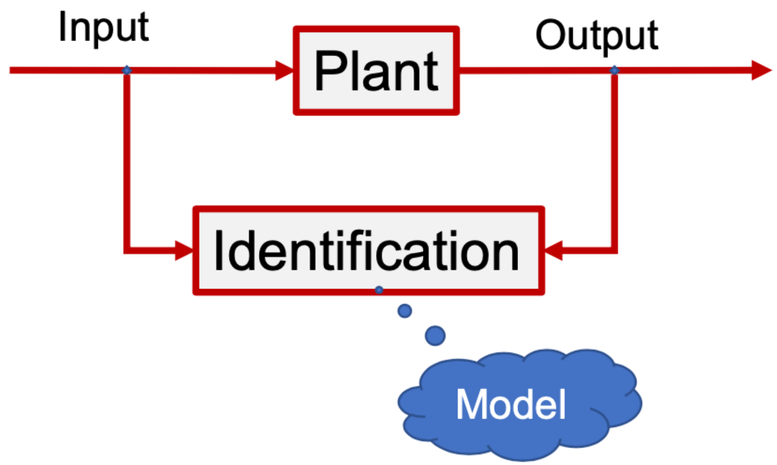
Apprentissage de modèles dynamiques

Learning flexible continuous-time models of linear dynamical systems

Hugues GARNIER

Aim of this lecture

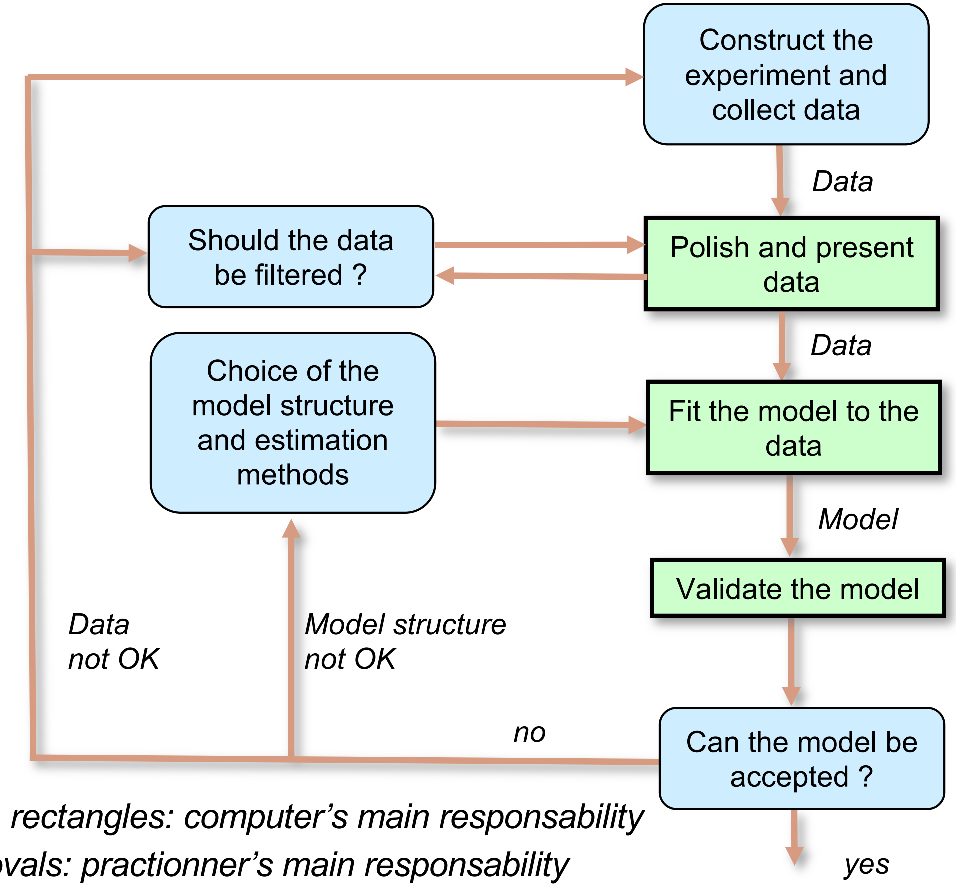


- ✓ To provide an introduction
  - theory of direct time-domain methods for ***continuous-time*** parametric linear black-box model identification

- ✓ The key computational method, we refer to, is
  - *Optimal Instrumental Variable (IV)* method

The system identification procedure

- System Identification: an ***iterative procedure***



Green rectangles: computer's main responsability
Blue ovals: practionner's main responsability
Adapted from Ljung 1999

The practionner has to make many choices:

- ✓ well-planned data acquisition
  - ✓ Sampling period, type of input, …
- ✓ data-preprocessing
  - ✓ Filtering, detrending …
- ✓ type of models to be estimated*:*
  - ✓ linear or non linear
  - ✓ ***continuous or discrete-time***
- ✓ estimation methods
  - ✓ ***PEM or IV***

These choices will impact the SYSID procedure and require active participation of a specifically trained practitioner !

H. Garnier

Continuous-time (CT) models of linear systems

- A model that describes the relationship between time continuous I/O signals is called a *continuous-time model*
  - *Differential equation / polynomial / transfer function model*

$$\frac{d^n y(t)}{dt^n} + a_1 \frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_n y(t) = b_0 \frac{d^m u(t)}{dt^m} + b_1 \frac{d^{m-1} u(t)}{dt^{m-1}} + \cdots + b_m u(t)$$

$$A(p)y(t) = B(p)u(t)$$

$pu(t) = \frac{du(t)}{dt}$ differentiation operator

$$A(p) = p^n + a_1 p^{n-1} + \cdots + a_n$$
$$B(p) = b_0 p^m + b_1 p^{m-1} + \cdots + b_m$$

$$G(s) = \frac{Y(s)}{U(s)} = \frac{B(s)}{A(s)}$$

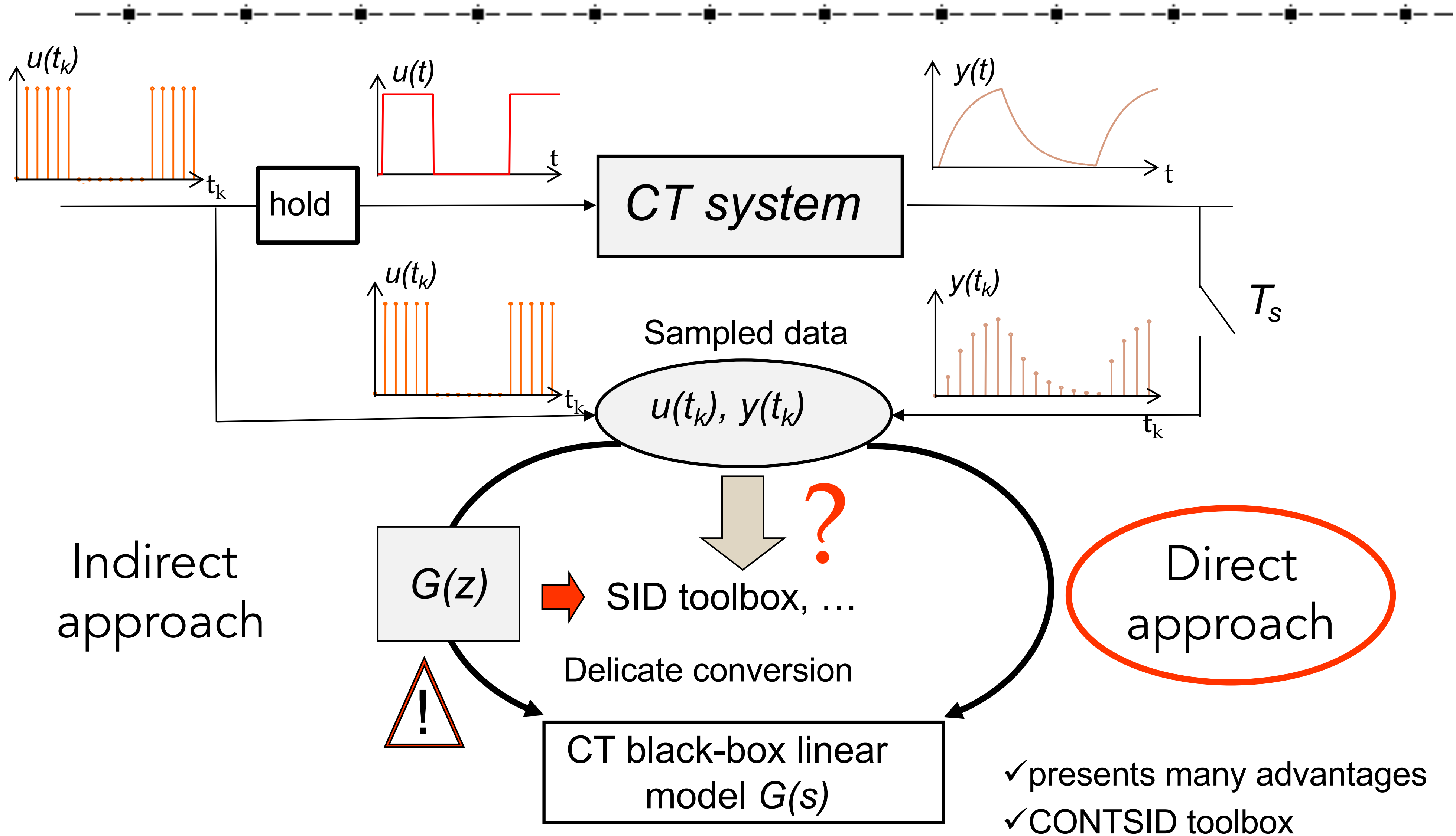
s: Laplace variable

  - *State-space model*

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

$$G(s) = C\left(sI - A\right)^{-1} B + D$$

Main approaches to identify a black-box CT linear models from time-domain sampled data ?



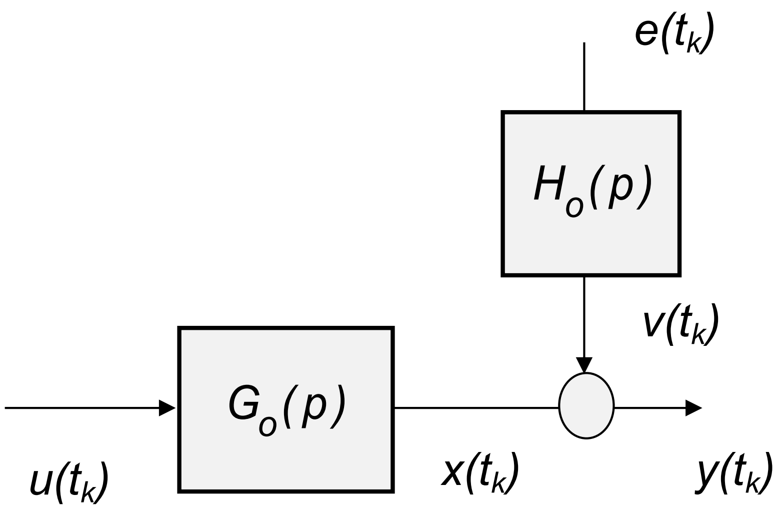
✓presents many advantages
✓CONTSID toolbox

The myth of the true data-generating system

- ✓ The mathematical model that will be identified from finite sampled data will be an approximation to the real system
- ✓ It is inexact and the data is never generated in practice from a system which "belongs to the model class"
- ✓ Nevertheless we shall find it convenient to assume such a true data-generating system to assist in deriving theoretical results
- ✓ But we do not believe that it truly captures the behavior of the physical system

True data-generating linear system

- ✓ Assumptions about the true system: $S = \{G_o(p); H_o(p)\}$



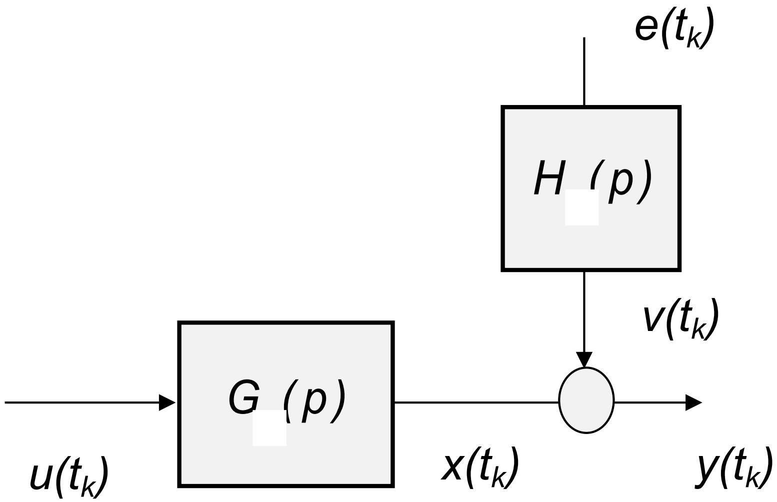
$$y(t_k) = G_o(p)u(t_k) + H_o(p)e(t_k)$$

$p = \dfrac{d}{dt}$ differentiation operator

- ✓ The measured output $y(t_k)$ is assumed to be made up of two distinct contributions:
  - $G_o(p)u(t_k)$: dependent of the choice of the input signal $u(t)$
  - the measurement noise $v(t_k)=H_o(p)e(t_k)$: independent of the input signal $u(t)$

The chosen model structure to capture the dynamics of the linear system

- ✓ Assumptions about the model class: $\mathcal{M} = \left\{ \left(G(p,\theta) \ ; \ H(p,\theta) \right), \ \theta \in R^{n_\theta} \right\}$



$$y(t_k) = G(p)u(t_k) + H(p)e(t_k)$$

$p = \dfrac{d}{dt}$ differentiation operator

- ✓ The model structure is assumed *a priori* known. 2 cases can be distinguished

$\mathcal{S} \in \mathcal{M}$
- ♦ Model form and order for *G* and *H* identical to G_o and H_o

$\mathcal{S} \notin \mathcal{M}, \mathcal{G} \in \mathcal{G}_o$
- ♦ Model form and order for *G* identical to G_o but *H different to* H_o

Black-box continuous-time model structures

✓ Model structure: $\mathcal{M} = \left\{ \left(G(p,\theta) \;\; ; \;\; H(p,\theta) \right), \;\; \theta \in R^{n_\theta} \right\}$

✓ General parametrization

Time-delay assumed known in the beginning

$$G(p,\theta) = \frac{B(p,\theta)}{A(p,\theta)} e^{-\tau p} \qquad H(p,\theta) = \frac{C(p,\theta)}{D(p,\theta)}$$

$$\theta^T = \begin{bmatrix} a_1 & \dots & a_n & b_0 & \dots & d_1 & \dots \end{bmatrix}$$

$$A(p,\theta) = p^n + a_1 p^{n-1} + \dots + a_n$$

$$B(p,\theta) = b_0 p^m + b_1 p^{m-1} + \dots + b_m$$

$$C(p,\theta) = p^{n_c} + c_1 p^{n_c-1} + \dots + c_{n_c}$$

$$D(p,\theta) = p^{n_d} + d_1 p^{n_d-1} + \dots + d_{n_d}$$

Main black-box CT model structures

- ✓ Main model structures used in practice: $\mathcal{M} = \left\{ \left(G(p,\theta) \;\; ; \;\; H(p,\theta) \right), \;\; \theta \in R^{n_\theta} \right\}$

CARX	$G(p,\theta) = \dfrac{B(p,\theta)}{A(p,\theta)} e^{-\tau p}$	$H(p,\theta) = \dfrac{1}{A(p,\theta)}$
COE	$G(p,\theta) = \dfrac{B(p,\theta)}{A(p,\theta)} e^{-\tau p}$	$H(p,\theta) = 1$
CBJ	$G(p,\theta) = \dfrac{B(p,\theta)}{A(p,\theta)} e^{-\tau p}$	$H(p,\theta) = \dfrac{C(p,\theta)}{D(p,\theta)}$
hybrid CBJ	$G(p,\theta) = \dfrac{B(p,\theta)}{A(p,\theta)} e^{-\tau p}$	$H(q,\theta) = \dfrac{C(q,\theta)}{D(q,\theta)}$

Distinction between model structures

✓ CARX model can be written in linear regression form

$$A(p,\theta)y(t_k) = B(p,\theta)u(t_k) + e(t_k)$$
$$y^{(n)}(t_k) = \varphi^T(t_k)\theta + e(t_k)$$

- ☹ The model is not very realistic in practice
  ⇒ There are common denominators in *G* and *H*

- ☺ The model is a linear function in θ
  ⇒ Important computational advantages

✓ COE and CBJ models have an independent parametrization of $G(p,\theta)$ and $H(p,\theta)$

$$y(t_k) = \frac{B(p,\theta)}{A(p,\theta)}u(t_k) + e(t_k) \qquad y(t_k) = \frac{B(p,\theta)}{A(p,\theta)}u(t_k) + \frac{C(p,\theta)}{D(p,\theta)}e(t_k)$$

- ☹ Models are no longer linear-in-the-parameters
- ☺ There are no common parameters in *G* and *H*
  ⇒ Advantages for independent identification of *G* and *H* and models more realistic in practice

Parameter estimation objective and assumptions

- ✓ Objective:
  - Find the best parametric models $G(p,\theta)$ and $H(p,\theta)$ $\mathcal{M} = \left\{ \left(G(p,\theta) \ ; \ H(p,\theta) \right), \ \theta \in R^{n_\theta} \right\}$
    for the unknown transfer functions $G_o(p)$ and $H_o(p)$
    using a set of measured data $u(t_k)$ and $y(t_k)$

- ✓ In the beginning, we will make the following assumption:

$$\exists \quad \theta_o \quad \text{such that} \quad G(p,\theta_o) = G_o(p) \quad \text{and} \quad H(p,\theta_o) = H_o(p)$$

i.e.

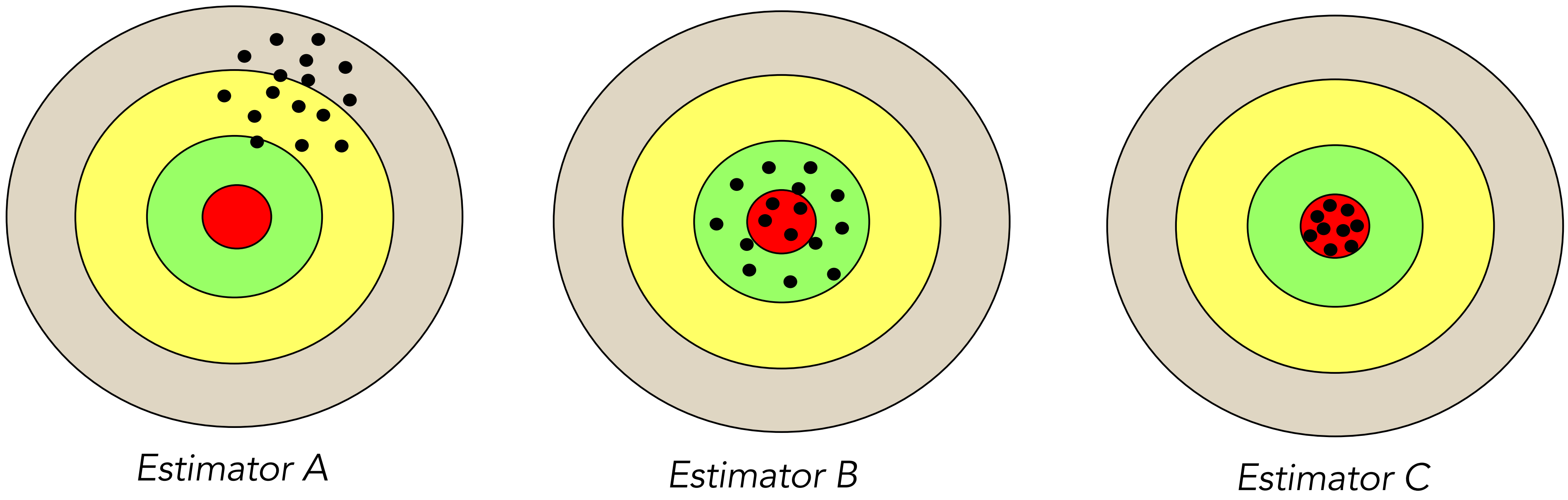
$$S \in \mathcal{M}$$

- ✓ The objective can therefore be restated as follows:
  - *Find an estimate of the unknown parameter vector θ_o using a set of N samples of the input and output data:*

$$Z^N = \{ u(t_k), y(t_k) \mid k = 1 \ldots N \}$$

generated by the true system, i.e. $y(t_k) = G_o(p)u(t_k) + H_o(p)e(t_k)$

Illustration of bias-variance trade-off for estimators



Center of the dartboard target *(in red)* represents θ_o

- ***Estimator A:*** biased *(average value of the estimates are not in the center of the target)*
- ***Estimator B:*** unbiased but quite large fluctuations around the mean value – large variance
- ***Estimator C***: unbiased and small variance

Issue in CT model identification: time-derivative measurement problem

✓ *DT* model identification - ***difference*** equation model

$$y(k)+a_1y(k-1)+\cdots+a_{n_a}y(k-n_a)=b_1u(k-1)+\cdots+b_{n_b}u(k-n_b-1)$$

✓ *CT* model identification - ***differential*** equation model

Unlike the *DT* model, where only sampled input and output data appear, the CT *differential* equation (DE) model contains I/O ***time-derivatives***

$$\frac{d^ny(t)}{dt^n}+a_1\frac{d^{n-1}y(t)}{dt^{n-1}}+\cdots+a_ny(t)=b_0\frac{d^mu(t)}{dt^m}+b_1\frac{d^{m-1}u(t)}{dt^{m-1}}+\cdots+b_mu(t)$$

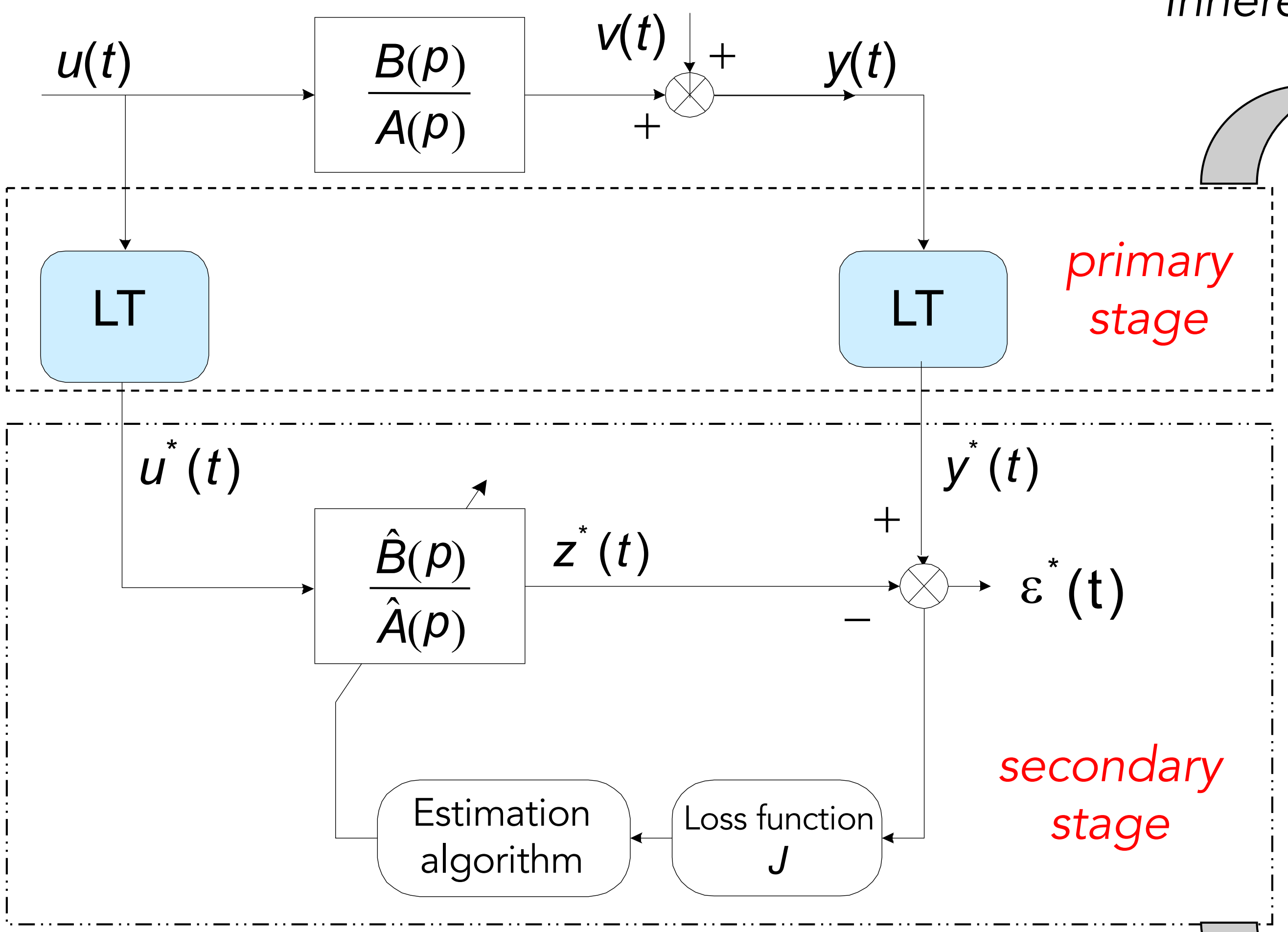
Not measured in most practical cases

Well-known approach to handle the time-derivative problem:
*Apply a **linear transform** to both I/O data can be seen as a **data prefiltering** strategy*

Two-stage approach for direct CT model identification

LT : *Linear Transforms*

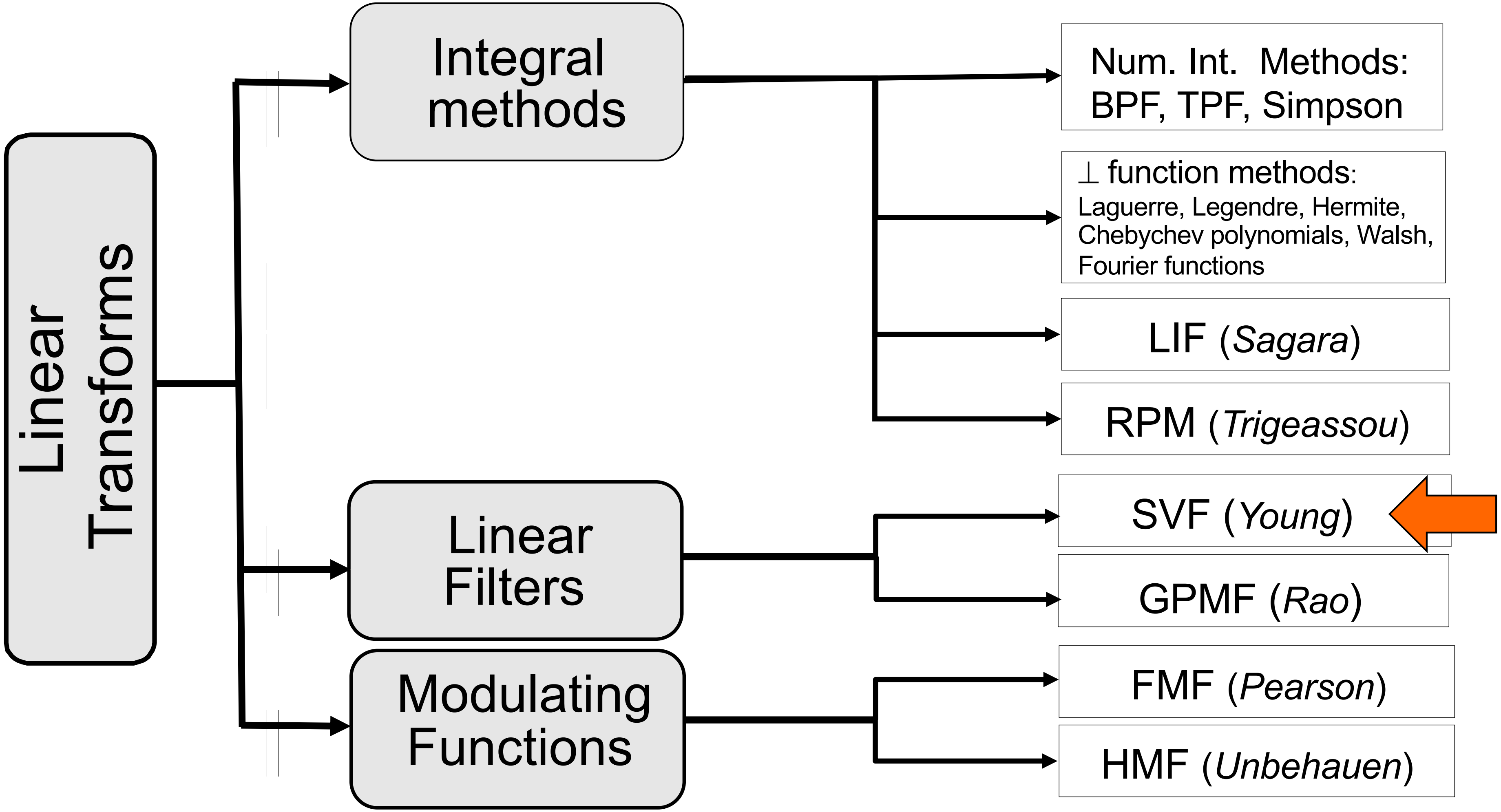
Inherent data prefiltering



Arises out of the non-measurable I/O time-derivatives problem

Most of the standard DT parametric model estimation methods can be used

Main linear transforms developed for the primary stage



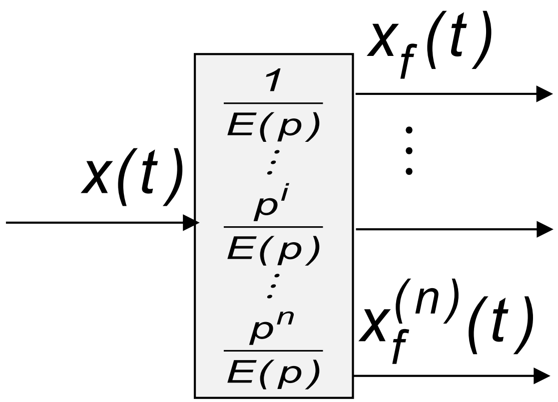
H. Garnier, M. Mensler, A. Richard, *Continuous-time model identification from sampled data: implementation issues and performance evaluation.* IJC, 76(13), 2003

Traditional State Variable Filtering (SVF) method

$$y^{(n)}(t)+a_1y^{(n-1)}(t)+\cdots+a_ny(t)=b_0u^{(m)}(t)+\cdots+b_mu(t)$$

Apply a stable SVF filter *L(p)=1/E(p)* on both sides, the prefiltered DE model obeys exactly *(except for a possible transient)*

$$y_f^{(n)}(t)+a_1y_f^{(n-1)}(t)+\cdots+a_ny_f(t)=b_0u_f^{(m)}(t)+\cdots+b_mu_f(t)$$



Bank of SVF filters

How to choose *L(p)=1/E(p)* ?

$$E(p)=\left(p+\lambda\right)^n$$

The filtered time-derivatives can then be exploited to estimate the parameters of the differential equation model

Bode plot of SVF filters

- ✓ The outputs of the SVF filter bank will provide a smoothed estimate of the I/O time-derivatives in the frequency band of interest

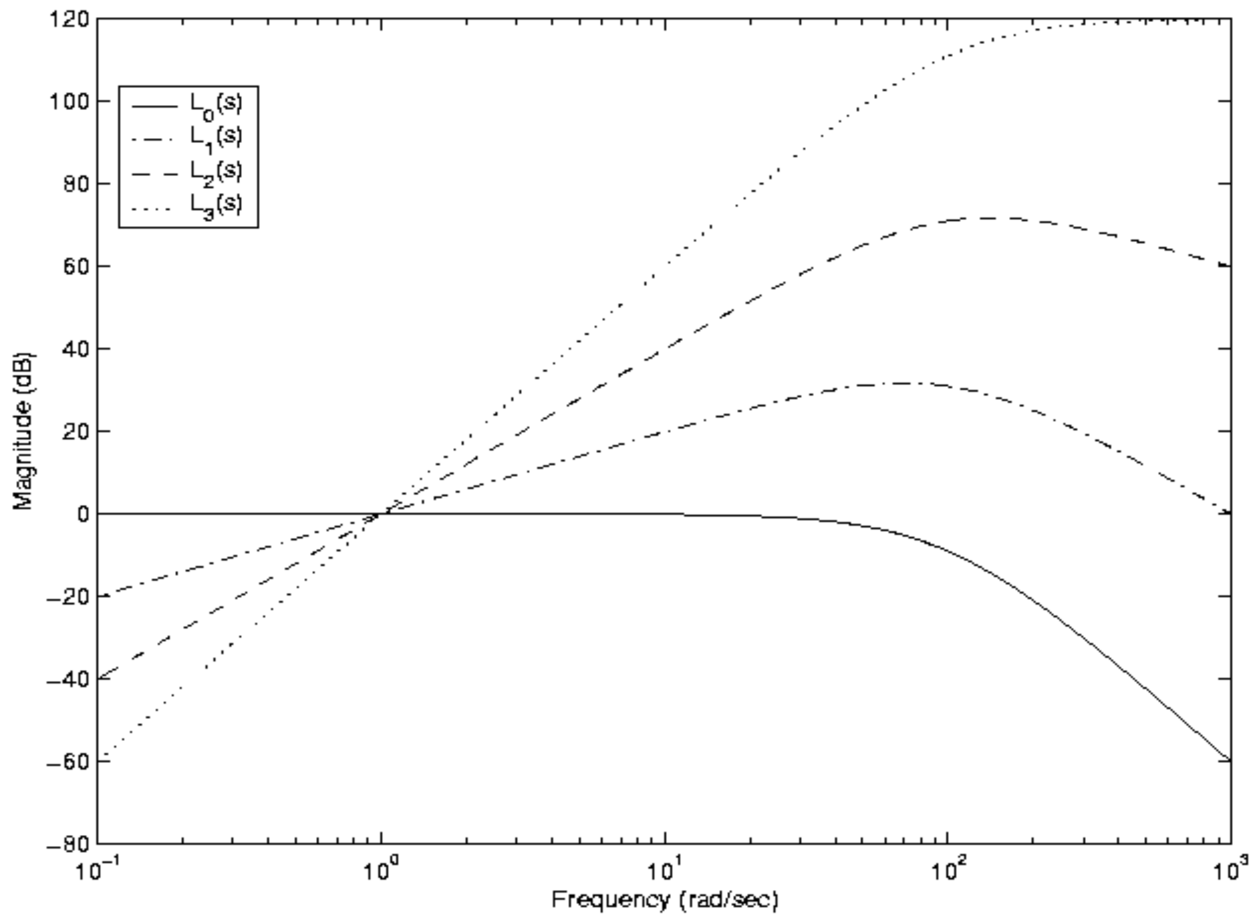
$$L_i(s) = \frac{s^i}{E(s)} = \frac{s^i}{(s+\lambda)^n}$$

$$L_0(s) = \frac{1}{(s+\lambda)^3}$$

$$L_1(s) = \frac{s}{(s+\lambda)^3}$$

$$L_2(s) = \frac{s^2}{(s+\lambda)^3}$$

$$L_3(s) = \frac{s^3}{(s+\lambda)^3}$$



Simple least squares-based SVF estimator

- ✓ At $t=t_k$, the prefiltered DE model can be rewritten in linear regression form

$$y_f^{(n)}(t_k) = \varphi_f^T(t_k)\theta + \varepsilon(t_k)$$

$$\varphi_f^T(t_k) = \begin{bmatrix} -y_f^{(n-1)}(t_k) & \cdots & -y_f(t_k) & u_f^{(m)}(t_k) & \cdots & u_f(t_k) \end{bmatrix}$$

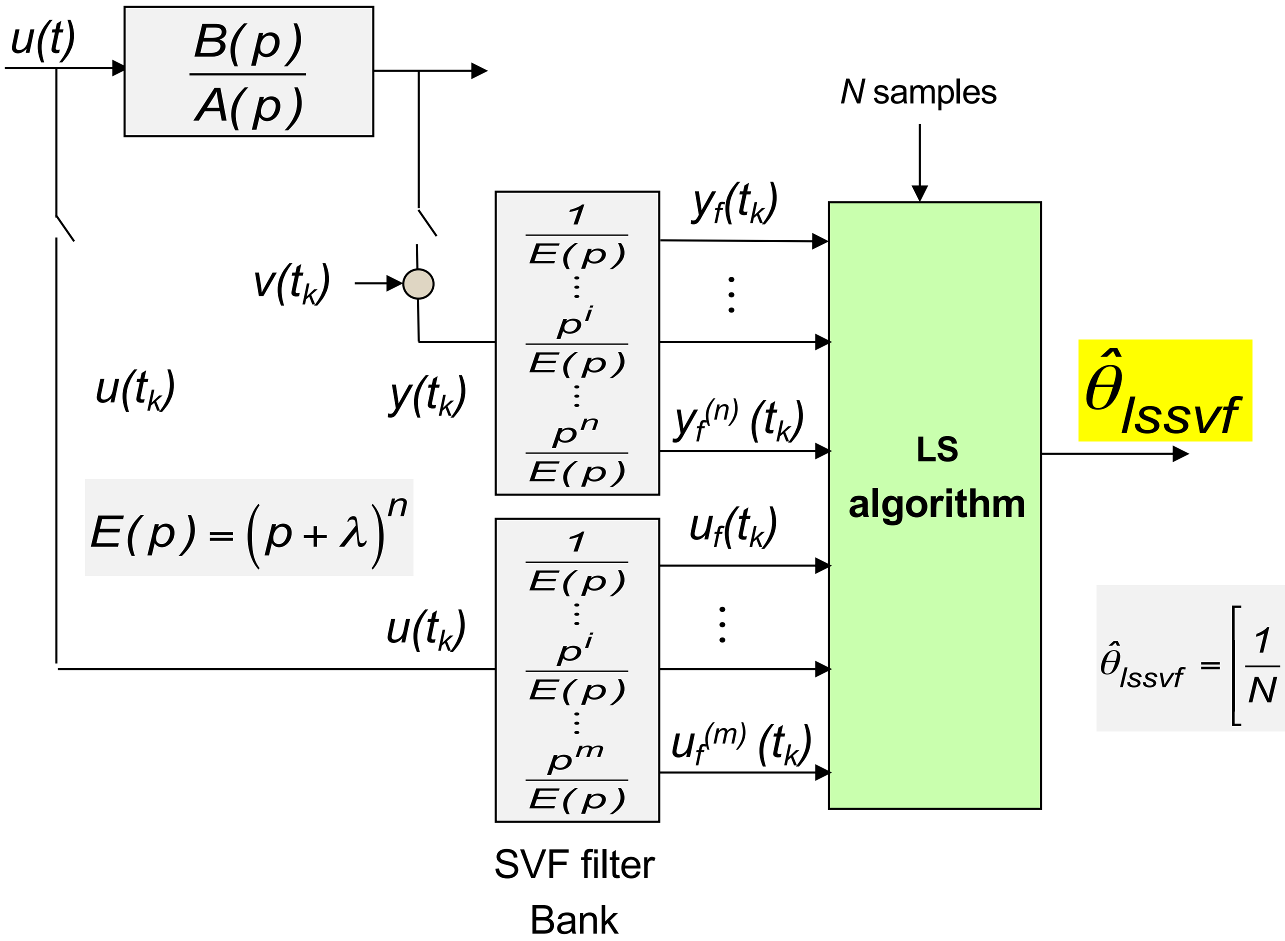
$$\theta = \begin{bmatrix} a_1 & \cdots & a_n & b_0 & \cdots & b_m \end{bmatrix}^T$$

- ✓ From N samples observed at $t_1, \ldots t_N$, the LS-based SVF parameter estimates are computed as

$$\hat{\theta}_{lssvf} = \arg\min_{\theta} \left(\frac{1}{N} \sum_{k=1}^{N} \left(y_f^{(n)}(t_k) - \varphi_f^T(t_k)\theta \right)^2 \right)$$

$$\hat{\theta}_{lssvf} = \left[\frac{1}{N} \sum_{k=1}^{N} \varphi_f(t_k)\varphi_f^T(t_k) \right]^{-1} \left[\frac{1}{N} \sum_{k=1}^{N} \varphi_f(t_k) y_f^{(n)}(t_k) \right]$$

Simple LS-based SVF estimator



$$\hat{\theta}_{lssvf} = \left[\frac{1}{N}\sum_{k=1}^{N}\varphi_f(t_k)\varphi_f^T(t_k)\right]^{-1}\left[\frac{1}{N}\sum_{k=1}^{N}\varphi_f(t_k)y_f^{(n)}(t_k)\right]$$

This simple LS-based SVF estimator represents the simplest archetype of CT model identification from sampled data

LSSVF method - Example

✓ Consider a second-order system

$$y^{(2)}(t)+a_1y^{(1)}(t)+a_2y(t)=b_0u(t)+e(t)$$

$$\left(p^2+a_1p+a_2\right)y(t)=b_0\,u(t)+e(t)$$

✓ Apply a second-order SVF filter $L(p)=1/(p+\lambda)^2$

$$\left(\frac{p^2}{(p+\lambda)^2}+a_1\frac{p}{(p+\lambda)^2}+a_2\frac{1}{(p+\lambda)^2}\right)y(t)=\left(b_0\frac{1}{(p+\lambda)^2}\right)u(t)+\frac{1}{(p+\lambda)^2}e(t)$$

$$y_f^{(2)}(t)+a_1y_f^{(1)}(t)+a_2y_f(t)=b_0u_f(t)+e_f(t)$$

✓ At $t=t_k$

$$y_f^{(2)}(t_k)=\begin{bmatrix}-y_f^{(1)}(t_k) & -y_f(t_k) & u_f(t_k)\end{bmatrix}\begin{bmatrix}a_1\\a_2\\b_0\end{bmatrix}+e_f(t_k)$$

LSSVF method - Example

- ✓ At $t=t_k$

$$y_f^{(2)}(t_k) = \begin{bmatrix} -y_f^{(1)}(t_k) & -y_f(t_k) & u_f(t_k) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_0 \end{bmatrix} + e_f(t_k)$$

- ✓ For $t=t_1,\dots t_N$, we have

$$\begin{bmatrix} y_f^{(2)}(t_1) \\ y_f^{(2)}(t_2) \\ \vdots \\ y_f^{(2)}(t_N) \end{bmatrix} = \begin{bmatrix} -y_f^{(1)}(t_1) & -y_f(t_1) & u_f(t_1) \\ -y_f^{(1)}(t_2) & -y_f(t_2) & u_f(t_2) \\ \vdots & \vdots & \vdots \\ -y_f^{(1)}(t_N) & -y_f(t_N) & u_f(t_N) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_0 \end{bmatrix} + \begin{bmatrix} e_f(t_1) \\ e_f(t_2) \\ \vdots \\ e_f(t_N) \end{bmatrix}$$

$$Y_N = \Phi_N \quad \theta \quad + \quad E_N$$

$$\hat{\theta}_{lssvf} = \left[\Phi_N^T \Phi_N\right]^{-1} \Phi_N^T Y_N$$

LS-based SVF estimator – Implementation aspects

$$\hat{\theta}_{lssvf} = \left[\Phi_N^T \Phi_N\right]^{-1} \Phi_N^T Y_N = \left[\sum_{k=1}^{N} \varphi_f(t_k)\varphi_f^T(t_k)\right]^{-1} \left[\sum_{k=1}^{N} \varphi_f(t_k) y_f^{(n)}(t_k)\right]$$

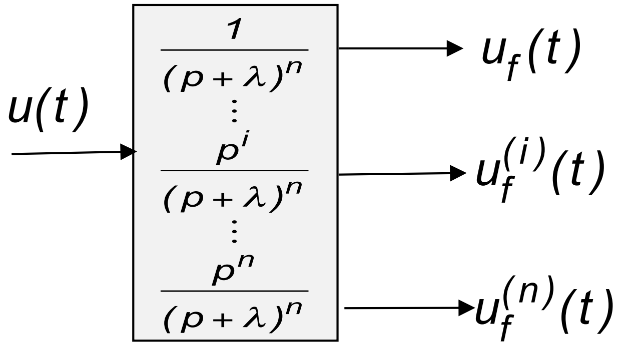
- ✓ Do not compute the normal equation solution above, but use instead numerically stable and computationally efficient algorithms for computing the LS-based SVF estimates :
  - SVD – Singular Value Decomposition (*pinv* in Matlab)
    - Θ=*pinv*(Φ)*Y computes the solution to $Y=\Phi\,\Theta$
  - QR factorization (matrix division \ in Matlab)
    - $\Theta = \Phi \backslash Y$ computes also the solution to $Y=\Phi\,\Theta$
- ✓ Recommended implementation of the LSSVF solution in Matlab

$$\hat{\theta}_{lssvf} = \Phi_N \backslash Y_N$$

SVF-based estimators – Implementation aspects

✓ Roles of the SVF filters

- Reconstruct the time-derivatives in the bandwidth of interest
- Improve the statistical efficiency of the estimates (filter out the high-frequency noise)



✓ User parameters of the SVF filter

- Filter order: should be chosen larger or equal than the system order *n*
  - *Simplest choice: minimal-order SVF,* $L(s)=1/(s+\lambda)^n$
  - *Note that so called minimal-order GPMF where* $L(s)=1/(s+\lambda)^{n+1}$ *is often more robust against the noise than basic SVF (see lsgpmf in CONTSID)*
- Cut-off frequency λ of the SVF filter $L(s)=1/(s+\lambda)^n$, chosen in order to emphasize the frequency band of interest

H. Garnier

SVF-based estimators – Implementation aspects

✓ Digital implementation of the CT SVF filtering operations

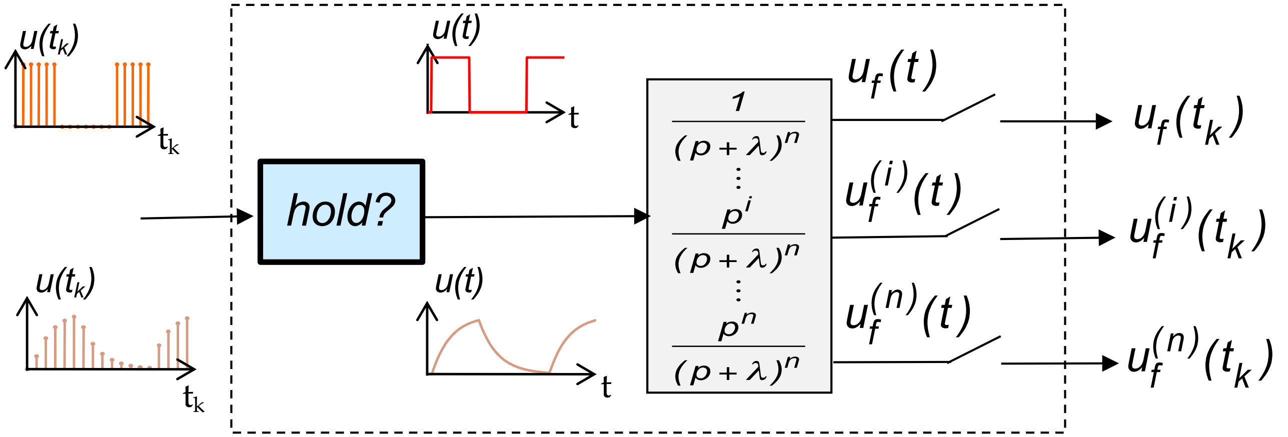
- The computation of the LSSVF parameter estimates requires the value of prefiltered signals at the time-instants t_k , $k = 1, \ldots, N$

$$\begin{bmatrix} y_f^{(2)}(t_1) \\ y_f^{(2)}(t_2) \\ \vdots \\ y_f^{(2)}(t_N) \end{bmatrix} = \begin{bmatrix} -y_f^{(1)}(t_1) & -y_f(t_1) & u_f(t_1) \\ -y_f^{(1)}(t_2) & -y_f(t_2) & u_f(t_2) \\ \vdots & \vdots & \vdots \\ -y_f^{(1)}(t_N) & -y_f(t_N) & u_f(t_N) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_0 \end{bmatrix} + \begin{bmatrix} e_f(t_1) \\ e_f(t_2) \\ \vdots \\ e_f(t_N) \end{bmatrix}$$

$$Y_N = \Phi_N \theta + E_N$$

$$\hat{\theta}_{lssvf} = \Phi_N \backslash Y_N$$

- The digital implementation method has to be selected carefully according to the assumption about the filter input intersample behavior: *choice of the hold block*



Digital implementation of the CT SVF filtering operations

- If the filter input intersample behavior is known (*e.g.* piecewise constant or piecewise linear) or if the input takes a particular form (*e.g.* a sine or sum of sines):
  - an exact solution to the filtering operation at specified time-instants can be obtained

- If the filter input intersample behavior is not known:
  - approximate solution to the filtering operation can be obtained only
    - approximation errors depend on T_s and fast sampling is often preferred in CT model identification
    - *Fast sampling is however not required for all CT methods, e.g. SRIVC (see later on)*

- One efficient approach is implemented in the Matlab ***lsim*** routine
  - where the state-space representation of the SVF filter bank is discretized assuming the best *zoh or foh* assumption for the input intersample behavior

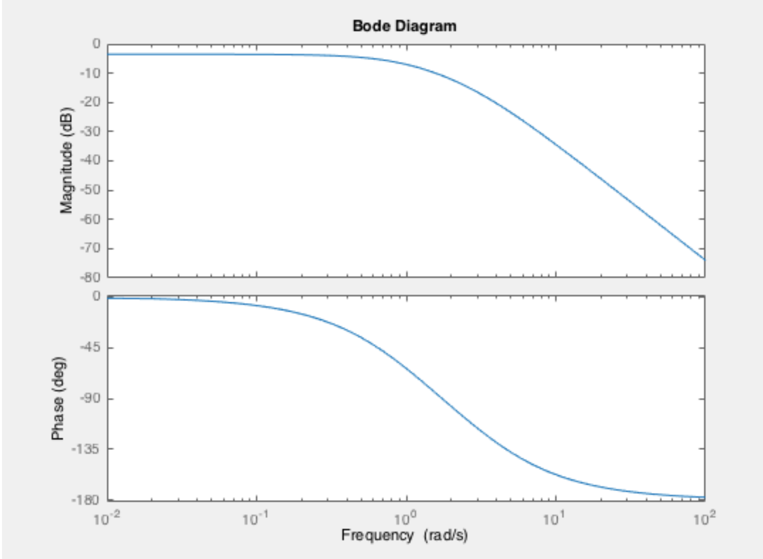
$$\begin{cases} \dot{x}(t) = A_c x(t) + B_c u(t) \\ y(t) = Cx(t) \end{cases} \xrightarrow[T_s]{\textit{hold}} \begin{cases} x(t_{k+1}) = F_d x(t_k) + G_d u(t_k) \\ y(t_k) = Cx(t_k) \end{cases}$$

LSSVF implementation in Matlab – 2nd-order example

✓ Simple second-order COE model

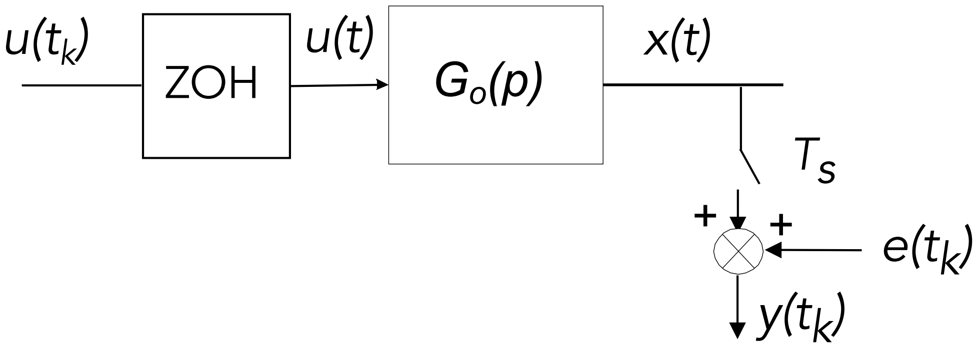
$$\begin{cases} x(t) = G_o(p)u(t) \\ y(t_k) = x(t_k) + e(t_k) \end{cases}$$

$$G_o(p) = \frac{2}{(p+3)(p+1)} = \frac{2}{p^2+4p+3}$$



✓ Simulations conditions

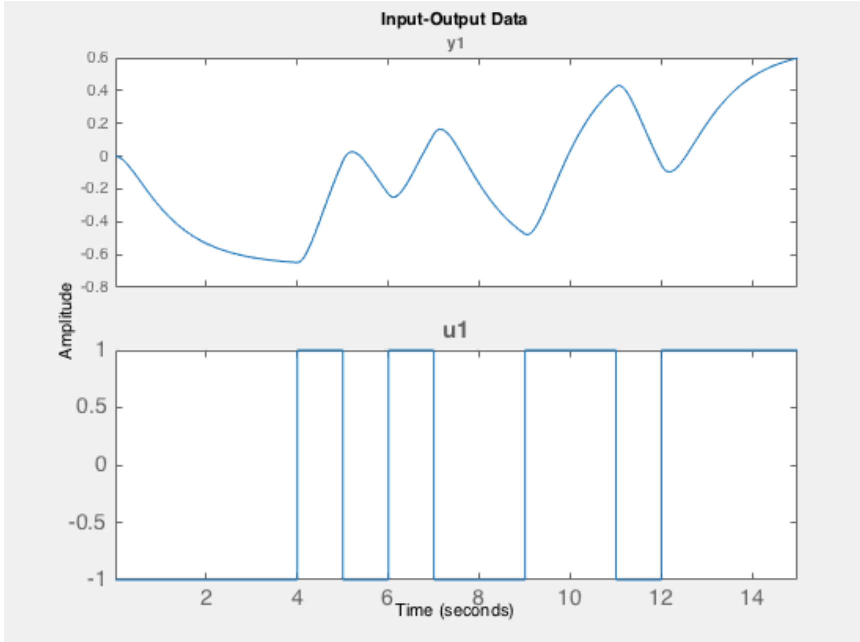
- $u(t)$: PRBS
- $T_s = 10$ ms
- $N = 1500$
- *2 output measurement situations*
  - *Noise-free*
  - $e(t_k)$: white Gaussian noise, $\sigma_e = 0.2$



H. Garnier

LSSVF estimator - Matlab implementation – Noise-free case

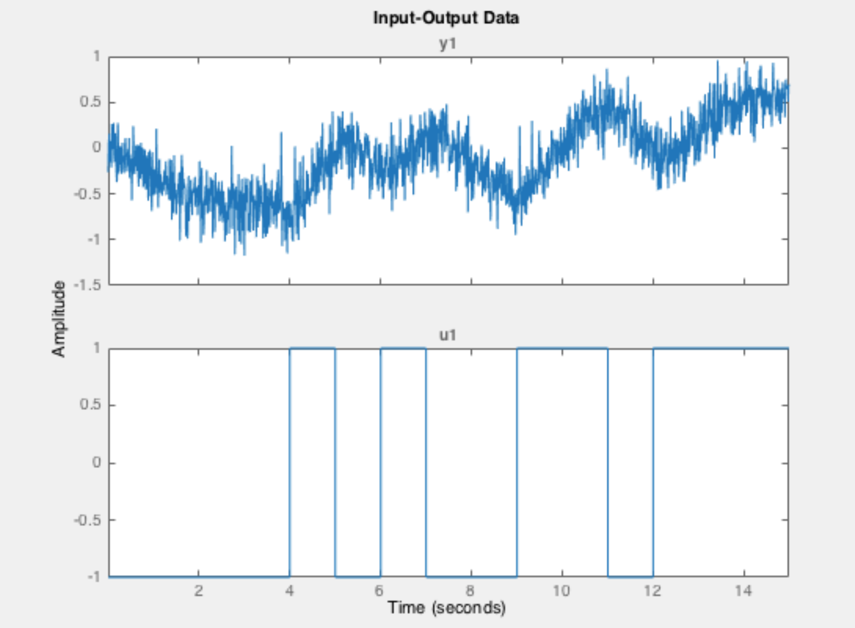
```
B=2;                            % B(p)=2
A=[1 4 3];                      % A(p)=p^2+4p+3 – True system
Ts=0.01;
u=prbs(4,100);                  % PRBS input  from the CONTSID
N=1500;
t=(0:N-1)'*Ts;
x=lsim(B,A,u,t);                % simulation of the noise-free output
data0=iddata(x,u,Ts);idplot(data0);
% Primary stage - SVF filtering
lambda=3;                       % l: SVF filter cut-off frequency
den_L=[1 2*lambda lambda^2];    % denominator of the SVF filter
num_L0=1;                       % numerator of L0(p)=1/(p+λ)^2
num_L1=[1 0];                   % numerator of L1(p) =p/(p+λ)^2
num_L2=[1 0 0];                 % numerator of L2(p) =p^2/(p+λ)^2
xf0=lsim(num_L0,den_L,x,t);     % Computation of the SVF filter bank outputs
xf1=lsim(num_L1,den_L,x,t);
xf2=lsim(num_L2,den_L,x,t);
uf0=lsim(num_L0,den_L,u,t);
% Secondary stage - LS estimates
Phi_N=[-xf1 -xf0 uf0];          % Regression matrix
Y_N=xf2;                        % Output vector
theta_lssvf=Phi_N\Y_N           % LSSVF estimates
theta_lssvf'                    % see also the LSSVF routine in the CONTSID toolbox
3.9997    2.9998    1.9999      % Mlssvf=lssvf(data0,[2 1 0],lambda)
```



LSSVF estimator - Matlab implementation – Noisy case

```
B=2;                        % B(p)=2
A=[1 4 3];                  % A(p)=p^2+4p+3 – True system
u=prbs(4,100);              % PRBS input  from the CONTSID
N=1500; Ts=0.01;
t=(0:N-1)'*Ts;
x=lsim(B,A,u,t);            % simulation of the noise-free output
y=x+0.2*randn(N,1);         % white noise added to the noise free output
data=iddata(y,u,Ts);idplot(data);
% Primary stage - SVF filtering
lambda=3;                   % l: SVF filter cut-off frequency
den_L=[1 2*lambda lambda^2]; % denominator of the SVF filter
num_L0=1;                   % numerator of L0(p)=1/(p+λ)^2
num_L1=[1 0];               % numerator of L1(p) =p/(p+λ)^2
num_L2=[1 0 0];             % numerator of L2(p) =p^2/(p+λ)^2
yf0=lsim(num_L0,den_L,y,t); % Computation of the SVF filter bank outputs
yf1=lsim(num_L1,den_L,y,t);
yf2=lsim(num_L2,den_L,y,t);
uf0=lsim(num_L0,den_L,u,t);
% Secondary stage - LS estimates
Phi_N=[-yf1 -yf0 uf0];      % Regression matrix
Y_N=yf2;                    % Output vector
theta_lssvf=Phi_N\Y_N       % LSSVF estimates
theta_lssvf'                % see also the LSSVF routine in the CONTSID toolbox
3.2542    2.6889    1.6865  % Mlssvf=lssvf(data,[2 1 0],lambda)
```

$\mathcal{S} \notin \mathcal{M}, \mathcal{G} \in \mathcal{G}_o$



H. Garnier

UNIVERSITÉ DE LORRAINE

POLYTECH NANCY

Basic LSSVF estimator – Statistical analysis

✓ Assume the data-generating system is described as

$$S: \quad y^{(n)}(t_k) = \varphi^T(t_k)\theta_o + v(t_k)$$

where θ_o is the true parameter vector

✓ Assume that ***v(t$_k$)*** is a stationary stochastic process **independent of *u(t$_k$)***. After the SVF filtering, the data-generating system can be rewritten as

$$y_f^{(n)}(t_k) = \varphi_f^T(t_k)\theta_o + v_f(t_k)$$

$$\hat{\theta}_{lssvf} = \left[\frac{1}{N}\sum_{k=1}^{N}\varphi_f(t_k)\varphi_f^T(t_k)\right]^{-1}\left[\frac{1}{N}\sum_{k=1}^{N}\varphi_f(t_k)y_f^{(n)}(t_k)\right]$$

$$\hat{\theta}_{lssvf} = \theta_o + \left[\frac{1}{N}\sum_{k=1}^{N}\varphi_f(t_k)\varphi_f^T(t_k)\right]^{-1}\left[\frac{1}{N}\sum_{k=1}^{N}\varphi_f(t_k)v_f(t_k)\right]$$

H. Garnier

Basic LSSVF estimator – Statistical analysis

$$\hat{\theta}_{lssvf} = \theta_o + \left[\frac{1}{N}\sum_{k=1}^{N}\varphi_f(t_k)\varphi_f^T(t_k)\right]^{-1}\left[\frac{1}{N}\sum_{k=1}^{N}\varphi_f(t_k)v_f(t_k)\right]$$

✓ Under weak conditions, the normalized sums tend to the corresponding expected values as *N* tends to infinity. Hence

$$\hat{\theta}_{lssvf} \xrightarrow[N\to\infty]{} \theta_o \quad \text{if} \begin{cases} \bar{E}\{\varphi_f(t_k)\varphi_f^T(t_k)\} & \text{is nonsingular} \\ \bar{E}\{\varphi_f(t_k)v_f(t_k)\} = 0 \end{cases}$$

- The first condition is satisfied in most cases
- The second condition is <u>never</u> satisfied

✓ ***LSSVF*** estimates are ***always biased*** because of the correlation between the regression vector $\varphi_f(t_k)$ and the noise $v_f(t_k)$

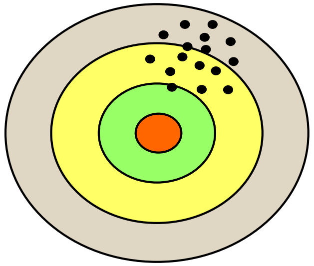
- even if $v(t_k)$ is white noise, $v_f(t_k)$ becomes colored due to the SVF filtering

Simple LSSVF estimator – Conclusions

- ✓ Simple LSSVF method has some attractive properties
  - Simple, analytical solution easy to compute, low computational complexity

- ✓ Main shortcomings
  - ***always biased*** in noisy output measurement situations

$$\bar{E}\left\{\hat{\theta}_{lssvf}\right\} \neq \theta_o \quad \text{since} \quad \bar{E}\left\{\varphi_f(t_k)v_f(t_k)\right\} \neq 0$$



  - ***quite sensitive*** to the SVF filter cut-off frequency

$$L(p) = \frac{1}{(p+\lambda)^n}$$

- ➢ Motivation for studying more advanced methods

We can do better !

Traditional solutions to get optimal estimates

✓ *Maximum Likelihood Method* (ML)

- If the disturbances on the system are Gaussian, the ML method coincides with the ***Prediction Error Method*** (PEM)

✓ *Instrumental Variable Method* (IV)

Prediction Error Method (PEM)

- ✓ Main idea: *model the noise !*
- ✓ General approach applicable to a wide range of model structures: *OE, BJ, …*
- ✓ Conditions to obtain optimal PEM estimates are well-established

$$\hat{\theta}_{pem} = \arg\min_{\theta} \sum_{k=1}^{N} \varepsilon^2(t_k,\theta) = \arg\min_{\theta} \sum_{k=1}^{N} \left\| y(t_k) - \hat{y}(t_k,\theta) \right\|^2$$

- ✓ If assumptions about the noise valid: delivers optimal estimates
- ✓ Involves often solving a non-convex optimization problem
  - relies on iterative nonlinear optimization *(computationally quite demanding)*
    - *Examples: gradient descent, Levenberg-Marquardt, … See TFEST in the SID toolbox*
  - special care required for the initialization of the iterative search
    - may be trapped in false solutions that correspond to local minima

Instrumental Variable (IV) method

- ✓ Main idea: *model the noise !*
- ✓ General approach applicable to a wide range of model structures: *OE, BJ, …*
- ✓ Conditions to obtain optimal IV estimates are well-established

$$\hat{\theta}_{iv}^{opt} = \arg\min_{\theta} \sum_{k=1}^{N} \left\| z_f^{opt}(t_k) L^{opt}(p) \left(y(t_k) - \varphi^T(t_k)\theta \right) \right\|_Q^2$$

  - Need to specify the *instrument* z_f and the *prefilter* $L(p)$
- ✓ If the assumptions about the noise are valid: delivers ***optimal*** estimates
- ✓ If the assumptions about the noise are not valid: delivers ***unbiased*** estimates
- ✓ Based on *(pseudo)* linear regression
  - *do not rely on nonlinear optimization : less risk to be trapped in false solutions*
  - low computational complexity *(comparable to the LS method)*

Solution of the Instrumental Variable (IV) method

- ✓ ***Recap***: LSSVF estimates always biased because of the correlation between the regression vector $\varphi_f(t_k)$ and the noise $v_f(t_k)$
- ✓ ***Main idea of IV:*** introduce a vector $z_f(t_k)$ called *instrument* or *instrumental variable* which components are uncorrelated with $v_f(t_k)$

$$E\{z_f(t_k)v_f(t_k)\} = 0$$

$$\frac{1}{N}\sum_{k=1}^{N} z_f(t_k)v_f(t_k) = 0 \qquad \text{with} \quad v_f(t_k) = y_f^{(n)}(t_k) - \varphi_f^T(t_k)\theta$$

$$\hat{\theta}_{iv} = sol_\theta \frac{1}{N}\sum_{k=1}^{N} z_f(t_k)\left(y_f^{(n)}(t_k) - \varphi_f^T(t_k)\theta\right) = 0$$

$$\hat{\theta}_{iv} = \left[\frac{1}{N}\sum_{k=1}^{N} z_f(t_k)\varphi_f^T(t_k)\right]^{-1}\left[\frac{1}{N}\sum_{k=1}^{N} z_f(t_k)y_f^{(n)}(t_k)\right]$$

- How should the instrument $z_f(t_k)$ be chosen ?

Basic two step IV-based SVF estimator

- The instrument must be chosen so that it is:
  - not correlated with the measurement noise $E\left\{z_f(t_k)v_f(t_k)\right\}=0$
  - sufficiently correlated with the filtered regression vector $E\left\{z_f(t_k)\varphi_f^T(t_k)\right\}\neq 0$

$$\varphi_f^T(t_k)=L(p)\begin{bmatrix} -y^{(n-1)}(t_k) & \cdots & -y(t_k) & u^{(m)}(t_k) & \cdots & u(t_k) \end{bmatrix} \qquad L(p)=\frac{1}{(p+\lambda)^n}$$

- In the basic two-step IVSVF estimator, the instrument is built as

$$z_f^T(t_k)=L(p)\begin{bmatrix} -\hat{x}^{(n-1)}(t_k) & \cdots & -\hat{x}(t_k) & u^{(m)}(t_k) & \cdots & u(t_k) \end{bmatrix}$$

$$\hat{x}(t_k)=G(p,\hat{\theta}_{lssvf})u(t_k)$$

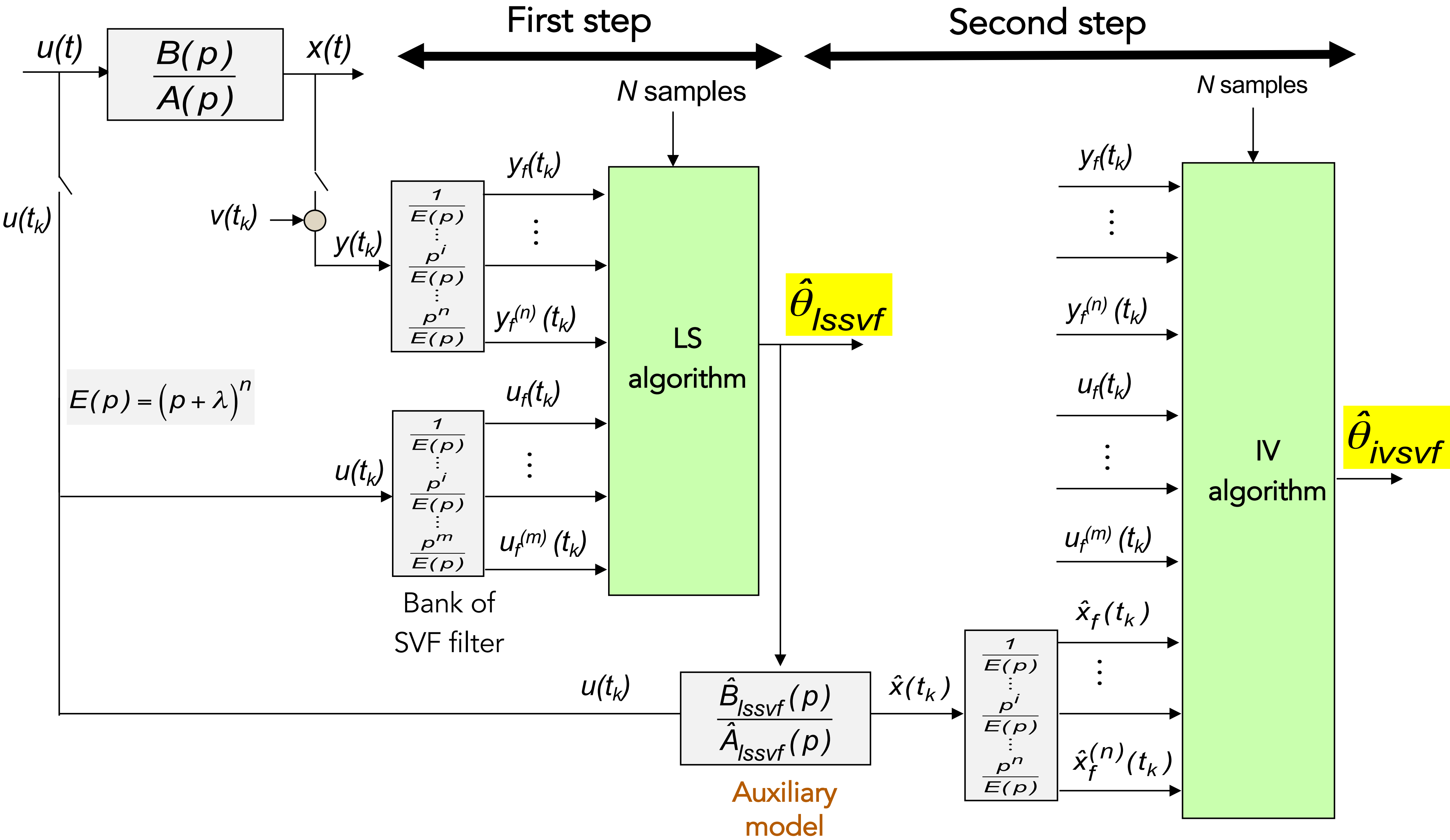
is the estimated noise-free output calculated from an *a priori* LSSVF estimate

- The basic ***IV-based SVF*** estimator can then be computed from

$$\hat{\theta}_{ivsvf}=\left[\frac{1}{N}\sum_{k=1}^{N} z_f(t_k)\varphi_f^T(t_k)\right]^{-1}\left[\frac{1}{N}\sum_{k=1}^{N} z_f(t_k)y_f^{(n)}(t_k)\right]$$

UNIVERSITÉ DE LORRAINE
POLYTECH NANCY

Two-step IV-based SVF estimator - Summary

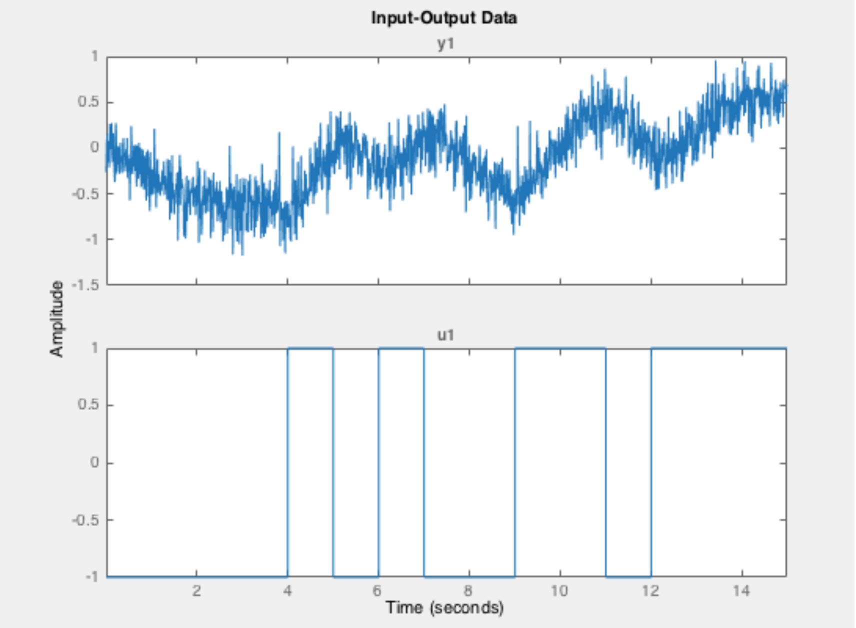


H. Garnier

IVSVF estimator - Matlab implementation – Noisy case

```
B=2;                      % B(p)=2
A=[1 4 3];                % A(p)=p²+4p+3 – True system
u=prbs(4,100);            % PRBS input from the CONTSID
N=1500; Ts=0.01;
t=(0:N-1)'*Ts;
x=lsim(B,A,u,t);          % simulation of the noise-free output
y=x+0.2*randn(N,1);       % white Gaussian noise added
data=iddata(y,u,Ts);idplot(data);

% First step – LSSVF estimation
lambda=3;                          % l: SVF filter cut-off frequency
den_L=[1 2*lambda lambda^2];       % denominator of the SVF filter
num_L0=1;                          % numerator of L0(p)=1/(p+λ)²
num_L1=[1 0];                      % numerator of L1(p) =p/(p+λ)²
num_L2=[1 0 0];                    % numerator of L2(p) =p²/(p+λ)²
yf0=lsim(num_L0,den_L,y,t);        % Computation of the SVF filter bank outputs
yf1=lsim(num_L1,den_L,y,t);
yf2=lsim(num_L2,den_L,y,t);
uf0=lsim(num_L0,den_L,u,t);
Phi_N=[-yf1 -yf0 uf0];             % Regression matrix
Y_N=yf2;                           % Output vector
theta_lssvf=Phi_N\Y_N              % LSSVF estimates
theta_lssvf'                       % see also the LSSVF routine in the CONTSID toolbox
3.2542    2.6889    1.6865         % Mlssvf=lssvf(data,[2 1 0],lambda)
```



IVSVF estimator - Matlab implementation – Noisy case

% Second step – IVSVF estimation

```
% Construction of the auxiliary model
Blssvf=theta_lssvf(3)';              % Auxiliary model
Alssvf=[1 theta_lssvf(1:2)'];
% Simulation of the auxiliary model output
xest=lsim(Blssvf,Alssvf,u,t);
% Computation of the SVF filter bank outputs for the auxiliary model
xestf0=lsim(num_L0,den_L,xest,t);    % filtered auxiliary model output
xestf1=lsim(num_L1,den_L,xest,t);    % 1st-order time-derivative of the filtered auxiliary model output
% Construction of the IV matrix
Z_N=[-xestf1 -xestf0 uf0];           % Instrumental variable matrix
% IVSVF estimates
theta_ivsvf=(Z_N'*Phi_N)\Z_N'*Y_N;   % IVSVF solution
theta_ivsvf'
3.9454    2.9977    1.9685           % see also the ivsvf routine in the  CONTSID toolbox
                                     % Mivsvf=ivsvf(data,[2 1 0],lambda)
```

% The estimation error has clearly been reduced

% Run several times your program to get a feel for the bias reduction (which can vary depending on the noise realization) or even better run a Monte Carlo simulation

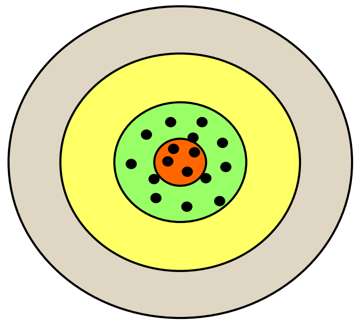
Basic IVSVF estimator – Conclusions

✓ Some attractive properties

- simple
- analytical solution
- low computational complexity
- unbiased estimates in output measurement noise situations $\mathcal{S} \notin \mathcal{M}, \mathcal{G} \in \mathcal{G}_o$

✓ But IVSVF estimates

- the method is not iterative, it has two steps only
- quite sensitive to the choice of the SVF filter
- not minimum variance



✓ Motivation for studying more advanced IV methods

We can still do better !

How to choose the instrument to get optimal estimates ?

Extended Instrumental Variable

- ✓ To improve the basic IV estimate accuracy, some extensions are introduced
  - operate a prefiltering by $L(\rho)$ on both I/O data
  - enlarge the instrument vector $z(t_k)$ such that $n_z \geq n_\theta$

$\rho = q \quad or \quad \rho = p$

- ✓ The so-called ***extended IV estimate*** is then given *(Söderström & Stoica 1983)*

$$\hat{\theta}_{xiv} = \arg\min_{\theta} \left\| \underbrace{\left(\frac{1}{N} \sum_{k=1}^{N} L(\rho)z(t_k)L(\rho)\varphi^T(t_k) \right)}_{R_N} \theta - \underbrace{\left(\frac{1}{N} \sum_{k=1}^{N} L(\rho)z(t_k)L(\rho)y(t_k) \right)}_{r_N} \right\|_Q^2$$

  - $L(\rho)$ is a stable prefilter, Q a positive definite weighting matrix $\|x\|_Q^2 = x^T Q x$

- ✓ It is the weighted LS solution of an overdetermined system of linear equations

$$\hat{\theta}_{xiv} = \left(R_N^T Q R_N \right)^{-1} \left(R_N^T Q r_N \right)$$

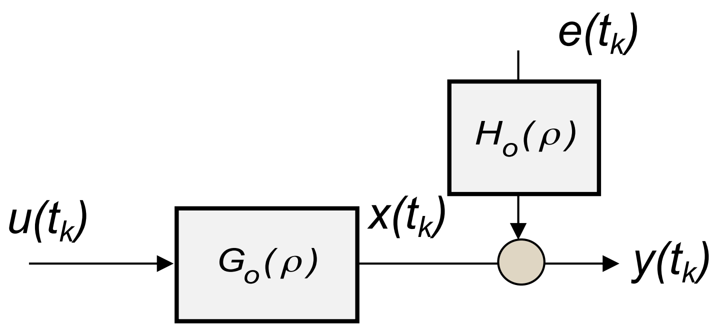
  - This solution is then well suited for the consistency analysis of the IV estimators

Optimal IV – General results

✓ Data-generating system (ρ=p or ρ=q)

$$y(t_k) = \frac{B_o(\rho)}{A_o(\rho)}u(t_k) + H_o(\rho)e(t_k)$$

$$y(t_k) = \varphi^T(t_k)\theta_o + v(t_k)$$



✓ IV estimates are ***consistent*** if

$$\begin{cases} \bar{E}\left\{L(\rho)z(t_k)L(\rho)v(t_k)\right\} = 0 \\ \bar{E}\left\{L(\rho)z(t_k)L(\rho)\varphi^T(t_k)\right\} \quad \text{is nonsingular} \end{cases}$$

✓ IV estimates are ***optimal*** if

(Söderström and Stoica 1983)

$$Q = I \qquad n_z = n_{\theta_o}$$

$$L^{opt}(\rho) = \frac{1}{H_o(\rho)A_o(\rho)}$$

$$z^{opt}(t_k) = \varphi_o(t_k) : \text{noise-free version of } \varphi(t_k)$$

✓ IV estimates are asymptotically ***Gaussian distributed***

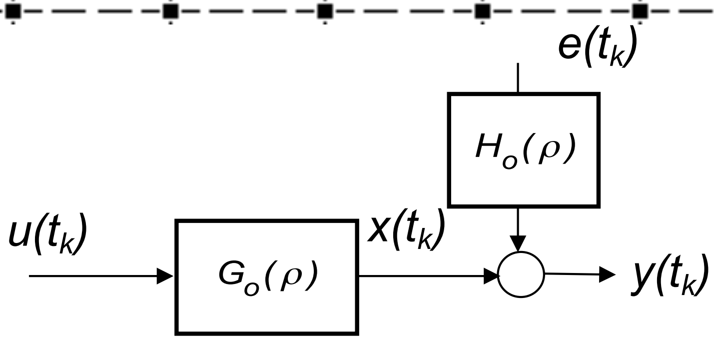
$$\sqrt{N}\left(\hat{\theta}_{iv} - \theta_o\right) \xrightarrow[N\to\infty]{dist} N(0, P_{iv})$$

$$P_{iv} = \sigma_e^2 \bar{E}\left\{\left(L(\rho)z(t_k)\right)\left(L(\rho)z(t_k)\right)^T\right\}$$

Optimal IV – General results

✓ Data-generating system (**ρ=p or ρ=q**)

$$y(t_k) = \frac{B_o(\rho)}{A_o(\rho)} u(t_k) + H_o(\rho) e(t_k)$$
$$y(t_k) = \varphi^T(t_k)\theta_o + v(t_k)$$

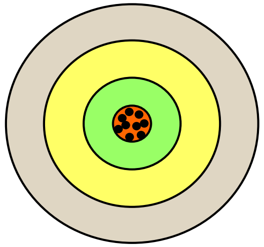


✓ *Optimal accuracy* if *(Söderström & Stoica 1983. See also Young 1976. Optimal IV derives from the ML equations. See the following recent paper*

P.C. Young, Refined instrumental variable estimation: ML optimization of a unified BJ model, Automatica, 2015)

$$L^{opt}(\rho) = \frac{1}{H_o(\rho) A_o(\rho)}$$

$$z_f^{opt}(t_k) = L^{opt}(\rho)\varphi_o(t_k)$$ $\varphi_o(t_k)$: noise-free version of $\varphi(t_k)$



✓ ***Inherent filtering:*** a distinguishing feature of optimal IV

- Interesting for CT model identification, the filtering
  - *ensures minimum variance estimates*
  - *provides a convenient way for generating the time-derivatives*
    - ➢ *can be **automatically (and optimally) chosen***

Implementation of the optimal IV solution

✓ *Usual dilemma met with accuracy optimization*

- requires the knowledge of the true plant and noise models !!
- $\varphi_o(t_k)$: noise-free version of $\varphi(t_k)$ requires the knowledge of the noise-free output $x(t_k)$

$$\begin{cases} z_f^{opt}(t_k) = L^{opt}(\rho)\varphi_o(t_k) \\ L^{opt}(\rho) = \dfrac{1}{H_o(\rho)A_o(\rho)} \end{cases}$$

✓ Two different main implementations have been suggested

- **Multistep procedure** *(Söderström & Stoica 1983)*
  - *Example: IV4 (4 steps) routine in the SID toolbox*
    - assumes a *(rather peculiar)* ARARX model structure
    - may be quite unreliable in practice *(see later on)*
- **Iterative *(or refined)* procedure** *(Young 1976, 1984)*
  - *Example: TFSRIVC routine in the CONTSID toolbox*
    - assumes a COE model structure
    - is particularly reliable in practice

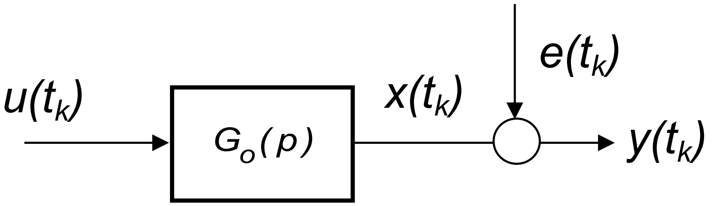
UNIVERSITÉ DE LORRAINE

POLYTECH NANCY

Iterative implementation of optimal IV: TFSRIVC for COE models

- ✓ Data-generating system: a CT output error (COE) model

$$\begin{cases} x(t) = \dfrac{B_o(p)}{A_o(p)} u(t) \\ y(t_k) = x(t_k) + e(t_k) \end{cases}$$



$$H_o(p) = 1$$

- ✓ Optimal choice for the instrument and filter

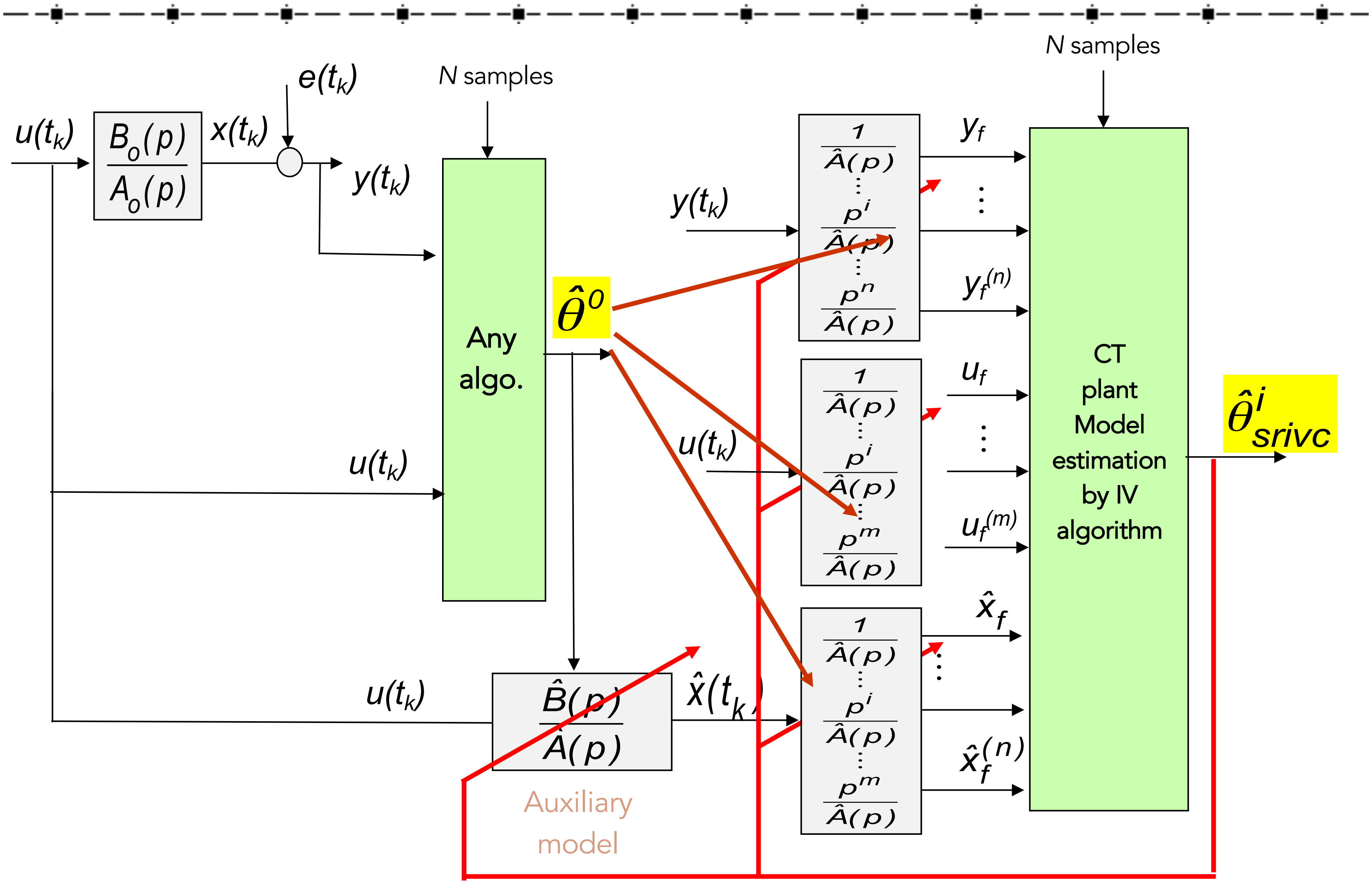
$$\begin{cases} z_f^{opt}(t_k) = L^{opt}(p)\varphi_o(t_k) \\ L^{opt}(p) = \dfrac{1}{A_o(p)} \end{cases}$$

$$\varphi_o^T(t_k) = \begin{bmatrix} -x^{(n-1)}(t_k) & \cdots & -x(t_k) & u^{(m)}(t_k) & \cdots & u(t_k) \end{bmatrix}$$

- ✓ Requires the knowledge of the true plant model and noise-free output
- ✓ Solution *(P.C. Young)*
  - use of an ***iterative procedure*** where the instrument and prefilter are iteratively adapted until they converge on their optimal value

$$\hat{\theta}_{srivc}^{i+1} = \left[\sum_{k=1}^{N} z_f(t_k, \hat{\theta}^i) \varphi_f^T(t_k, \hat{\theta}^i) \right]^{-1} \left[\sum_{k=1}^{N} z_f(t_k, \hat{\theta}^i) y_f^{(n)}(t_k, \hat{\theta}^i) \right]$$

Optimal TFSRIVC method for COE models



The learning rate is usual very fast

Convergence occurs in about 4 to 5 iterations

TFSRIVC parametric error covariance matrix estimate

✓ Good empirical estimates of the uncertainty in the TFSRIVC parameter estimates

- Provided by the parametric error covariance matrix estimate

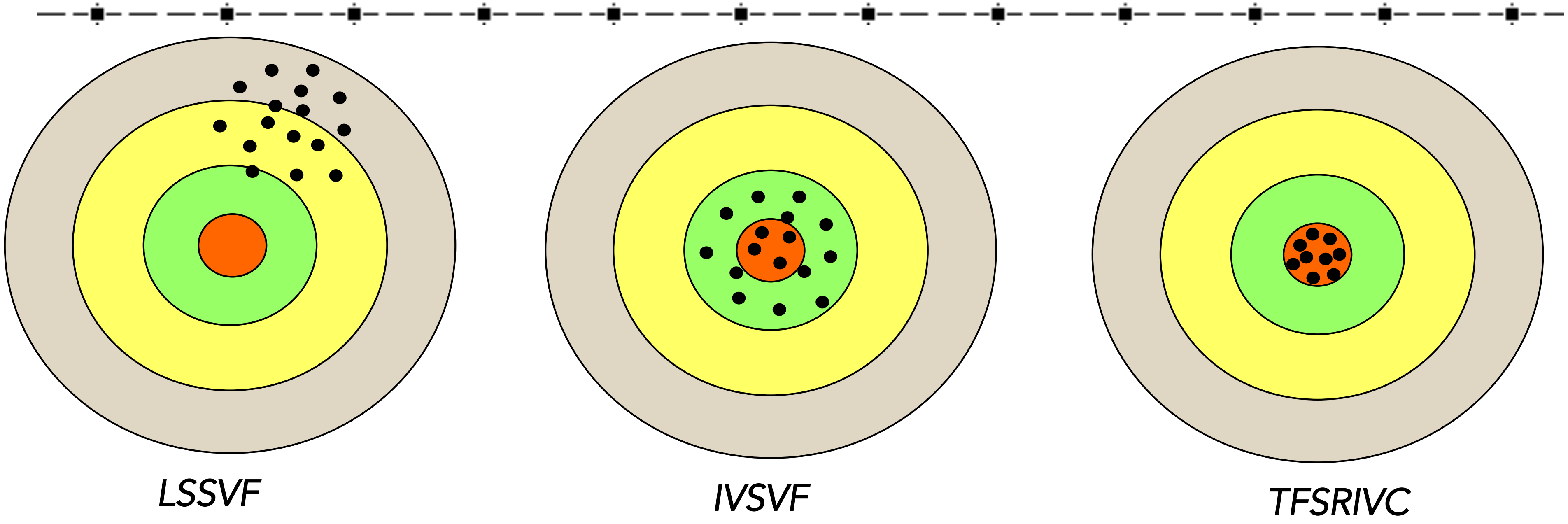
$$P_{\hat{\theta}_{srivc}} = E\left\{\left(\hat{\theta}_{srivc} - \theta_o\right)\left(\hat{\theta}_{srivc} - \theta_o\right)^T\right\} \geq J^{-1} \qquad J: \text{Fischer Inf. Matrix}$$

$$\hat{P}_{srivc} = \frac{\sigma_{\hat{e}}^2}{N}\left[\frac{1}{N}\sum_{k=1}^{N} z_f(t_k,\hat{\theta}_{srivc}) z_f^T(t_k,\hat{\theta}_{srivc})\right]^{-1}$$

where $\hat{e} = y(t_k) - \hat{y}(t_k,\hat{\theta}_{srivc})$

- even for small sample size N
- can be used in the procedure to select the best model structure
  (see YIC criterion later)

To sump up



- *Simple* ***LSSVF:*** always biased
- Two-step ***IVSVF:*** unbiased *but* not minimum variance
- *Iterative* ***TFSRIVC***: optimal *(unbiased & minimum variance)* for COE models
  unbiased with low *(but not minimum)* variance when the additive noise is colored

The TFSRIVC algorithm provides a reliable and robust approach to CT model identification

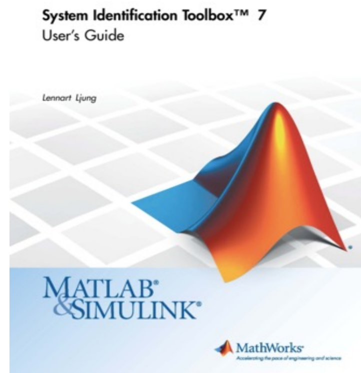
It is recommended for day-to-day use

Instrumental variable: take-home messages

- ✓ Include inherent *(possibly optimal)* data prefiltering
- ✓ Conditions to obtain optimal IV estimates are well-established
- ✓ Provide consistent estimates even for an imperfect noise structure $\mathcal{S} \notin \mathcal{M}, \mathcal{G} \in \mathcal{G}_o$
  - Choice of the instrument and prefilters influences the variance only, while the consistency properties are secured
- ✓ Implementation of the optimal IV solution
  - ***Iterative*** algorithms: much more preferable than ***multistep*** algorithms
- ✓ Offer similar good performance as PEM methods in general
- ✓ ***Iterative*** IV implementations present one major advantage over PEM
  - are much less sensitive to the initialization stage

Software aspects

- ✓ Several actively maintained toolboxes are available
  - Comprehensive Mathworks SID toolbox *(L. Ljung)*
  - FDIDENT toolbox *(I. Kollar, J. Schoukens)*
  - UNIT toolbox *(B. Ninness)*
  - CAPTAIN toolbox *(P. Young)*



- ✓ No software entirely dedicated to direct CT approaches

CONTSID TOOLBOX

  - first released in 1999

CONtinuous-Time System IDentification

Key features

- ✓ Supports ***direct CT identification*** approaches
  - ***Basic linear*** *black-box* models
    - Transfer function and state-space models
    - regularly and irregularly sampled data
    - Time-domain or frequency domain data
  - ***More advanced*** *black-box* models
    - ***On-line, errors-in-variables*** and ***closed-loop*** situations
    - Nonlinear systems: ***block-structured***, ***LPV*** or ***LTV*** models

- ✓ May be seen as an add-on to the Matlab System Identification toolbox
  - Uses the same syntax, data and model objects

    M=tfsrivc(data,np,nz)

- ✓ P-coded version freely available from: www.cran.univ-lorraine.fr/contsid

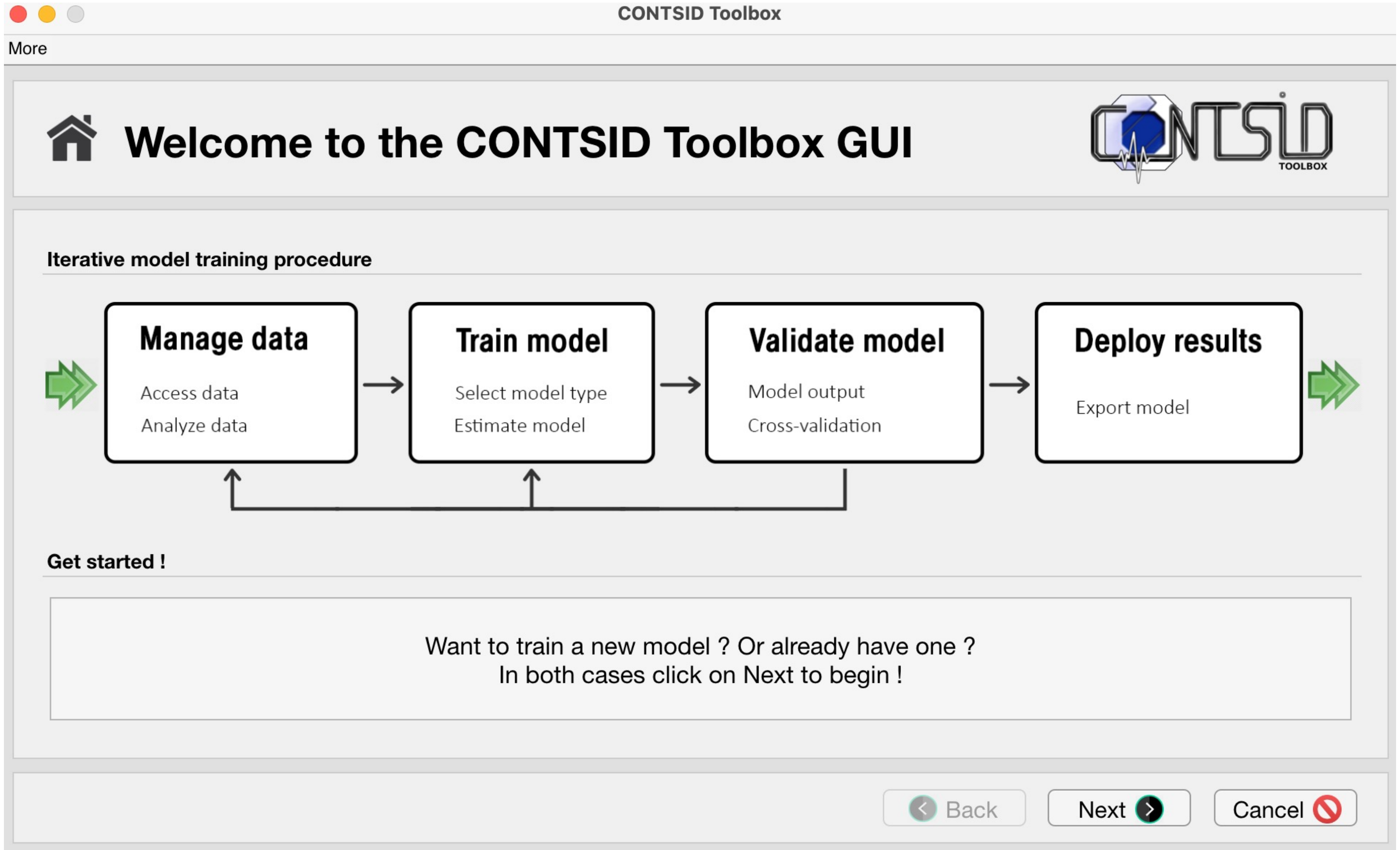
Main features of the latest version 7.4

- ✓ Core of the routines mainly based on **iterative optimal IV**: *SRIVC*
  - CONTSID includes also a few PEM and subspace-based methods

- ✓ *SRIVC-based parameter estimation schemes for more advanced identification*
  - ***simple process models***: *PROCSRIVC*
  - ***Transfer function + delay models:*** *TFSRIVC*
  - ***Transfer function + delay + noise models:*** *TFRIVC*
  - ***Time Varying Parameter models***: *recursive RSRIVC*
  - ***Closed-loop identification***: *CLSRIVC*
  - ***LPV models***: *LPVSRIVC*
  - ***Hammerstein models***: *HSRIVC*, …

- ✓ Includes a new flexible ***GUI*** and many ***demos*** to illustrate its use and the recent developments

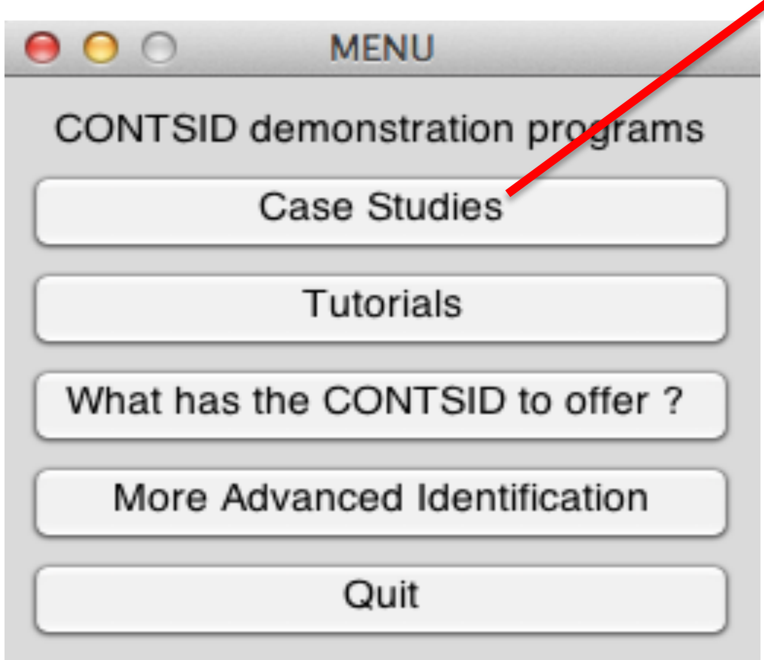
CONTSID graphical user interface

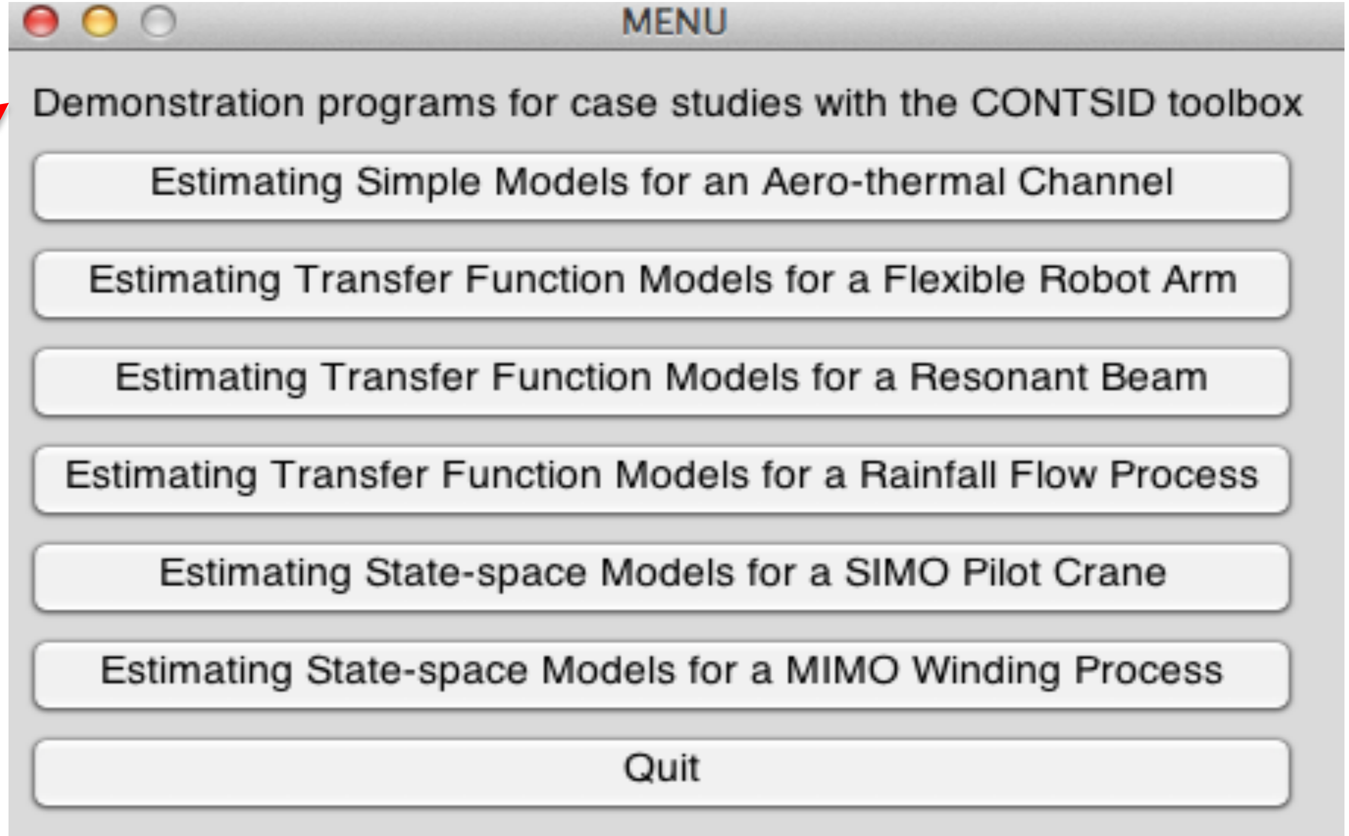


- Allows the user to easily apply the iterative process of system identification

CONTSID toolbox – Demonstration programs

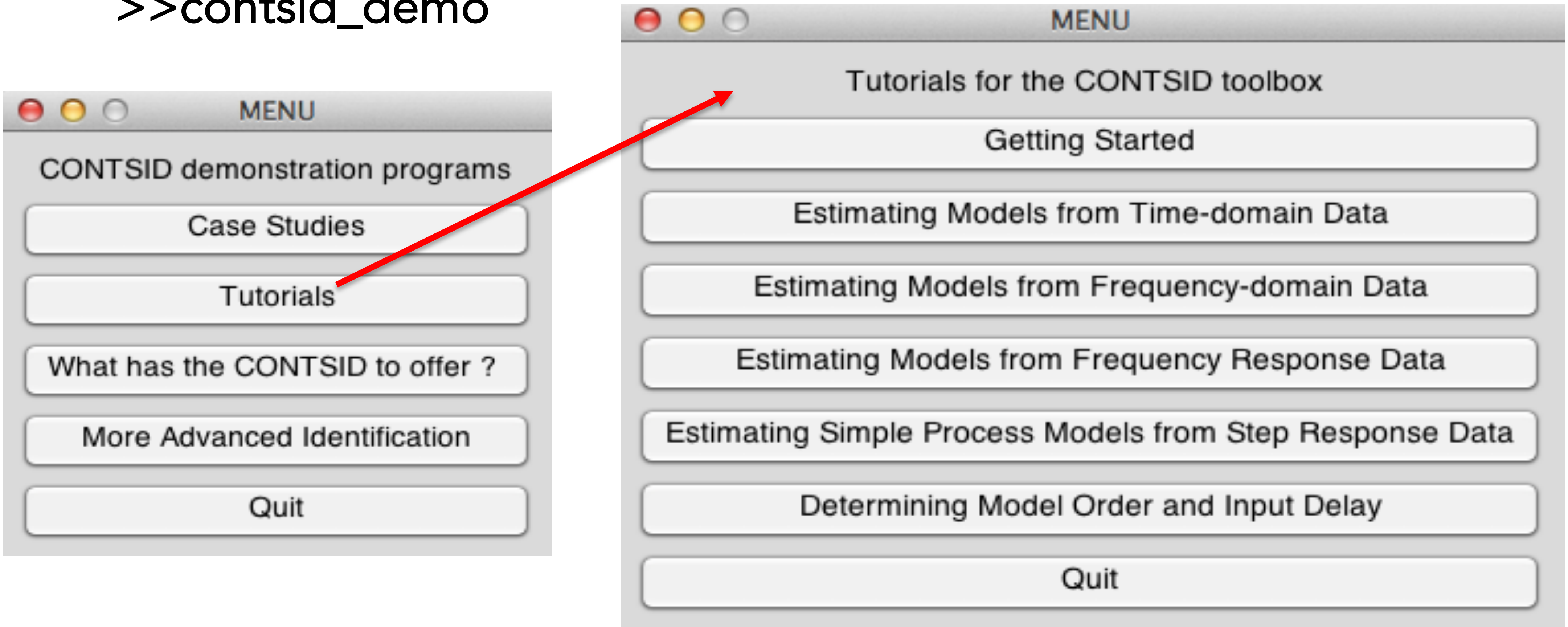
>>contsid_demo





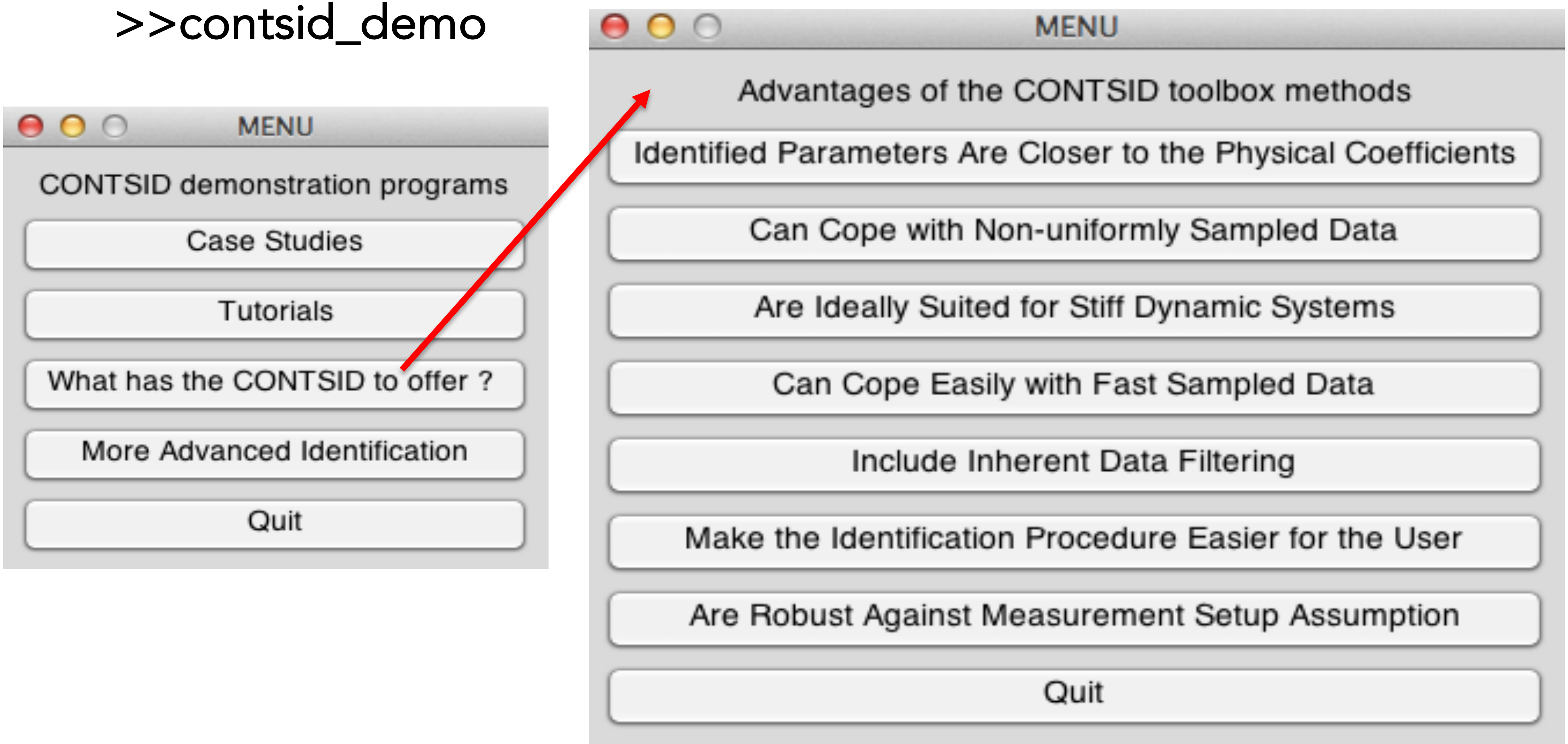
CONTSID toolbox – Demonstration programs

>>contsid_demo



CONTSID toolbox – Demonstration programs

>>contsid_demo



CONTSID toolbox – Demonstration programs

>>contsid_demo