Identification of linear time-invariant models

A refresher on practical aspects

Hugues GARNIER

hugues.garnier@univ-lorraine.fr

The error made by the majority of novices in data-driven model learning

- Many beginners work for a long time their full understanding of algorithms at the expense of their problem solving skills
- To carry out a data-driven model learning project, it is not enough to know optimisation algorithms but it is also necessary to know how to use these algorithms, which few people know how to do !
- The practical side of data-driven model learning is therefore crucial to be successful
- This is what you will learn in the following, which is probably one of the *most important part from the practical point of view*

Model identification of linear time-invariant systems

- For ***control design and analysis***, Linear Time-Invariant (LTI) models have been hugely important, mainly motivated by
  - their simplicity
  - their performance and robustness properties are well understood
- Classical SYSID methods of linear models
  - share these properties in many regards
  - have a relatively low computational complexity
  - have strong systems-theoretical background, with well developed concepts such as identifiability, input design, informative data selection

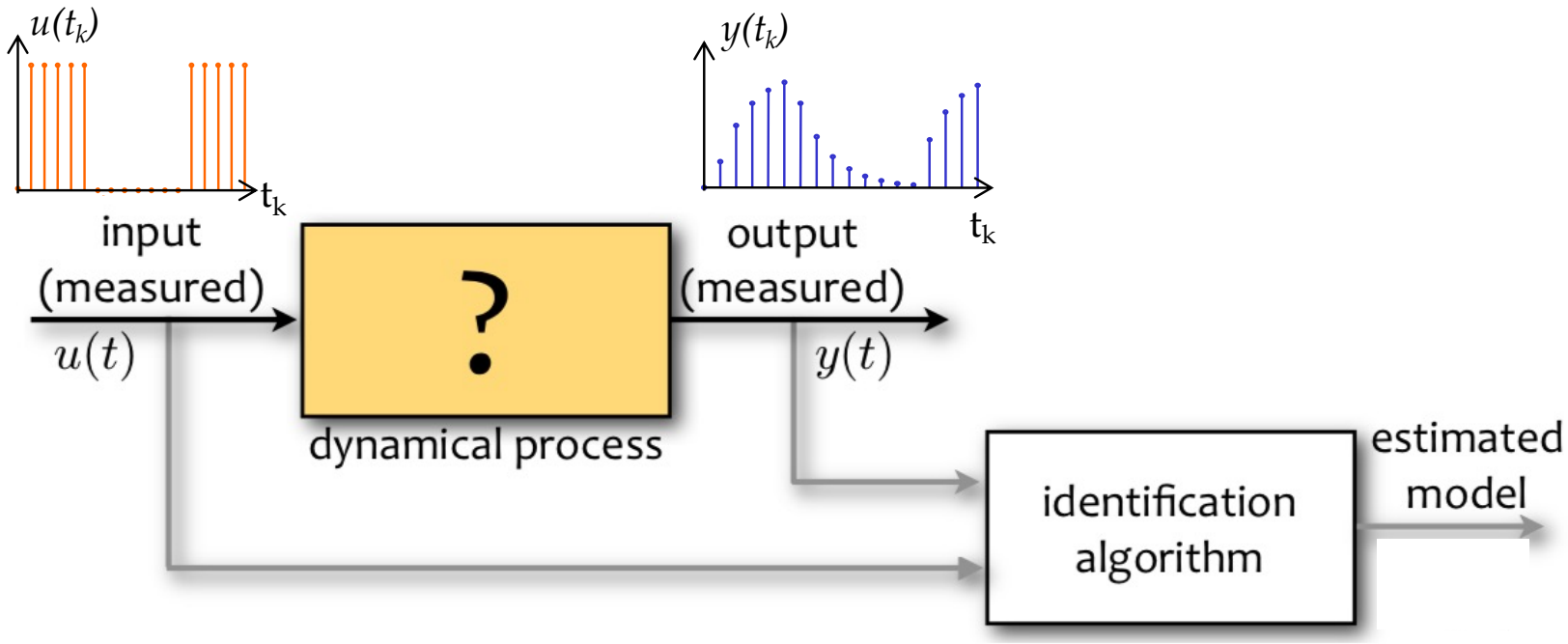
➢ Always try first data-driven methods for identifying

linear time-invariant (LTI) models

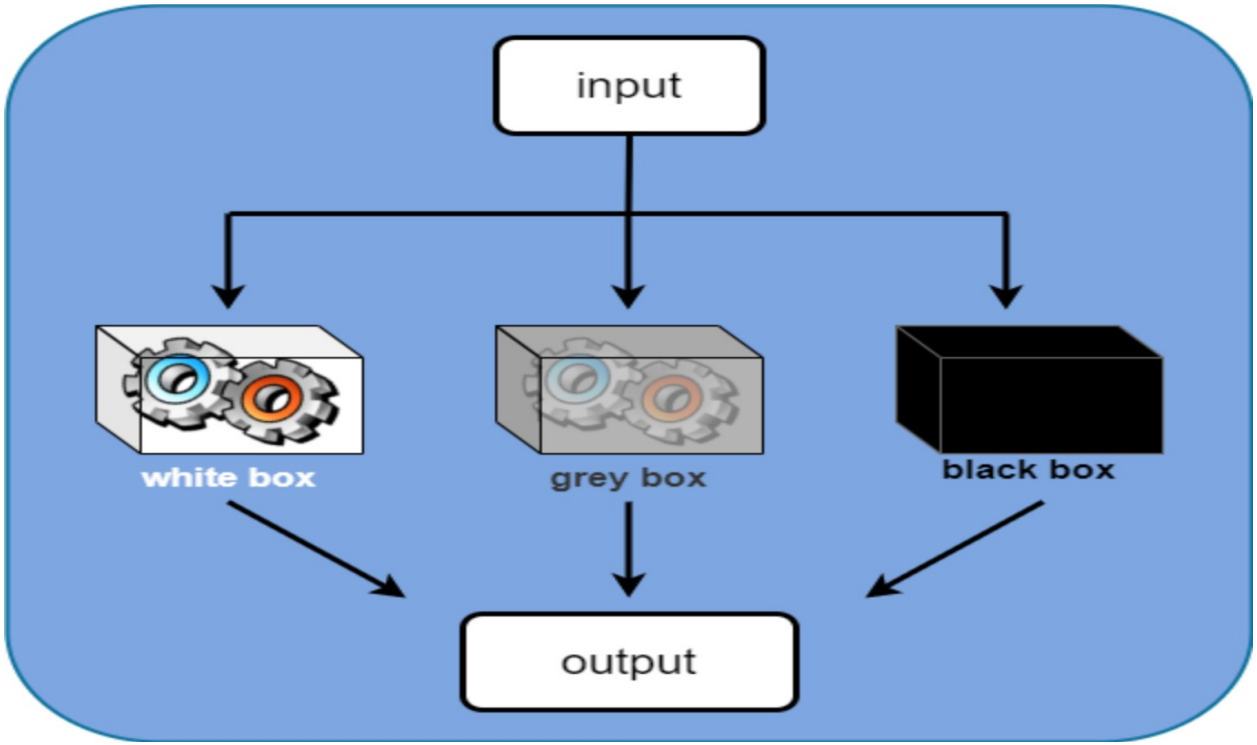
➢ If the fit is not good, try to estimate nonlinear or time-varying models

System identification – Brief recap

- System identification is an iterative process, where you identify models with different structures from sampled data and compare model performance
- Ultimately, choose the simplest model that best describes the dynamics of your system
- *A priori* physical knowledge of the system is often a key for success



A palette of color for the models (structure + parameters)



White-box model

- Structure built from Physics
- Parameters a priori known

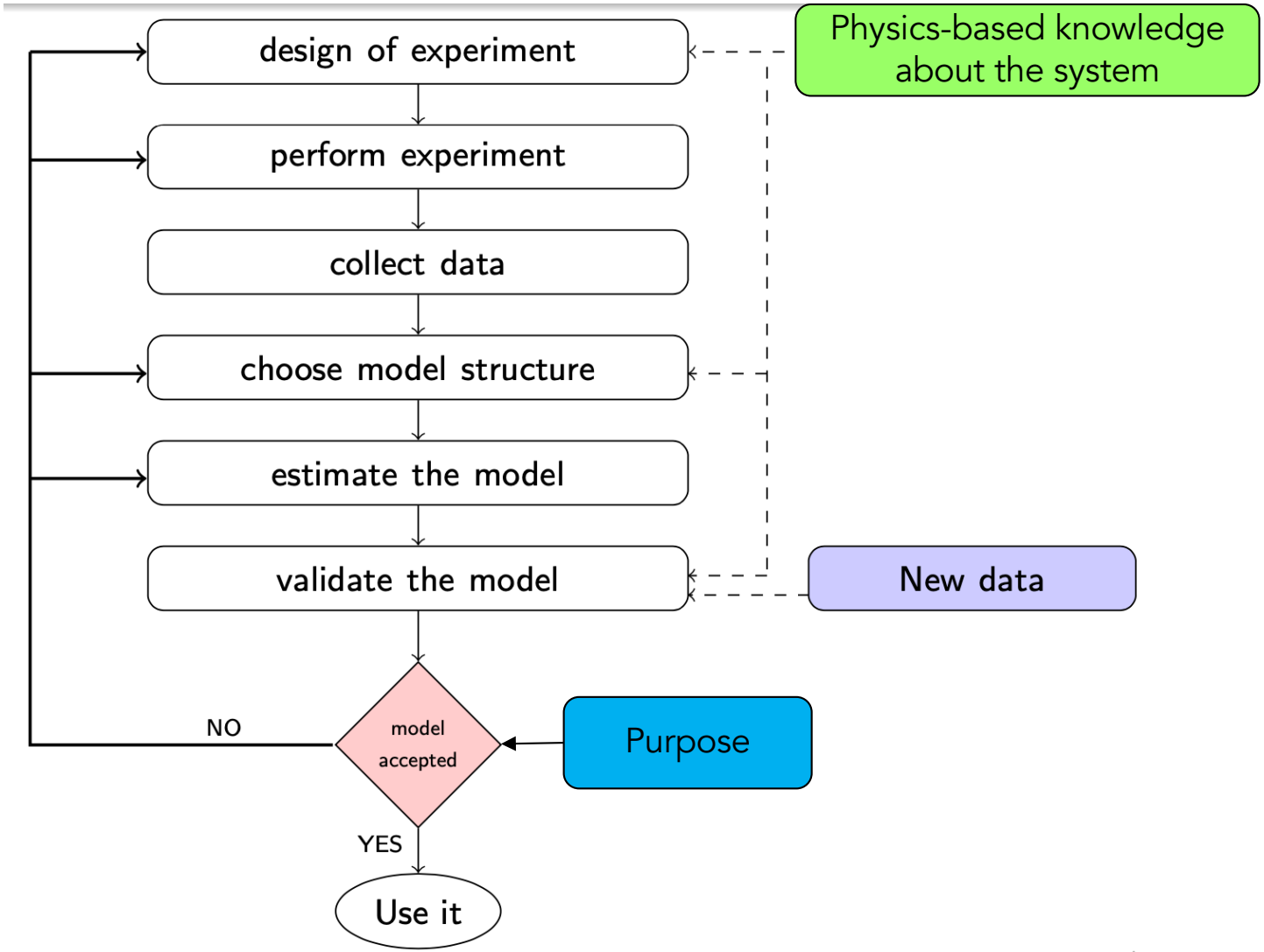
Grey-box model

- Structure built from Physics
- Some parameters are estimated from data

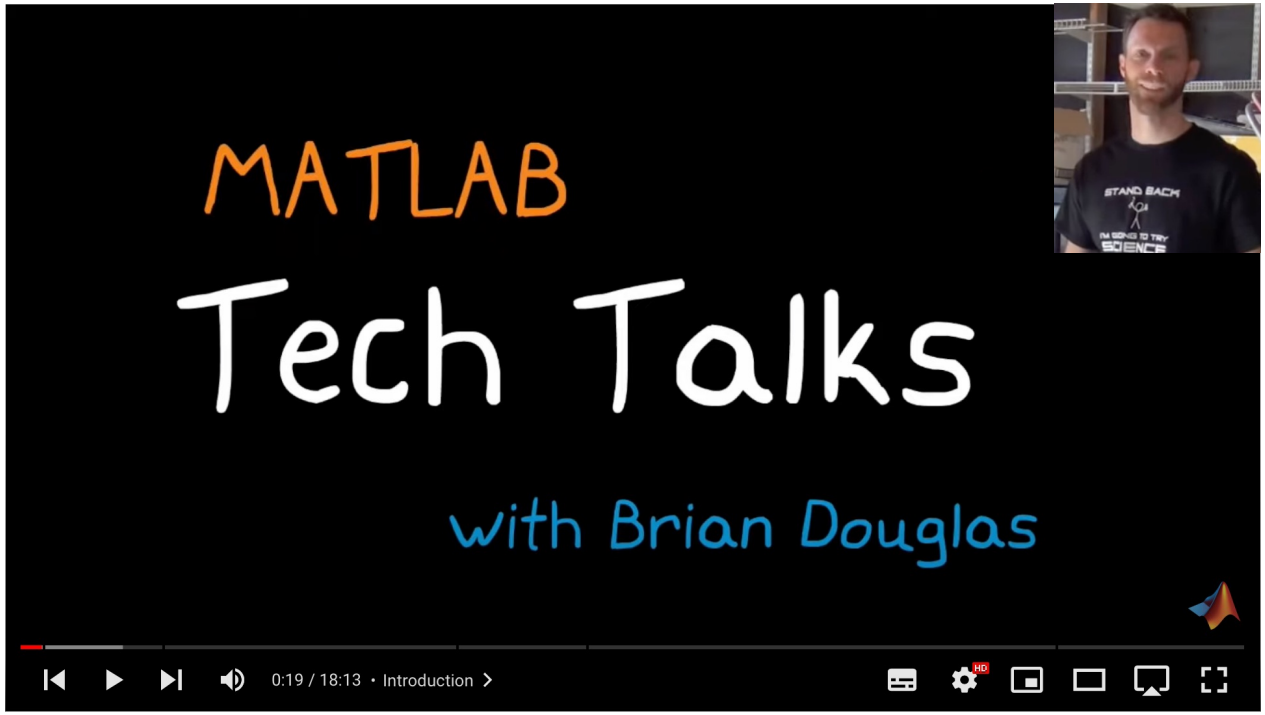
Black-box model

- Structure built mainly from data
- Parameters estimated from data

The iterative system identification workflow



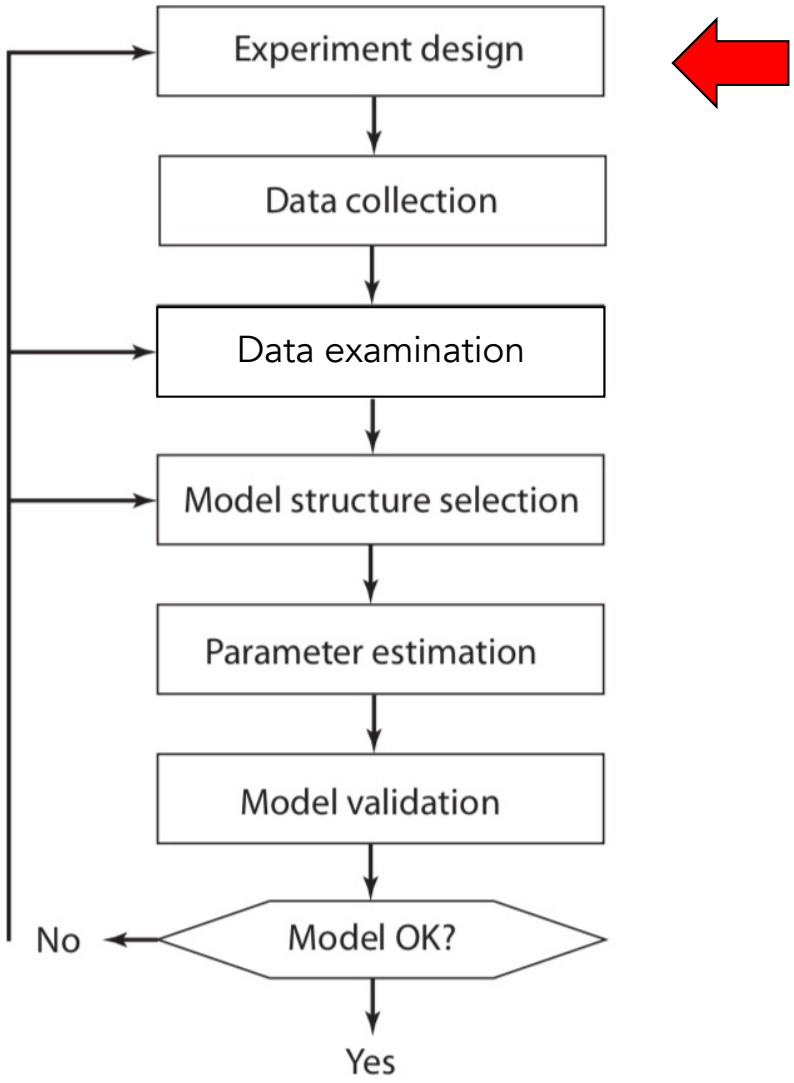
Linear System Identification – A refresher with Brian



Linear System Identification | System Identification, Part 2 16 mn

https://www.youtube.com/watch?v=qC_C04SEV1E

Data-driven system identification
An iterative procedure



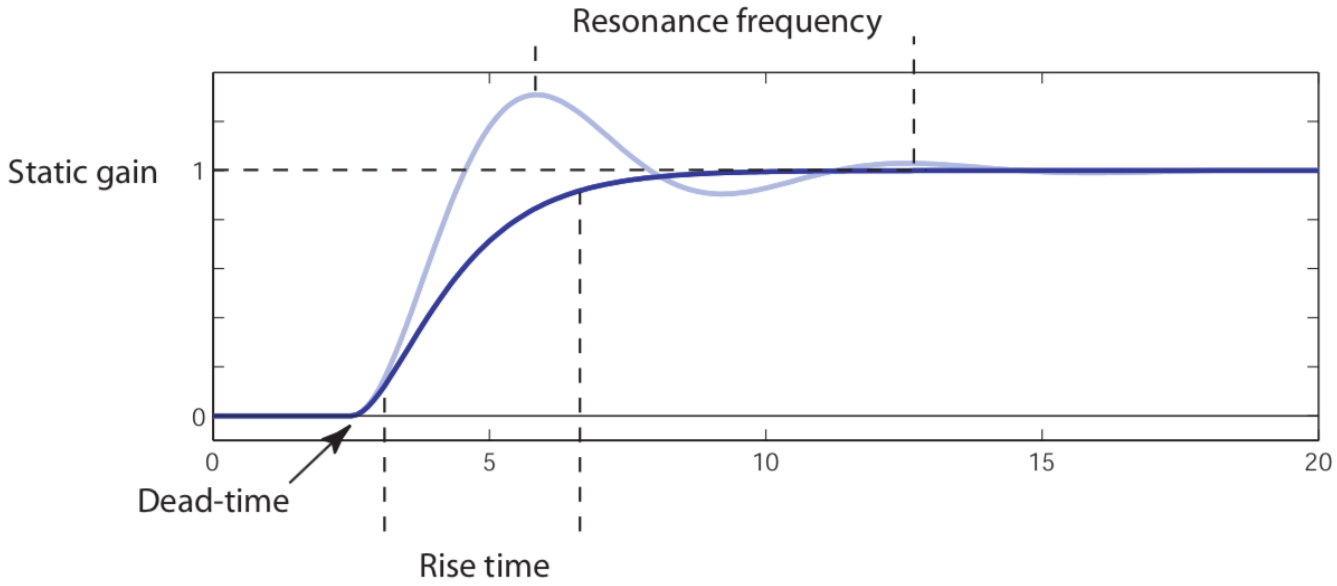
Experiment design for data collection

- To obtain a good model of your system, you must have measured data that reflects the dynamic behavior of the system
- The accuracy of the model depends on the quality of the measurement data, which in turn depends on the experiment design

Often good to use a two-stage approach

1. Preliminary experiments
   - step/impulse response tests to get basic understanding of system dynamics
   - linearity, stationary gains, time delays, time constants, sampling interval
2. Data collection for model estimation
   - carefully designed experiment to enable good model fit
   - operating point, input signal type, number of data points to collect, etc.

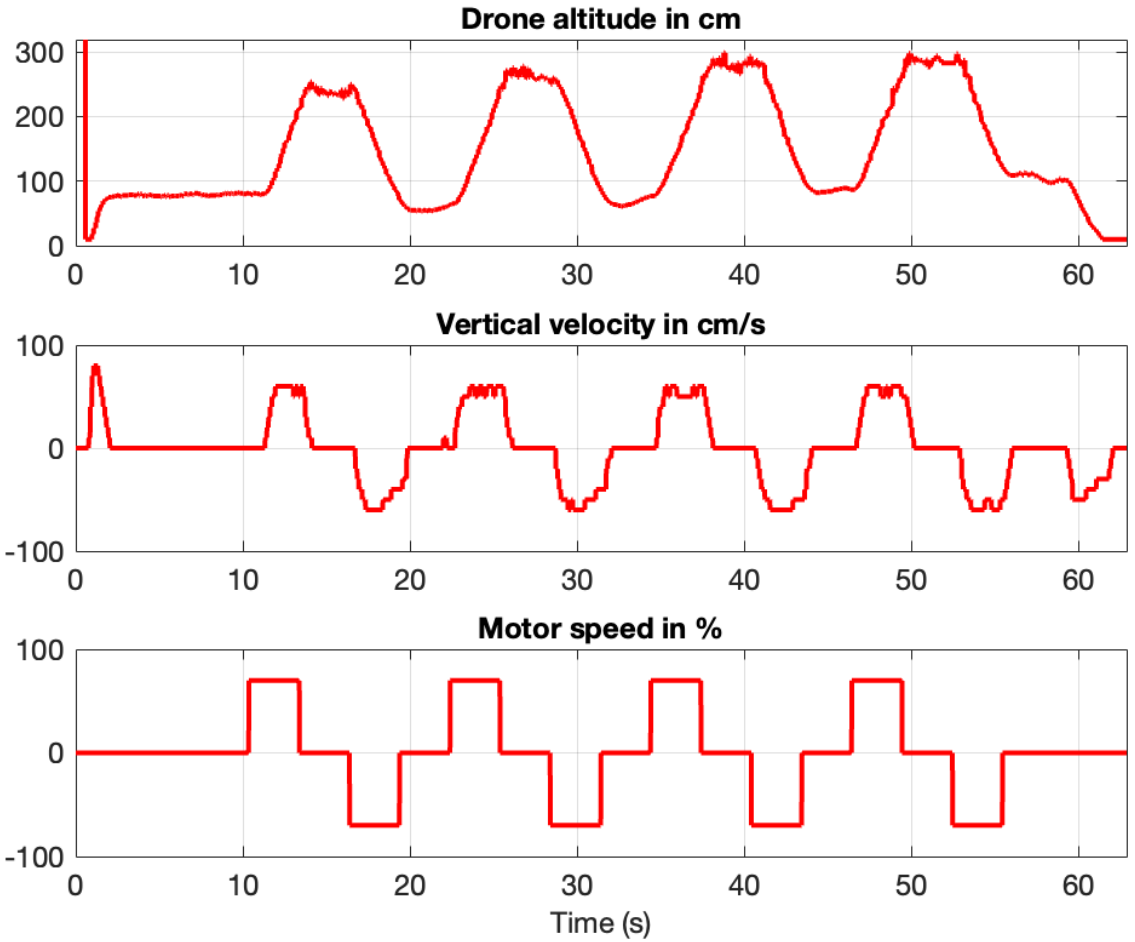
Preliminary test: step response experiment



Useful for obtaining qualitative information about system

- indicates dead-times, static gain, time constants and resonances

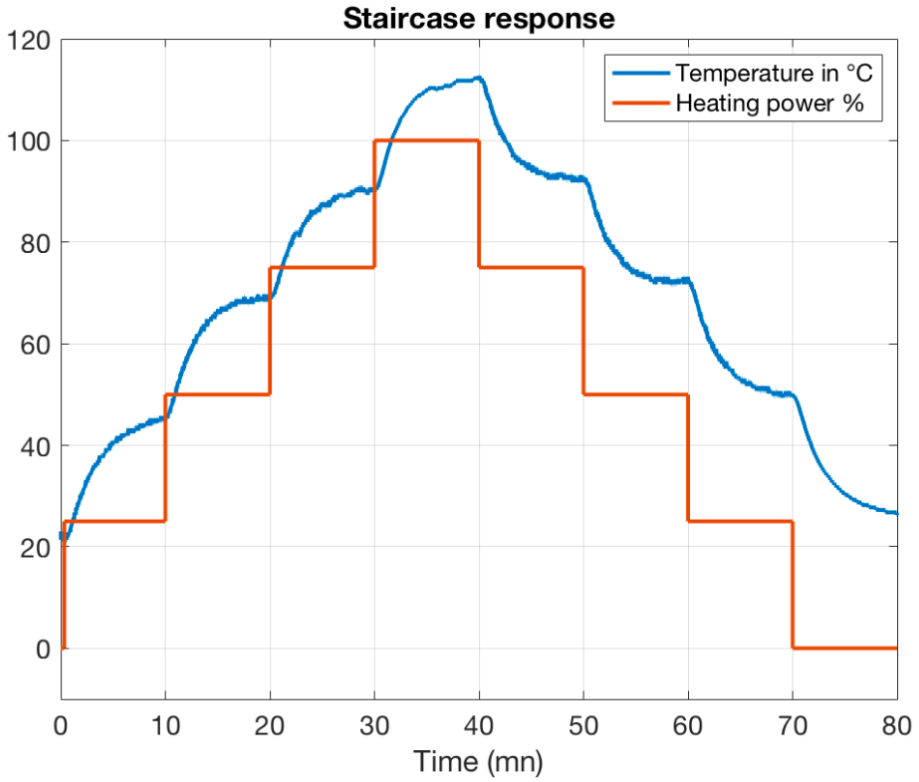
Preliminary test: square wave response experiment
Apply, when ever possible, periodic square wave input



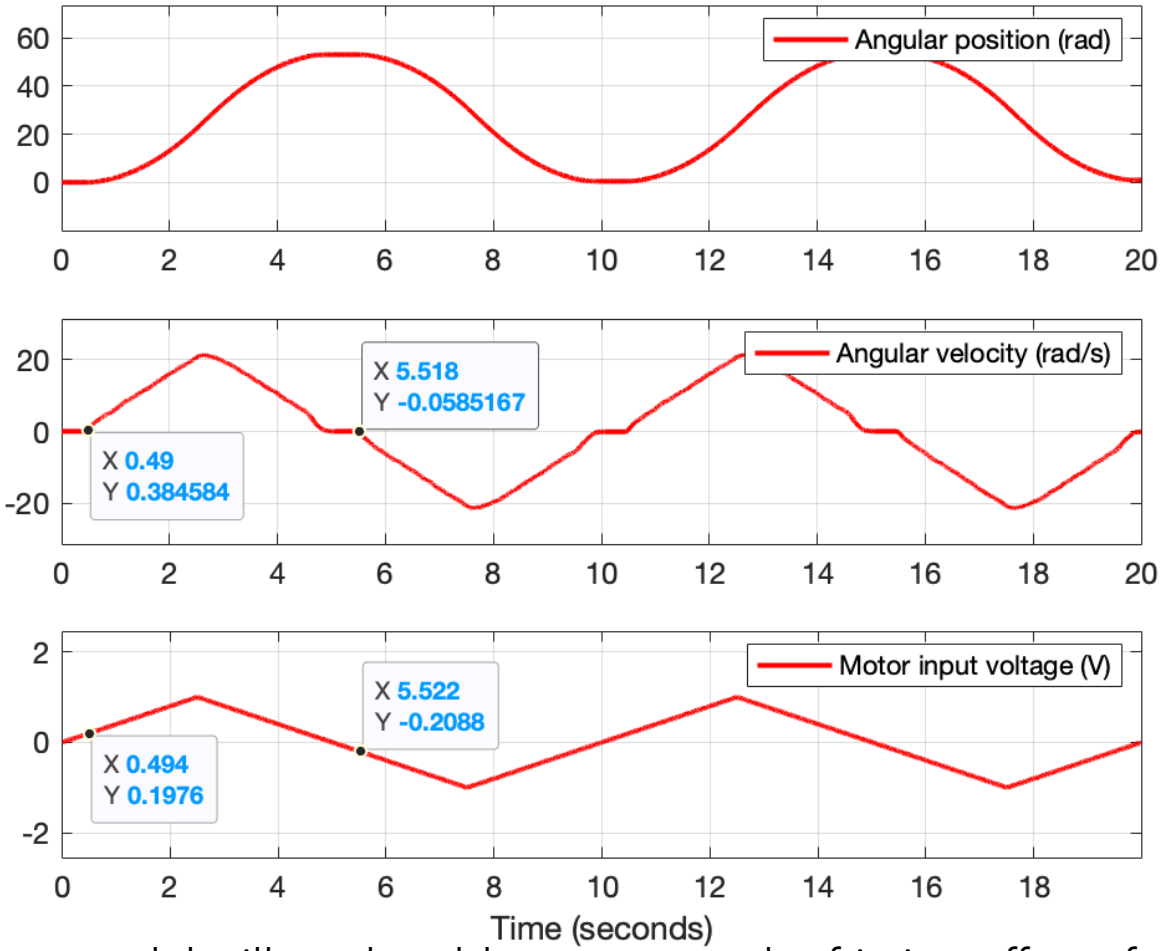
Give insights about non-linearity effects in the system

Tests for verifying linearity

- a linear system has the same response independent of the operative point
- test step response in different operating points



Tests for detecting dry friction

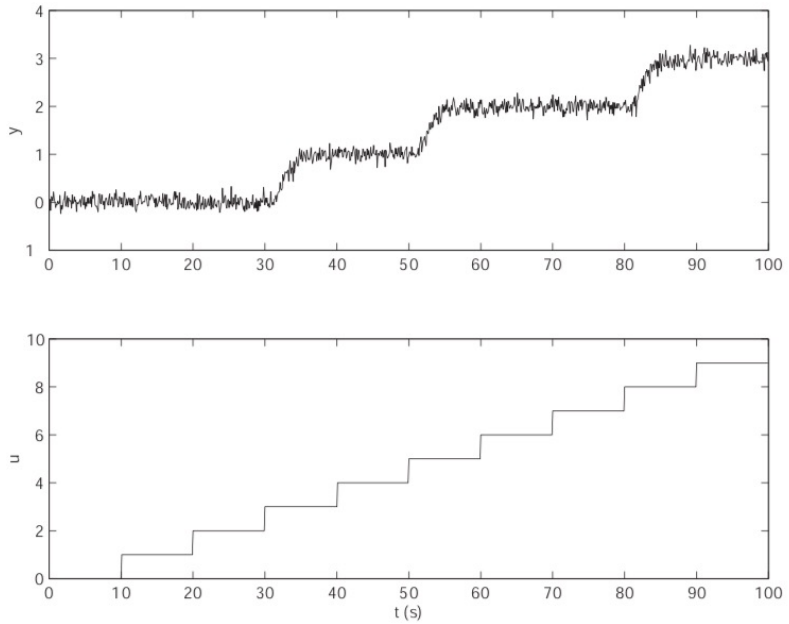


A simple linear model will not be able to capture the friction effects from these data

- Use a nonlinear model to better capture the friction effects
- Use amplitude of the steps >0.2 to cancel out the friction effects

Tests for detecting friction

Friction can be detected by using small step increases in input



Input moves every two or three steps.

A simple linear model will not be able to capture the friction effects from these data

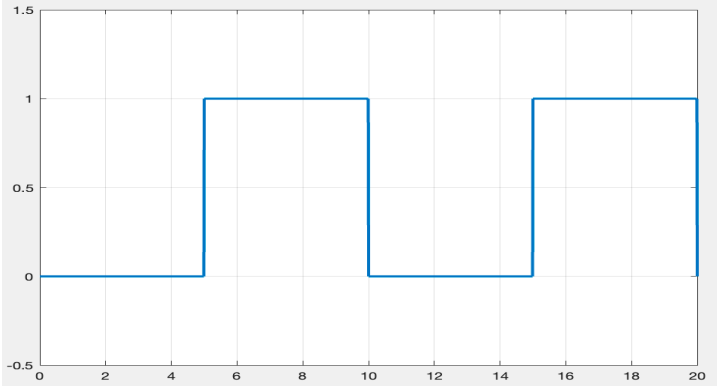
- Use a nonlinear model to better capture the friction effects
- Increase the amplitude of the steps to cancel out the friction effects

Choice of inputs for informative data

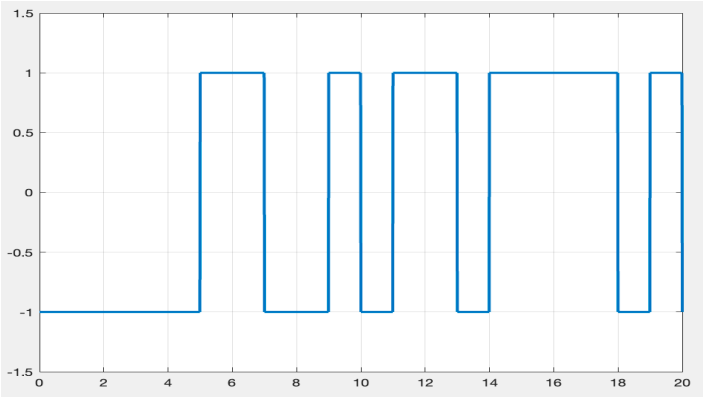
- Needs to be sufficiently "rich."
  - Input signal is needed to excite the system
  - The experiment should be carried out under conditions that are similar to those under which the model is going to be used
- Amplitude
  - Trade-off is needed
    - Large amplitude gives good signal-to-noise ratio, low parameter estimate variance
    - But most systems enter into nonlinear regimes for large input amplitude
- Number of data points
  - The larger, the better (…if data is informative)

Common choices of input for linear system identification

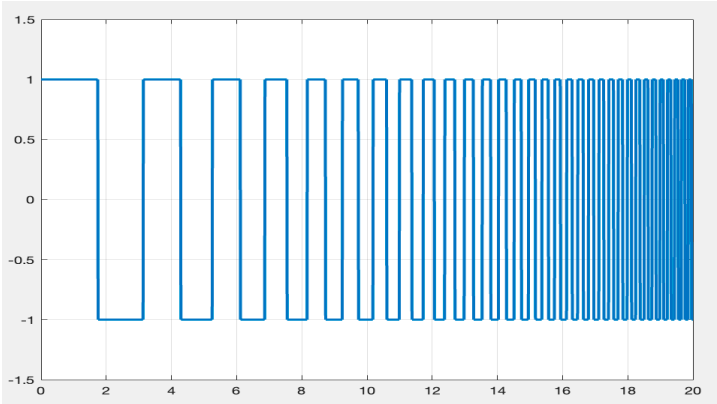
Step input/square wave



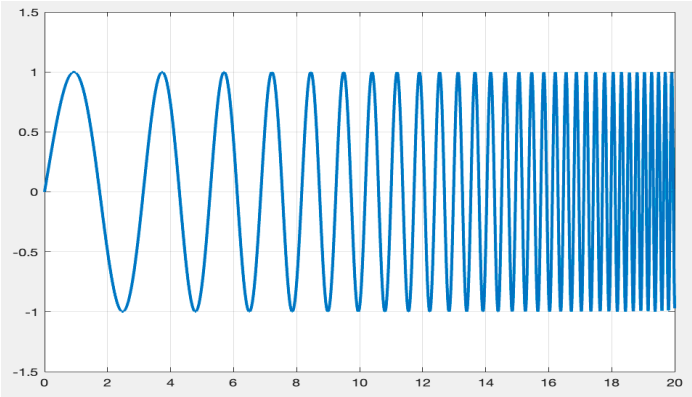
PRBS



Square chirp



Sine chirp



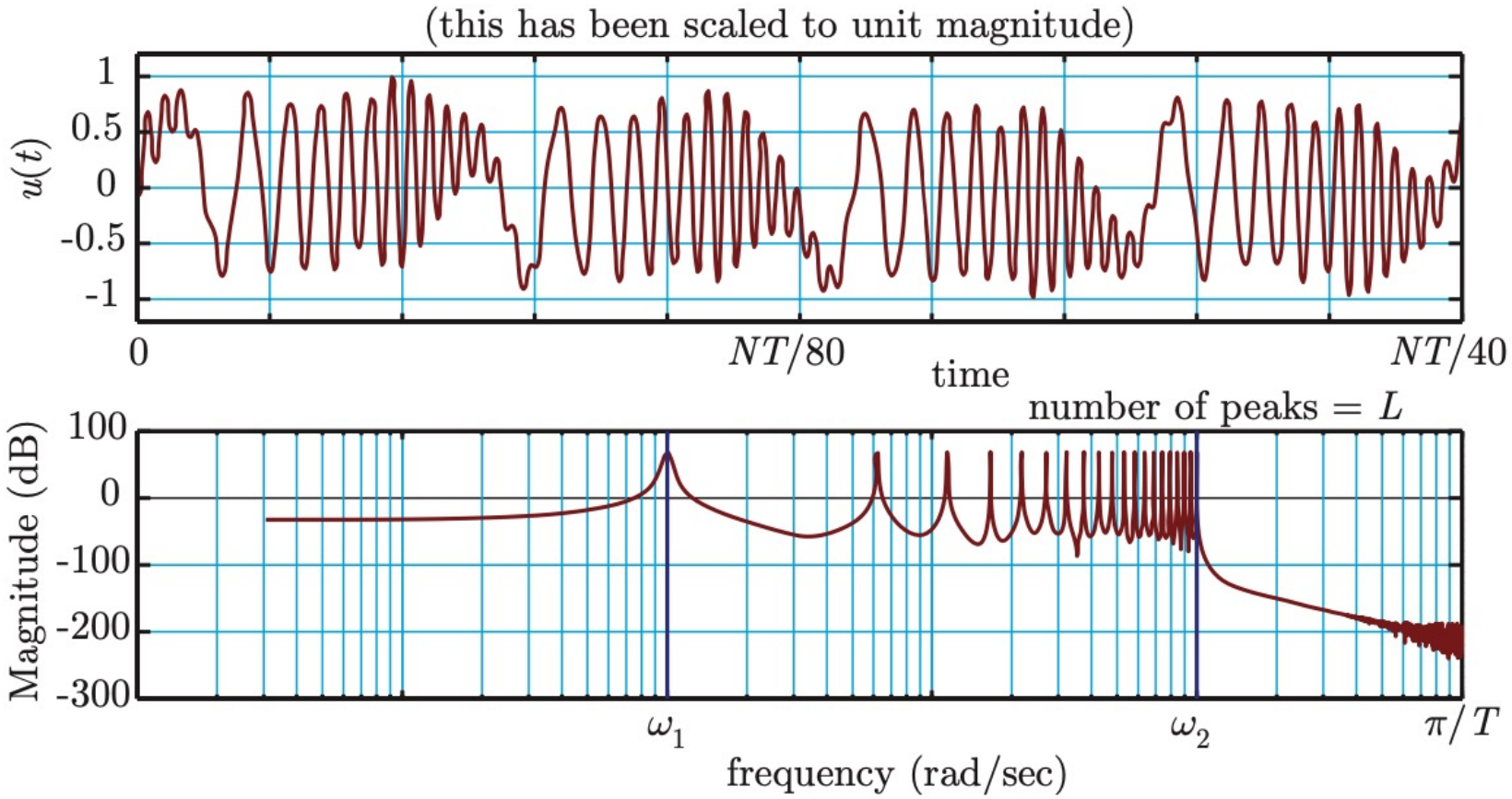
Multisine are also common

Exciting input signal ?

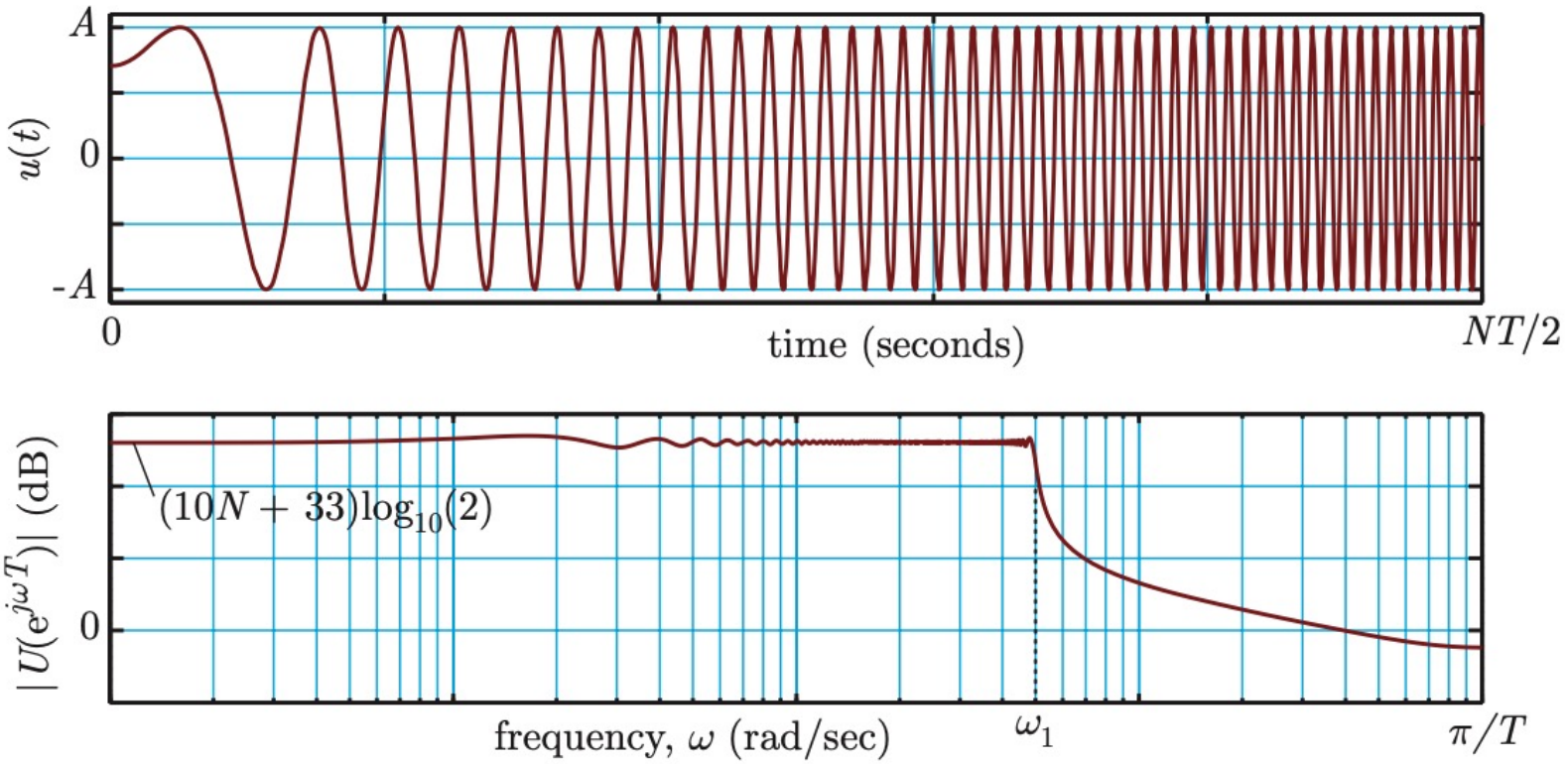
What makes a input signal exciting?

The issue boils down to the " frequency content" of the signal

Multi-sine signal



Chirp signal



Chirp signal

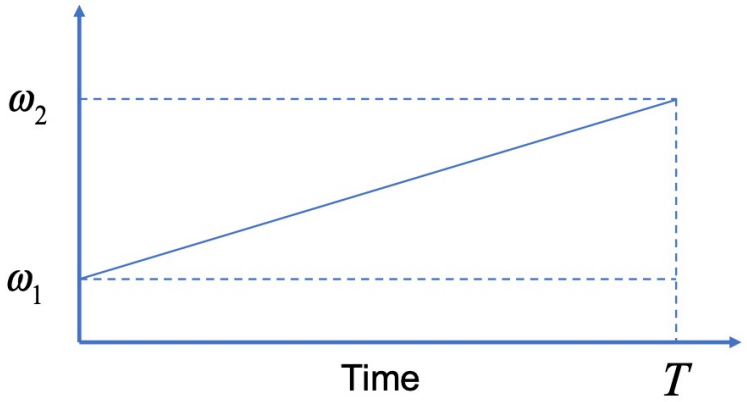
Continuously varying the frequency between ω_1 and ω_2 over a time period of $0 \le t \le T$ creates a Chirp Signal:

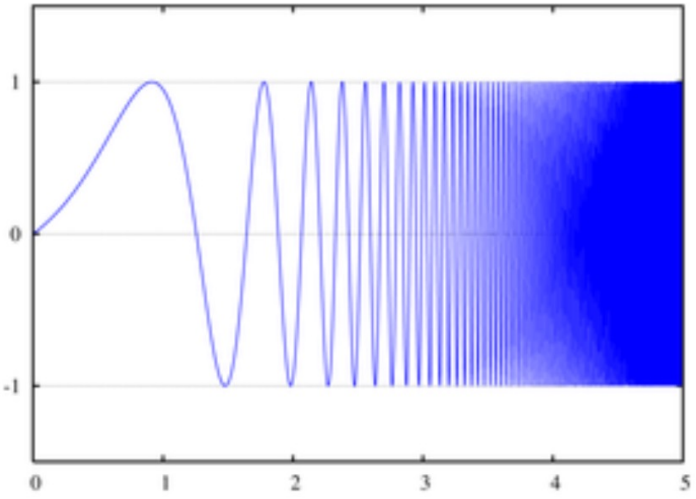
$$u(t) = A\cos\left(\omega_1 t + (\omega_2 - \omega_1)\frac{t^2}{2T}\right)$$

This is one of the most frequently used technique

The instantaneous frequency, ω_i, is obtained by differentiating $\varphi(t)$

$$\omega_i = \omega_1 + \frac{t}{T}(\omega_2 - \omega_1)$$





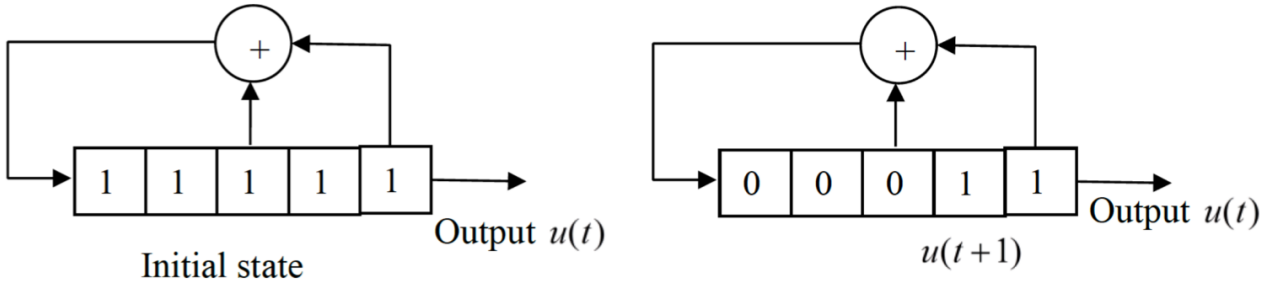
In Matlab, `chirp` function can generate a chirp input

- `u = chirp(t,fmin,tfinal,fmax,method,-90)`

Pseudo Random Binary Sequence (PRBS)

Now that random binary signals are useful for system identification, how can we generate them? A Pseudo-Random Binary Signal (PRBS) is a periodic, deterministic signal with white noise like properties. It has been widely used for system identification as well as for spread spectrum wireless communication and GPS.

Example: A 5 bit shift register



Binary shift register with feedback

Initial state

1 1 1 1 1 0001101110101000010010110 0 1111100011011101010000100101100

31 bits = $2^5 - 1$ | Repeat the same 31 bits

How to design PRBS ?

1. Choice of magnitude with a trade-off between signal to noise ratio and system nonlinearity.
2. Number of data N should be large enough to filter out the noise
3. The length of shift register n is related to the number of parameters in model and desired frequency resolution. n is chosen greater than 6 to have at least 32 frequency points excited. If system contains very low-damped modes a greater n should be chosen.
4. Number of periods is chosen such that 2 is satisfied.
5. A PRBS can be enriched in low frequencies by using a frequency divider (the clock frequency of the shift register is divided by p). A rule of thumb is :

   length of the largest pulse in PRBS $>$ settling time of the system

   $$n\, pT_s > t_{settling}$$

   This rule should be used with caution because a large value of p reduces the frequency contents of PRBS in high frequencies.

Input design in Matlab

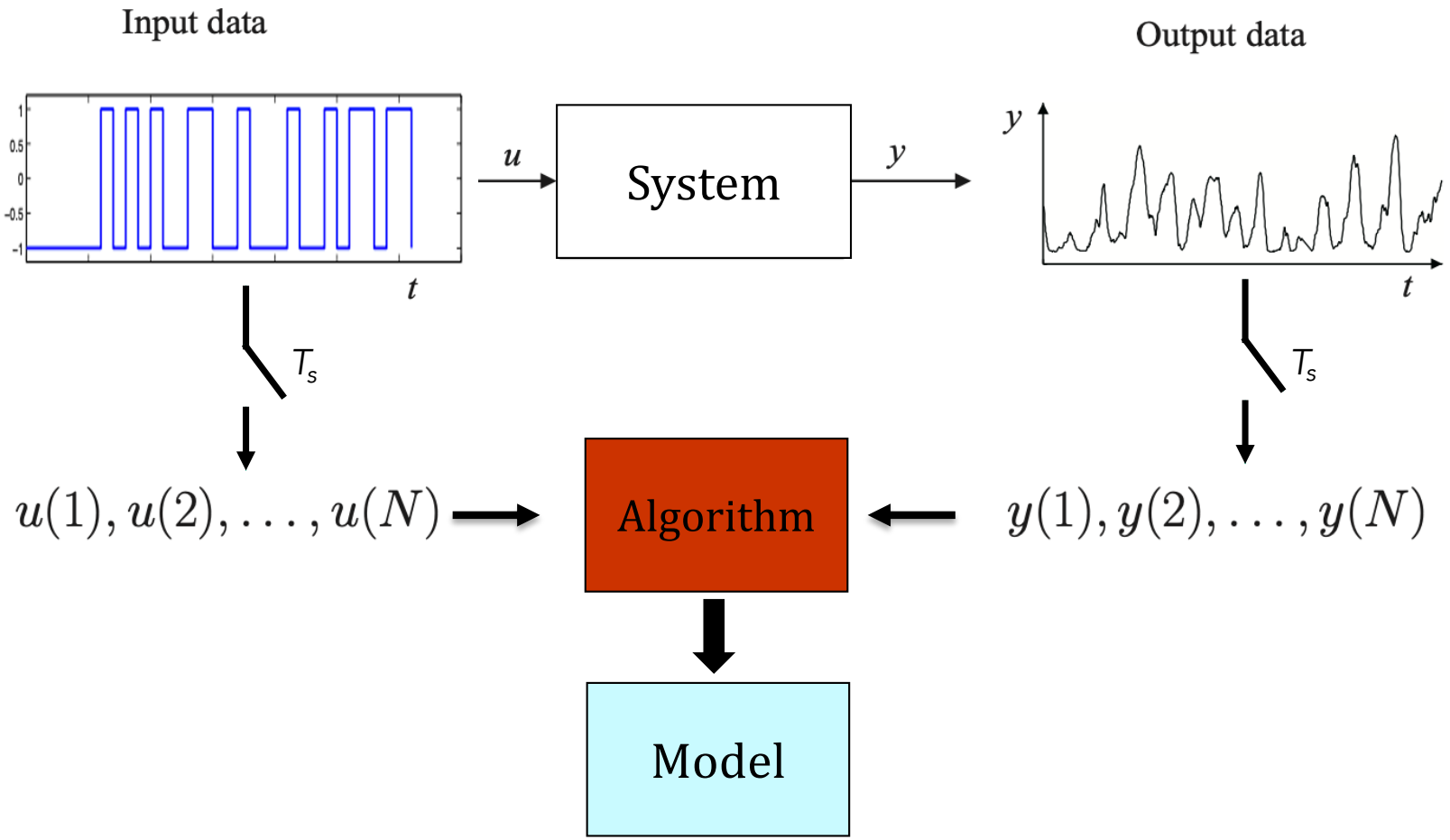
- In the Matlab SYSID toolbox, `idinput` function can generate various types of input
  - `u = idinput(N, type, band, levels)`

    where
    - N: number of data points
    - type:
      - RGS: generates a Random, Gaussian Signal
      - RBS: generates a Random, Binary Signal
      - PRBS: generates a Pseudo-random Binary Signal (PRBS)
      - SINE: generates a sum-of-sinusoid signal

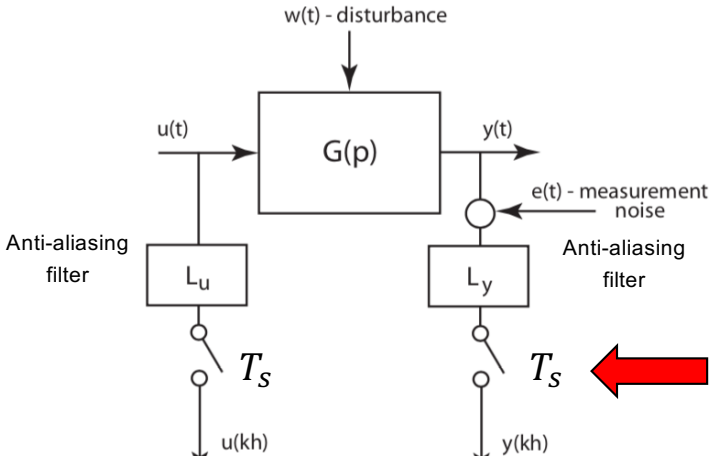
- In CONTSID toolbox, `prbs` function can generate a maximum-length PRBS input
  - `u = prbs(n,p,levels)`
    - n: order (number of stages) of the shift register. n must be between 6 and 18
    - p: coefficient such that the prbs signal remains constant over intervals of length p

$$n\,pT_s > t_{settling\_time}$$

Data collection



Data collection – Sampling period selection



A common rule of thumb is to choose the sampling frequency such as:

$$\omega_s > 100 \times \omega_{band} \qquad \text{or} \qquad T_s < \frac{2 \times \pi}{100 \times \omega_{band}}$$

where ω_{band} is the estimated system bandwidth (rad/s)

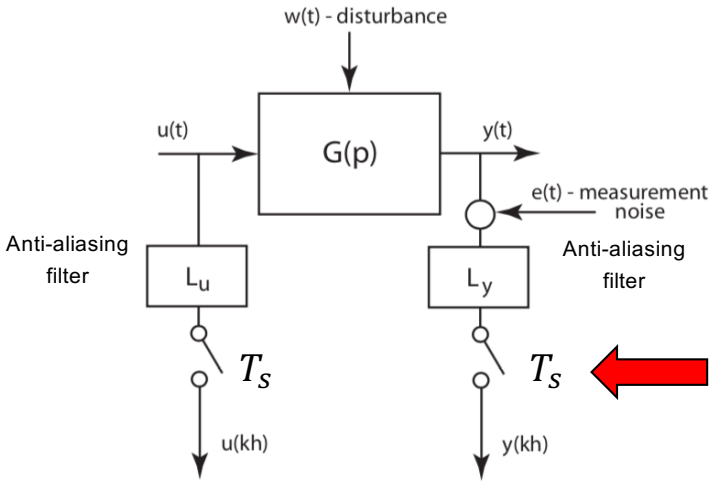
In practice: record the step response of the system, approximate it as a first-order model and determine the time-constant T, then

$$\omega_{band} = \frac{1}{T}$$

which leads to the constraint

$$T_s < \frac{2 \times \pi \times T}{100}$$

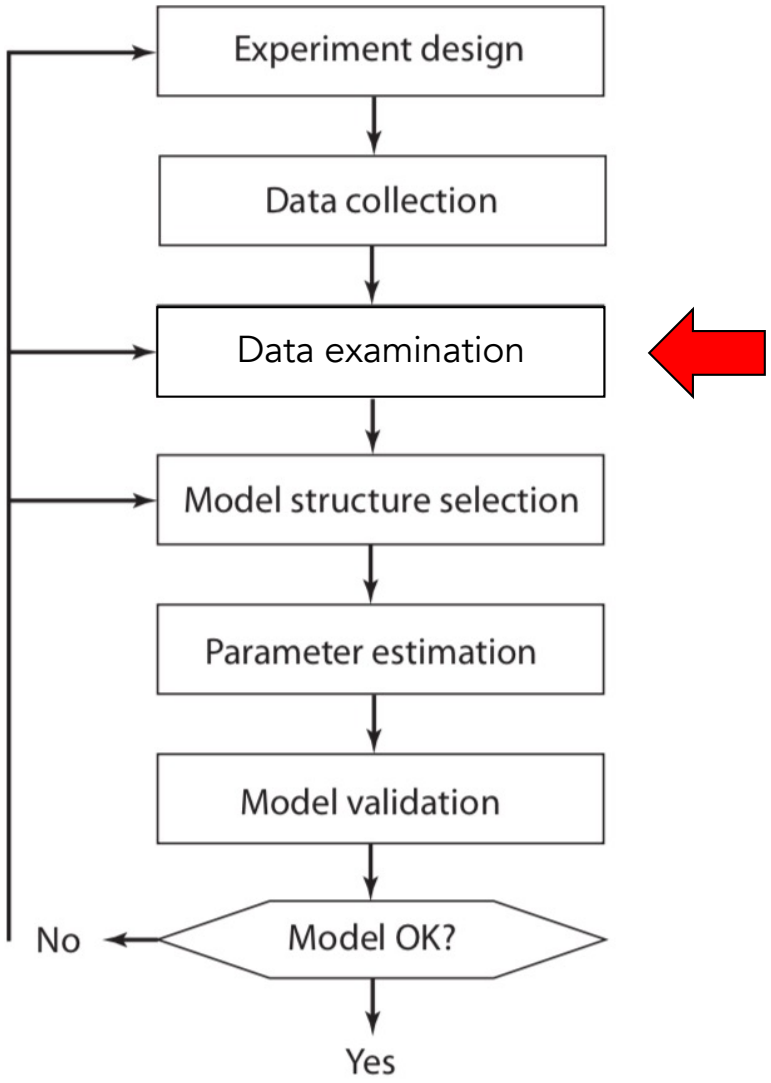
Data collection – Sampling period selection



If you are uncertain of what an appropriate sampling frequency for the system should be, keep in mind that it is better to sample a bit too fast than a little too slow

Notice that sampling too quickly is not always good for discrete-time model identification while oversampling is not critical for continuous-time methods

Data-driven system identification
An iterative procedure



Examination of the collected data

ALWAYS plot FIRST the input/output data !

Examine carefully the measured data and identify possible problems

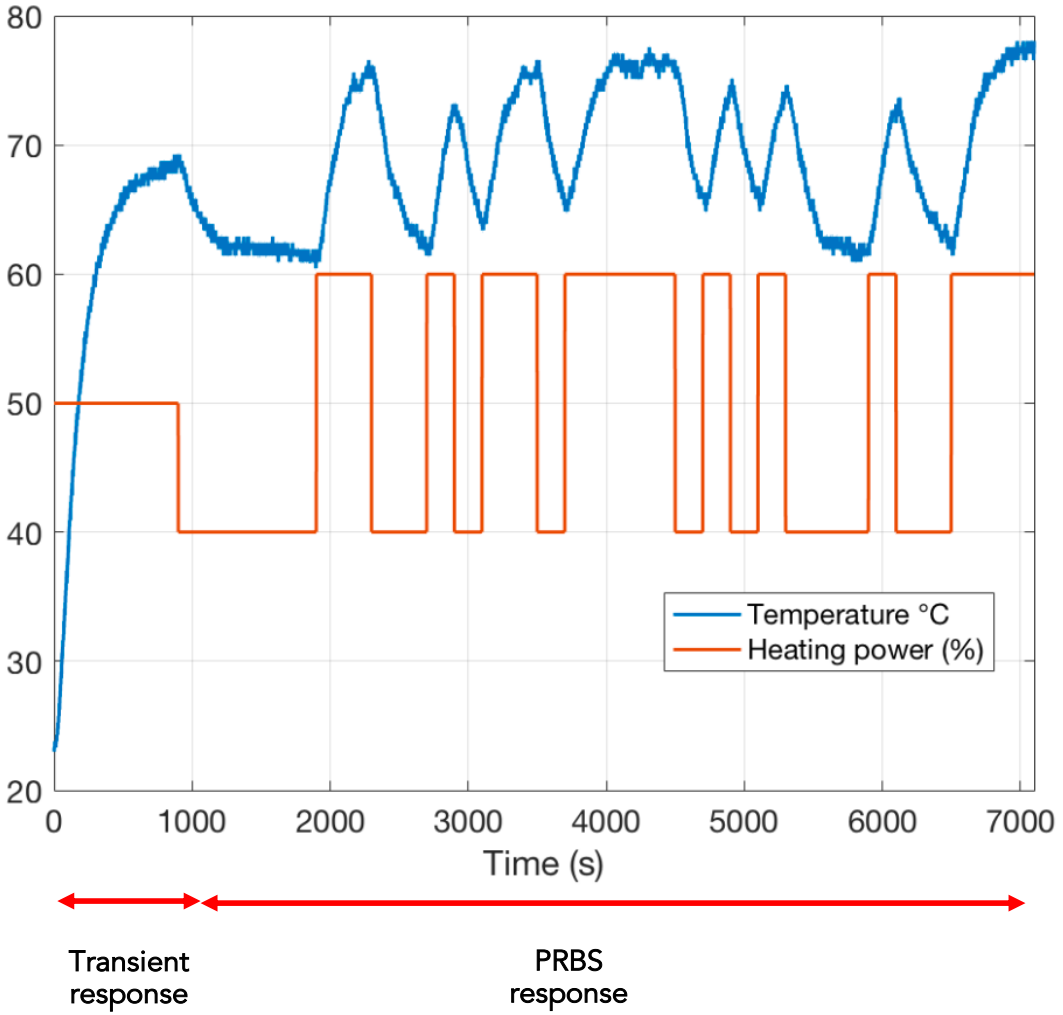
- Offset and drift (low-frequency disturbances)
- Occasional bursts and outliers
- High-frequency noise/disturbance

Select good/relevant segments of data for model estimation and validation

Examination of the collected data
What do you observed?

Operating Point for y

Operating Point for u



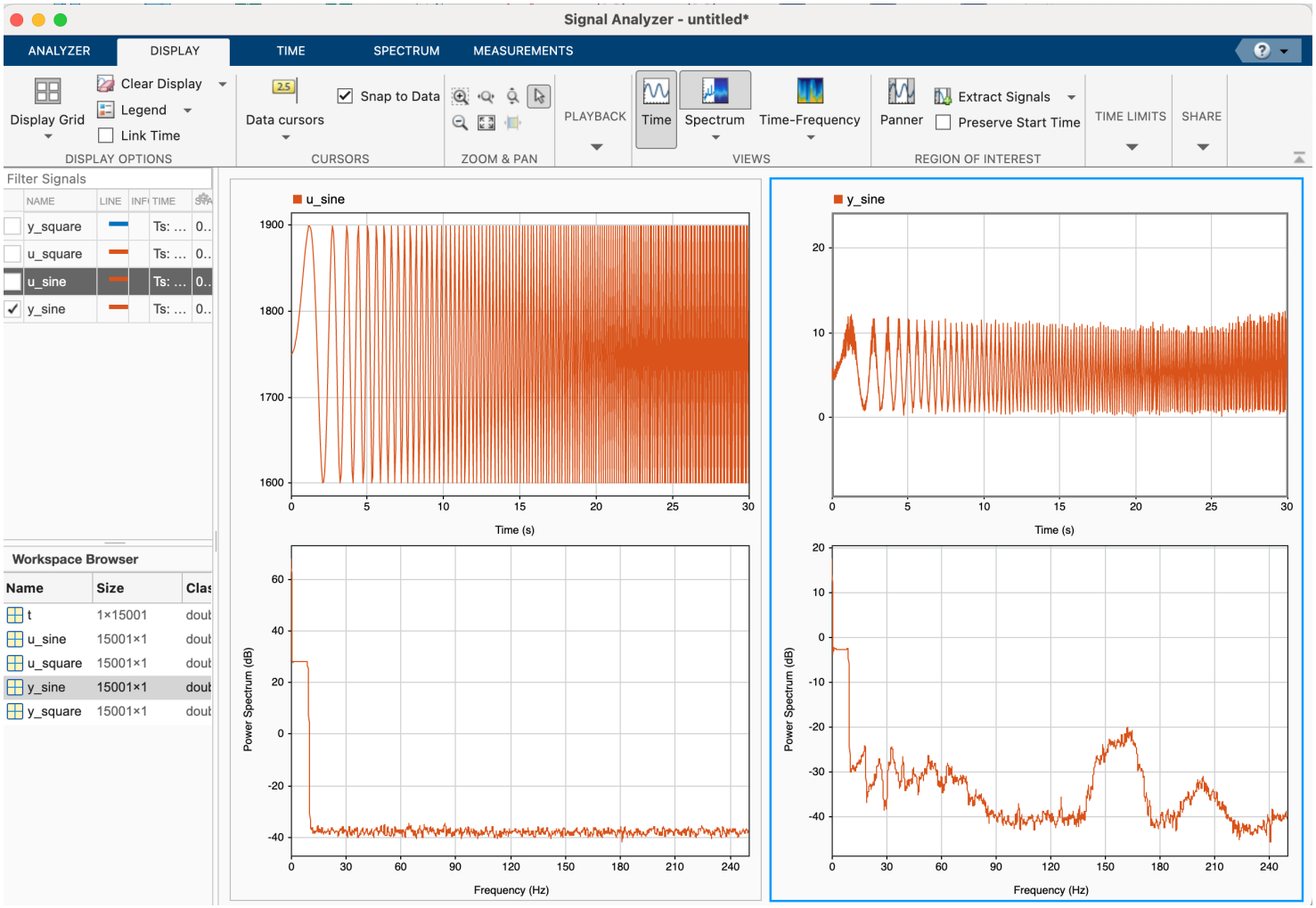
Post processing of the collected data

- Input and output should be scaled to have approximately the same magnitude to avoid the numerical problems
- Look at the data:
  - Proper signal levels, frequencies, disturbance...
- Remove
  - transients before reaching the correct operating point (for nonlinear
  - signal mean (if not a physical model)
  - drift and slow trends (`detrend` in MATLAB)
  - high frequency noise should have been removed by antialiasing filter. In case apply a low-pass filter (keeping the breakpoints of the Bode plot)
  - "Outliers" (manifestly erroneous measurements): check plot of residuals

$$y(t) - \hat{y}(t|\theta)$$

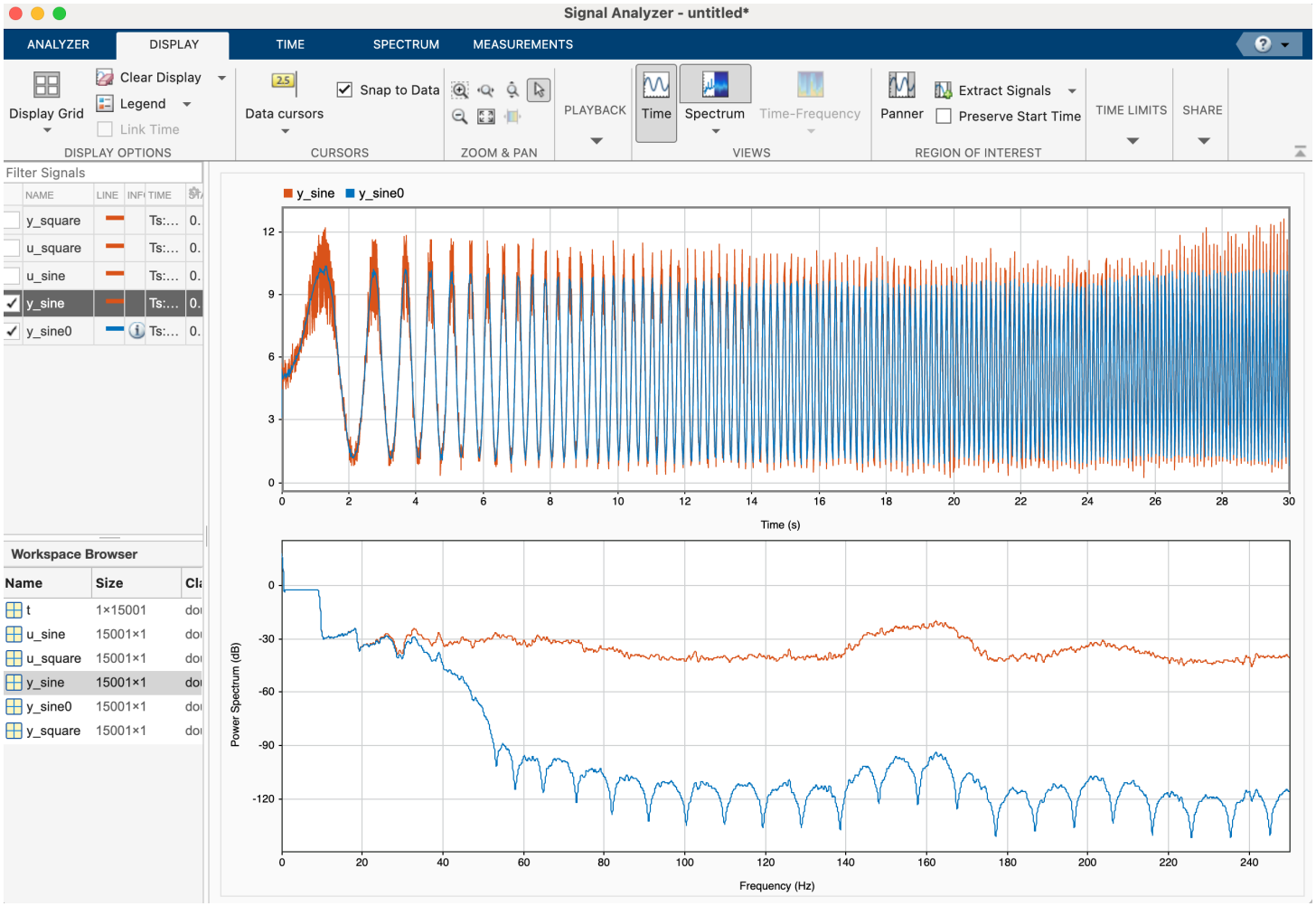
Post low-pass filtering of the collected data

- Use of the SignalAnalyzer App to display both input/output spectrum

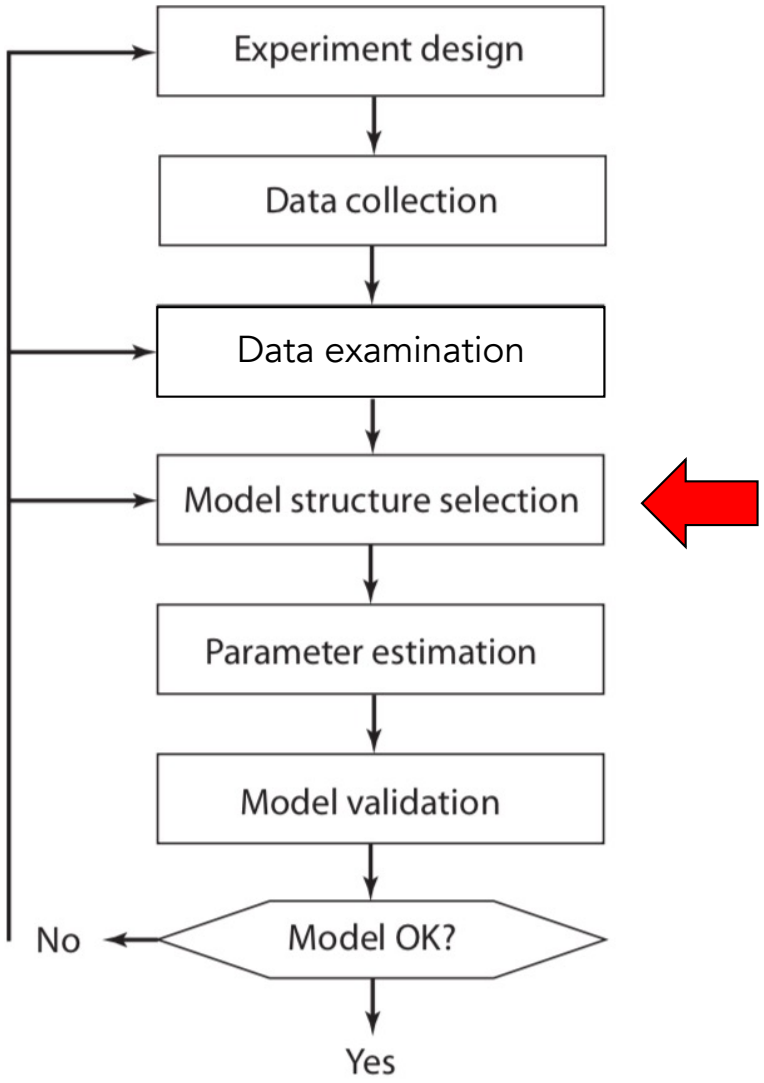


Post low-pass filtering of the collected data

- Use of the SignalAnalyzer App to apply low-pass filtering on both input/output data. Use the filtered data to fit a model



Data-driven system identification
An iterative procedure



Family of linear model structures

- A model structure is a mathematical relationship between input and output variables that contains unknown parameters

- Common examples of linear model structures are:

➢ Input/output polynomial models	$Ay = \frac{B}{F}u + \frac{C}{D}e$	CT/DT
➢ Low-order process models plus delay	$G(s) = \frac{K_p}{1 + T_{p1}s} e^{-T_d s}$	CT
➢ Transfer function models plus delay	$G(s) = \frac{(b_0 + b_1 s + b_2 s^2 + \ldots)}{(1 + f_1 s + f_2 s^2 + \ldots)} e^{-T_d s}$	CT/DT
➢ State-space models	$\dot{x} = Ax + Bu$ $y = Cx + Du$	CT/DT

- Most of these model structures can be expressed in continuous-time (CT/DT) and in discrete-time (DT)

Discrete-time versus continuous-time model ?

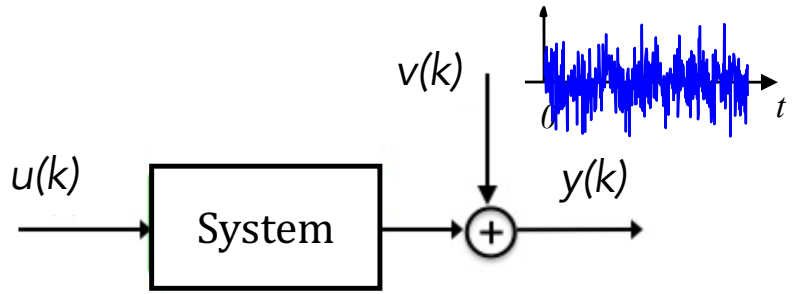
- ✓ For several decades, the general mainstream identification approach has been to identify DT models from sampled data

- ✓ To obtain satisfying results, DT model identification
  - Often requires the active participation of an experienced practitioner

- ✓ **Direct** CT model identification includes many advantages and is therefore recommended:
  - well adapted to the current sampling situations: **fast or irregular**
  - requires less participation from the user ***(inherent pre-filtering)***
  - makes the ***application*** of the SYSID procedure much ***easier***

Models for measurement noise

So far: only deterministic models



$u(k)$ = input

$v(k)$ = measurement noise

$y(k)$ = output

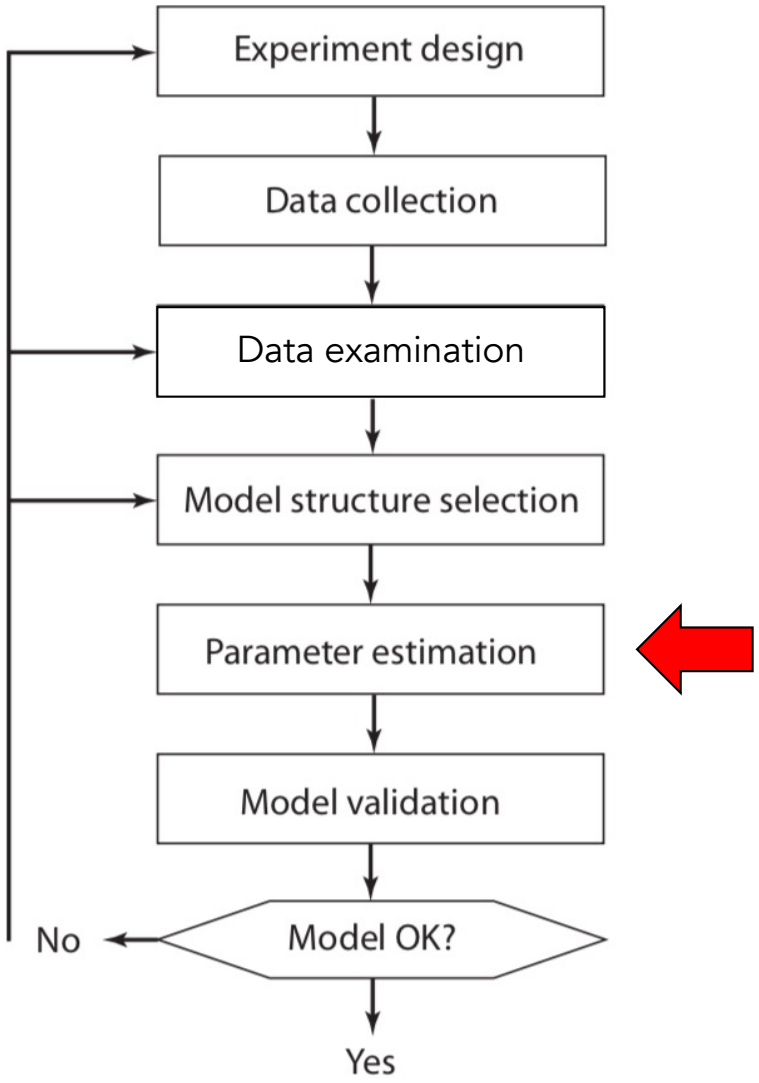
- Measurement noise modelled as a stochastic discrete-time signal
- Stochastic models:
  - means, covariances
  - spectra (energy or power)

$$v(k) = H(q)e(k) = \frac{C(q^{-1})}{D(q^{-1})}e(k)$$

$e(k)$ is a white Gaussian noise

q^{-1}: *delay operator*

Data-driven system identification
An iterative procedure



Optimization methods for parameter model estimation

- Common optimization algorithms for estimating the parameters of the models are:
  - Least-squares (LS) method
  - Instrumental variable (IV) method
  - Prediction error method (PEM)
  - Subspace method

Least squares (LS) method

System:

$$y(t) = \boldsymbol{\varphi}^T(t)\boldsymbol{\theta} + v(t), \quad t = 1, \ldots, N$$
$$\boldsymbol{Y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \boldsymbol{v}$$

where $v(t)$ is a disturbance and $E\boldsymbol{v} = 0$, $E\boldsymbol{v}\boldsymbol{v}^T = \boldsymbol{R}$.

Estimate:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{Y} = \Big[\sum_{t=1}^{N}\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^T(t)\Big]^{-1}\Big[\sum_{t=1}^{N}\boldsymbol{\varphi}(t)y(t)\Big]$$

Instrumental variable (IV) methods

System:

$$y(t) = \boldsymbol{\varphi}^T(t)\boldsymbol{\theta} + v(t), \quad t = 1, \ldots, N$$

where $v(t)$ is a disturbance with $E\boldsymbol{v} = 0$.

Estimate: Modify the least squares solution. We get:

$$\hat{\boldsymbol{\theta}} = \Big[\sum_{t=1}^{N} \boldsymbol{z}(t)\boldsymbol{\varphi}^T(t)\Big]^{-1}\Big[\sum_{t=1}^{N} \boldsymbol{z}(t)y(t)\Big]$$

where $\boldsymbol{z}(t)$ is the vector of instruments.

- ✓ Amongst the different IV versions, one is particularly recommended:
  - SRIVC: Simple Refined Instrumental Variable algorithm for COE models
    - robust to noise assumptions and algorithmic aspects

Prediction error methods (PEM)

Idea: Model the noise as well. General methodology applicable to a broad range of models.

The following choices have to be made:

- Choice of model structure. Ex: ARMAX, OE.
- Choice of predictor $\hat{y}(t|t-1,\boldsymbol{\theta})$.
- Choice of criterion function. Ex: $V(\boldsymbol{\theta}) = \frac{1}{N}\sum \varepsilon^2(t,\boldsymbol{\theta})$.

Estimate:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} V(\boldsymbol{\theta})$$

Subspace methods

- Use linear algebra to estimate state-space models
  - Well adapted to multivariable-input multivariable-output (MIMO) systems
  - More to come next week

Main estimation routines in the SID and CONTSID toolboxes

Matlab SID toolbox **CONTSID toolbox**

Commands for Offline Estimation

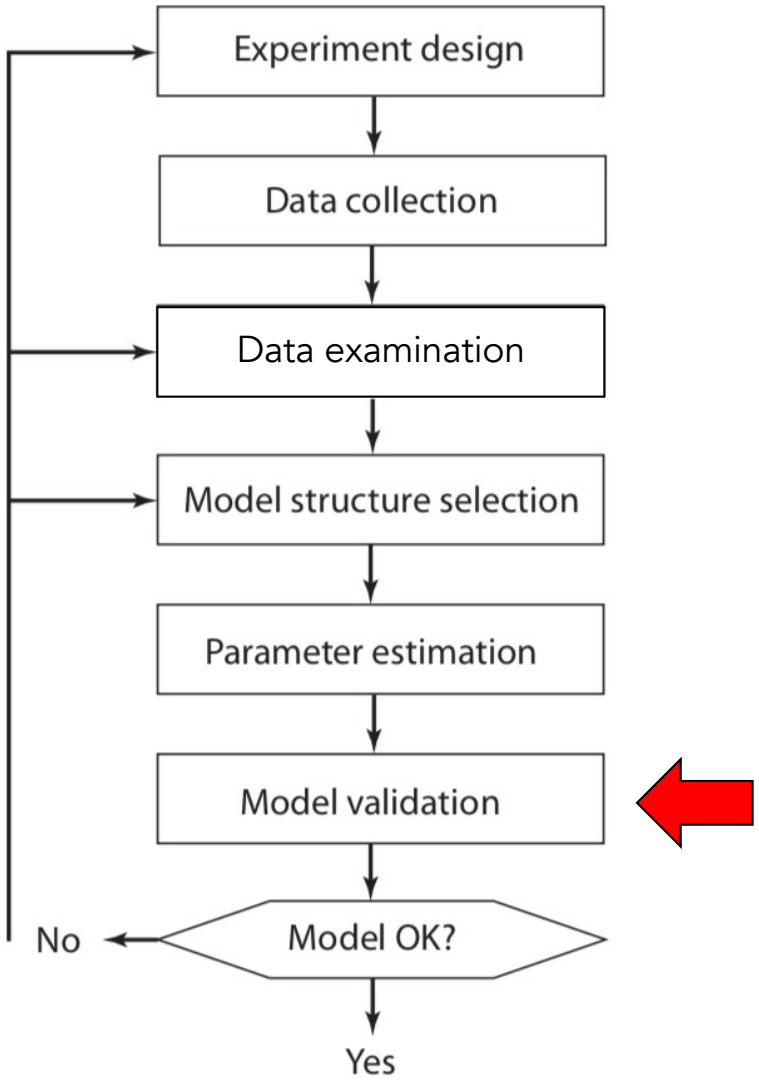
Model Type	Estimation Commands	
Transfer function models	`tfest`	`tfsrivc/tfrivc`
Process models (low-order transfer functions expressed in time-constant form)	`procest`	`procsrivc`

Linear input-output polynomial models	`armax` (ARMAX and ARIMAX models)
	`arx` (ARX and ARIX models)
	`bj` (BJ only)
	`iv4` (ARX only)
	`ivx` (ARX only)
	`oe` (OE only)
	`polyest` (for all models)
		`lssvf (CARX)`
		`rivc (CBJ)`
		`srivc (COE)`
		`coe (COE)`

State-space models	`n4sid`
	`ssest`
	`ssregest`
		`sidgpmf`

- The majority of these estimation algorithms are iterative
- As these routines can estimate different models quickly, you should try as many different structures as possible to see which one produces the best results

Data-driven system identification
An iterative procedure

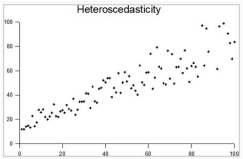


Recommended workflow of model validation

1. Compare simulated model output with estimation data
   - ➢ use a model fit criterion: *e.g.* the FIT value or the coefficient of determination R_T^2

2. Compare simulated model output with validation data

1. Perform statistical tests on prediction errors
   - ➢ Plot the autocorrelation of the residuals and the cross-correlation between the input and the residuals
   - ➢ This statistical test is often difficult to pass for real-life data
     - ➢ Non-linear effects, time-varying effects, noise heteroskedasticity, …



Plot with random data showing heteroscedasticity: The variance of the *y*-values of the dots increase with increasing values of *x*.

2. Interpret the main features of the identified model in a physical sense
   - Steady-state gain, time-constants, time-delay, damping coefficient, natural frequencies

Estimation data versus validation data

Split your data:

Estimation data

data use for parameter estimation

$\Downarrow$

compute $\hat{\theta}_N$

Validation data

data not used for computing $\hat{\theta}_N$

use them to evaluate the quality of the fit

$\Downarrow$

Cross-validation

Common model accuracy measures

Let $\hat{y}_k$ be the model prediction. Calculate the *residuals/prediction errors* as

$$\varepsilon_k = y_k - \hat{y}_k$$

Common model accuracy measures are:

- the FIT percentage

$$\text{FIT} = 100 \times \left(1 - \frac{\|y_k - \hat{y}_k\|}{\|y_k - \bar{y}_k\|}\right) \quad \text{(expressed in \%)}$$

  - indicates the agreement between the model and measured output
    - 100% means a perfect fit, and 0 indicates a poor fit

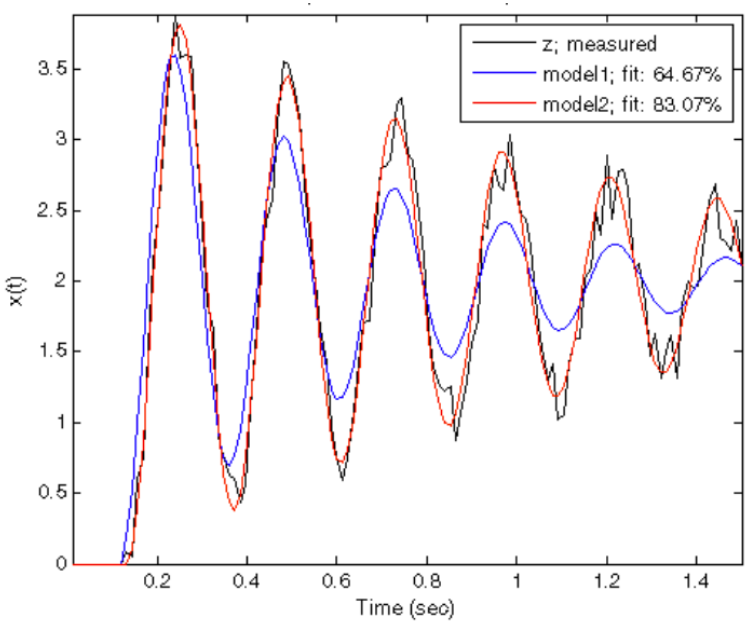
- the coefficient of determination

$$R_T^2 = 1 - \frac{\sigma_\varepsilon^2}{\sigma_y^2}$$

  - indicates the agreement between the model and measured output
    - the closer R_T^2 to 1, the better the fit

Comparison of model response to measured response with the estimation data

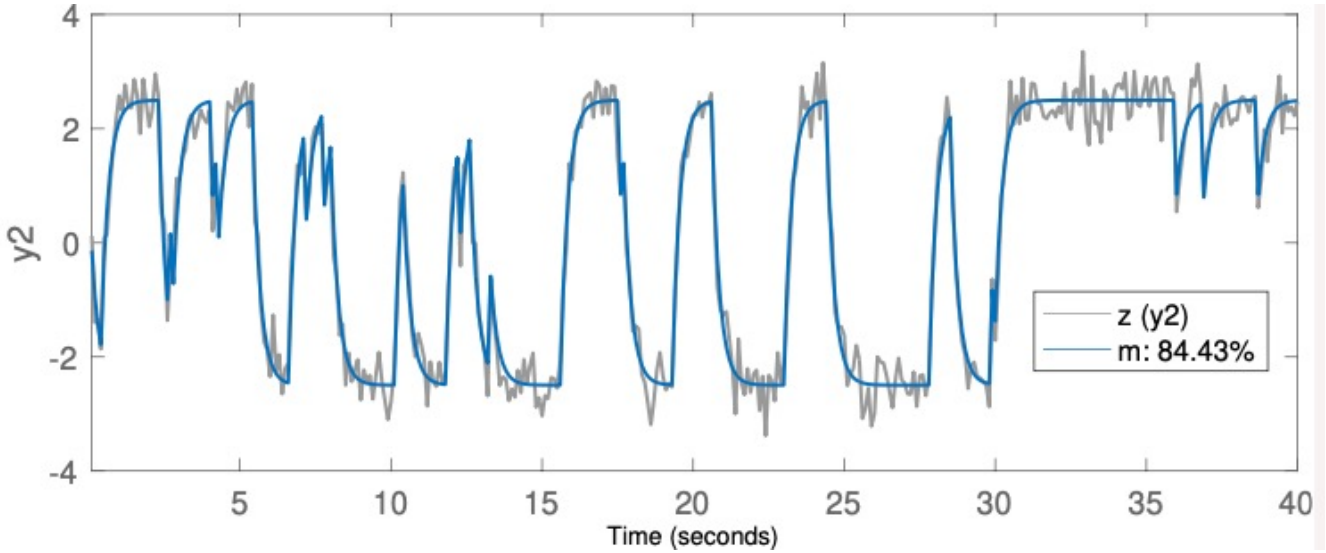
- Typically, you evaluate first the quality of models by comparing their model responses to the measured output with the estimation data



➢ model2 above is better than model1 because model2 better fits the data (83% vs. 65%)

Comparison of model response to measured response with the estimation data: Warning message

- Do not be impressed by a good fit to data on a simulation test with the estimation data

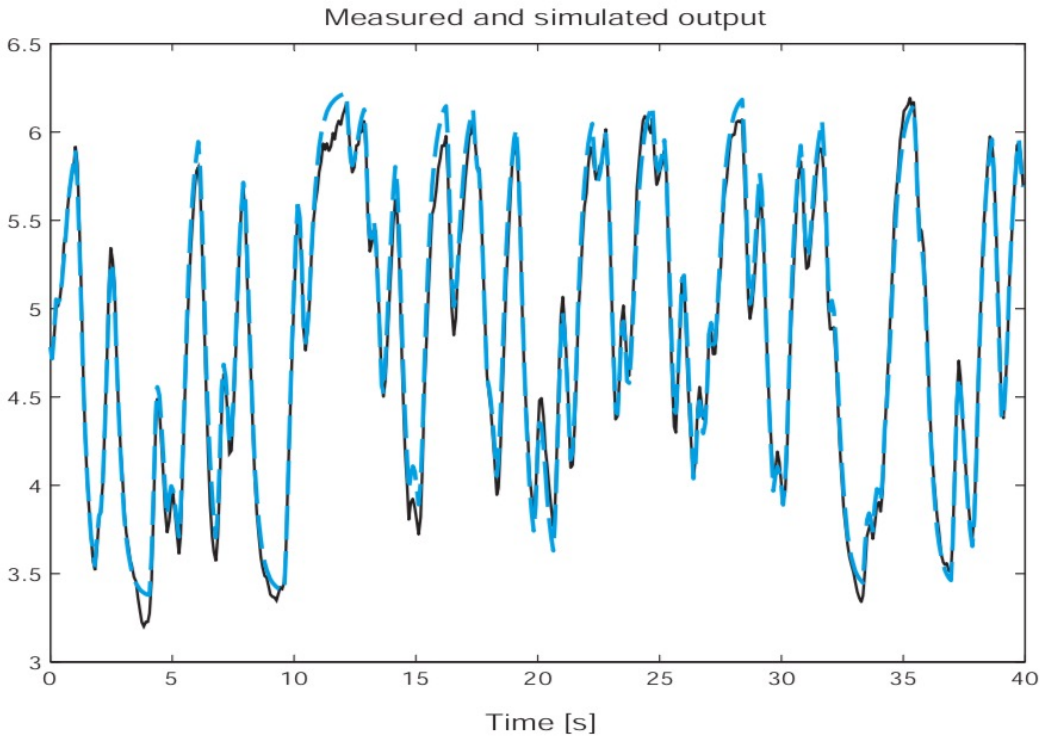


- The real test is to see how well the model can reproduce the validation data: cross-validation data test

Comparison of model response to measured response with the validation data

Apply input signal in validation data set to estimated model

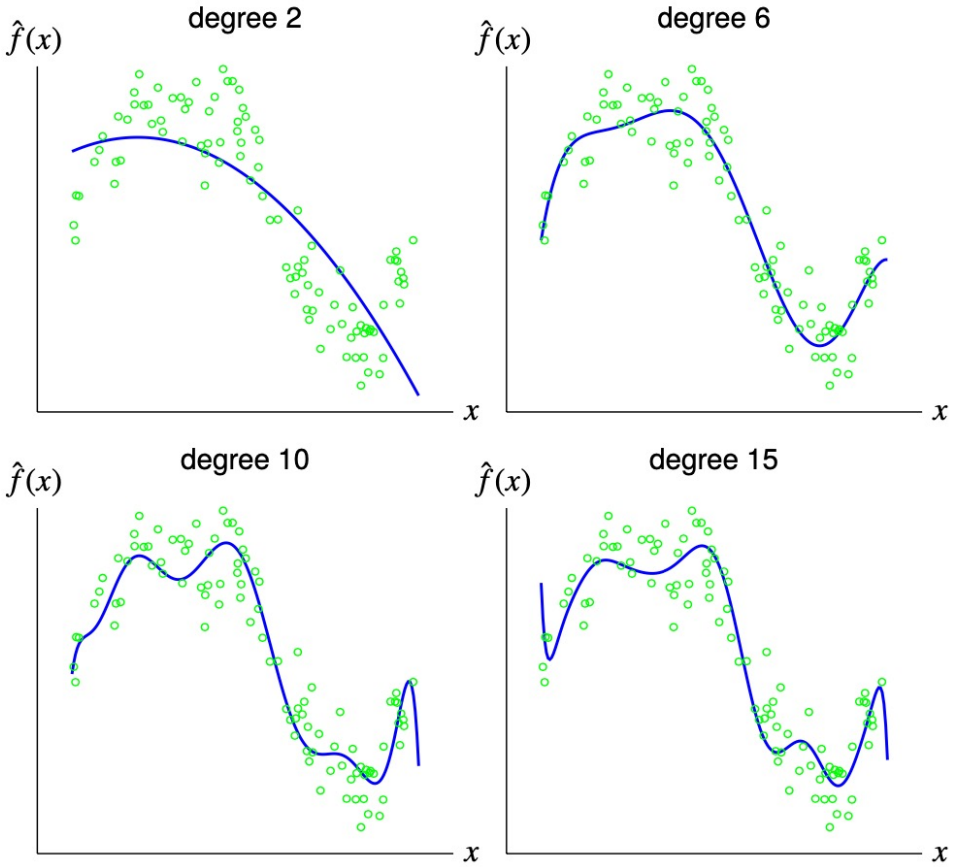
Compare simulated output with output stored in validation data set.



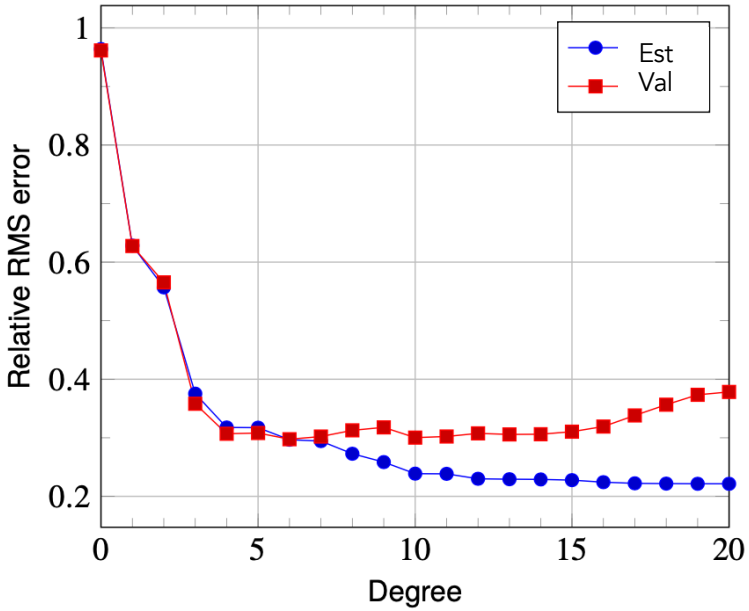
Choice of the model order
Example: LS polynomial model fit

$$\hat{f}(x) = \theta_1 + \theta_2 x + \cdots + \theta_p x^{p-1}$$

- Model fit using estimation data of 100 noisy points
- Plots below show simulation results on validation data of 100 points



RMS error versus polynomial degree for both estimation and validation data



Interpreting results:

- With a 6-th degree polynomial, the relative RMS test error for both estimation and validation data is around 0.3. It is a good sign, in terms of generalization ability, that the estimation and validation errors are similar
- RMS error plot suggest polynomial degree 4, 5, or 6 as reasonable choices

- Too few parameters: model fails to capture the function
- Too many parameters, the model captures the noise
- If validation RMS errors are larger than estimation RMS errors, model is over-fit
- Methods for avoiding overfit:
  - Keep the model simple
  - Use regularization

Traditional criteria for model order selection

If fresh validation data is not available (=no cross-validation)

- A loss function $J(n_p,Z^N)$ is formulated from two functions:
  - one term measuring the model fit based on the loss function
  - one term penalizing the model complexity

$$J(n_p,Z^N) = \log V(\hat{\theta}_{n_p},Z^N) + \beta(n_p,Z^N)$$

  - *$\beta(n_p,Z^N)$ is a function which should increase with the model order but decrease to zero when $N \longrightarrow \infty$*

Traditional criteria for model order selection

- Usual approach: pick the model that minimizes
  - AIC (Akaike's Information Criterion)

$$AIC(n_p, Z^N) = \log V(\hat{\theta}_{n_p}, Z^N) + \frac{2n_p}{N}$$

  - FPE (Final Prediction Error)

$$FPE(n_p, Z^N) = \frac{1 + \frac{n_p}{N}}{1 - \frac{n_p}{N}} V(\hat{\theta}_{n_p}, Z^N)$$

  - YIC (Young's Information Criterion)

$$YIC = \log\left(\frac{\sigma_\varepsilon^2}{\sigma_y^2}\right) + \log \frac{1}{n_p} \sum_{j=1}^{n_p} \frac{\sigma_\varepsilon^2 \hat{p}_{jj}}{\hat{\theta}_j^2}$$

Choosing among different model orders

- One approach is to fit multiple models to the same data

  Which is the best model among these ?

- Assuming the goal is to make good predictions on the validation data
  - Select the model order that has the best YIC, AIC, FPE with the highest associated FIT/ R_T^2 on the validation data

np	m	n	nk	RT2	YIC	Niter	FPE	AIC
5	2	3	0	0.83	-8.33	10	2.35	0.85
6	2	4	0	0.92	-8.19	5	1.13	0.12
7	1	5	0	0.53	-7.51	10	6.90	1.93
4	1	3	0	0.51	-3.72	10	8.40	2.12
5	1	4	0	0.03	-0.89	10	13.9	2.63

  - If several model candidates achieve similar performance, you should choose the simplest (lowest-order) one among these candidates

Model order selection from subspace state-space model estimation

A pragmatic and interesting way to choose the model order is to use the susbpace-based estimation method n4sid directly, as an alternative to ssest

The algorithm automatically estimates a discrete-time state-space model of the best order in the 1:10 range with the estimation data

```
>>M = n4sid(data)
```

More about n4sid to come

Choice of the model order
Take-home message

- Choosing the model order is difficult
  - Start with low-order candidate models, and so on. You can compare higher order models against these
  - Compare candidate models using validation data
  - Increasing the model order will always increase the FIT/ R_T^2 on the estimation data, but the important question is whether or not it substantially increases the FIT/ R_T^2 on the validation data sets
  - Increasing the model order can easily lead to over-fit. To avoid the over-fit:
    - keep the model simple (low-order)
    - use information criteria
    - use regularization

Model assumption verification via residual statistical tests

- When you choose a model structure (ARX, ARMAX, OE, BJ, …), you make the implicit assumption that the input/output has been generated by the chosen model (assumption about the noise model in particular)
- With this assumption, residual tests coming from probability/statistics can be used

If we fit the parameters of the model

$$y[t] = G(q;\theta)u[t] + H(q;\theta)e[t]$$

to data, the *residuals*

$$\varepsilon[t] = H(q;\theta)^{-1}\left\{y[t] - G(q;\theta)u[t]\right\}$$

explains mismatch between model and observed data.

If the model is correct, the residuals should be

- white, and
- uncorrelated with u

Autocovariance of the residuals

Residuals

$$\epsilon(t, \hat{\theta}_N) = y(t) - \hat{y}(t|\hat{\theta}_N)$$

- Ideally these should be independent (if noise model is well-estimated)
- Estimating Autocovariance of the residuals

$$\hat{R}_\epsilon(\tau) = \frac{1}{N} \sum_{t=1}^{N} \epsilon(t+\tau)\epsilon(t)$$

- Plot $\hat{R}_\epsilon$
- independent residuals: $\hat{R}_\epsilon(\tau)$ is a δ function

- Large components indicate unmodelled dynamics for the noise model

Cross-covariance between the residuals and the input

Residuals:

$$\epsilon(t, \hat{\theta}_N) = y(t) - \hat{y}(t|\hat{\theta}_N)$$

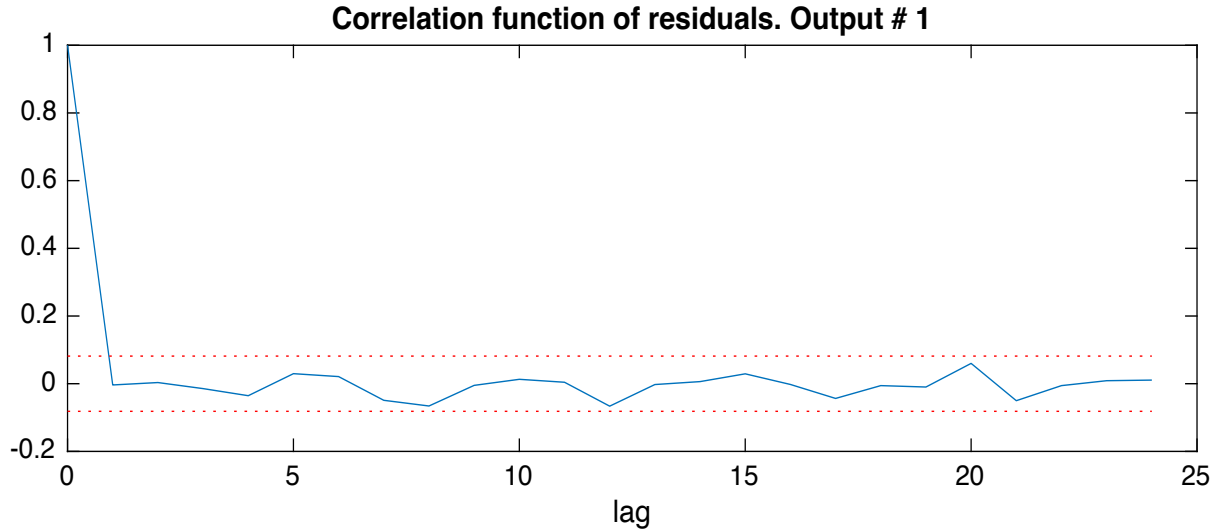
- Ideally ϵ should be independent of u
- Estimating Cross-covariance between ϵ and u

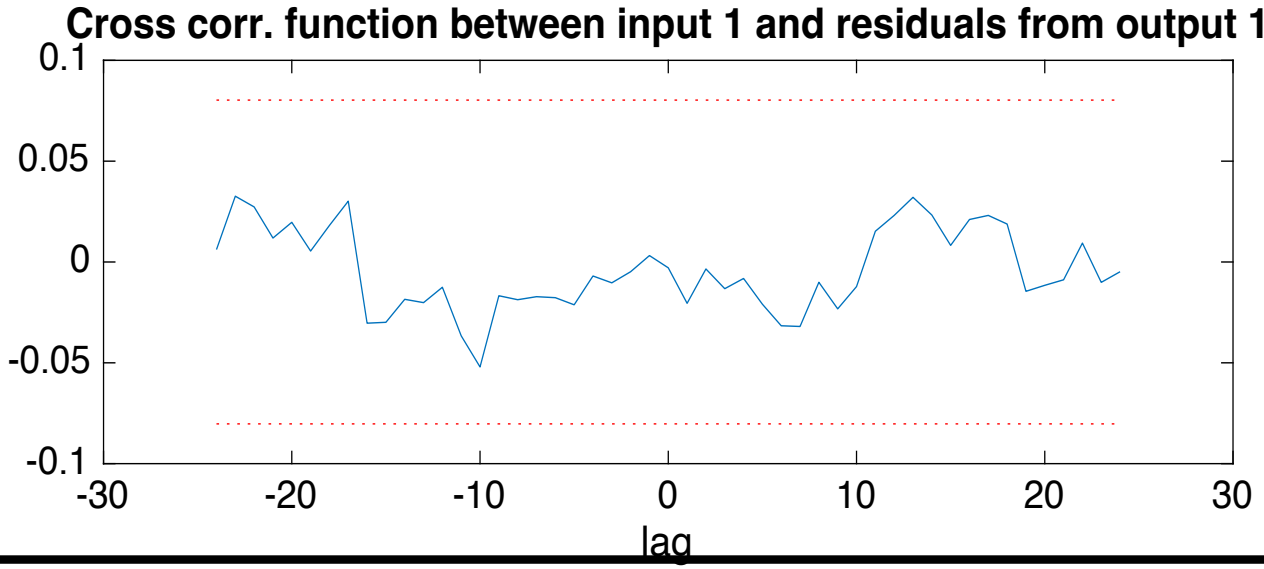
$$\hat{R}_{\epsilon u}(\tau) = \frac{1}{N} \sum_{t=1}^{N} \epsilon(t+\tau) u(t)$$

- Plot $\hat{R}_{\epsilon u}$
- Correlation for negative τ: perhaps there is a feedback ($u(t)$ "depends" on $\epsilon(t-\tau)$)

- Large components indicate unmodelled dynamics for the plant model

Statistical tests on the residuals





Sotfware available

Most of the theory covered in the course for continuous-time model identification is implemented in:

- the CONtinuous-Time System IDentification (CONTSID) toolbox for Matlab

- *A lot can be learned from the demos available*

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

63

Main features of the latest version 7.4

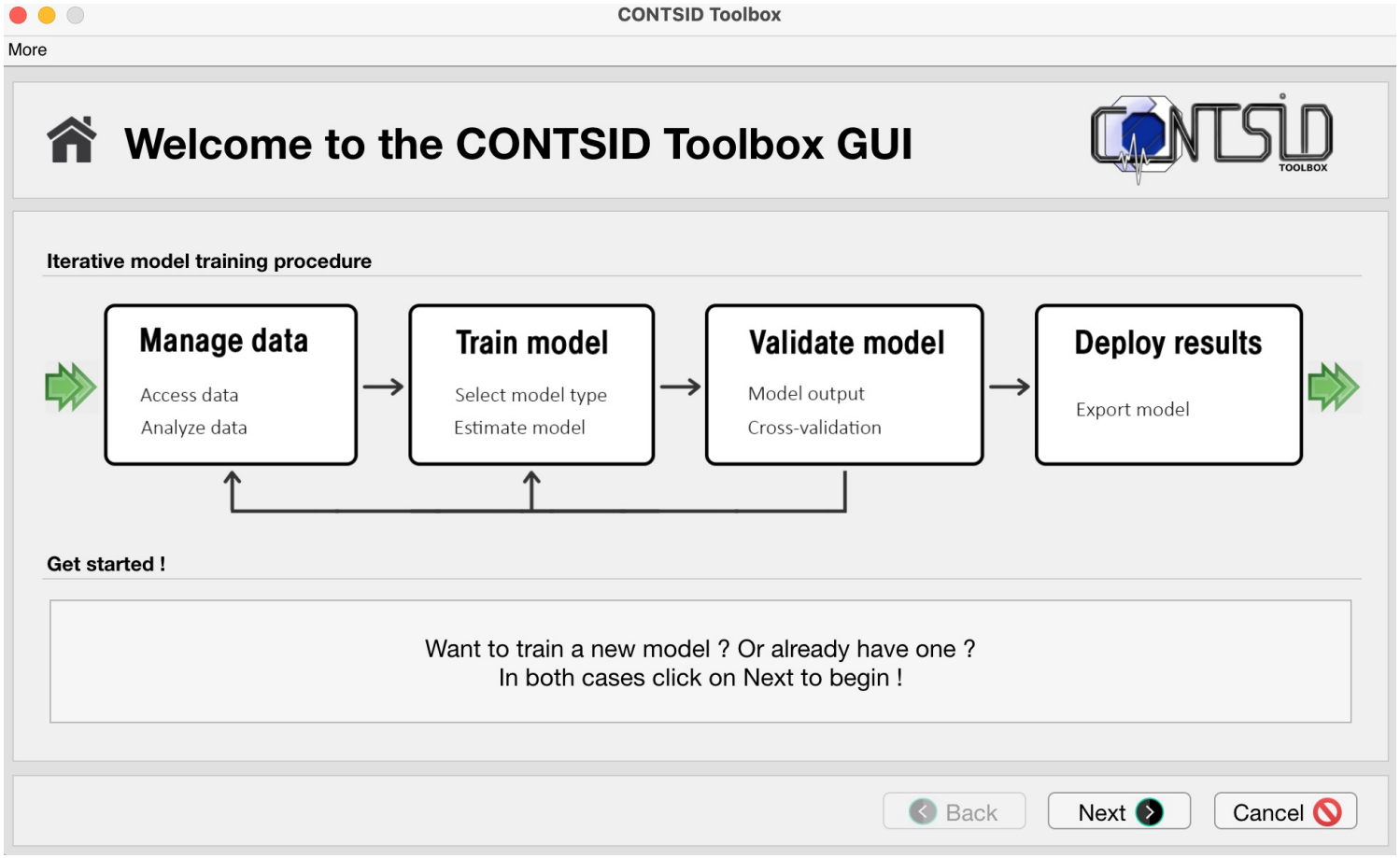
✓ Core of the routines mainly based on **refined optimal IV**: *SRIVC*

– CONTSID includes also a few PEM and subspace-based methods

- *SRIVC-based parameter estimation schemes for more advanced identification*
  - ***Polynomial models:*** *SRIVC*
  - ***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

64

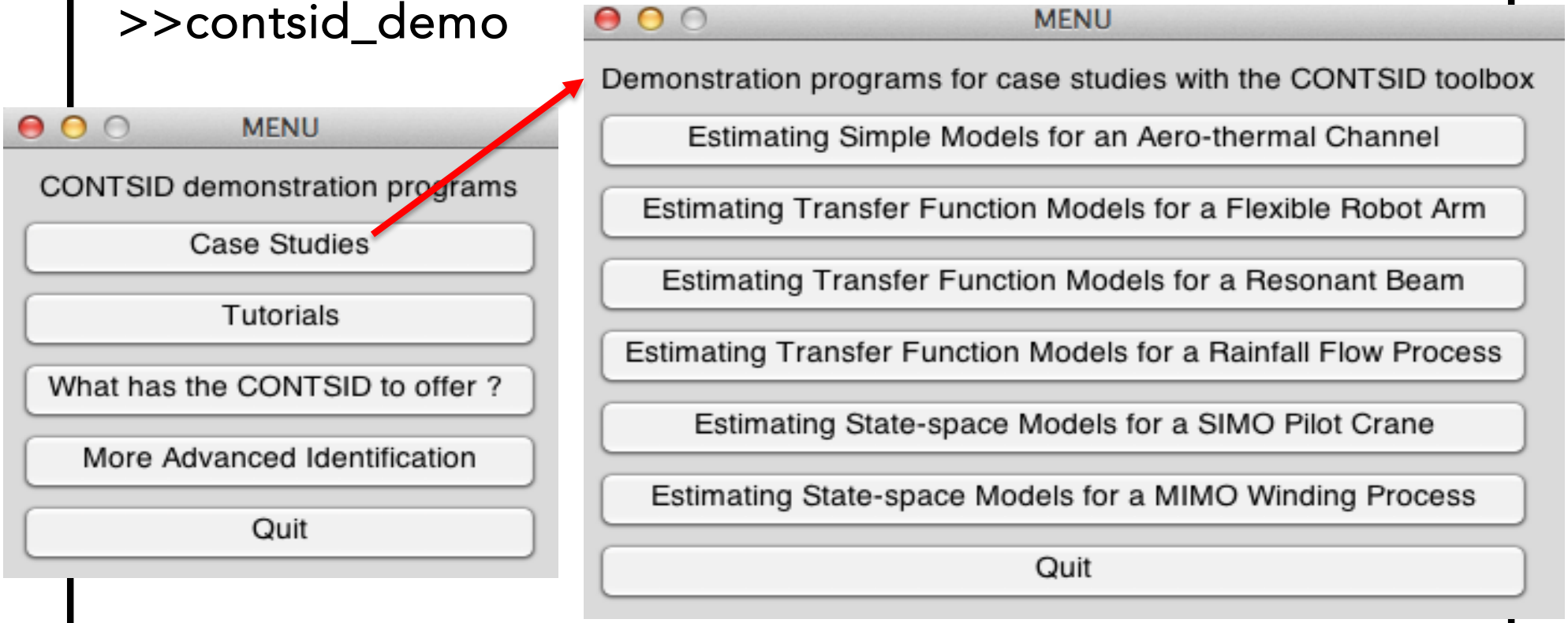
CONTSID graphical user interface



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

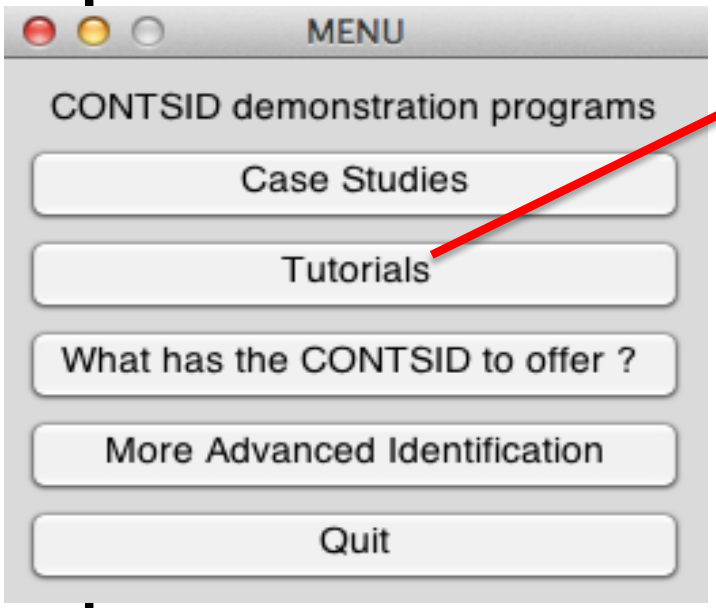
CONTSID toolbox – Demonstration programs

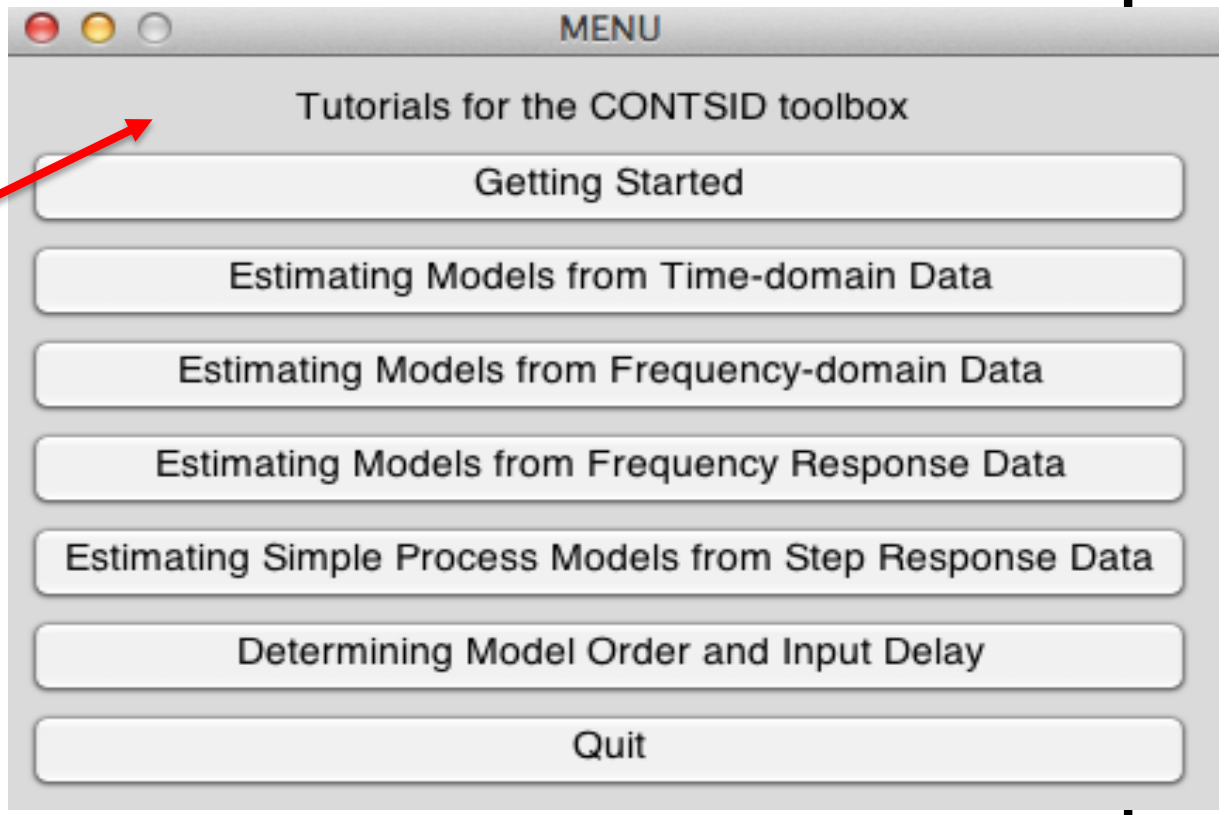
>>contsid_demo



CONTSID toolbox – Demonstration programs

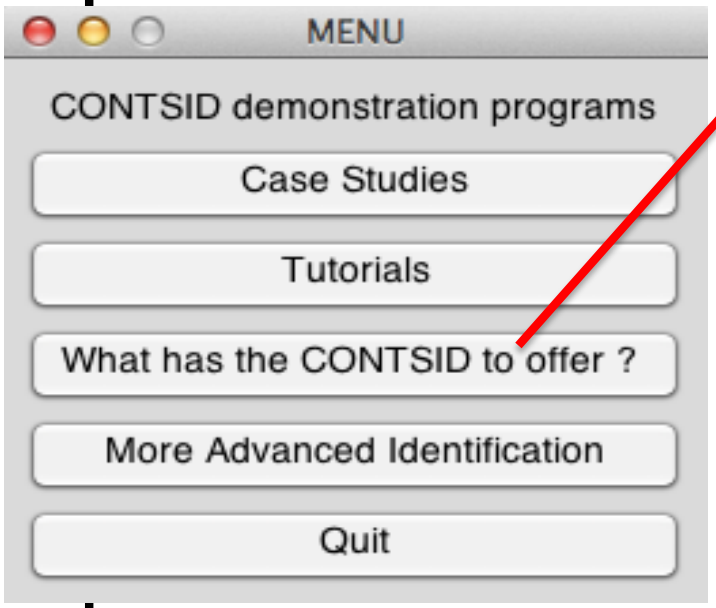
>>contsid_demo

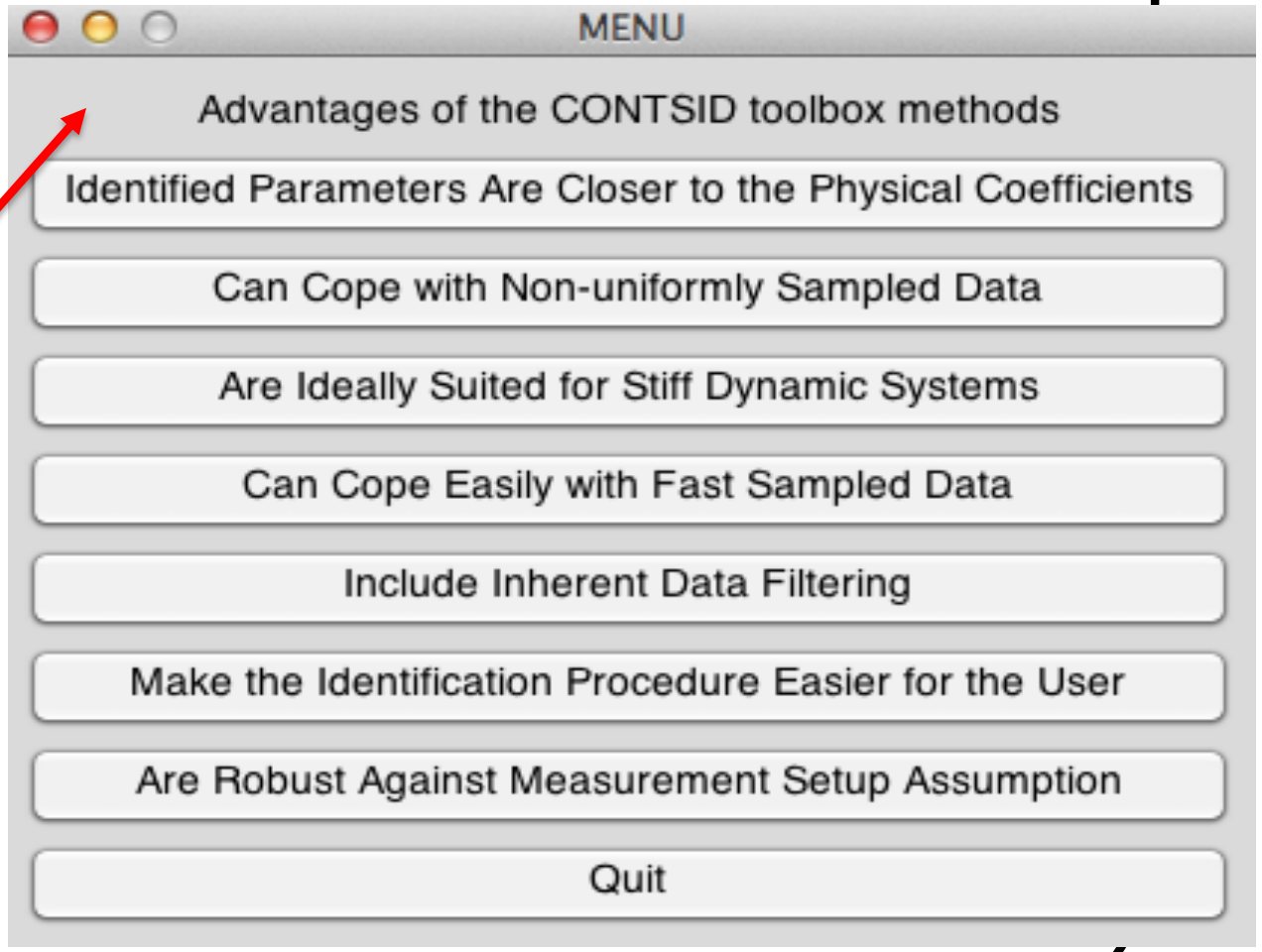




CONTSID toolbox – Demonstration programs

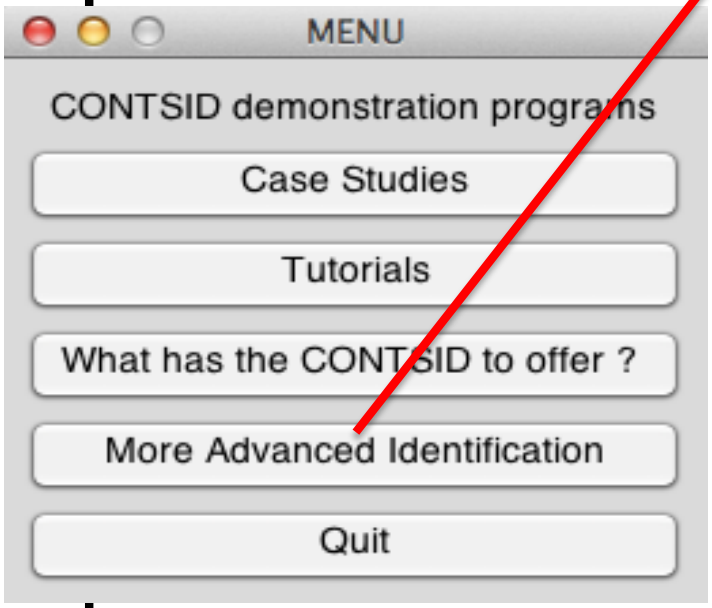
>>contsid_demo

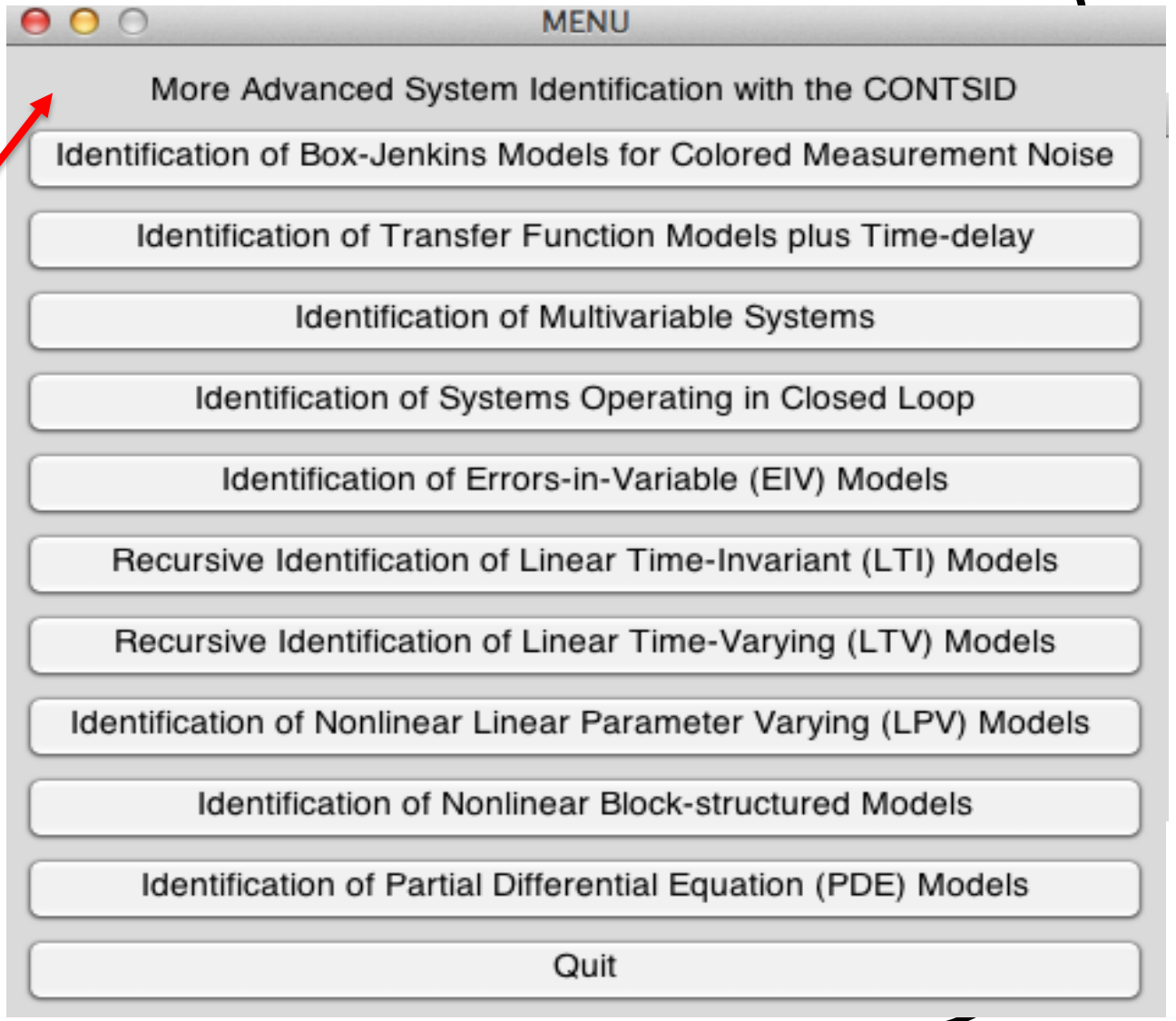




CONTSID toolbox – Demonstration programs

>>contsid_demo





The simulated *Rao-Garnier* benchmark
See *contsid_tutorial1.m*

- 4th order simulated system *p=d/dt*

$$G_o(p) = \frac{K(1-Tp)}{\left(\frac{p^2}{\omega_{n,1}^2} + \frac{2\zeta_1}{\omega_{n,1}} p + 1\right)\left(\frac{p^2}{\omega_{n,2}^2} + \frac{2\zeta_2}{\omega_{n,2}} p + 1\right)}$$

$$= \frac{-6400p + 1600}{p^4 + 5p^3 + 408p^2 + 416p + 1600}$$

$K = 1$

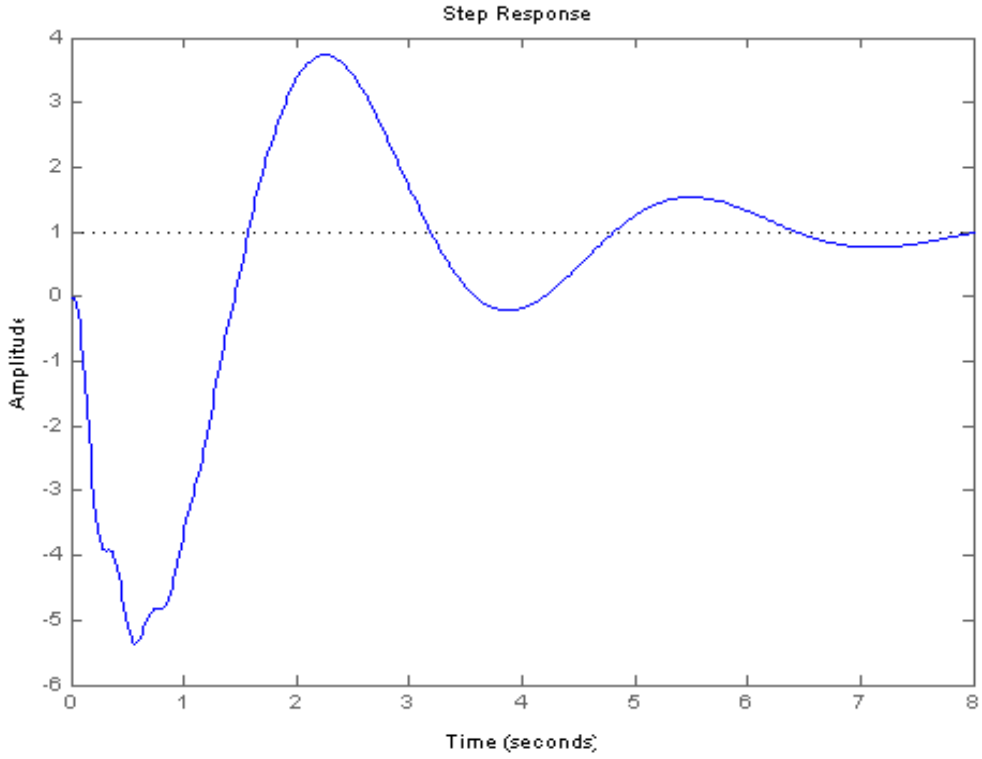
$\omega_{n,1} = 2\ rad/s;\quad \omega_{n,2} = 20\ rad/s$

$\zeta_1 = 0.1;\quad \zeta_2 = 0.25$

- ✓ *1 unstable zero*
- ✓ *2 pseudo-oscillatory modes*

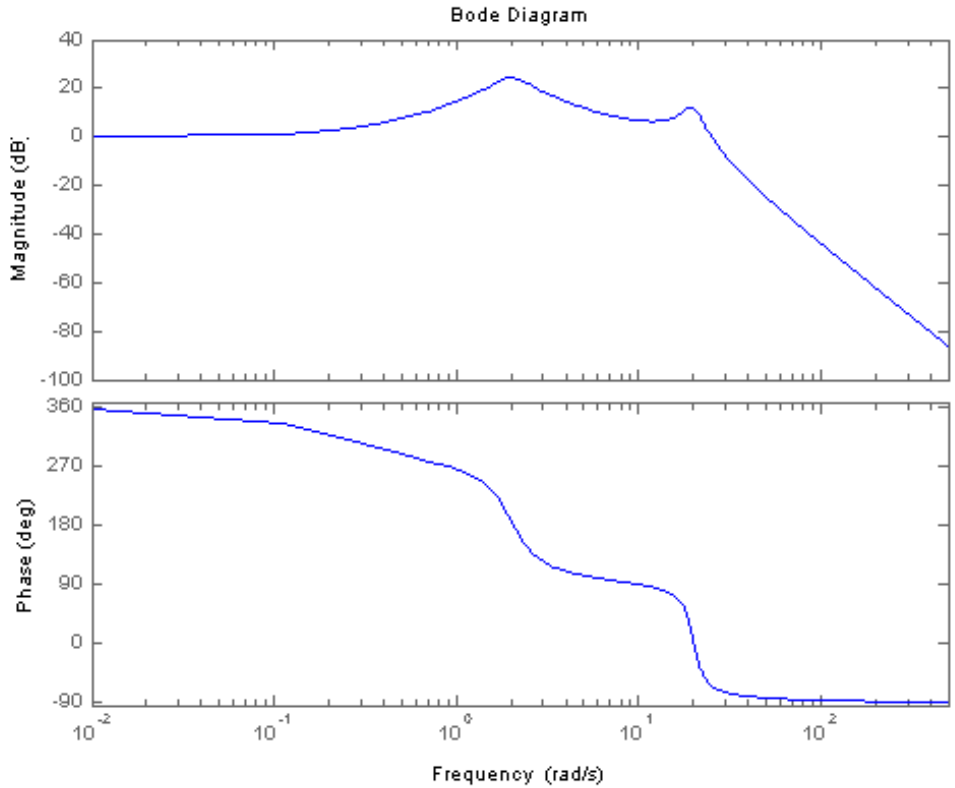
The simulated *Rao-Garnier* benchmark

- The step response

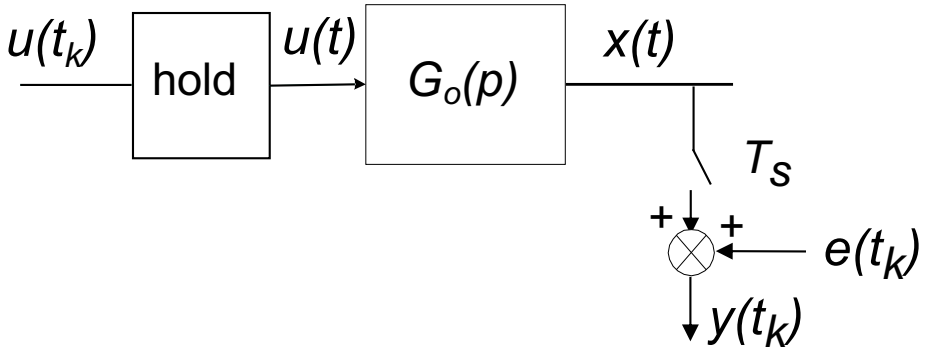


The *Rao-Garnier* benchmark

- The Bode diagram

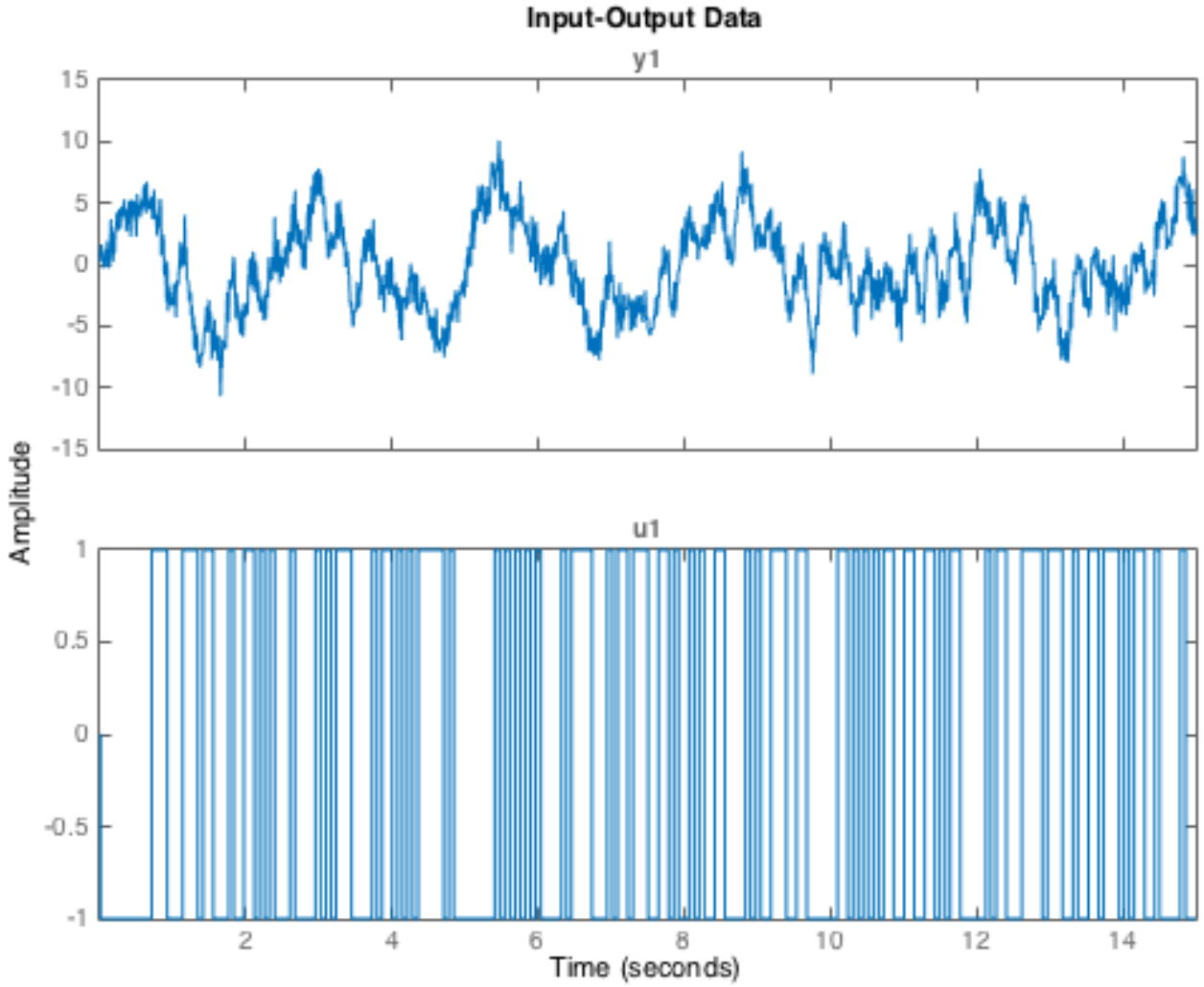


Simulation setup



- $u(t_k)$: PRBS *(respects the ZOH assumption)*
- T_s=10 ms
  - f_s ≈ 10 times the system bandwidth, an often given rule
- $e(t_k)$: DT white noise, SNR=10 dB

I/O data – Rao-Garnier benchmark



Model order determination

Different model structures in the range [m n nk] = [1 3 0] to [2 5 0] have been computed for the given data set

$$G(s) = \frac{\sum_{i=0}^{m} b_i s^i}{\sum_{i=0}^{n} a_i s^i} e^{-n_k T_s s}$$

5 best models sorted according to YIC

np	m	n	nk	RT2	YIC	Niter	FPE	AIC
5	2	3	0	0.83	-8.33	10	2.35	0.85
6	2	4	0	0.92	-8.19	5	1.13	0.12
7	1	5	0	0.53	-7.51	10	6.90	1.93
4	1	3	0	0.51	-3.72	10	8.40	2.12
5	1	4	0	0.03	-0.89	10	13.9	2.63

The second model with [m n nk]= [2 4 0] seems to be quite clear cut

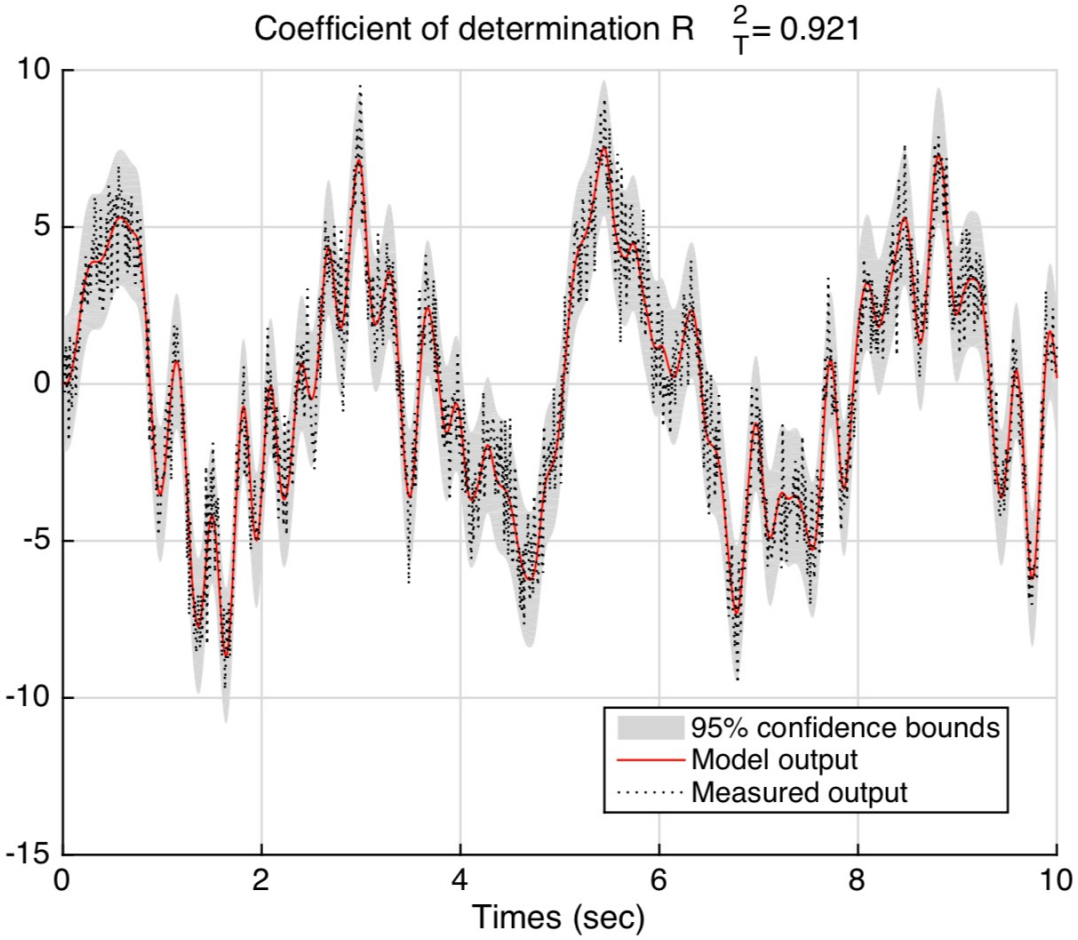
It has the second most negative YIC=−8.19, with the highest R^2_T = 0.92

See srivcstruc and selcstruc in the CONTSID toolbox

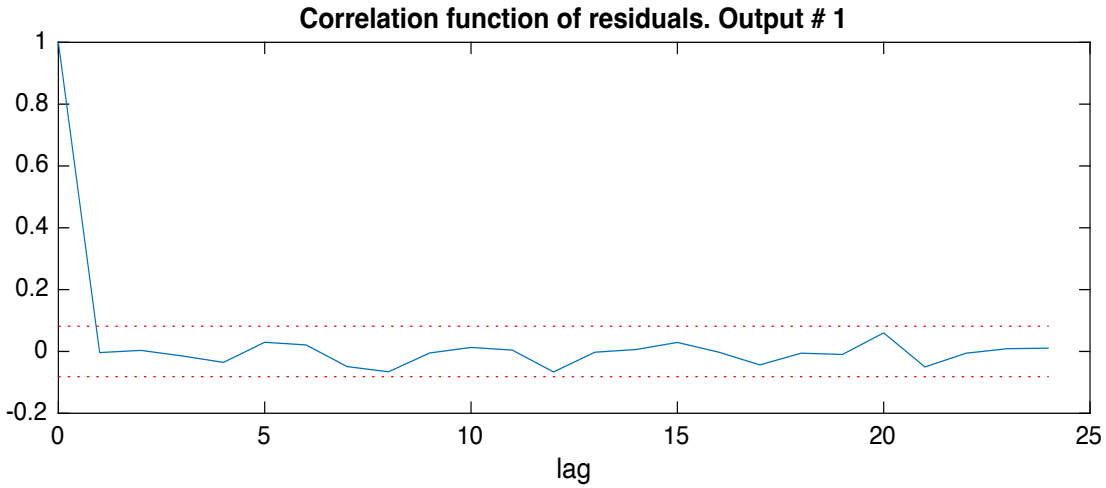
Estimated model parameters and their uncertainties via TFSRIVC

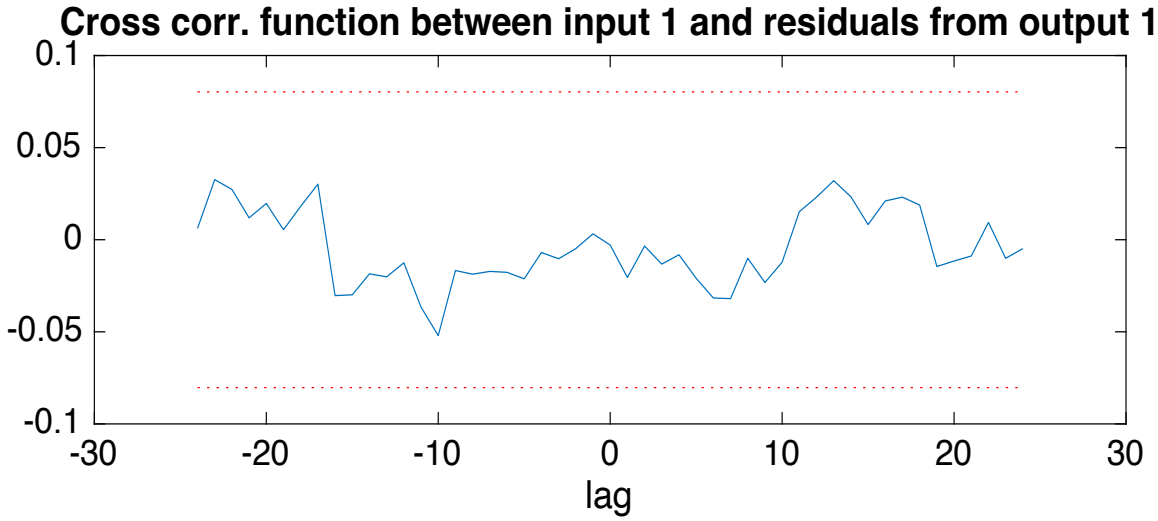
```
>> present(Mtfsrivc)

Mtfsrivc =

  From input "u1" to output "y1":
                            -6480 (+/- 131.8) s + 1880 (+/- 251.4)
  ---------------------------------------------------------------------------------------
  s^4 + 5.382 (+/- 0.2094) s^3 + 407.6 (+/- 3.565) s^2 + 424.2 (+/- 11.53) s + 1566 (+/- 26.24)

Continuous-time identified transfer function.

Parameterization:
   Number of poles: 4   Number of zeros: 1
   Number of free coefficients: 6
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using Contsid TFSRIVC method on time domain data.
Fit to estimation data: 71.91
FPE: 1.140e+00, MSE 1.124e+00
```

Comparison of model and measured output



Statistical tests on residuals





Data-driven linear model identification/learning
Takehome message

- Several choices by the practitioners have to be made and often revised
  - Many of these choices have to be taken with the intended model use in mind and thus have a subjective flavour
  - The more *a priori* knowledge from physics you can exploit in the SYSID workflow, the better
  - Interpretability of the identified models in meaningful physical terms is essential

- Always keep in mind
  - Good models cannot be obtained from bad data !
  - All models are approximation of the real system !
  - Good models are simple !