Modeling and Learning for Dynamical Systems

Lecture 11

Martin Enqvist

LINKÖPING UNIVERSITY

Nonlinear Identification

Nonlinear Identification

Nonlinear system identification is like the zoology of non-elephant animals

- A rich fauna: Many nonlinear model structures have been suggested (and many are even *universal approximators* (able to approximate any (well-behaved) function) when the model order is increased)
- Many estimation frameworks are available
- No general standards (concerning terminology, modeling objectives, software tools, etc.)
- Results and methods have sometimes been rediscovered/reinvented under new names

Here, we will go on a guided tour in this djungle and encounter 7 categories of nonlinear models. . .

Prediction-Error Minimization

The Prediction-Error Minimization (PEM) idea,

$$\hat{\theta}_N = \arg\min_{\theta} V_N(\theta)$$

$$\text{where } V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} \|y(t) - \hat{y}(t|\theta)\|^2$$

can be used also when the model structure is nonlinear.

- We need to be able to derive and compute the predicted output $\hat{y}(t|\theta)$ using the chosen model structure
- Computation of the gradient

$$\psi(t,\theta) = \frac{d}{d\theta}\hat{y}(t|\theta)$$

is often needed/useful for the numerical minimization of $V_N(\theta)$

Nonlinear Grey-Box Models (Category 1)

Consider a first-principles state-space model (with unknown parameters θ and additive white measurement noise):

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), \theta) \\ y(t) &= h(x(t), u(t), \theta) + e(t) \end{aligned}$$

The corresponding output prediction (a model simulation):

$$\begin{aligned} \dot{\hat{x}}(t) &= f(\hat{x}(t), u(t), \theta) \\ \hat{y}(t|\theta) &= h(\hat{x}(t), u(t), \theta) \end{aligned}$$

The PEM can be used to estimate θ. This is known as a **nonlinear grey-box model**.

Nonlinear Grey-Box Models in Matlab

Specify the model structure in a Matlab function (p1, ...pN are parameters)

```
function [dx,y] = myfunc(t,x,u,p1,p2,...,pN,aux)
dx = ...
y = ...
end
```

The model is then defined using the command

```
m0 = idnlgrey('myfunc',[ny nu nx],par);
```

where nx, nu, nx defines the number of outputs, inputs and states and par contains initial guesses for the parameters p1, ...pN (and possibly more information like intervals for each parameter)
The model is then estimated using

```
m = pem(ze,m0);
```

Nonlinear Observers

The presence of process noise and instabilty of the true system require a more general approach to the computation of the predicted output

- In this case, also previous outputs should be used to compute the optimal predictor based on the model
- *Nonlinear observers* are needed in these cases (beyond the scope of this course)

Example: Nonlinear Grey-Box Model

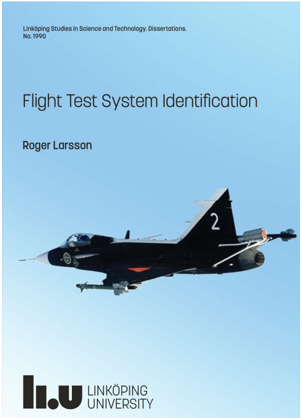
The pitch dynamics of the JAS 39 Gripen fighter jet can be modeled as

$$\dot{x}(t) = \underbrace{\begin{pmatrix} a_{11}x_1(t) + a_{12}x_2(t) \\ f(x_1(t)) + a_{22}x_2(t) \end{pmatrix} + Bu(t)}_{=\tilde{f}(x(t),u(t),\theta)} + w(t)$$

$$y(t) = x(t) + e(t)$$

where x_1 $(= \alpha)$ is the angle of attack, x_2 is the pitch rate, and u_1 and u_2 are the elevator and canard control signals, respectively.

This is an unstable system with process noise where the nonlinear function f is particularly interesting to estimate.



(PhD thesis from 2019)

Example: Nonlinear Grey-Box Model...

Discretization using the standard Euler method gives ($x_k = x(kT_S)$, etc.)

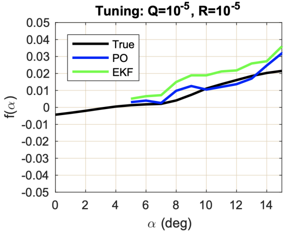
$$x_{k+1} = x_k + T_S \tilde{f}(x_k, u_k, \theta) + \tilde{w}_k$$
$$y_k = x_k + \tilde{e}_k$$

and an approximate nonlinear observer that seems to work well can be defined as

$$\hat{x}_{k+1} = \hat{x}_k + T_S \tilde{f}(\hat{x}_k, u_k, \theta) + K(y_k - \hat{y}_k(\theta))$$
$$\hat{y}_k(\theta) = \hat{x}_k$$

Here, both θ and K are estimated from flight test data where the aircraft did a wind-up turn

Results (Larsson, 2019):



(the described method is denoted PO and EKF is an approach based on an extended (linearized) Kalman filter, present model in black)

Block-Oriented Models (Category 2)

One common approach to nonlinear modeling is to define the model structure as consisting of a few linear dynamical and static nonlinear blocks in series. This is known as **block-oriented modeling**.

- Hammerstein model:

$$y(t) = G(q)z(t), \quad z(t) = f(u(t))$$

  (both G and f contain parameters)

- Wiener model:

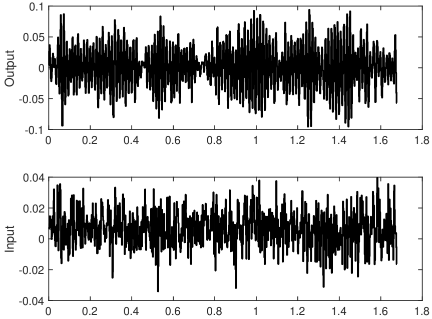
$$y(t) = f(z(t)), \quad z(t) = G(q)u(t)$$

  (both G and f contain parameters)

- Wiener-Hammerstein model: L+NL+L
- Hammerstein-Wiener model: NL+L+NL

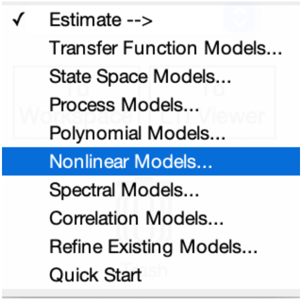
Silverbox Data Example

Consider a particular subset of a benchmark dataset known as the Silverbox data, which has been collected from a nonlinear electric circuit.



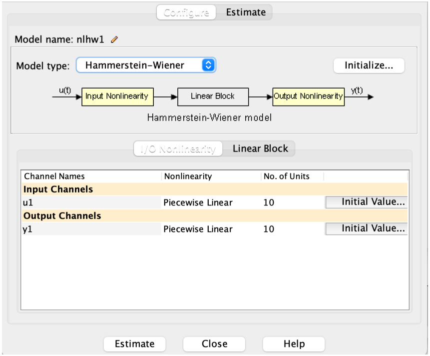
Silverbox Data Example: Block-Oriented Models

Estimate a nonlinear model…



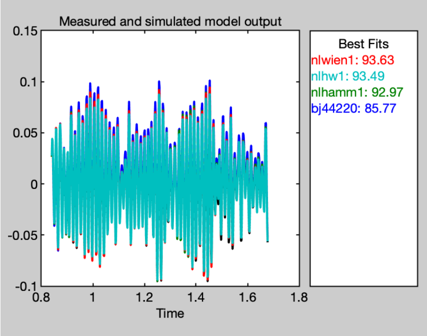
Silverbox Data Example: Block-Oriented Models...

Configure the model structure



Silverbox Data Example: Block-Oriented Models...

In this case, a Wiener model seems to perform best on validation data



Nonlinear Regression Models

A **nonlinear ARX (NARX)** model:

$$\hat{y}(t|\theta) = g(\varphi(t), \theta)$$

where

$$\varphi(t) = f_{\varphi}(y(t-1), \ldots, y(t-n), u(t), \ldots, u(t-m))$$

Common special case:

$$\varphi(t) = \begin{pmatrix} y(t-1) & \ldots & y(t-n) & u(t) & \ldots & u(t-m) \end{pmatrix}$$

Nonlinear FIR (NFIR) models are obtained if φ does not depend on y
We will now look at four types of NARX models: Semi-physical models, local models, local linear models and black-box NARX models

Semi-Physical Models (Category 3)

A **semi-physical model** is a model where the measured data have been transformed in a nonlinear way such that a linear model is enough to describe the transformed input-output data.

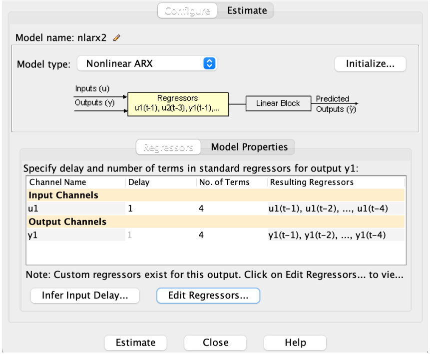
- This transformation can be based on intuition, fundamental laws of nature, high-school physics, etc.
- Example: Water is heated with an immersion heater: A linear model from voltage to temperature will not be accurate, but a linear model from the squared voltage (which is proportional to the power) to temperature will work fine
- Sometimes, the nonlinear transformation is introduced by using nonlinear regressors in a linear (in the parameters) regression model: $\hat{y}(t|\theta) = \varphi(t)^T\theta$ with

$$\varphi(t) = f_\varphi(y(t-1), \ldots, y(t-n), u(t), \ldots, u(t-m))$$

(Here, θ can be estimated using the least-squares method.)

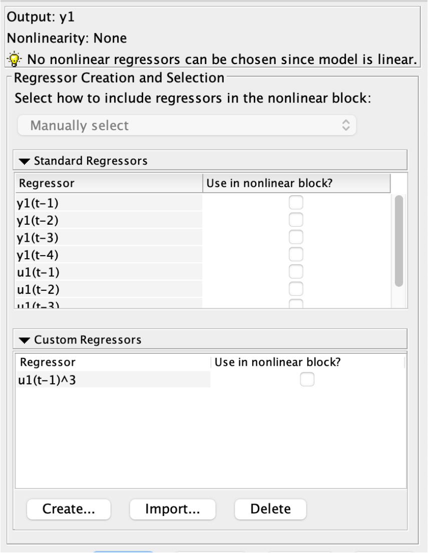
Silverbox Data Example: Semi-Physical Models

A linear model with nonlinear regressors can be estimated by selecting a NARX model, turning off the nonlinear part and selecting `Edit regressors`



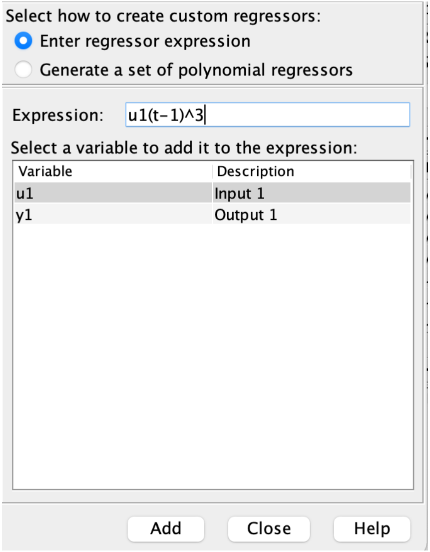
Silverbox Data Example: Semi-Physical Models. . .

Custom regressors can then be added. . .



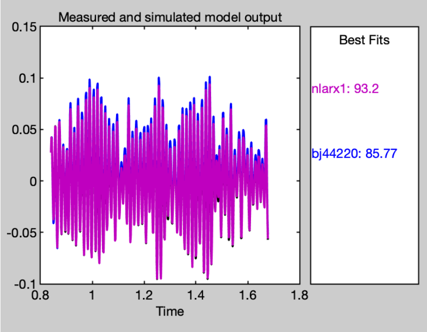
Silverbox Data Example: Semi-Physical Models. . .

. . . in a straightforward way



Silverbox Data Example: Semi-Physical Models...

In this case, the result is rather good



Local Models (Category 4)

A **local model** is obtained by approximating the output at a point φ^* as a weighted average of the available output observations:

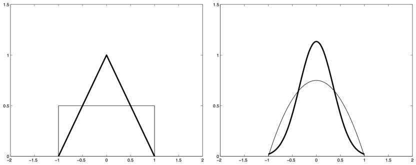
$$\hat{y} = \hat{g}(\varphi^*) = \sum_{k=1}^{N} w_k y(k)$$

Here, w_k are normalized (sum = 1) weights.

- The nearest-neighbor model is one (extreme) example
- It is often useful to include more averaging by using weights on the form

$$w_k(\varphi^*, \varphi(k)) = \mathcal{N} \cdot \kappa(\alpha_k \|\varphi^* - \varphi(k)\|)$$

  where $\mathcal{N}$ is a normalization factor and α_k are parameters
- Some examples of κ-functions:



Local Models (Category 4)...

- $\frac{1}{\alpha_k}$ acts as a bandwidth:
  - Small α_k (large bandwidth): Many w_k will be nonzero $\Rightarrow$ small variance, large bias
  - Large α_k (small bandwidth): Few w_k will be nonzero $\Rightarrow$ large variance, small bias
- Local models are used frequently in statistics

Local Linear Models (Category 5)

A **local linear model** is a combination of several linear models using a measureable scheduling variable ρ that defines the operating point of the system.

- A number of representative values are chosen for ρ: ρ_k, $k = 1, 2, \ldots, d$
- At each point ρ_k, an accurate linear model is found, which gives $\hat{y}^{(k)}(t)$. For example, these models can be ARX models

$$\hat{y}^{(k)}(t|\theta^{(k)}) = \varphi(t)^T \theta^{(k)}$$

- The total model can be written

$$\hat{y}(t) = \sum_{k=1}^{d} w_k(\rho(t), \rho_k)\hat{y}^{(k)}(t)$$

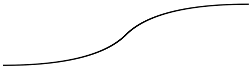
- For example, such models are used frequently in aerospace applications (where velocity and altitude can be used as scheduling variables)

A General Nonlinear Black-Box Model

One general class of nonlinear black-box models can be written

$$\hat{y}(t|\theta) = g(\varphi(t), \theta) = \sum_{k=1}^{d} \gamma_k \kappa(\alpha_k(\varphi(t) - \beta_k))$$

Functions of *sigmoid* type are common choices of the activation (basis) function κ:



One example is the *logistic* function

$$\kappa(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

Another common choice is the *rectified linear unit (ReLU)*

$$\kappa(x) = x_+ = \max(x, 0)$$

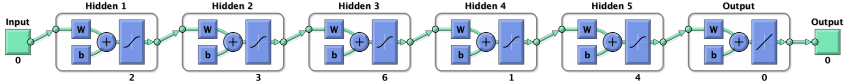
These models are building blocks in **neural networks**

Neural Networks (Category 6)

- A neural network is obtained by combining many neurons in parallel (one hidden layer) or in cascade (several hidden layers, deep networks)
- The term *hidden* refers to the fact that only the weighted sum of the neuron outputs are available, not their individual outputs
- The result is a nonlinear function from $\mathbb{R}^n$ (n inputs) to $\mathbb{R}^m$ (m outputs)
- A continuous function can be approximated arbitrarily well by a neural network (true also for polynomials and many other model structures)
- Neural networks have a risk for overfitting due to their flexibility. One remedy for this is *early stopping* (restricting the maximum number of iterations in the iterative search for optimal parameters)
- Neural networks with one hidden layer are available in the System Identification Toolbox in Matlab

Feedforward Nets

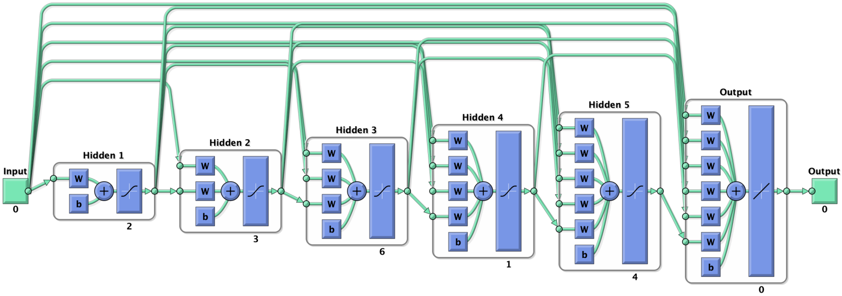
Feedforward nets are obtained if the outputs of the units (neurons) in a previous hidden layer are processed further in another layer. This can be repeated many times (deep networks).



Feedforward nets are available in the Deep Learning Toolbox in Matlab and can be used together with the System Identification Toolbox to facilitate comparisons with other model types

Cascade Forward Nets

Cascade forward nets are obtained if the outputs of the units (neurons) in all earlier hidden layers are processed further in another layer. This can be repeated many times (deep networks).



Cascade forward nets are available in the Deep Learning Toolbox in Matlab and can be used together with the System Identification Toolbox to facilitate comparisons with other model types

Silverbox Data Example: Neural Networks

Consider the Silverbox dataset again, and let us try some neural network models:

`mNN1ff` (feedforward net, one layer with 12 units):

```
net1ff=feedforwardnet([12]);
NN1ff=neuralnet(net1ff);
mNN1ff=nlarx(edat,[4 4 0],NN1ff);
```

`mNN4ff` (feedforward net, 4 layers with 3 units each):

```
net4ff=feedforwardnet([3,3,3,3]);
NN4ff=neuralnet(net4ff);
mNN4ff=nlarx(edat,[4 4 0],NN4ff);
```

Silverbox Data Example: Neural Networks...

`mNN1cf` (cascade forward net, one layer with 12 units):

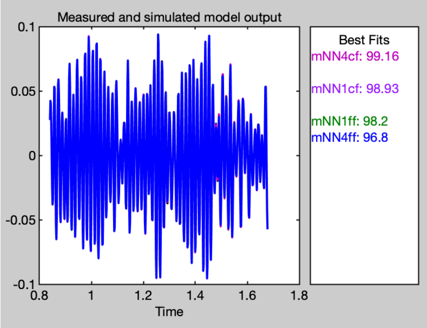
```
net1cf=cascadeforwardnet([12]);
NN1cf=neuralnet(net1cf);
mNN1cf=nlarx(edat,[4 4 0],NN1cf);
```

`mNN4cf` (cascade forward net, 4 layers with 3 units each):

```
net4cf=cascadeforwardnet([3,3,3,3]);
NN4cf=neuralnet(net4cf);
mNN4cf=nlarx(edat,[4 4 0],NN4cf);
```

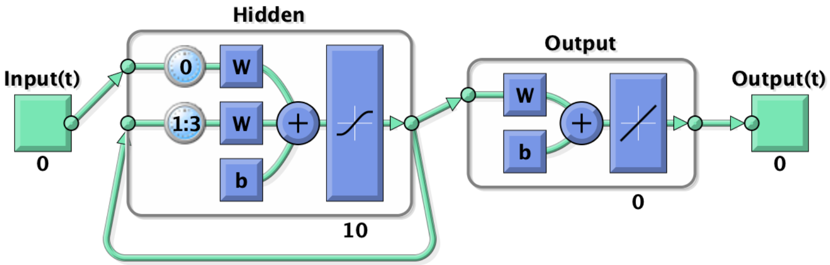
Silverbox Data Example: Neural Networks...

All four neural network models give high model fits for validation data, and the `mNN4cf` model achieves an impressive 99.16%



Recurrent Neural Nets (RNNs) (Category 7)

The neural networks described so far have been of NARX type, where the resulting model is a static function of the regression vector φ. A **recursive neural net (RNN)** contains internal feedback paths where computed variables are fed back and used as to define the output prediction in the next step.



Recurrent Neural Nets (RNNs) (Category 7)...

- Feeding back the output results in a nonlinear output-error (NOE) model structure:

$$\hat{y}(t|\theta) = g(\hat{y}(t-1|\theta), \ldots, \hat{y}(t-n|\theta), u(t), \ldots, u(t-m))$$

- In general, an RNN has a nonlinear state-space structure
- RNN structures provide more flexibility, but might pose problems due to instability, poor numerical properties, etc.
- RNNs are available in the Deep Learning Toolbox in Matlab

Summary

- Nonlinear grey-box models
- Block-oriented models (Hammerstein, Wiener, etc.)
- Semi-physical models
- Local models
- Local linear models
- Neural networks
- Recurrent neural networks

www.liu.se

LINKÖPING UNIVERSITY