1.1 Introduction to System Identification

System identification is the task of inferring a mathematical description, a model, of a dynamical system from a series of measurements on the system.

There can be several reasons for establishing mathematical descriptions of dynamical systems. Typical applications encompass simulation, prediction, fault detection and control system design. If the burden associated with building a model using laws of physics, chemistry, economics, etc., is overwhelming, system identification techniques are naturally of particular interest.

In this lesson the attention is drawn to identification of neural networks for nonlinear dynamical systems, which results in higher computation burden compared to identification of linear systems. However many of the results known from conventional system identification apply as well to neural-network-based identification.

Depending on the level of a priori insight into the system, the identification problem can be approached in different ways:

1. If the identification is based exclusively on measured data, assuming no or only little knowledge about the physics of the system, the identification process is called black-box modelling.
2. If the modelling is based totally on pure physical laws, the identification process is called white-box modelling.
3. When a certain level of insight (or a priori knowledge) about the system exists and is used to improve the empirical modelling the identification process is called gray-box modelling.

The lesson deals mainly with black-box modelling, however a fundamental understanding of the system behaviour is always useful as it facilitates the identification process. Such insight can include:

- Order of the system
- Slow or fast dynamics
- Adequate sampling frequency
- Stability properties
- Operating range
- Time delay
- Degree of nonlinearity (is it almost linear?)
- Basic characteristics of nonlinearities (hard/smooth)

Although all systems are in principle nonlinear, most of the literature on system identification deals with identification of linear systems. There are many reasons for this, e.g.:
1. Many systems can be described as linear or piecewise linear, that is different linear models are employed for different regions of the operating range
2. From a computational standpoint it is less complicated to perform
3. The analysis is less complicated from a statistical perspective
4. It is much simpler to design controllers for a system described by a linear model.

1.2 The Procedure

When attempting to identify a model of a dynamical system, it is common practice to follow the procedure depicted in fig. 1.1.

![Diagram](image)

**Fig. 1.1 The basic system identification procedure**

**EXPERIMENT**
The purpose of the experiment is to collect a set of data that describes how the system behaves over its entire range of operation. The idea is to vary the inputs $u$ and observe the impact on the outputs $y$ (open-loop experiment). It is assumed that experimental data describing the underlying system in its entire operating region have been obtained beforehand with a proper choice of sampling frequency:
\( \{u(t)\} \) is no longer just a set of inputs but it is now a signal, the control signal. Likewise \( \{y(t)\} \) now represents the measured output signal. ‘\( t \)’ specifies sampling instant number \( t \). If the system under consideration has more than one input and/or output, \( u(t) \) and \( y(t) \) are simply vectors.

If the system to be identified is unstable or contains lightly damped dynamics, it may be necessary to conduct the experiment in closed-loop, e.g. by introducing a manually tuned, stabilizing feedback controller.

Some of the main issues in the experiment stage are:

- Choice of the sampling frequency
- Design of a suitable input signal
- Pre-processing of the data (nonlinearity tests, removal of disturbances or noise)

**MODEL STRUCTURE SELECTION**

Assuming that a data set has been acquired, the next step is to select a model structure. Unfortunately, this issue is much more difficult in the nonlinear case than in the linear case. Not only is it necessary to choose a set of regressors but also a network architecture is required. The approach used here is to select the regressors based on inspiration from linear system identification and then determine the best possible network architecture with the given regressors as inputs.

A model structure is a set of candidate models, that is a set inside which one should search for a model. In general the problem of selecting a model is twofold:

- Selection of a “family” of model structures considered appropriate for describing the systems, e.g. linear model structures, or in the case of nonlinear modelling, choice of the neural network (multi-layer perceptron, radial basis function, etc)
- Inside the family chosen above, selection of a subset. For instance, in the family of linear structures choice of an ARX \((2,3,1)\) where \((2,3,1)\) signifies a time delay of one sampling period and that the present output depends on two past measured outputs and three past measured inputs

Both input-output models and state space models can be considered. Here attention will be paid to the former and particularly to systems that have only one output, whose considerations can be easily extended to multi-output systems. See next section for a thorough explanation of this issue.

**ESTIMATE MODEL**

Once a set of candidate models is chosen, the next step is to pick one particular model from this set. One will typically choose the model that performs best according to some type of criterion. The most common strategy is to pick the model that provides the best one-step ahead predictions in terms of the smallest expected squared error between observed outputs and predictions. The process of picking a model from the model structure is the statistical literature known as estimation, while in the neural network community is usually called training or learning.
VALIDATION
After that a model has been estimated/trained it must be evaluated to investigate whether or not it meets the necessary requirements; The validation is closely connected to the intended use of the model and it coincides with the validation procedure already explained in neural networks: verify that unknown input data are well approximated.

GOING BACKWARDS IN THE PROCEDURE
The paths going from the validation block to the previous stages indicate that the procedure is iterative, i.e. it is necessary to go back to determine a number of different models to try out various model structures or even, in the worst case, to redo the experiment.

• Path leading back to model estimation: sometimes it is hard to guarantee that the training algorithm will converge to the best model in the model structure according to a simple criterion. The problem is that a criterion can have several local minima and finding the global minimum is not easy. Sometimes it can be necessary to augment the criterion by its regularization.
• Path leading back to model structure selection: sometimes it is necessary to modify the structure of the selected model, e.g. by modifying the number of the hidden neurons (pruning): an initial model structure that is large enough as for hidden neurons is chosen and then it is reduced gradually until the optimal structure is achieved
• Path leading back to experiment: if it seems impossible to determine a decent model regardless of how the model structure is selected, this may indicate that the data are not enough, either because additional processing, as filtering, is necessary, or because there is simply a lack of information in the data set. This last issue implies that certain operation ranges are not reflected in the data set, so it is necessary that an additional experiment be made to acquire more information about the missing operation ranges or regimes.

1.3 Model Structure Selection

The purpose of this section is to introduce model structures for identification of nonlinear dynamical systems. First a brief review of linear model structures will be given, as a basis for developing neural-network-based model structures, seen as generalization of the linear model structure.

1.3.1 Linear Model Structures

A dynamical system is called linear if it is possible to describe it by the model

\[ y(t) = G(q^{-1})u(t) + H(q^{-1})e(t) \] (1.1)

Where

• \( G \) and \( H \) are transfer functions
• \( e(t) \) is a white noise signal independent of past inputs, in general characterized by some probability density functions
• \( q^{-1} \) is the time delay operator, which works on the signal as follows:

\[ q^{-d}x(t) = x(t - d) \] (2.2)

where \( d \) is some multiple of the sampling period
If these variables are considered as vectors, (1.1) comprises also the MIMO case.

If the system is linear the goal is to determine good estimates of $G$ and $H$. The criterion defining the meaning of “good” estimate primarily relates to the model ability to produce ONE-STEP ahead predictions with errors of LOW variance.

For the general linear system (2.1) it can be proved that the minimum variance (one-step ahead) prediction is given by:

$$\hat{y}(t | t-1) = H^{-1}(q^{-1})G(q^{-1})u(t) + [1 - H^{-1}(q^{-1})]y(t)$$  \hspace{1cm} (1.3)

called also “predictor form of the model”.

Now some definitions follow:

**True system**
The true system $S$ is assumed to be described by:

$$y(t) = G_o(q^{-1})u(t) + H_o(q^{-1})e_o(t)$$  \hspace{1cm} (1.4)

where $e_o(t)$ is a white noise signal independent of the input signal $u(t)$

**Model Structure**
The model structure $M$ is a parametrized set of candidate models

$$M : \big\{ G(q^{-1}, \theta), H(q^{-1}, \theta) \big| \theta \in D_m \big\}$$

$$y(t) = G(q^{-1}, \theta)u(t) + H(q^{-1}, \theta)e(t)$$  \hspace{1cm} (1.5)

where $\theta$ denotes the $p$ adjustable parameters, $D_m$ is some subset of $R^p$ inside which the search of the parameters is to be carried out (search space). The predictor form of (1.5) is given by

$$\hat{y}(t | t-1, \theta) = H(q^{-1}, \theta)G(q^{-1}, \theta)u(t) + [1 - H^{-1}(q^{-1}, \theta)]y(t)$$  \hspace{1cm} (1.6)

Here the $\theta$ introduced a san argument specifies that the model structure represents a set of models. As only one-step ahead predictors are considered here the $t-1$ symbol will be left off. The predictor form (1.6) is often written in alternative form as

$$\hat{y}(t | \theta) = \phi^T(t)\theta$$  \hspace{1cm} (1.7)

Where $\theta$ is the parameter vector and $\phi(t)$ is the regression vector, which contains past inputs, past outputs, or signals derived from past inputs and outputs.

It can be proved that the basic requirement for the search space $D_m$ is
The stability condition obviously implies that the predictor remains stable. The assumption $G(0, \theta) = 0$ is included to ensure that the predictions depend only on past inputs (since the constant term of the polynomial is null), while the condition $H(0, \theta) = 1$ ensures that the predictions depend only on past output measurements.

A basic requirement is that the true system can be represented by the model structure, that is that the model structure is “large enough” to describe it. This means that:

$$S \in M$$

(1.9)

A **MODEL** is simply a particular choice of parameter vector, i.e. $\theta = \hat{\theta}$

### 1.3.2 Simplification of the Linear Model Structures

The general form (2.5) can be rewritten in the following form, considering that $G$ and $H$ are ratios of polynomials. From this form some important simplifications can be made according to the different assumptions that can be made about the spectral density of the noise and how the noise can be accounted for in the system.

$$A(q^{-1})y(t) = q^{-d} \frac{B(q^{-1})}{F(q^{-1})} u(t) + \frac{C(q^{-1})}{D(q^{-1})} e(t)$$

(1.10)

where

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_n q^{-n}$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \ldots + b_m q^{-m}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + \ldots + c_k q^{-k}$$

$$D(q^{-1}) = 1 + d_1 q^{-1} + \ldots + d_l q^{-l}$$

$$F(q^{-1}) = 1 + f_1 q^{-1} + \ldots + f_r q^{-r}$$

(1.11)

Remark that the polynomials $A, C, D, F$ are monic, while $B$ is not.

According to the different assumptions on these polynomials different model structures can be envisaged.

### FIR (FINITE IMPULSE RESPONSE) model structure

In this case $C=D=1, A=1, F=1$, i.e.

$$G(q^{-1}, \theta) = q^{-d} B(q^{-1}) \text{ and } H(q^{-1}, \theta) = 1$$

(1.12)

and the predictor is given by
\hat{y}(t | \theta) = q^{-d} B(q^{-1}) u(t) \quad (1.13)

This can be then rewritten as

\hat{y}(t | \theta) = \varphi^T(t) \theta \quad (1.14)

and in this case the regression vector is given by:

\varphi(t) = [u(t - d), \ldots, u(t - d - m)]^T \quad (1.15)

And the parameter vector is

\theta = [b_0, \ldots, b_m]^T \quad (1.16)

Note that FIR models have no poles, this means that a system with poles cannot be described exactly by it. However if the system is stable, and the impulse response decays quite fast, the system can be approximated by a FIR model if the \( B(q^{-1}) \) is chosen as the first \( m \) coefficients of the impulse response.

**ARX (AutoRegressive, eXternal input) model structure**

In this case \( C=D=F=1 \), i.e.,

\[ G(q^{-1}, \theta) = q^{-d} \frac{B(q^{-1})}{A(q^{-1})} \quad \text{and} \quad H(q^{-1}, \theta) = \frac{1}{A(q^{-1})} \quad (1.17) \]

and the predictor is given by

\[ \hat{y}(t | \theta) = q^{-d} B(q^{-1}) u(t) + [1 - A(q^{-1})] y(t) = \varphi^T(t) \theta \quad (1.18) \]

with:

\[ \varphi(t) = [y(t-1), \ldots, y(t-n), u(t-d), \ldots, u(t-d-m)]^T \quad (1.19) \]

\[ \theta = [-a_1, \ldots, -a_n, b_0, \ldots, b_m]^T \quad (1.20) \]

In this case \( G \) has poles, but there is only an algebraic relationship between the prediction and the past inputs and measured outputs. Consequently the predictor **will always be stable**, even if the
system is unstable. This is the main feature of the ARX model structure. Sometimes this model is also called *series-parallel model*.

**ARMAX (AutoRegressive, Moving Average, eXternal input) model structure**

In this case $D=F=1$, i.e.,

$$G(q^{-1}, \theta) = q^{-d} \frac{B(q^{-1})}{A(q^{-1})} \quad \text{and} \quad H(q^{-1}, \theta) = \frac{C(q^{-1})}{A(q^{-1})} \quad (1.20)$$

and the predictor is given by

$$\hat{y}(t | \theta) = q^{-d} \frac{B(q^{-1})}{C(q^{-1})} u(t) + [1 - \frac{A(q^{-1})}{C(q^{-1})}] y(t) \rightarrow$$

$$C(q^{-1}) \hat{y}(t | \theta) = q^{-d} B(q^{-1}) u(t) + [C(q^{-1}) - A(q^{-1})] y(t) + \hat{y}(t | \theta) - \hat{y}(t | \theta) \rightarrow \quad (1.21)$$

$$\hat{y}(t | \theta) = q^{-d} B(q^{-1}) u(t) + [C(q^{-1}) - A(q^{-1})] y(t) - C(q^{-1}) \hat{y}(t | \theta) + y - \varepsilon(t, \theta) \rightarrow$$

$$\hat{y}(t | \theta) = q^{-d} B(q^{-1}) u(t) + [1 - A(q^{-1})] y(t) + [C(q^{-1}) - 1] \varepsilon(t, \theta) = \varphi^T(t) \theta$$

where $\varepsilon(t, \theta) = y - \hat{y}(t | \theta)$ represents the prediction error or residual. The regression vector and the parameter vectors are defined therefore by:

$$\varphi(t) = [y(t-1),...,y(t-n),u(t-d),...,u(t-d-m),\varepsilon(t-1,\theta),...\varepsilon(t-k,\theta)]^T \quad (1.22a)$$

$$\theta = [-a_1,...,-a_n,b_0,...,b_m,c_1,...,c_k]^T \quad (1.22b)$$

Because of the polynomial $C$, the predictor now has poles, this means that the roots of $C$ must lie inside the unit circle for the predictor to be stable. Also the poles imply that the regression vector depend on the model parameter, which makes the estimation more complicated. This model dependency is well shown by the presence of $\theta$ in the regression vector.

**OE (Output Error) model structure**

The output error (also called *parallel model*) model structure is used if the only noise affecting the system is white measurement noise

$$y(t) = q^{-d} \frac{B(q^{-1})}{F(q^{-1})} u(t) + e(t) \quad (1.23)$$

In this case $C=D=0$ and $A=1$, i.e.

$$G(q^{-1}, \theta) = q^{-d} \frac{B(q^{-1})}{F(q^{-1})} \quad \text{and} \quad H(q^{-1}, \theta) = 1 \quad (1.24)$$
and the predictor is given by
\[
\hat{y}(t | \theta) = q^{-d} \frac{B(q^{-1})}{F(q^{-1})} u(t) \rightarrow 
\]
\[
F(q^{-1}) \hat{y}(t | \theta) = q^{-d} B(q^{-1}) u(t) + \hat{y}(t | \theta) - \hat{y}(t | \theta) \rightarrow 
\]
\[
\hat{y}(t | \theta) = q^{-d} B(q^{-1}) u(t) - F(q^{-1}) \hat{y}(t | \theta) + \hat{y}(t | \theta) \rightarrow 
\]
\[
\hat{y}(t | \theta) = q^{-d} B(q^{-1}) u(t) + [1 - F(q^{-1})] \hat{y}(t | \theta) = \varphi^T(t) \theta 
\]

where \( e(t, \theta) = y - \hat{y}(t | \theta) \) represents the prediction error or residual. The regression vector and the parameter vectors are defined therefore by:

\[
\varphi(t) = [\hat{y}(t - \| \theta), \ldots, \hat{y}(t - r | \theta), u(t - d), \ldots, u(t - d - m)]^T 
\]

\[
\theta = [-f_1, \ldots, -f_r, b_0, \ldots, b_m]^T 
\]

Because of the polynomial \( F \) the predictor now has poles, this means that the roots of \( F \) must lie inside the unit circle for the predictor to be stable.

### 1.3.3 Nonlinear Linear Model Structures based on Neural Networks

When dealing with nonlinear dynamical systems, the problem of selecting the model structure becomes increasingly difficult. In any case, since multi-layer perceptrons (MLP) are suitable for learning nonlinear relationships from a set of data, it will be the neural network chosen in the following for nonlinear system identification. By making this choice, the model structure selection is basically reduced to the following two issues:

- Choice of the inputs to the neural network
- Choice of the topology of the neural network (number of hidden neurons or layers)

The approach is to use the input structure of the above linear models and to employ the MLP as the neural network. This results in the following advantages:

- Natural extension of the linear model structure
- The internal structure of the neural network (number of hidden neurons or layers) can be easily and gradually expanded to model more complex nonlinear relationships
- The structural decisions required by the user are reduced
- It is suitable for the design of control systems

Also here some definitions holds as in the linear counterpart

The true system \( S \) is assumed to be described by

\[
y(t) = g_0 [\varphi(t, \theta), \theta] + e_0(t) 
\]

Where \( g_0 \) is a nonlinear function and \( e_0(t) \) is the noise defined earlier. This true system from which the data are obtained is called “a teacher”.
The model structure $M$ is again defined as a parametrized set of candidate models

$$M : \{g[\varphi(t, \theta), \theta] \mid \theta \in D_m \}$$

$$y(t) = g[\varphi(t, \theta), \theta] + e(t) \tag{1.25}$$

Where the symbols are the same as in the linear case. A particular model is called “a student”. The predictor form is then derived as

$$\hat{y}(t \mid \theta) = g[\varphi(t, \theta), \theta] \tag{1.26}$$

In the above formulae $\varphi(t, \theta)$ is obviously the regression vector while $\theta$ is the vector containing the adjustable parameters in the neural network, i.e. the weights. $G$ is the nonlinear function realized by the neural network and it is assumed to have a feedforward structure.

As in the earlier case the basic requirement is that the true system can be represented by the model structure, that is that the model structure is “large enough” to describe it. This means that the teacher lies in the student space, i.e.

$$S \in M$$

Depending on the choice of the of the regression vector, different nonlinear model structures are created: if for example the regression vector is selected as for ARX models the model structure is called NNARX as the acronym for Neural Network ARX, and likewise NNFIR, NNARMAX, NNOE are introduced.

**NNFIR**

In this case the vector regression is:

$$\varphi(t) = [u(t - d),...,u(t - d - m)]^T$$

The predictor is always stable because a pure algebraic relationship exists between predictions and past inputs. See fig. 1.2

![NNFIR model structure](image)
**NNARX**
In this case the vector regression is:

\[ \varphi(t) = [y(t-1),...,y(t-n),u(t-d),...,u(t-d-m)]^T \]

As in the NNFIR, the predictor is always stable because a pure algebraic relationship exists between predictions and past inputs and outputs. These outputs are those measured from the real system. This is the so-called “prediction mode”. See fig. 1.3 for NNARX.

![Fig. 1.3: The NNARX model structure](image)

The NNARX structure is the preferred choice when the system is deterministic or the level of noise is insignificant.

**NNARMAX**
In this case the vector regression is:

\[ \varphi(t) = [y(t-1),...,y(t-n),u(t-d),...,u(t-d-m), \varepsilon(t-1, \theta),...,\varepsilon(t-k, \theta)]^T \]

Although the function \( g \) in (1.25) is realized by a feedforward network, the predictor will have a feedback due to the residual. See Fig. 1.4

The past residuals depend on the model output and consequently establish a feedback. In this case a network model is referred to as a *recurrent network*. In the linear case, as explained; the analysis of the root polynomials permits to check the stability of the ARMAX model. This is not the case for the NNARMAX model. Typically it is more relevant to consider the stability as a local property. It may happen that an NNARMAX model is stable when operated in some regimes and unstable in others. Thus whether or not this can lead to problems in practice will depend on how the model is operated.

**NNOE**
In this case the vector regression is:

\[ \varphi(t) = [\hat{y}(t-1|\theta),...,\hat{y}(t-n|\theta),u(t-d),...,u(t-d-m)]^T \]

See fig. 1.5
Some of the regressors in the NNOE structure are the predictions of past outputs and it is therefore subject to the same problems as the NNARMAX.
1.3.4 Variations and hybrid model structures

It is often proposed to use variations of the above mentioned model structures to create mixtures of linear and nonlinear model structures. Some examples are shown in the following:

- It is sometimes seen that the NNARX model structure is made up of two separate networks or by a network and a linear term

\[
\hat{y}(t|\theta) = g_y[\varphi_y(t), \theta_y] + g_u[\varphi_u(t), \theta_u] 
\]

1.27

\[
\hat{y}(t|\theta) = g_y[\varphi_y(t), \theta_y] + q^{-d} B(q^{-1})u(t) 
\]

1.28

\[
\hat{y}(t|\theta) = [1 - A(q^{-1})]y(t) + g_u[\varphi_u(t), \theta_u] 
\]

1.29

where

\[
\varphi_y(t) = [y(t-1),...,y(t-n)]^T 
\]
\[
\varphi_u(t) = [u(t-d),...,u(t-d-m)]^T 
\]

- Likewise various output error model structures can be generated; a modification of (1.29) yields

\[
\hat{y}(t|\theta) = [1 - A(q^{-1})]\hat{y}(t|\theta) + g_u[\varphi_u(t), \theta_u] 
\]

1.30

Which overcomes the problems with unknown stability characteristics of OR-models, since the feedback is controlled by a linear transfer function.

- Following the same logical deduction a variation of the NNARMAX model structure can be derived as

\[
\hat{y}(t|\theta) = g_{uy}[\varphi_{uy}(t), \theta_{uy}] + C(q^{-1})e(t, \theta) 
\]

1.31

where

\[
\varphi_{uy}(t) = [y(t-1),...,y(t-n),u(t-d),...,u(t-d-m)]^T 
\]

- Physical knowledge may obviously influence the choice of the regressors, which could imply raising some of the signals to different powers or applying combinations of different signals

- It is possible to implement the model structures so that they can contain their linear counterpart as a subset. This can be obtained by introducing direct connections from input to outputs

- Except for NNOE structures, it is easy to modify the model structures for time series analysis (i.e. no external input). The past inputs are simply omitted in the regression vector. In time series analysis an experiment is not conducted as in system identification since the system cannot be influenced.
Conclusion

The regressors are a number of past signals. Some commonly model structures are obtained by the following choice

- FIR/NFIR: past control inputs
- ARX/NNARX: past control inputs and observed outputs
- OE/NNOE: past control inputs and predicted outputs
- ARMAX/NNARMAX: past control inputs, observed outputs and predicted residuals

Sometimes a linear combination of nonlinear and linear model structures is required. For nonlinear model structures one has also to select the architecture of the neural network, that is the number of hidden neuron or layers.

1.3.5 Stability

Stability plays an important role in control theory: it is a necessary condition for feasibility of the control system that the closed-loop system (controller + stable) be stable. Besides being stable the closed-loop system must also satisfy various requirements, e.g. speed of response and damping. Also in system identification stability plays an important role, even if it is not so crucial as in control theory. In system identification plays a key role in connection to asymptotic analysis of estimation methods and in connection to practical implementation of training methods: if the predictor is unstable for a certain choice of model parameters, numerical problems may occur during training.

The basic block in the nonlinear model structures described earlier is in the form

\[ \hat{y}(t | \theta) = g[\phi(t, \theta), \theta] \] (1.32)

As shown in fig. 1.6

![Fig. 1.6 Example of a neural network model with feedback](image)

This description can be easily transformed into the form

\[ X(t + 1) = F_x[X(t), U(t)] \quad X(t_0) = X_0 \] (1.33)
Where $X(t)$ is the part of the regression vector $\phi(t)$ containing past outputs from the neural network $U(t)$ is the part of the regression vector $\phi(t)$ containing past inputs from the neural network and $F_u$ consists of the neural network function $g$ and the shift function.

Consider now that $U(t)$ is a function of $t$, then (1.33) can be transformed into the the following discrete-time state space model:

$$X(t+1) = F[X(t),t] \quad X(t_0) = X_0 \quad (1.34)$$

Where $t \in \mathbb{Z}$. Now let $X_0(t;X_0,t_0)$ be the unique solution to (1.34) with initial value given by $X_0$.

For brevity we indicate this solution $X_0(t)$ and call it nominal solution. Then by introducing $x(t) = X(t) - X_0(t)$, i.e. the deviation from the nominal solution, the above equation can be described by:

$$x(t+1) = f(x(t),t) \quad (1.35)$$

where the nominal solution corresponds to the initial condition $x(t_0) = 0$ in this case. We will call then an equilibrium point a constant point $x_e$, if it exists, satisfying (1.35), that is such that

$$f(x_e,t) = 0 \quad \forall t \geq t_0 \quad (1.36)$$

One such equilibrium point of (1.35) is $x_e = 0$ in the case under study, that is

$$f(0,t) = 0 \quad \forall t \geq t_0$$

Now the stability of a solution can now be defined for a generic $x(t_0)$

**Definition 1 (Liapounov stability)**

The equilibrium point of (1.35) $x_e$ is stable if, given $x(t_0) = x_0$,

$$\forall \varepsilon > 0 \text{ and } \forall t_0, \exists \delta(\varepsilon,t_0) > 0 : \|x(t_0) - x_e\| \leq \delta \Rightarrow \|x(t) - x_e\| \leq \varepsilon \quad \forall t \geq t_0 \quad (1.37)$$

where $x(t)$ is the solution of (1.35) associated with the initial condition $x(t_0) = x_0$.

Remark: the Liapounov stability requires that the solution $x(t), \forall t \geq t_0$, remains inside the sphere of centre $x_e$ and radius $\varepsilon$, if the initial condition $x(t_0) = x_0$ is chosen inside the sphere of centre $x_e$ and radius $\varepsilon$.

This definition is only valid in sampling times, which is of interest in system identification (not in control theory where the controlled system is continuous). In most practical situations stability in discrete-time implies continuous-time stability, but it is not always this way and thus a rigourous treatment of the stability issue in continuous-time should always take place.

**Definition 2**

If the statement (1.37) is valid for a $\delta$ independent of $t_0$, then the solution is said to be uniformly stable.
However one is not often satisfied with the fact that the solution $x(t)$ is arbitrarily close to the equilibrium point $x_e$: asymptotically the two solutions are required to be identical.

**Definition 3**

The equilibrium point of (1.35) $x_e$ is asymptotically stable if it is stable (according to definition 1) and moreover there exist a $\rho > 0$ such that:

$$\lim_{t \to \infty} x(t) = x_e \quad \forall x_0 \quad \text{with} \quad \|x(t_0) - x_e\| \leq \rho$$

(1.38)

Sometimes the stability of the solution is not an issue, what is important is to get a bounded output if the input is bounded; examples of norms used in this cases are:

$$\|x(t)\|_\infty = \sup_{t \geq 0} \|x(t)\|$$

$$\|x(t)\|_2 = \left( \sum_{t=0}^{\infty} x(t)^T x(t) \right)^{1/2}$$

$$\|x(t)\|_p = \left( \sum_{t=0}^{\infty} \|x(t)\|^p \right)^{1/p}$$

**Definition 4**

The solution $x(t)$ is BIBO stable if there exists constants $c_u$, $c_x$, such that:

$$\|u(t)\| < c_u \quad \Rightarrow \quad \|x(t)\| < c_x$$

In a neural network with only bounded activation functions, such as hyperbolic tangents, the output will always be bounded.

For linear time invariant (LTI) systems it is well-known that asymptotic and BIBO stability is obtained if the eigenvalues of the system matrix, or the poles of the transfer function, are strictly inside the unit circle.

For autonomous nonlinear systems (if in the equation (1.35) the time does not appear explicitly), the origin is asymptotically locally stable if the corresponding linearized system matrix has eigenvalues strictly less than one.

For time-varying nonlinear systems, such as neural networks models with time-varying inputs, the analysis is much more complicated: under the assumption that the time variation is sufficiently slow, the stability of the neural network can be established by evaluating the eigenvalues of the linearized system.

In practical terms, in system identification with neural networks, what is usually meant for stability is that the signals should not “explode” and deviate too far from the desired ones. Typically stability problems will be that the network or system output goes into excessive oscillations instead of being smooth, or that the signals saturate.

**Conclusion**

The predictors must be stable. Instability might occur for model structures that have feedback. NNFIR or NNARX model structures are usually to be be preferred as a first choice as their
predictors cannot become unstable. Although a predictor in principle should be stable, stability problems might occur during training or afterwards if the obtained model is inaccurate.

1.3.6 Selecting the Lag Space (Method of the Lipschitz quotient)

The model structure selection is here treated at a quite general level. Often it is of little importance that the network architecture has been selected a few parameters too small or too large. However a wrong choice of the lag space, that is the number of delayed signals used as regressors, may have a disastrous impact on some control applications. Too small obviously implies that the essential dynamics will not be modelled, too large a lag space, e.g. in linear systems, may manifest itself as common factors in the identified transfer function, so a similar behaviour should be expected in the nonlinear case: although this is not always a problem, common factors (corresponding to hidden modes) can lead to control problems in some controller designs.

The specific choice of the network architecture and regressors is a difficult task. While however it is difficult to apply physical insight for determining the optimal number of hidden neurons or layers, it can often dictate the proper lag space: if the lag space selection is made well, the model structure selection is substantially reduced.

One method is now described which can be used for deterministic systems. It is based on a prior assumption that the system can be represented accurately by a function which is reasonably smooth in the regressors.

Hypothesis
1) Let the system be described by a noise-free NNARX model:

\[
y(t) = g_0[\varphi(t), \theta] \tag{1.39}
\]

\[
\varphi(t) = [\varphi_1, \varphi_2, \ldots, \varphi_z]^T = [y(t-1), \ldots, y(t-n), u(t-d), \ldots, u(t-d-m)]^T
\]

2) Let a data set consisting of \(N\) input-output pairs be given (how to get them is explained later on):

\[
Z^N = \{\varphi(t), y(t) \mid t = 1, \ldots, N\} \tag{1.40}
\]

3) Let the magnitude of the system derivative with respect to each of the regressors be bounded by some positive value \(B\) (Lipschitz condition).

\[
|g_{l,i}| = \left|\frac{\partial g_0}{\partial \varphi_{l,i}}\right| \leq B \quad l = 1, 2, \ldots, z \tag{1.41}
\]

For all combinations of input-output pairs, the Lipschitz quotient is now introduced

\[
q_{y} = \left\|\frac{y(t_i) - y(t_j)}{\varphi(t_i) - \varphi(t_j)}\right\|, \quad i \neq j \quad i, j = 1, \ldots, N, \quad \tag{1.42}
\]

where \(\|\|\) specifies the Euclidean norm. The Lipschitz condition then states that \(q_y\) is always bounded if the function \(g_0\) is continuous, that is \(0 \leq q_y \leq L\).
Consider now the differences $\delta y = y(t_i) - y(t_j)$ and $\delta \varphi_i = \varphi_i(t_i) - \varphi_i(t_j)$. If the differences $\delta \varphi_i$ are small, the following approximation is possible:

$$
\delta y \approx \frac{\partial g_0}{\partial \varphi_1} \delta \varphi_1 + \frac{\partial g_0}{\partial \varphi_2} \delta \varphi_2 + ... + \frac{\partial g_0}{\partial \varphi_z} \delta \varphi_z = g_1 \delta \varphi_1 + g_2 \delta \varphi_2 + ... + g_z \delta \varphi_z
$$  \hspace{1cm} (1.43)

Moreover

$$
|\varphi(t_i) - \varphi(t_j)| = \left|\varphi_1(t_i), \varphi_2(t_i), ..., \varphi_z(t_i)\right| - \left|\varphi_1(t_j), \varphi_2(t_j), ..., \varphi_z(t_j)\right| = |\delta \varphi^T| = \\
\left|\delta \varphi_1, \delta \varphi_2, ..., \delta \varphi_z^T\right| = \sqrt{(\delta \varphi_1)^2 + (\delta \varphi_2)^2 + ... + (\delta \varphi_z)^2}
$$

Consequently the Lipschitz quotient must obey, if in a first approximation all $\delta \varphi_i$ are small and almost equal:

$$
q^{(z)}_y = \frac{|\delta y|}{\sqrt{(\delta \varphi_1)^2 + (\delta \varphi_2)^2 + ... + (\delta \varphi_z)^2}} = \frac{|g_1 \delta \varphi_1 + g_2 \delta \varphi_2 + ... + g_z \delta \varphi_z|}{\sqrt{(\delta \varphi_1)^2 + (\delta \varphi_2)^2 + ... + (\delta \varphi_z)^2}} \leq \sqrt{Z}B
$$  \hspace{1cm} (1.44)

$Z$ refers to the total number of regressors.

It is interesting to apply this inequality on two different cases: the regression vector is insufficient and the regression vector is too large:

**The regression vector is insufficient**

Assume that the $z$-th regressor is missing, then:

$$
q^{(z-1)}_y = \frac{|\delta y|}{\sqrt{(\delta \varphi_1)^2 + (\delta \varphi_2)^2 + ... + (\delta \varphi_{z-1})^2}} = \frac{|g_1 \delta \varphi_1 + g_2 \delta \varphi_2 + ... + g_{z-1} \delta \varphi_{z-1}|}{\sqrt{(\delta \varphi_1)^2 + (\delta \varphi_2)^2 + ... + (\delta \varphi_{z-1})^2}} = \alpha q^{(z)}_y
$$  \hspace{1cm} (1.45)

As an extreme example, imagine that the differences $\delta \varphi_l = 0$ for all $l$ except for $l = \alpha \to \infty$ from (1.45). If the output depends on the $z$-th regressor, there will be obviously points where the difference $\delta y = y(t_i) - y(t_j) \neq 0$ and $g_z$ is different from zero, which implies $q^{(z-1)}_y \to \infty$.

Disregarding the regressor $\varphi_z$ will thus lead to an infinite Lipschitz quotient. In the general case it must be expected that the lack of a regressor most often will lead to very large quotients, and moreover as more regressors are missing, the quotient will increase significantly.

**The regression vector is too large**

If too many signals are included in the regression vector, the vector will contain redundant information. Consider for instance the case where one additional regressor is included:
\[ q^{(z+1)}_y = \frac{|\delta_1|}{\sqrt{(\delta\phi_1)^2 + (\delta\phi_2)^2 + \ldots + (\delta\phi_{z+1})^2}} = \frac{\sqrt{(\delta\phi_1)^2 + (\delta\phi_2)^2 + \ldots + (\delta\phi_z)^2}}{\sqrt{(\delta\phi_1)^2 + (\delta\phi_2)^2 + \ldots + (\delta\phi_z)^2}} \beta q^{(z)}_y \]  

(1.46)

Then \( \beta<1 \), which means that the Lipschitz quotient will have an insignificant reduction. So an additional superfluous regressor will have only a minor impact on the Lipschitz quotient.

These properties can be used as a criterion for the determination of the optimal regressor structure. The procedure is as follows:

1. For a given choice of the lag space, determine the Lipschitz quotients for all combinations of input-output pairs.
2. Select the \( p \) largest quotients, e.g. \( p=0.01N \div 0.02N \). The largest quotients typically occur when the differences \( \delta\phi_i \) are small.
3. Evaluate the index value:

   \[ \bar{q}^{(z)} = \left( \prod_{k=1}^{p} \sqrt{z}q^{(z)}(k) \right)^{1/p} \]  

(1.47)

where \( q^{(z)}(k) \) is the Lipschitz quotient based on the \( z \) regressors for the \( k \)-th largest quotient.
4. Repeat the calculations for a number of different lag structures.
5. Plot the criterion as a function of lag-space and select the optimal number of regressors as the “nee-point” of the curve.

In any case it is very time-consuming to compute all the quotients. In particular if \( N \) is the large and one wishes to explore a large number of lag structures. Therefore one must consider letting the number of past inputs and outputs be increased simultaneously. As long as they are not both too large, “common factors” are avoided. Fig. 1.7 shows the method applied to a data set obtained from a simulated experiment with a nonlinear dynamic system. The number of past inputs and outputs are increased simultaneously from \( n=m=1 \) to \( n=m=8 \). It is seen that the index value recommends using the lag space \( n=m=2 \), which is actually the correct dimension.
Conclusion
In the noise-free case it is sometimes possible to determine the lag space (number of inputs and outputs) automatically with the so-called Lipschitz coefficients.

1.3.7 Experiment

The primary purpose of an experiment is to produce a set of examples of how the dynamical system to be identified responds to various control inputs; It is the first stage in the identification process. One must therefore be extremely careful to collect a set of data that describes how the system behaves over the entire range of operation. Although the choice and location of sensors also could be considered part of the experiment, here it is assumed that the complete system, including sensors, is given. It is also assumed that a certain control over the inputs is available.

Nonlinearity tests
If one has insufficient physical insight into deciding whether a linear system can be used instead of a linear one, with resulting easiness to design and implement controllers, one can simply estimate a number of different linear models to investigate this. Even this is the most logical path to follow, it is the most laborious way to check this. To avoid therefore an unnecessary waste of time it is desirable to have a collection of simple tests at disposal for determining “how nonlinear” the system is actually.
In the following some simple nonlinearity tests are presented.

**Superposition check.** Two main characteristics of nonlinear systems are that superposition

\[ y(t) = g[\varphi_1(t) + \varphi_2(t)] = g[\varphi_1(t)] + g[\varphi_2(t)] \]  \hspace{1cm} (1.48)

and homogeneity

\[ y(t) = g[\alpha \varphi(t)] = \alpha g[\varphi(t)] \]  \hspace{1cm} (1.49)

Are NOT satisfied, at least not over the whole operating range. If no disturbances affect the system these conditions are quite easy to check. It is necessary to assume the system AT REST for (1.48) and (1.49) to hold for linear systems, otherwise one must also take into account the initial conditions. If the system is stable one can also wait until the transient has died off before checking the conditions.

As an example one can try out the following procedure:

- Apply a zero input signal and wait for the steady state
- Is there a DC offset D?
- Apply two different input signals \( u_1(t) \) and \( u_2(t) \) such that \( u_2(t) = c \) \( u_1(t) \) to which correspond \( y_1(t) \) and \( y_2(t) \)
- If the system is linear then the ratio

\[ r(t) = \frac{y_2(t) - D}{y_1(t) - D} \]

Should be equal to \( c \) at all times.

As a nonlinearity test for the system one can therefore use

\[ v = \max_i \left| \frac{r(t) - c}{c} \right| \]

Which for linear systems should be zero.

**Frequency response check.** It is well known that the frequency response is unique for linear systems regardless of the amplitude of the input signal. In order to check for linearity one can thus apply different sinusoids to the system. The frequency and amplitudes should be varied. If the system is LINEAR, the stationary output should be a sinusoid with the same frequency and the amplitude proportional to the input.

By checking the presence of subharmonics by a Fourier analysis of the output signal, one can get an idea of the nonlinearity. One should take some care of disturbances. If the system is heavily affected by measurement noise it is advisable that the output sequence be averaged over a number of trials.

Example:

Consider the system

\[ \ddot{y}(t) + \dot{y}(t) + y(t) + y^3(t) = u(t) \]
If three different square waves with increasing amplitude were given to the system; the response of the system becomes less damped. If a sinusoid with increasing amplitude were given the response is not even sinusoidal for higher input signal values. See the two following figures. The homogeneity condition is not satisfied.
Experiment design

Choice of sampling frequency: if the sampling frequency is chosen too high compared to the dynamics of the system under consideration, numerical ill-conditioning can occur in system identification.

If the identification is performed for control system design, the sampling frequency should also be chosen in accordance with the desired dynamics of the closed-loop system (controller + system). A high sampling frequency permits a rapid reference tracking and a smoother control signal, but the problems with numerical ill-conditioning become more pronounced. Thus the selection of the sampling frequency should be a compromise between favouring the identification and favouring the controller design.

Conclusion: the choice of the sampling frequency. Not too slow, but not fast either.

The curse of dimensionality: since, unlike linear systems, the superposition and homogeneity properties do not hold for nonlinear systems, this imposes an immense increase in the demand for excitation in the input signal. While for identification of linear systems it is sufficient to apply a signal containing a finite number of frequencies, a nonlinear system requires that all combinations of frequencies and amplitudes in the system operating range be represented in the signal. As a consequence the necessary size of the data set increases dramatically with the number of inputs and outputs. This is another example of the curse of dimensionality and there is no remedy. This is the main drawback for nonlinear black-box approaches and prevents the use of neural networks for the identification of large scale systems.

Designing the input signal: before an input signal is selected it is important to identify the operating range of the system. Special care must be taken not to excite dynamics that one does not intend to incorporate in the model, e.g. resonances. Typically this implies that the input signal be kept at sufficiently low frequencies.

In the identification of linear systems it is customary to apply a signal consisting of a number of sinusoids of different amplitudes.

When working with nonlinear systems it is important that all amplitudes and frequencies are represented, as mentioned above. Some input signals that attempt to meet this demand are proposed below:

\[ N\text{-samples constant: } u(t) = e(t) \text{ for } t = 1, 2, \ldots, \]

Will then jump to a new level at each \( N \)th sampling instant. An example is given in fig.1.8
Fig. 1.8. The “N-samples constant” signal for $N=15$. The signal is scaled to fit the interval $[-1;+1]$.

**Level change at random instances:** an extension to the N-samples constant signal is obtained by introducing an additional random variable for deciding when to change the level

$$u(t) = \begin{cases} 
  u(t-1) & \text{with probability } \alpha \\
  e(t) & \text{with probability } 1 - \alpha
\end{cases}$$

See fig. 1.9 for an example. The higher $\alpha$, the smoother the input signal.

Fig 1.9 : Input signal where the level is changed at random instances.
Chirp signal: a chirp signal is a sinusoid with a gradually increasing frequency. With such a signal one can accurately excite the desired frequency range. The signal can be generated by the following procedure:

\[
\omega_t = \omega_{\text{start}} + \frac{t}{N}(\omega_{\text{final}} - \omega_{\text{start}})
\]

\[
u(t) = u_0 + A\sin(\omega_t T_s) \quad t = 1,2,\ldots, N
\]

Where \(T_s\) is the sampling frequency. See fig. 1.10 where the frequency is increased from \(\omega_{\text{start}} = 0.01/T_s\) to \(\omega_{\text{final}} = 0.2/T_s\)

When applying this signal to a nonlinear system, it should be repeated for a number of different DC-values \((u_0)\) and amplitudes \((A)\).

Conclusion: it is important that the input signal applied to the system really “shakes” the system so that all the operating range is covered. Make sure to collect plenty of data, but try to avoid collecting too much redundant information.

Collecting data in closed-loop

When a system is unstable or poorly damped, in order to keep the system inside the range in which it is intended to operate, it is necessary to use

1. a stabilizing controller
2. or, alternatively, a manually tuned PID controller
3. or a human operator for controlling the system

In fact if the model is going to be used for control system design, it is a good idea to collect data in closed-loop. It has been proved that identifiability of systems operating in closed-loop under certain circumstances can be lost. One way to preserve identifiability is either using a nonlinear controller or an additional input in the loop (fig. 1.11): even if the system is identifiable, it is often a good idea to add an additional signal anyhow. A PID controller or a manually tuned controller typically perform so as to give input signals at low frequencies. This may have the effect that the high
frequency portion of the operating signal is not excited properly. Adding a signal with more energy in high frequencies may then compensate for this deficiency.

Fig. 1.11: a new input signal is added to the output of a manually tuned controller to further excite the system

Preparing the data for modelling

Intelligent pre-processing of the data is often much more important than trying a large number of different model structures and training schemes. Pre-processing implies the extraction of most valuable information from the measured data to make it suitable for neural network modelling. Some techniques are suggested in the following:

Filtering: filtering can be used to remove from the measured signals noise, periodic disturbances, off-sets, and the effects of “uninteresting” dynamics. When high frequency noise or disturbances cause problems, it is advisable to remove them by using an analog presampling filter to avoid aliasing problems. Off-sets, drifts, and low frequency disturbances can be removed by digital filters.

Removing redundancy and outliers from the data set: sometimes a large number of input-output pairs from a small regime of the whole operating range dominates the data set. When training on this data set, it is likely that the model obtained will be very accurate in the regime that was over-represented at the expense of a poor performance outside this regime. So a little “surgery” is necessary to eliminate redundant information: apart from obtaining a more equal weighting of the information, a reduction of the data set size will also reduce the training time.

It is also recommended to remove outliers from the data set, which can have a detrimental effect while training.

Note that NNOE and NNARMAX models (i.e. recurrent networks) can have problems in the identification process if portions of the data are removed: the feedback connections in the model give rise to a transient when an abrupt change in the output signal occurs. These transients will need some time to disappear and can corrupt the training.

Scaling: It is highly recommended to remove the mean and scale all signals to the same variance, maybe by using a transformation to a standard variable (zero mean and unit variance): in fact the signals are usually measured in different physical units and without scaling there is tendency that the signals of largest magnitude will be too dominating. Moreover scaling makes the training algorithm numerically robust and leads to a faster convergence and finally better models.

If the network model is a 2-layer neural network with linear output units, it is straightforward to rescale the weights after the training sessions is completed. In this way the final network model can work in the productive phase on unscaled data.