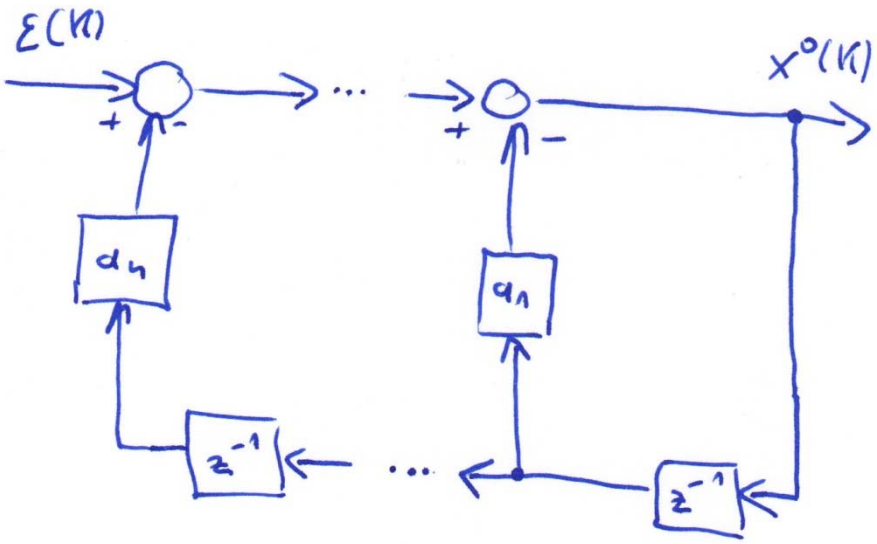
Fachgebiet Systemanalyse

Lecture Notes
System Identification

Summer Semester 2017
RCSE course

Dr.-Ing. Th. Glotzbach
Prof. Dr.-Ing. habil. Ch. Ament



Date 03/2017 (V10)

Content

Basic methods

Experimental analysis

Text books

- R. Isermann, M. Münchhof: Identification of Dynamic Systems – An Introduction with Applications, Springer, 2011 (93,80 €)
- L. Ljung: System Identification – Theory for the User, 2nd edition, Prentice Hall, 1998 (92 €)
- D. Simon: Optimal State Estimation – Kalman, H Infinity, and Nonlinear Approaches, Wiley, 2006 (112 €)

1 Introduction

1.1 Motivation

For which purposes do we need models of technical processes?

- Analysis of a system (stability, time constants, etc.)
- Prediction of system behavior (by numerical simulation)
- Controller design
- Diagnosis and monitoring
- Optimization of system design

1.2 Models

Definition

A *mathematical model* is a description of a system using mathematical concepts and language. […]

Mathematical models are used not only in the natural sciences (such as physics, biology, earth science, meteorology) and engineering disciplines (e.g. computer science, artificial intelligence), but also in the social sciences (such as economics, psychology, sociology and political science); physicists, engineers, statisticians, operations research analysts and economists use mathematical models most extensively.

A model may help to explain a system and to study the effects of different components, and to make predictions about behaviour. [Wikipedia]

Classification by the type of mapping

Physical models	*Conceptual models*
are down-scaled or simplified physical realizations of the real system, e.g. architectural models, or rapid prototyping components.	exist only in our mind, e.g. describing a system by mathematical equations or by a theoretical structure.

Classification by the type of description

White box models	*Black box models*
describe a system by causally determined relations. Such an analytical description is derived from physical laws („first principles“) and is mostly the result of theoretical modelling.	characterize the relation of inputs and outputs in a descriptive way. General model approaches or non-parametric models are used as a result of experimental modeling.

Also mixed models are used; they can be addressed as *grey box models.*

Parametric models

are built from a structured model approach and a finite number of model parameters, which quantify the model approach. Compared to non-parametric models these models achieve a more compact representation.

Non-parametric models

do not have a certain structure and characterize the system by data that is received from measurement, e.g. a step response. In principle they have an infinite number of parameters.

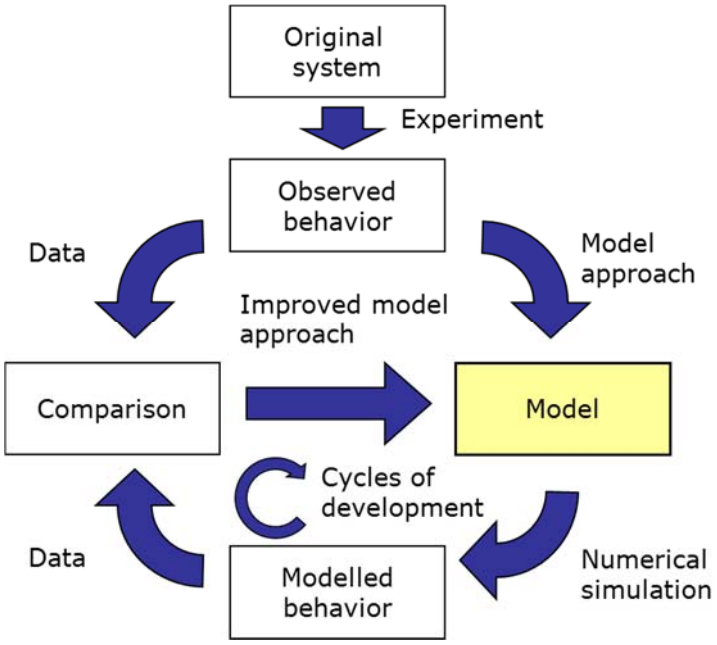
1.3 Modelling

Definitions

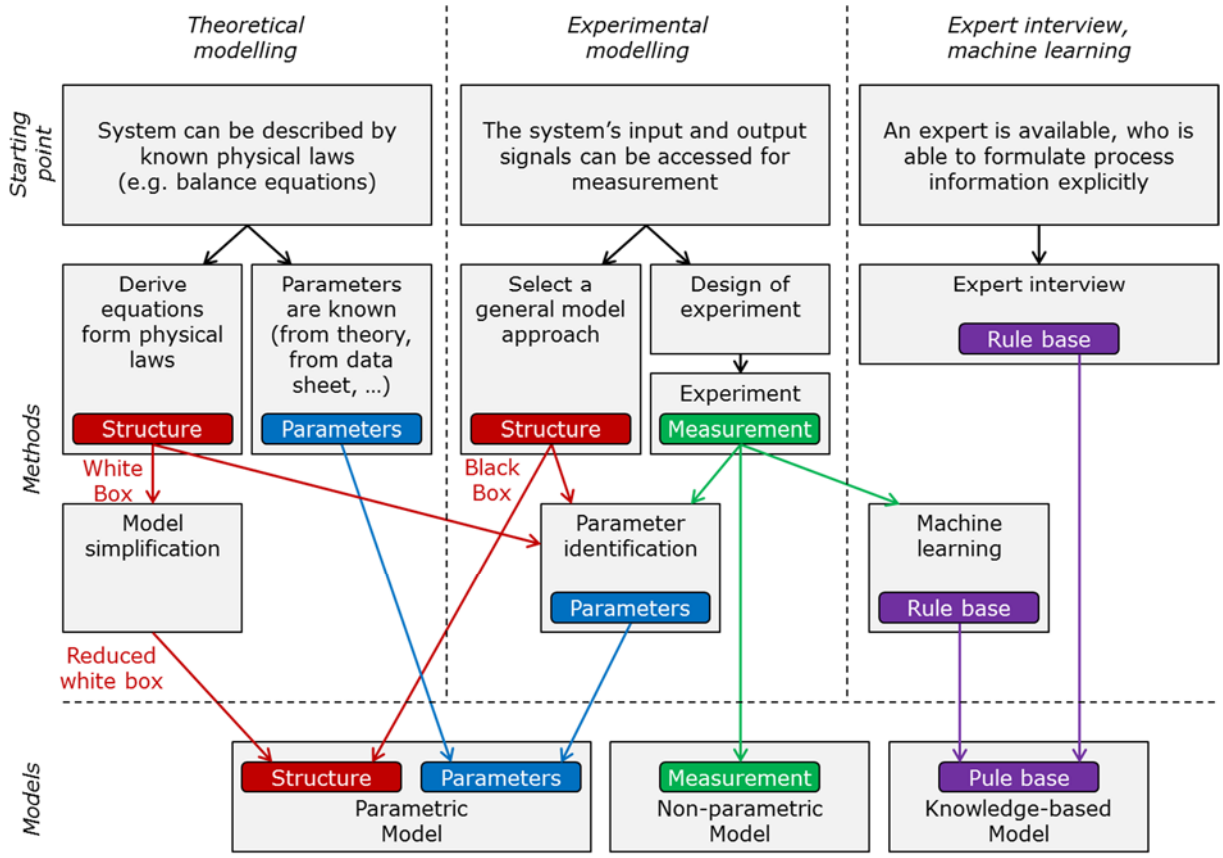
The process of developing a mathematical model is termed *mathematical modelling*. [Wikipedia].

The field of *system identification* uses statistical methods to build mathematical models of dynamical systems from measured data. System identification also includes the optimal design of experiments for efficiently generating informative data for fitting such models as well as model reduction. [Wikipedia]

The modeling process



Modeling strategies

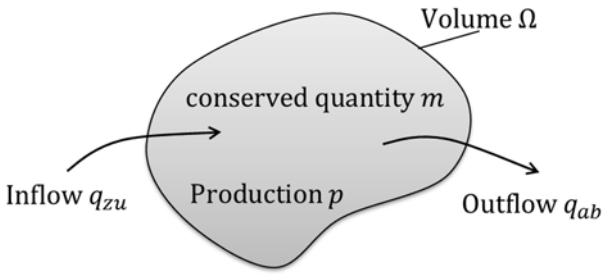


2 Physical model approaches: White box models

2.1 Balance equations

Balancing a conserved quantity over time is a very universal modelling principle, which can be applied e.g. for process plants or biological systems.

The conserved quantity m is balanced within a specified volume Ω:



The balance equation in integral form is:

$$m(t) = m(0) + \int_0^t \left[q_{zu}(\tau) - q_{ab}(\tau) + p(\tau)\right] d\tau$$

The balance equation in differential form is:

$$\dot{m}(t) = q_{zu}(t) - q_{ab}(t) + p(t)$$

Typical conserved quantities are:

- Mass or volume
- Energy
- Number of cells
- Electrical charge
- Momentum

For each volume Ω we obtain one first order linear differential equation.

2.2 Electrical systems

Equations of RLC Networks

Kirchhoff voltage law:

$$\sum_j u_j(t) = 0$$

Kirchhoff current law:

$$\sum_j i_j(t) = 0$$

Current voltage relation of network elements:

Element	*Current voltage relation* (time domain)	Ohm's law (frequency domain)
Resistance *R*	$u(t) = R \cdot i(t)$	$U(s) = R \cdot I(s)$
Capacitor *C*	$u(t) = \frac{1}{C}\int_0^t i(\tau)\, d\tau$	$U(s) = \frac{1}{s\,C} \cdot I(s)$
Inductor *L*	$u(t) = L \cdot \frac{di(t)}{dt}$	$U(s) = s\,L \cdot I(s)$

The general Ohm's law with impedance $Z(s)$ is:

$$U(s) = Z(s) \cdot I(s)$$

Mesh current analysis of RLC Networks

1. If applicable, *current sources* have to be transformed to *voltage sources*.
2. Plot the network *graph*.
   Select a *tree* out of this graph.

   Definitions:
   Nodes correspond to the connecting wires, *edges* to the devices.
   A closed loop within the graph is called a *mesh*.
   A *tree* connects all nodes of a graph without any meshes.

3. Introduce currents I_i with i=1,...,n in all edges of the graph that do not belong to the tree.

4. The system of equations is:

$$\underbrace{\begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n1} & Z_{n2} & \cdots & Z_{nn} \end{bmatrix}}_{\underline{Z}} \cdot \underbrace{\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix}}_{\underline{I}} = \underbrace{\begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{bmatrix}}_{\underline{U}}$$

- $\underline{Z}$ impedance matrix of the system with the main diagonal elements Z_{ii} and all other elements Z_{ik}:

  Z_{ik} : Impedance that is passed by I_i and I_k commonly. If the direction of both currents is similar use positive sign, otherwise negative sign.

  Z_{ii} : Sum of all impedances along mesh i

- $\underline{I}$ vector of currents

  I_i : Current of mesh i (with $i=1,\ldots,n$)

- $\underline{U}$ vector of voltage sources

  U_i : Sum of independent voltage sources in mesh i. If the voltage source supports the mesh current, use positive sign.

2.3 Modeling analogies

The analysis of systems with lumped elements (devices) can be generalized. For its characterization two complementary variables are used:

- *Flow* $q(t)$: The flow is passing a network device and can be measured within the device. Due to the Kirchhoff current law the sum of all currents in relation to a network node is zero.
- *Potential* $V(t)$: The potential is assigned to the connectors of a network device; a potential difference can be measured between two connectors. Due to the Kirchhoff voltage law the sum of potential differences within a network mesh is zero.

A *generalized Ohm's law* characterizes the relation of flow and potential difference regarding a lumped network element.

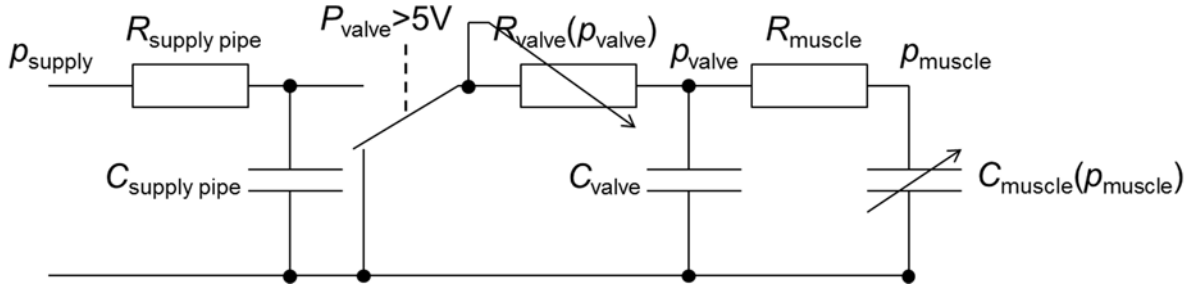
Application in different physical domains:

Domain	Flow $q(t)$	Potential (difference) $V(t)$	Resistance	Capacitor (Storage of flow)	Inductor (Storage of potential diff.)
Electrical system	El. current $I(t)$ in A	El. voltage $U(t)$ in V	El. resistance R in Ω = V/A	El. capacitor C in F = As/V	El. inductor L in H = Vs/A
Mechanical system (translatory)	Force $F(t)$ in N	Velocity $v(t)$ in m/s	Reciprocal friction constant $1/k$ in m/Ns	Mas m in kg	Reciprocal spring constant $1/c$ in m/N
Mechanical system (rotatory)	Torque $M(t)$ in Nm	Angular velocity $\omega(t)$ in 1/s	Reciprocal friction constant $1/k$ in 1/Nms	Moment of inertia J in kg m^2	Reciprocal spring constant $1/c$ in 1/Nm
Pneumatic system	Mass flow $\dot{m}(t)$ in kg/s	Pressure $p(t)$ in Pa= N/m^2	Pneumatic resistance in 1/ms	Storage of mas in kg m^2/N	–
Thermal system	Heat flow $q(t)$ in W=J/s	Temperature $T(t)$ in K, °C	Thermal resistance in K/W	Thermal capacity in Ws/K	–

The product of flow and potential (difference) is usually a power: $P(t) = q(t)\, V(t)$

Example: Pneumatic system

In the following a pneumatic system (including a supply pipe, a valve and a pneumatic muscle) is described using electrical symbols:



2.4 Block diagram

Advantages of block diagrams:

- Clear graphical representation.
- Model implementation can be done step by step.
- Substructures may help to make it clearer.
- Within a project blocks can be exchanged easily.
- Numerical simulations can be carried out based on block diagrams (e.g. by Simulink)

- C-Code can be generated from block diagrams.

Classification:

Object orientated modeling: Nodes are physical components; edges are physical interactions between these components. Description is close to physical reality. Software: e.g. Modelica/Dymola	*Causal modeling:* Nodes are transfer functions; edges are signals. Useful for system analysis. Software: e.g. Matlab/Simulink

2.5 State space description

The state of a dynamical system at time t is completely characterized by a set of states $x_1(t)$, … $x_n(t)$. These states are summarized as state vector $\underline{x}(t)$.

System dynamics is defined by the state differential equation. It describes the change of states in time $\dot{\underline{x}}(t)$ as a function of the current state $\underline{x}(t)$ and the system's input $\underline{u}(t)$. It is a system of first-order differential equations.

Measurable outputs are summarized as output vector $\underline{y}(t)$. The output equation defines $\underline{y}(t)$ based on the current state $\underline{x}(t)$ and (in rare cases) the system's input $\underline{u}(t)$. The output equation is an algebraic equation.

System description

State differential equation:	$\dot{\underline{x}}(t) = \underline{f}\left(\underline{x}(t), \underline{u}(t)\right)$
Output equation:	$\underline{y}(t) = \underline{g}\left(\underline{x}(t), \underline{u}(t)\right)$

with the following variables:

State vector $\underline{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}$ with n: Number of states

Input vector $\underline{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}$ with m: Number of inputs

Output vector $\underline{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_p(t) \end{bmatrix}$ with p: Number of outputs

3 General model approaches: Black box models

If no physical laws (first principles) are available or their derivation is too time-consuming, we can use general (nonphysical) model approaches. They describe the input output relation of the real process. In the following commonly used general model approaches will be presented.

The model parameters have no physical interpretation. They are unknown at that point and have to be identified based on measurement (see chapter 4).

3.1 Look-up table models

Look-up tables define a functional relation by sampling the whole range of an input u and storing the corresponding values for the output $\hat{y}$ explicitly. This might be a simple and useful method to map e.g. a calibration curve.

It can be considered as a non-parametric model (see chapter 1).

Example of a look-up table model

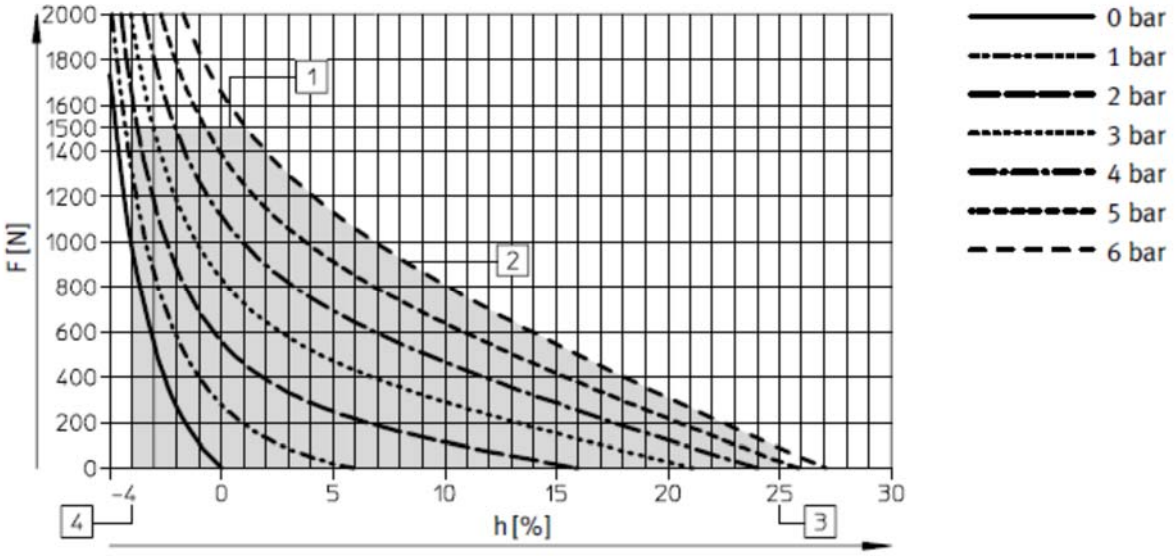
Resistance values of a PT100 sensor as a function of temperature [from www.pt100.de]:

[-200...+39 °C] [+40...+289 °C] [+290...+339 °C] [+340...850 °C]

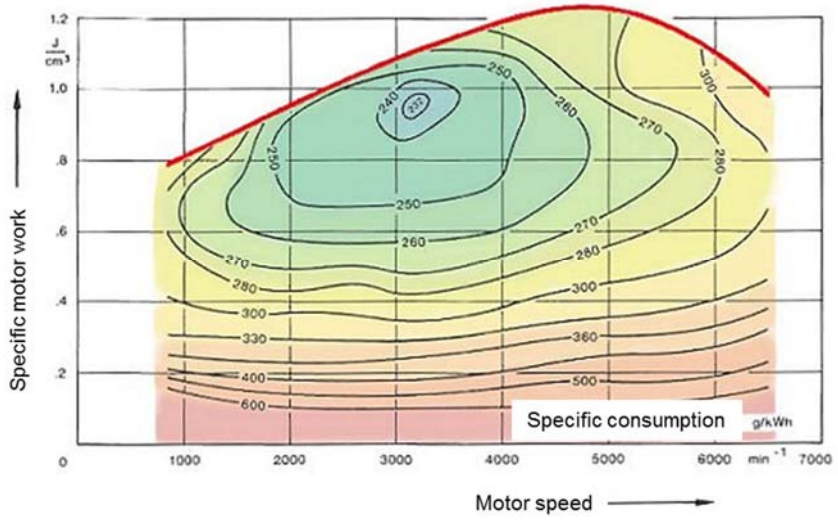
	0	1	2	3	4	5	6	7	8	9
40	115,539	115,925	116,311	116,697	117,083	117,469	117,854	118,240	118,625	119,010
50	119,395	119,780	120,165	120,550	120,934	121,319	121,703	122,087	122,471	122,855
60	123,239	123,623	124,007	124,390	124,774	125,157	125,540	125,923	126,306	126,689
70	127,072	127,454	127,837	128,219	128,602	128,984	129,366	129,748	130,130	130,511
80	130,893	131,274	131,656	132,037	132,418	132,799	133,180	133,561	133,941	134,322
90	134,702	135,083	135,463	135,843	136,223	136,603	136,982	137,362	137,741	138,121
100	138,500	138,879	139,258	139,637	140,016	140,395	140,773	141,152	141,530	141,908

Examples for a graphical representation of models:

1. Pneumatic muscle: Functional relation of muscle air pressure and muscle contraction h to the resulting muscle force F. Data sheet of the pneumatic muscle with 20 mm diameter from Festo:



2. Performance diagram of a combustion engine: Functional relation of motor speed and specific motor work to motor consumption:



3.2 Polynomial models

Approach for a system with single input (m=1) and single output (SISO system) with the order n of the polynomial approach:

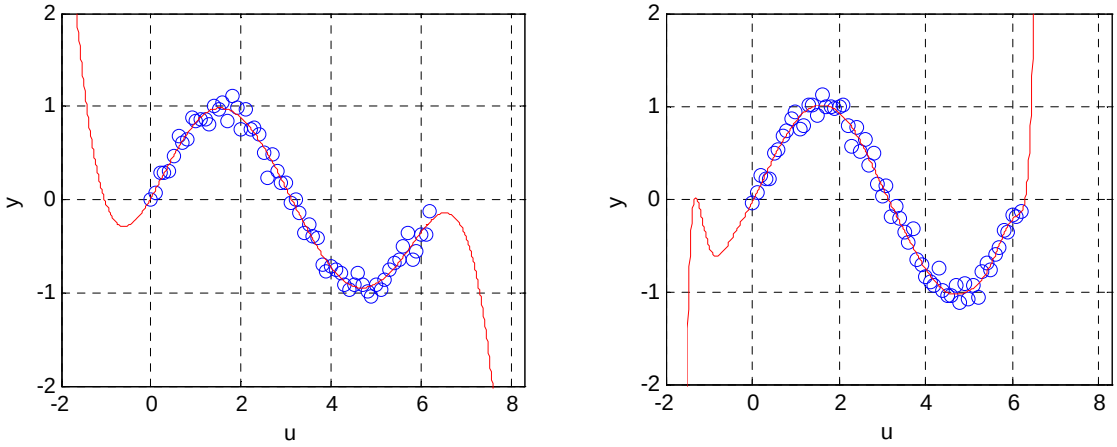
$$\hat{y} = a_0 + a_1\, u + a_2\, u^2 + \ldots + a_n\, u^n = \sum_{i=0}^{n} a_i\, u^i$$

Approach for a system with multiple inputs (m>1), and single output (MISO system):

$$\hat{y} = a_0 + \sum_{j=1}^{m} a_i\, u_j + \sum_{j=1}^{m} \sum_{k=1}^{m} a_{j,k}\, u_j\, u_k + \ldots$$

The output $\hat{y}$ of this approach is a nonlinear function of the input in case $n>1$. Hence, nonlinear input output relations can be describes with this model approach. On the other hand, output $\hat{y}$ is a linear function of the model parameters a_i, which will be turn out to be a major advantage in the following chapter 4.

An unknown function can be characterized by a polynomial approach throughout its whole input range. But, local aberration can be characterized only insufficiently. Moreover, it cannot be expected to obtain good extrapolation properties as the following example will illustrate [from Nelles: Nonlinear System Identification]. A sin function is approximated by a polynomial approach with n=5 (left) und n=15 (right).



3.3 Radial basis function models

A radial basis function approximates a given system within a local area of the input space that is defined by the input $\underline{u}$. For the description of the entire input space multiple of these local descriptions are superimposed.

Suitable *radial basis functions* θ have the following properties:

- The functional values are non-negative.
- It has a maximum value in the centroid point $\underline{u}_0$.
- With increasing distance to the centroid $\underline{u}_0$ the functional values are monotonically decreasing.

The following figure shows two commonly used radial basis functions. In the upper part the one-dimensional case is shown. The bell-shaped Gaussian function

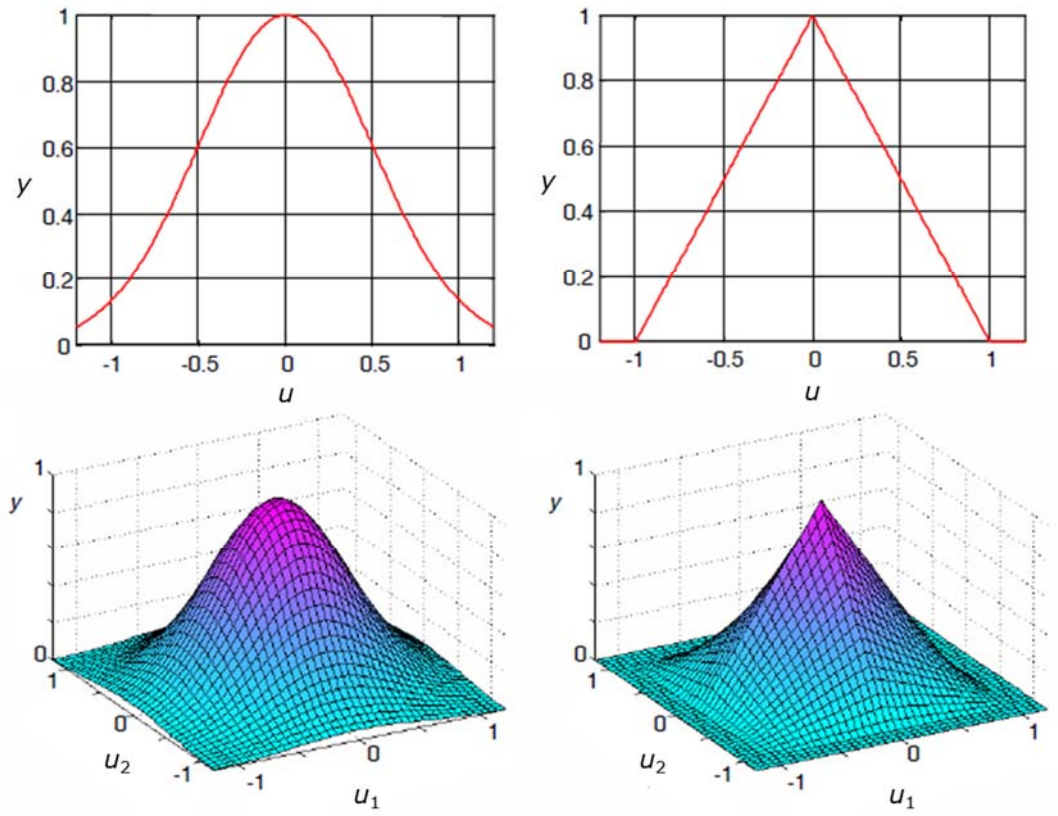
$$\hat{y} = \theta(u) = e^{-\frac{1}{2}\nu^2} \quad \text{with} \quad \nu = \frac{1}{\sigma}(u - u_0)$$

is continuously differentiable, whereas the triangular function

$$\hat{y} = \theta(u) = \begin{cases} 1 - \dfrac{1}{2\sigma}(u - u_0) & \text{with } u_0 \le u \le u_0 + 2\sigma \\ 1 + \dfrac{1}{2\sigma}(u - u_0) & \text{with für } u_0 - 2\sigma \le u \le u_0 \\ 0 & \text{otherwise} \end{cases}$$

is not differentiable in the corner points. In both functions, parameter σ defines the width of the basis function.

In the lower part of the figure the extension for the two-dimensional case is plotted for both radial basis functions.

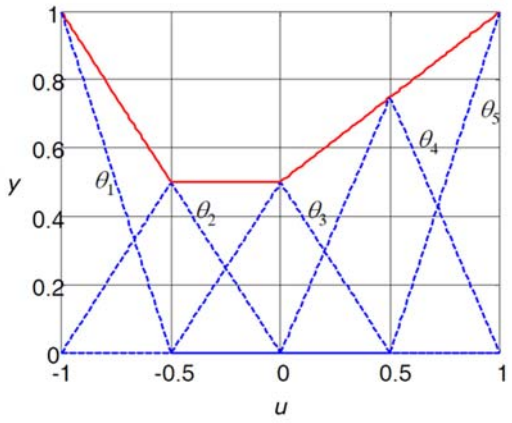


In order to approximate a functional relation, multiple radial basis functions θ_i with $i = 1,\ldots,n$ will be superimposed. Each basis function has a different centroid $\underline{u}_{0,i}$ and is scaled with a weighing factor a_i:

$$\hat{y} = \sum_{i=1}^{n} a_i \cdot \theta_i(\underline{u})$$

This approach is able to describe a nonlinear function relations between the input $\underline{u}$ and the output $\hat{y}$. Similar to the polynomial approach in section 3.2 the model output $\hat{y}$ is a linear function of the model parameters.

As an example, the following figure shows the superposition of a function (red line) by five triangular radial basis functions (dashed blue lines).



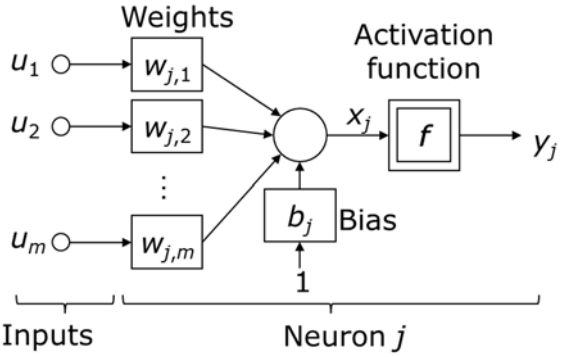
3.4 Neural networks

Artificial neural networks are an abstraction of biological neural networks. They are used to obtain a parallel and meshed approach to the modeling of technical processes. This architecture can also be implemented and evaluated in parallel, which can be advantageous if subsystems fail. Then, the model behaves more robust such that the basic function remains available in most cases, although the overall performance is reduced.

In the following, a very common network architecture is introduced, in which the *neurons* are arranged in *layers* that do not of any feedbacks. It is called *multi layer perceptron (MLP) or feed forward network.*

Neurons

The inputs $u_1,\ldots,u_m$ at the neuron j are weighed with the network weights $w_{j,1},\ldots,w_{j,m}$. Together with the bias b_j they are summed up to the signal x_j, which is the input of the activation function f. [see also Matlab, Neural Network Toolbox, User's Guide]:



The function f can be a nonlinear function, in many cases functions are used that model a threshold as

$$f(x) = \tanh(x) \quad \text{oder} \quad f(x) = \frac{1}{1+e^{-x}}.$$

Especially in the last layer of a network a linear function $f(x) = x$ may be used. Introducing an input vector $\underline{u} = [u_1 \quad \dots \quad u_m]^T$ and a vector of weights $\underline{w}_j = [w_{j,1} \quad \dots \quad w_{j,m}]^T$ this relation can be written as a compact equation:

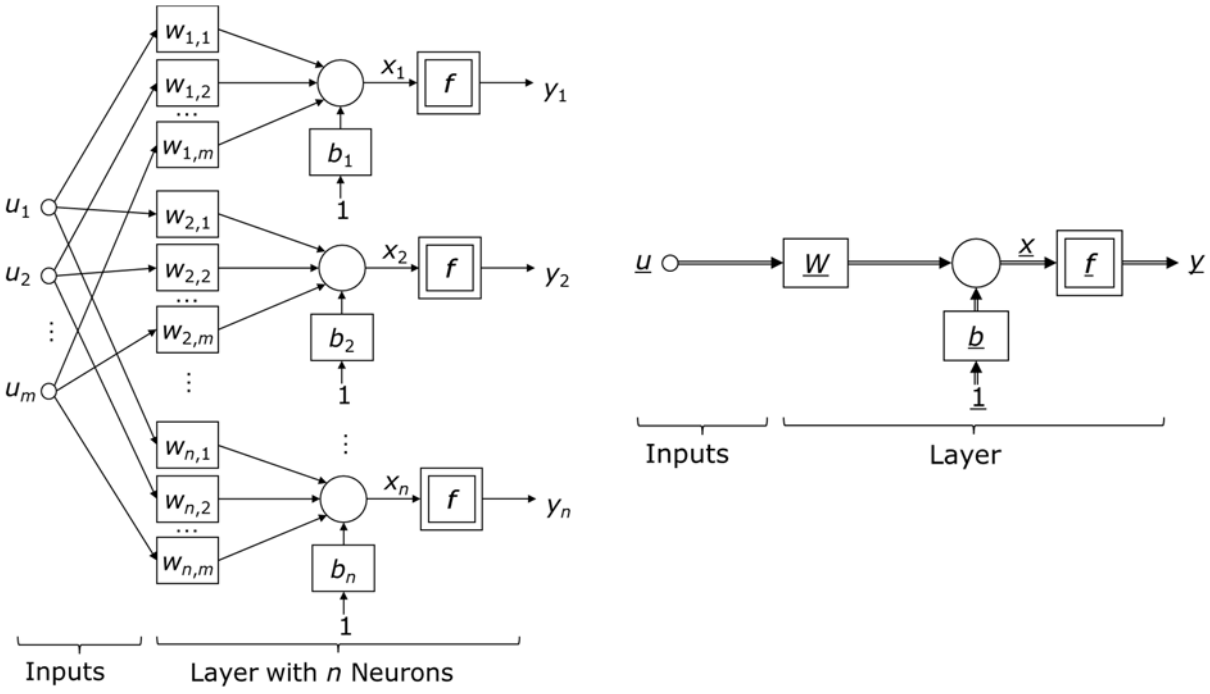
$$y_j = f(\underline{w}_j^T \underline{u} + b_j)$$

Layers

A layer is built up out of n neurons in parallel. With the bias vector $\underline{b} = [b_1 \quad \dots \quad b_n]^T$ and the matrix of network weights $\underline{W} = [\underline{w}_1^T \quad \dots \quad w_n^T]^T$ the functional relation for the vector of outputs $\underline{y} = [y_1 \quad \dots \quad y_n]^T$ can be written as:

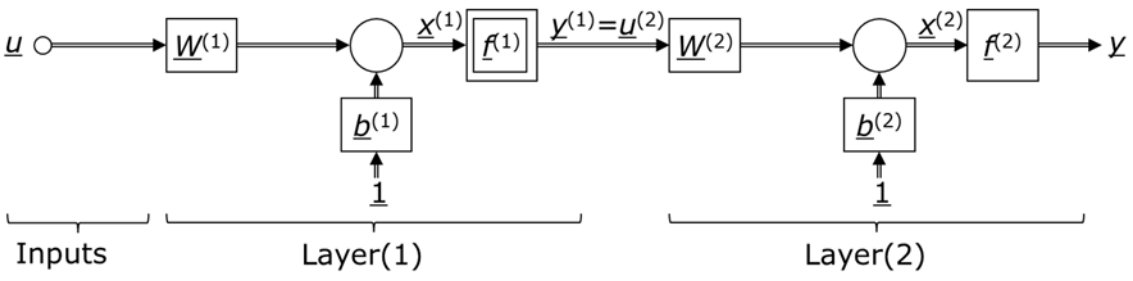
$$\underline{y} = \underline{f}(\underline{W}\,\underline{u} + \underline{b})$$

The following figures show the block diagram of one layer – on the right hand side using n single neurons, on the right hand side using vector signals:



Multi Layer Perceptron (MLP)

A typical MLP consists of two layers: For the first layer a nonlinear activation function $\underline{f}^{(1)}$, for the second layer a linear activation function $\underline{f}^{(2)}$ is used:



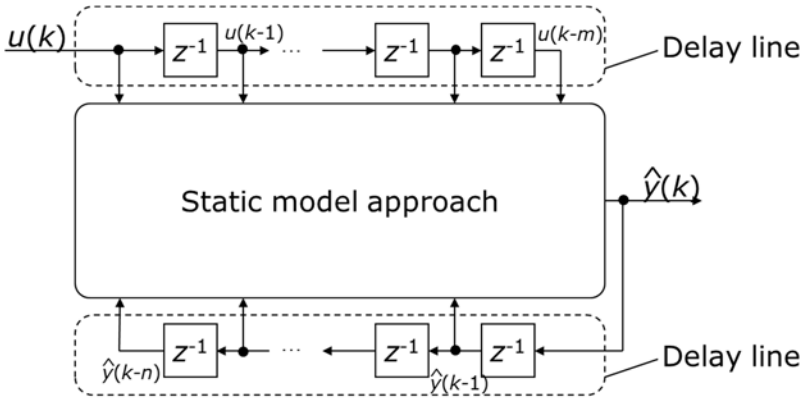
The corresponding functional relation is:

$$\underline{y} = \underline{f}^{(2)}\left[\underline{W}^{(2)}\,\underline{f}^{(1)}(\underline{W}^{(1)}\,\underline{u} + \underline{b}^{(1)}) + \underline{b}^{(2)}\right]$$

Parameters of this model approach are the network weights $\underline{W}^{(\cdot)}$ and the bias values $\underline{b}^{(\cdot)}$. They can be obtained from a nonlinear parameter optimization that will be introduced in the next chapter 5.

3.5 Extension to dynamical models

The model approaches presented in this section are static approaches. In order to characterize dynamical systems, external dynamics (as a "memory") has to be added. For time-discrete systems this can be realized by storing passed values of the input $u(k)$ and the output $y(k)$ in a delay line:



Hence, the static model approaches can be used again. But, the length of the delay lines n and m as well as the sample time ΔT have to be defined in an appropriate way, such that the dynamical properties are included and the number of $(n+m+1)$ inputs is not too high.

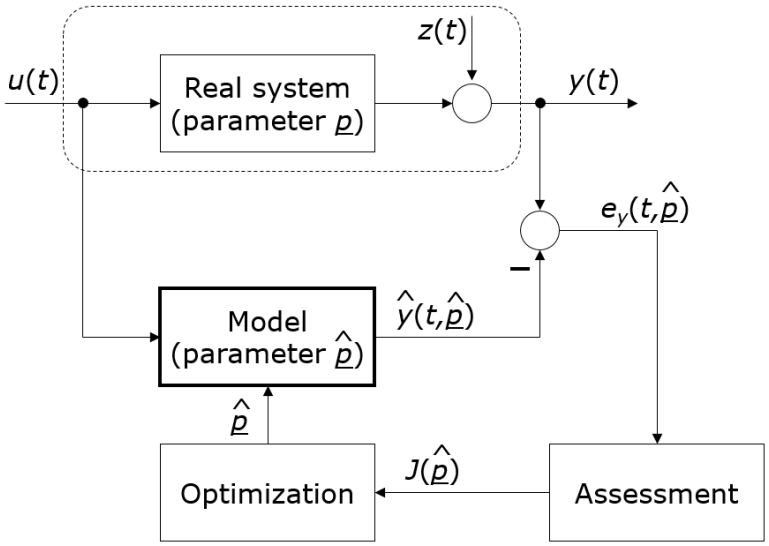
4 Parameter identification

4.1 Problem definition

The real system is defined by a set of parameter, which is summarized in a parameter vector $\underline{p}$. For the identification of this set of parameter, a model is built in parallel to the real system that uses an estimation $\underline{\hat{p}}$ of these parameters. The measured values $y(t)$ of the real system is compared with the output $\hat{y}(t, \underline{\hat{p}})$ of the model and the *output error* $e_y(t, \underline{\hat{p}})$ is determined.

An *object function* $J(\underline{\hat{p}})$ will assess the output error within a time interval. Usually the object function is defined such that its value will increase, if the output error rises. Now, optimization methods are required in order to minimize the objective function $J(\underline{\hat{p}})$ as a function of the model parameters $\underline{\hat{p}}$ and to adapt the model behavior to the real system. Hence, the problem of parameter identification is converted into an optimization problem.

Block diagram:



In principle the objective function can be defined independently in different ways. In most cases the squared output error is integrated over a time interval T:

$$J(\underline{\hat{p}}) := \int_0^T e^2(t, \underline{\hat{p}})\, dt$$

4.2 Direct optimization

Linear parameter estimation

It is assumed that the output *y* is a linear function of the unknown parameter vector $\underline{x}_p$:

$$y(t, \underline{p}) = \underline{m}^T(t) \cdot \underline{p}$$

The model is defined in the same way:

$$\hat{y}(t, \underline{\hat{p}}) = \underline{m}^T(t) \cdot \underline{\hat{p}}$$

Hence, the output error between the model $\hat{y}$ and the real measurement *y* is:

$$e_y(t, \underline{\hat{p}}) = y(t) - \hat{y}(t, \underline{\hat{p}}) = y(t) - \underline{m}^T(t) \cdot \underline{\hat{p}}$$

A set of measurements with $k = 1,...,N$ is obtained and summarized as vectors:

$$\underline{e}_y(\underline{\hat{p}}) = \begin{bmatrix} e(t_1, \underline{\hat{p}}) \\ \vdots \\ e(t_N, \underline{\hat{p}}) \end{bmatrix}, \quad \underline{y} = \begin{bmatrix} y(t_1) \\ \vdots \\ y(t_N) \end{bmatrix}, \quad \underline{M} = \begin{bmatrix} \underline{m}^T(t_1) \\ \vdots \\ \underline{m}^T(t_N) \end{bmatrix}$$

Then, we can write the output error for all *N* measurements as:

$$\underline{e}_y(\underline{\hat{p}}) = \underline{y} - \underline{M} \cdot \underline{\hat{p}}$$

As objective function *J* the sum of the squared errors

$$J(\underline{\hat{p}}) = \underline{e}_y^T(\underline{\hat{p}}) \cdot \underline{e}_y(\underline{\hat{p}})$$

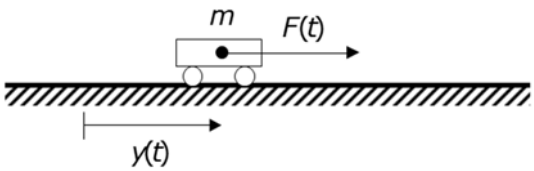
has to be minimized. It is also called "least square error" (Carl Friedrich Gauß: „Methode der kleinsten Quadrate“).

The solution for the optimal parameter vector $\underline{\hat{p}}$ is:

$$\underline{\hat{p}} = \left(\underline{M}^T \cdot \underline{M}\right)^{-1} \underline{M}^T \cdot \underline{y}$$

The system will be called *identifiable*, if $\left(\underline{M}^T \cdot \underline{M}\right)$ is regular. If this is not the case, it might be that not enough measurements or only very similar measurements are available.

Example: Parameter identification of a vehicle with constant acceleration



A vehicle with an unknown mass m will be accelerated with a constant force F_0 starting at $t = 0$:

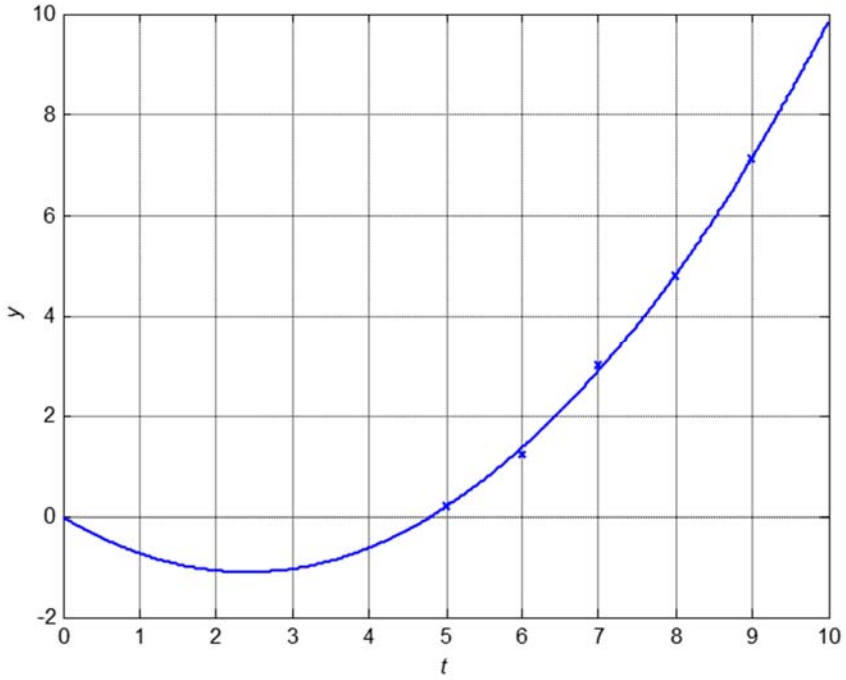
$$F(t) = F_0 \cdot \sigma(t) \quad \text{with} \quad F_0 = 4$$

At the times t_k the vehicle position $y(t_k)$ is measured:

k	1	2	3	4	5
t_k	5,0	6,0	7,0	8,0	9,0
$y(t_k)$	0,2	1,2	3,0	4,8	7,1

For modeling Newton's law $F(t) = m \cdot \ddot{y}(t)$ can be applied. Initially the vehicle is located in $y(0) = 0$. The unknown mass m and the unknown initial vehicle velocity $\dot{y}(0) = v_0$ should be determined by parameter identification.

In the figure below the model output $\hat{y}$ (as solid line) and the measured values (as cross-marks) are plotted.



4.3 Iterative Optimization

Simplex search (from Nelder and Mead)

This optimization method does not require any information about the derivatives of the objective function $J(\underline{p})$. Therefore, it is especially suitable for discontinuous and nonlinear functions. But, it is less effective than gradient methods.

In Matlab the function `fminsearch` implements this method.

The *n* components of the parameter vector $\underline{p} = [p_1,..., p_n]^T$, which should be determined by minimizing the objective function $J(\underline{p})$, span an *n*-dimensional vector space. In this parameter vector space a simplex *S* with *n*+1 corners is defined by a set of corner points $\underline{p}_1$, …, $\underline{p}_{n+1}$:

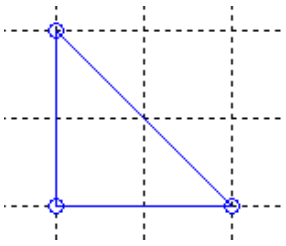
$$S = \{\underline{p}_1,..., \underline{p}_n, \underline{p}_{n+1}\}$$

In the case *n* = 2 it is a triangle, in case of *n* = 3 it is a triangular pyramid.

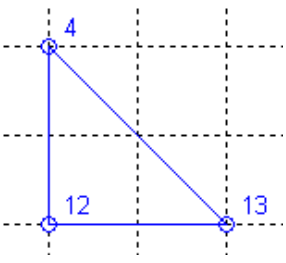
Applying heuristic rules the location of the simplex in the parameter vector space is modified, such that the simplex will finally contract in a minimum.

Algorithms:

(1) Choose the corner points of the initial simplex $S^{(0)}$. The supposed parameter vector should be included in the simplex volume.



(2) Calculate the objective function values for all corner points $J(\underline{p}_1),..., J(\underline{p}_{n+1})$.



(3) Find the corner points $\underline{p}_{\max}$ and $\underline{p}_{\min}$ with the maximum and the minimum objective function values, respectively: $J_{\max} = \max_i J(\underline{p}_i)$ and $J_{\min} = \min_i J(\underline{p}_i)$

(4) Calculate the centroid of all corner points excluded $\underline{p}_{\max}$:

$$\underline{p}_C = \frac{1}{n} \cdot \sum_{\substack{i=1 \\ i \neq \max}}^{n+1} \underline{p}_i$$

(5) Try **reflection**: Mirror $\underline{p}_{\max}$ to a new position:

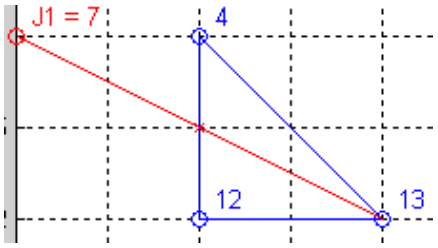
$\hat{\underline{p}} = \underline{p}_C + \alpha \cdot (\underline{p}_C - \underline{p}_{\max})$ with $\alpha = 1$

and determine the objective function value $J(\hat{\underline{p}})$.

If $J(\hat{\underline{p}}) < J_{\min}$: continue with (6) expansion.

If $J(\hat{\underline{p}}) > J_{\max}$: continue with (7) contraction.

Otherwise: use $\hat{\underline{p}}$ instead of $\underline{p}_{\max}$ as new corner point and continue with (8).



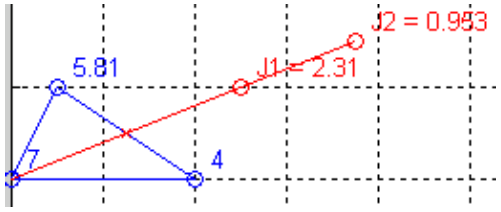
(6) Try **expansion**: Expend $\hat{\underline{p}}$ to a new position:

$\tilde{\underline{p}} = \underline{p}_C + \gamma \cdot (\hat{\underline{p}} - \underline{p}_C)$ with $\gamma = 2$

and determine the objective function value $J(\tilde{\underline{p}})$.

If $J(\tilde{\underline{p}}) < J_{\min}$: take $\tilde{\underline{p}}$ instead of $\underline{p}_{\max}$ as new corner point and continue with (8).

Otherwise: take $\hat{\underline{p}}$ instead of $\underline{p}_{\max}$ as new corner point and continue with (8).



(7) Try **contraction**: Reduce $\underline{p}_{\max}$ to a new position:

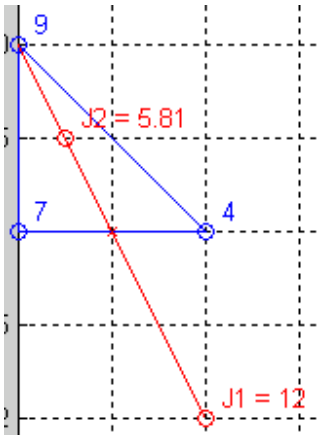
$\tilde{\underline{p}}_p = \underline{p}_C + \beta \cdot (\underline{p}_{\max} - \underline{p}_C)$ with $\beta = 0.5$

and determine the objective function value $J(\tilde{\underline{p}})$.

If $J(\tilde{\underline{p}}) < \min(J(\hat{\underline{p}}), J_{\min})$: take $\tilde{\underline{p}}$ instead of $\underline{p}_{\max}$ as new corner point.

Otherwise: reduce simplex

$\underline{p}_i = (\underline{p}_i - \underline{p}_{\min})/2 + \underline{p}_{\min}$ for all corner points *i*.



(8) Stop, if minimum simplex volume has been reached.
Otherwise: continue with (3).

Line search

General structure of the iterative optimization algorithm:

(1) Find an initial parameter vector $\underline{p}^{(0)}$

(2) Calculate the direction of descent $\underline{\eta}^{(k)}$

(3) Calculate the step size $\alpha^{(k)}$

(4) Iteration step: $\underline{p}^{(k+1)} = \underline{p}^{(k)} + \alpha^{(k)} \cdot \underline{\eta}^{(k)}$

(5) Stop, if $\left\| \underline{\eta}^{(k)} \right\| \leq \varepsilon$

Otherwise: continue with step (2)

If the initial parameter vector $\underline{p}^{(0)}$ is defined disadvantageously, it may happen that the algorithm does not find the global minimum, but only a local minimum of the objective function $J(\underline{p})$.

Directions of descent:

1. Negative gradient: $$\underline{\eta} = -\frac{\partial J(\underline{p})}{\partial \underline{p}}$$

with the elements of the gradient vector $\eta_i = -\frac{\partial J(\underline{p})}{\partial p_i}$ for i = 1,..,n components in in $\underline{p}$. The negative gradient is a confident direction of descent.

2. Newton direction: $$\underline{\eta} = -\left[\underline{H}(\underline{p})\right]^{-1} \frac{\partial J(\underline{p})}{\partial \underline{p}}$$

with the elements of the Hesse matrix: $H_{i,j}(\underline{p}) = \frac{\partial J(\underline{p})}{\partial p_i \, \partial p_j}$ for i = 1,..,n and j = 1,..,n

The Newton direction is only a confident direction of descent, if the Hesse matrix is positive definite: $\underline{H} > 0$.

3. Modified Newton direction: $$\underline{\eta} = -\left[\underline{H}(\underline{p}) + \delta \, \underline{I}\right]^{-1} \frac{\partial J(\underline{p})}{\partial \underline{p}}$$

The expression $\delta \underline{I}$ is added to the Hesse matrix. The constant $\delta > 0$ has to be chosen such that $\left[\underline{H}(\underline{p}) + \delta \, \underline{I}\right] > 0$ yields and a confident descent is obtained.

Step size control:

1. Optimal step size: This is again a (1-dimensional) optimization problem:

$$J\left(\underline{p}^{(k)} + \alpha^{(k)} \cdot \underline{\eta}^{(k)}\right) \rightarrow \min$$

3. Armijo rule: Choose $\alpha^{(k)} = \beta^m$, with m as the smallest integer value that holds the inequality

$$J\left(\underline{p}^{(k)}\right) - J\left(\underline{p}^{(k)} + \beta^m \cdot \underline{\eta}^{(k)}\right) \geq \sigma \cdot \beta^m \left[\left.\frac{\partial J(\underline{p})}{\partial \underline{p}}\right|_{\underline{p}^{(k)}}\right]^T \cdot \left(-\underline{\eta}^{(k)}\right)$$

(For $m-1$ the inequality is not fulfilled.)

Start with $m = 0$ and decrease, if the inequality is fulfilled. Or increase, if the inequality is not fulfilled.

Set $\beta = 1/4$ and $\sigma = 1/10$.

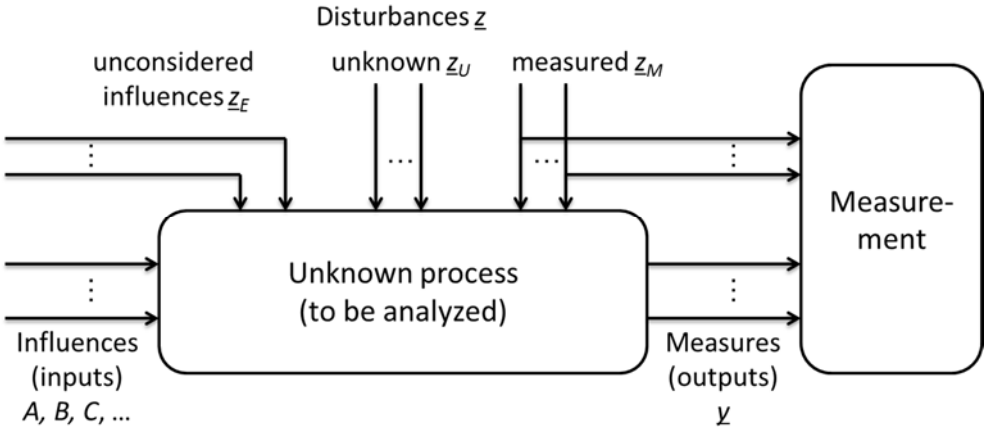
5 Design of experiment

5.1 Introduction

In order to characterize an unknown process, an experimental analysis of the system can be performed (also in parallel to simulative or theoretical studies). Methods for the *Design of Experiment (DoE)* have been developed, in order to provide an efficient experimental setup and analysis.

System point of view

Measures (outputs) $\underline{y}$ of the unknown system characterizes the performance or quality of the process or its products. We would like to analyze the impact of the influences (inputs) *A*, *B*, *C*, … to the output $\underline{y}$. All remaining signals are considered as disturbances $\underline{z}$. They include influences that were excluded as an input $\underline{z}_E$, unknown disturbances $\underline{z}_U$, and disturbances that could be measured $\underline{z}_M$.



Principles

In the following basic principles of an experimental analysis are summarized:

- *Replication*: In order to reduce the impact of unknown random disturbances, each experiment should be repeated for a number of times *m*. As a result one can use the mean value

$$\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$$

  as result. Its variance will decrease with

$$\sigma\{\bar{y}\} = \frac{1}{\sqrt{m}} \cdot \sigma \quad \text{and} \quad \sigma^2 = \frac{1}{m-1} \cdot \sum_{i=1}^{m} \left(y_i - \bar{y}\right)^2 .$$

- *Randomization*: The order of experiments should be randomized in order to avoid confounded effects.
- *Blocking*: Experiments that could be performed within constant experimental constraints (e.g. constant measurable disturbances $\underline{z}_M$) should be analyzed within separated groups (blocks).

Procedure

The experimental procedure should be divided into the following steps:

1. System analysis: What is the objective of the analysis? Which are the inputs, outputs and disturbances (according to the block diagram above)? What are expected impacts and interactions?
2. Experimental design
3. Execution of experiments
4. Experimental analysis

Each step should be completed, before the next step is started! A *sequential analysis* may be useful, e.g., a first run as a screening experiment and a second run for main analysis.

5.2 Experimental Design

For each influence *A*, *B*, *C*, … a number of steps has to be defined. The simplest and most common case is $s = 2$ steps, which will lead to a linear experimental design.

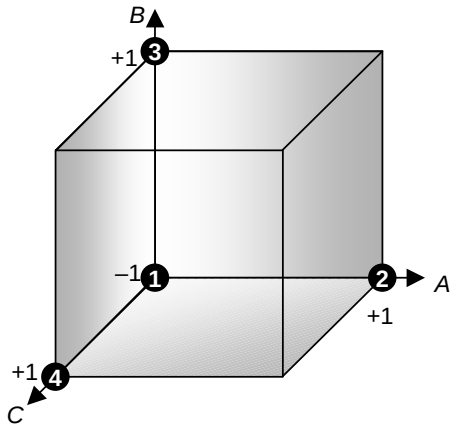
The following table gives an example of the choice of $s = 2$ steps for a turning process with three influences *A*, *B*, and *C*.

Influences (inputs)		Original steps		Normalized steps	
A	Cutting depth a_e	0,5 mm	1,0 mm	– 1	+ 1
B	Feed *f*	0,05 mm	0,1 mm	– 1	+ 1
C	Cutting speed v_c	2 000 min^{-1}	3 000 min^{-1}	– 1	+ 1

For the subsequent analysis these steps were normalized, e.g., to "–1" and "+1" in case of two steps.

Single factorial design

The single factorial design is a straight forward approach: Starting with an initial experiment (all steps in "–1") the step of each influence is changed to "+1", whereas the steps of all other steps remain in "–1".

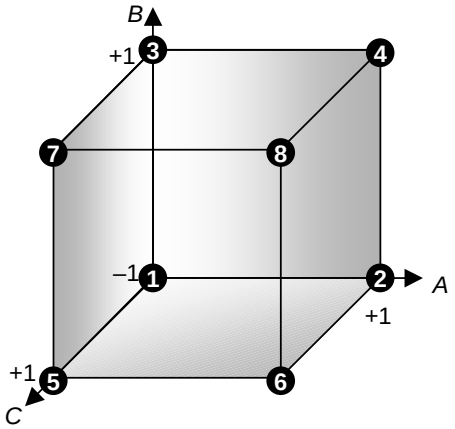


Number of experiment	Choice of steps		
	A	*B*	*C*
1	–1	–1	–1
2	+1	–1	–1
3	–1	+1	–1
4	–1	–1	+1

The number of experiments grows only linear with the number of influences. But, no interactions between influences can be analyzed. Therefore, this design may only be used for screening experiments.

Full factorial design

This design covers all possible combinations of the steps of the inputs. In the following case of 3 inputs with 2 steps each we have to execute 8 experiments. It is called 2^3 design.



Number of experiment	Choice of steps		
	A	*B*	*C*
1	−1	−1	−1
2	+1	−1	−1
3	−1	+1	−1
4	+1	+1	−1
5	−1	−1	+1
6	+1	−1	+1
7	−1	+1	+1
8	+1	+1	+1

All possible interactions can be determined. The following table shows the main effects (ME) of the inputs *A*, *B*, *C*, their second-order interactions (2 IA) *AB*, *BC*, *AC* and their third-order interaction (3 IA) *ABC*. The normalized steps of the interactions can be calculated formally by multiplying the steps of the corresponding main effects row by row.

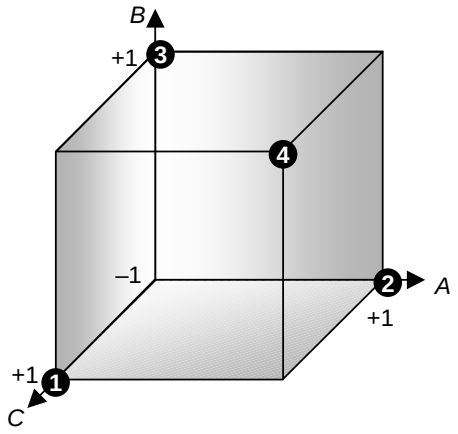
Number of experiment	Choice of steps			Possible interactions			
	A (ME)	*B* (ME)	*C* (ME)	*AB* (2 IA)	*BC* (2 IA)	*AC* (2 IA)	*ABC* (3 IA)
1	−1	−1	−1	+1	+1	+1	−1
2	+1	−1	−1	−1	+1	−1	+1
3	−1	+1	−1	−1	−1	+1	+1
4	+1	+1	−1	+1	−1	−1	−1
5	−1	−1	+1	+1	−1	−1	+1
6	+1	−1	+1	−1	−1	+1	−1
7	−1	+1	+1	−1	+1	−1	−1
8	+1	+1	+1	+1	+1	+1	+1

The disadvantage of this design is the “exploding” number of experiments that have to be executed for a growing number of influences.

Fractional factorial design

If we know that an interaction between main effects is zero (from additional insights regarding the process), we could use this information to reduce the number of experiments. Starting with a full factorial design a new influence is introduced that has the same profile of steps of this interaction. The following example starts with a 2^2 design, then *C* is introduced as interaction *AB*, and we receive a 2^{3-1} fractional design.

Number of experiment	Choice of steps		Interaction	New Influence
	A (Me)	*B* (ME)	*AB* (2 IA)	*C* (ME)
1	–1	–1	+1	+1
2	+1	–1	–1	–1
3	–1	+1	–1	–1
4	+1	+1	+1	+1

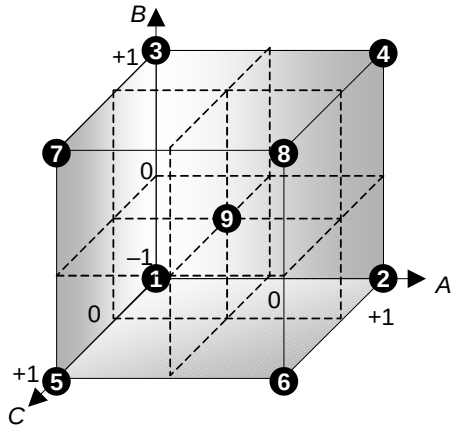


For larger designs the situation will even improve. The following table shows a 2^3 design. To introduce a new influence *D*, now only a third-order interaction *ABC* has to be replaced. A combined impact of *A*, *B*, *C* is very rare, therefore, the assumption that the interaction *ABC* is zero will be true in most cases.

Number of experiment	Choice of steps			Possible interactions			
	A (ME)	*B* (ME)	*C* (ME)	*AB = CD* (2 IA)	*BC = AD* (2 IA)	*AC = BD* (2 IA)	*ABC=D* (3 IA, ME)
1	–1	–1	–1	+1	+1	+1	–1
2	+1	–1	–1	–1	+1	–1	+1
3	–1	+1	–1	–1	–1	+1	+1
4	+1	+1	–1	+1	–1	–1	–1
5	–1	–1	+1	+1	–1	–1	+1
6	+1	–1	+1	–1	–1	+1	–1
7	–1	+1	+1	–1	+1	–1	–1
8	+1	+1	+1	+1	+1	+1	+1

Verification of linear behavior

With two steps (“–1” and “+1”) only a linear relation between inputs and output can be covered, whereas the real process relation may be nonlinear. To verify the assumption of linearity, we can add one further experiment at the normalized step “0”. If the output at this point is equal to the mean of the steps “–1” and “+1” for all inputs, the linear design is justified, otherwise a higher order design ($s>2$) should be considered.



5.3 Analysis

Up to now, we looked at the inputs (influences) and discussed an efficient design of experiment in preparation to the experiment. For analysis after the experiment has been completed we will consider now also the output (measurement).

Linear effects of inputs and interactions

Action A is considered as the change of the influence from step "−1" to "+1". Then, the *linear effect* of action *A* is the resulting chance of the output *y.* If $\underline{A}$ is the list of normalized steps of the input, $\underline{y}$ the list of corresponding measurements and *N* the number of experiments the linear effect if *A* is determined as:

$$e(A) = \frac{2}{N} \cdot \underline{A}^T \cdot \underline{y}$$

Resuming the example of the cutting process the depth of roughness could be chosen as measurement, which is added in the following table:

Number of experiment	Choice of steps		Interaction	Measurement
	A (Cutting depth)	*B* (Feed)	*AB*	*y* (depth of roughness in µm)
1	−1	−1	+1	2.8
2	+1	−1	−1	3.5
3	−1	+1	−1	3.0
4	+1	+1	+1	5.5

For action *A* the linear effect is $e(A)$ = 1,6 µm; and for action *B* it yields $e(B)$ = 1,1 µm. This means that the depth of roughness increases for both actions. If the linear effect would be zero, there would be no impact of the input; a negative value would describe a decreasing depth of roughness as consequence of an action.

The linear effect of an interaction is calculated in a similar way. In the example the linear effect of the interaction *AB* is

$$e(AB) = \frac{2}{N} \cdot \left(\underline{A}. * \underline{B}\right)^T \cdot \underline{y} = 0{,}9\,\mu\text{m}\,.$$

A positive linear effect of the interaction means that simultaneous actions *A* and *B* have an additional positive impact on the output that cannot be assigned to the separate linear effects of *A* or *B*.

If a third influence *C* would be introduced on *AB*, this would lead to misinterpretations, because $e(AB)$ is not close to zero.

Significances of linear effects

The value of the linear effects gives no information on its relevance. The *significance* of effects can be assessed using statistical tests.

Within one column all experiments with step value „–1“ are considered as control group, all experiments with „+1“ are part of the experimental group (assuming a linear design of experiments with s = 2 steps). To prove significance with have to look at the difference of means between these two groups and put this into relation to the variance S_i^2 of the measurement i (with i = 1,…, N measurements). To determine S_i^2 the experiment in one row of the plan will be repeated m times.

Calculation of significances

The difference of the means between the experimental and the control group is the *linear effect* as introduced before

$$e(A) = \frac{2}{N} \cdot \underline{A}^T \cdot \underline{y} \qquad (N\text{: Number of experiments})$$

and $e(A)$, $e(B)$, … can be taken from the corresponding columns in the table.

The total variance is:

$$S^2 = \frac{4}{m \cdot N^2} \sum_{i=1}^{N} S_i^2 \quad (m\text{: Number of repetitions within one experiment})$$

To ably the t-test, the relation of e(A) and S will be compared the threshold that can be taken from a table (see page 5-8). The effect of action A is significant with the probability p_{twosided} of significance, if

$$\left| \frac{e(A)}{S} \right| > t_{\text{tab}}(p_{\text{twosided}}, \nu)$$

yields. Therein, ν is the degree of freedom with

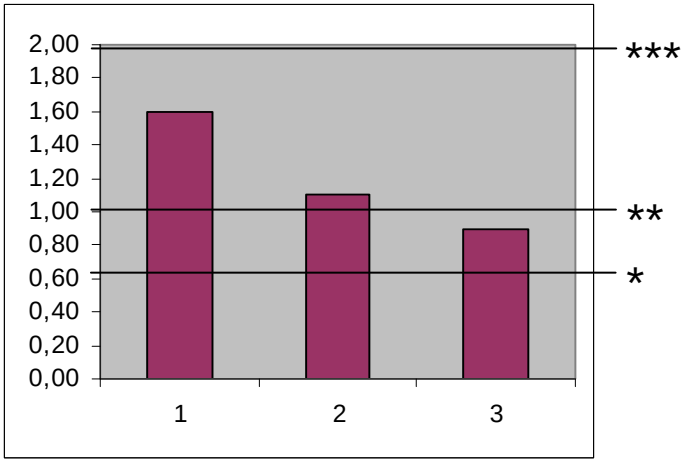
$$\nu = N \cdot (m-1)$$

For illustration we could expend the inequality above by the denominator S and introduce an effect threshold $\pm t_{\text{tab}}(p_{\text{twosided}}, \nu) \cdot S$, which must be exceeded by the absolute value of $e(A)$ to be significant.

Example: Turning process (continued)

(Number of repetitions is $m = 2$)

Number of experiment	Choice of steps		Interaction			Mean	Variance
	A	B	AB	$y_{i,1}$	$y_{i,2}$	$\overline{y}_i$	S_i^2
1	−1	−1	+1	2.6	3.0	2,8	0.08
2	+1	−1	−1	3.4	3.6	3,5	0.02
3	−1	+1	−1	3.0	3.0	3,0	0
4	+1	+1	+1	5.1	5.9	5,5	0.32
	$e(A)$=1.6	$e(B)$=1.1	$e(AB)$=0.9				Sum = 0.42
Significance	**	**	*				



Calculation:

- Total variance: $S^2 = \frac{4}{2 \cdot 4^2} \cdot 0.42 = 0.053$ and standard deviation: $S = 0.229$
- Degree of freedom: $\nu = 4 \cdot (2-1) = 4$
- Effect thresholds:
  (weakly) significant * : $t_{\text{tab}}(\nu = 4, p_{\text{towsided}} = 0.95) \cdot S = 2.776 \cdot 0.229 = 0.636$
  significant ** : $t_{\text{tab}}(\nu = 4, p_{\text{towsided}} = 0.99) \cdot S = 4.604 \cdot 0.229 = 1.055$
  highly significant *** : $t_{\text{tab}}(\nu = 4, p_{\text{twosided}} = 0.999) \cdot S = 8.610 \cdot 0.229 = 1.972$
- Thresholds can be introduced into the bar diagram and the significances can be classified

Percentiles of the Student (t) distribution $t_{tab}(p,\nu)$:

	singlesided: p =									
	0,6	0,75	0,90	0,95 *	0,975	0,99 **	0,995	0,9975	0,999 ***	0,9995
	twosided:									
	0,3	0,50	0,80	0,90	0,95 *	0,98	0,99 **	0,995	0,998	0,999 ***
ν										
1	.325	1.000	3.078	6.314	12.706	31.821	63.657	127.32	318.31	636.62
2	.289	.816	1.886	2.920	4.303	6.965	9.925	14.089	23.326	31.598
3	.277	.765	1.638	2.353	3.182	4.541	5.841	7.453	10.213	12.924
4	.271	.741	1.533	2.132	2.776	3.747	4.604	5.598	7.173	8.610
5	.267	.727	1.476	2.015	2.571	3.365	4.032	5.773	5.893	6.869
6	.265	.718	1.440	1.943	2.447	3.143	3.707	4.317	5.208	5.959
7	.263	.711	1.415	1.895	2.365	2.998	3.499	4.029	4.785	5.408
8	.262	.706	1.397	1.860	2.306	2.896	3.355	3.833	4.501	5.041
9	.261	.703	1.383	1.833	2.262	2.812	3.250	3.690	4.297	4.781
10	.260	.700	1.372	1.812	2.228	2.764	3.169	3.581	4.144	4.587
11	.260	.697	1.363	1.796	2.201	2.718	3.106	3.497	4.025	4.437
12	.259	.695	1.356	1.782	2.179	2.681	3.055	3.428	3.930	4.318
13	.259	.694	1.350	1.771	2.160	2.650	3.012	3.372	3.852	4.221
14	.258	.692	1.345	1.761	2.145	2.624	2.977	3.326	3.787	4.140
15	.258	.691	1.341	1.753	2.131	2.602	2.947	3.286	3.733	4.073
16	.258	.690	1.337	1.746	2.120	2.583	2.921	3.252	3.686	4.015
17	.257	.689	1.333	1.740	2.110	2.567	2.898	3.222	3.646	3.965
18	.257	.688	1.330	1.734	2.101	2.552	2.878	3.197	3.610	3.922
19	.257	.688	1.328	1.729	2.093	2.539	2.861	3.174	3.579	3.883
20	.257	.687	1.325	1.725	2.086	2.528	2.845	3.153	3.552	3.850
21	.257	.686	1.323	1.721	2.080	2.518	2.831	3.135	3.527	3.819
22	.256	.686	1.321	1.717	2.074	2.508	2.819	3.119	3.505	3.792
23	.256	.685	1.319	1.714	2.069	2.500	2.807	3.104	3.485	3.767
24	.256	.685	1.318	1.711	2.064	2.492	2.797	3.091	3.467	3.745
25	.256	.684	1.316	1.708	2.060	2.485	2.787	3.078	3.450	3.725
26	.256	.684	1.315	1.706	2.056	2.479	2.779	3.067	3.435	3.707
27	.256	.684	1.314	1.703	2.052	2.473	2.771	3.057	3.421	3.690
28	.256	.683	1.313	1.701	2.048	2.467	2.763	3.047	3.408	3.674
29	.256	.683	1.311	1.699	2.045	2.462	2.756	3.038	3.396	3.659
30	.256	.683	1.310	1.697	2.042	2.457	2.750	3.030	3.385	3.646
40	.255	.681	1.303	1.684	2.021	2.423	2.704	2.971	3.307	3.551
60	.254	.679	1.296	1.671	2.000	2.390	2.660	2.915	3.232	3.460
120	.254	.677	1.289	1.658	1.980	2.358	2.617	2.860	3.160	3.373
200	.254	.676	1.286	1.653	1.972	2.345	2.601	2.839	3.131	3.340
∞	.253	.674	1.282	1.645	1.960	2.326	2.576	2.807	3.090	3.291

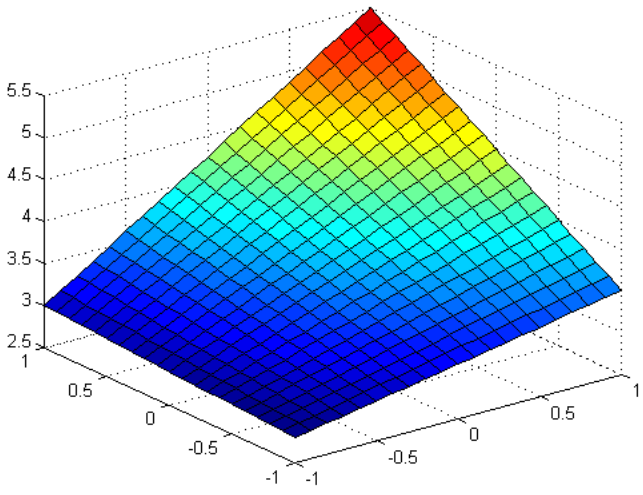
alternatively in Excel: = `TINV(1-p; nu)`

Modeling

For further analysis we can setup a model, that characterizes the impact of the influences *A*, *B*, *C*, … to the measurement *y*. Such a model is able to predict the output *y* also in cases of intermediate steps as *A* = 0.5. If we have already obtained the effects, the model for two inputs is

$$y(A,B) = \frac{e(A)}{2}A + \frac{e(B)}{2}B + \frac{e(AB)}{2}AB + \bar{y}\,,$$

where $\bar{y}$ is the mean of the measurement y (in the example of the turning process it is $\bar{y} = 3{,}7\mu m$). The figure shows a two dimensional plat of y(A,B).

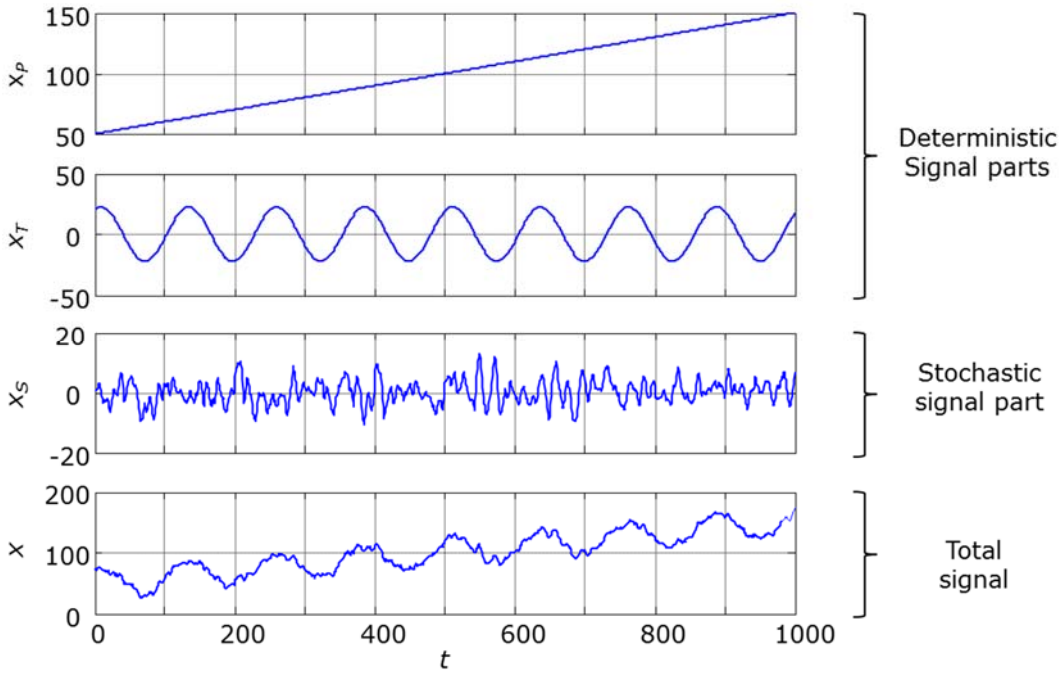


6 Identification of signal models

6.1 Introduction

A signal model is able to characterize a signal in a more comprehensive way than using its primary data. It can be applied for signal prediction, analysis and adjustment.

We assume that the measured signal is built as total signal by the superposition of different signal parts. The subsequent example shows the addition of the signal parts x_P, x_T, and x_S to the total signal x.



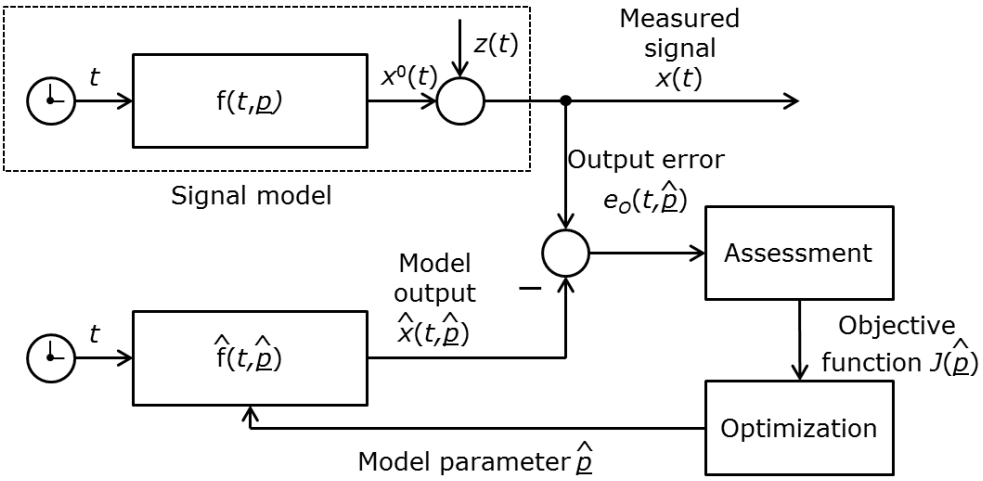
A signal model is obtained is obtained in the following steps:

1. Based on the measured signal it has to be decided, which are the relevant signal parts. For each signal part a model approach has to be defined.
2. The model parameters of these model approaches are identified from measurement.
3. The signal parts are superposed to the required total signal model.

In the following section 6.2 polynomial x_P and periodic x_T signal models are presented. In order to model stochastic signals AR, MA, and ARMA model approaches are introduced in section 6.3.

6.2 Deterministic signal models

The measured signal $x(t)$ is considered to be generated from a signal source. First, the ideal signal $x^0(t)$ is generated from function f, which is then disturbed by $z(t)$.



The signal source is modelled by the function $\hat{f}$ in parallel such that the measured signal $x(t)$ and the model output $\hat{x}(t)$ can be compared. The model parameters are summarized to the vector $\underline{\hat{p}}$. The output error is:

$$e_O(t, \underline{\hat{p}}) = x(t) - \hat{x}(t, \underline{\hat{p}})$$

Now, we have to find the parameter vector $\underline{\hat{p}}$ that minimizes the output error $e_O(t, \underline{\hat{p}})$.

Assessment

For the assessment of the output error an objective function has to be defined. In most cases a least square approach is used:

$$J(\underline{\hat{p}}) := \int_0^T e_O^2(t, \underline{\hat{p}})\, dt$$

Practically, time discrete measurements in the points $t = kT$ with k=1,…,N and sample time T are available. We introduce a new notation:

$$x(k) := x(t = kT)$$

The objective function is

$$J(\underline{\hat{p}}) := \sum_{k=1}^{N} e_O^2(k, \underline{\hat{p}}) = \underline{e}_O^T(\underline{\hat{p}}) \cdot \underline{e}_O(\underline{\hat{p}}) \text{ with } \underline{e}_O(\underline{\hat{p}}) = \begin{bmatrix} e_O(1, \underline{\hat{p}}) \\ \vdots \\ e_O(N, \underline{\hat{p}}) \end{bmatrix}$$

Furthermore, the *root mean squared error (RMSE)* can be calculated:

$$RMSE(\underline{\hat{p}}) = \sqrt{\frac{1}{N} J(\underline{\hat{p}})}$$

Optimization

A direct solution of the optimization problem can be obtained, if the model output is a linear function of the parameters:

$$\hat{x}(t, \underline{\hat{p}}) = \underline{m}^T(t) \cdot \underline{\hat{p}}$$

For all k=1,…,N measurements it is

$$\underbrace{\begin{bmatrix} e_A(1) \\ \vdots \\ e_A(N) \end{bmatrix}}_{\underline{e}_A} = \underbrace{\begin{bmatrix} x(1) \\ \vdots \\ x(N) \end{bmatrix}}_{\underline{x}} - \underbrace{\begin{bmatrix} \underline{m}^T(1) \\ \vdots \\ \underline{m}^T(N) \end{bmatrix}}_{M} \cdot \underline{\hat{p}}$$

and the direct solution is:

$$\underline{\hat{p}} = \left(\underline{M}^T \cdot \underline{M}\right)^{-1} \cdot \underline{M}^T \cdot \underline{x}$$

Polynomial model approach

It is assumed that the undisturbed signal is characterized by a polynomial time function:

$$\mathrm{f}(kT) = \sum_{i=0}^{n} a_i \, (kT)^i$$

It is a linear model approach that can be segmented as

$$\mathrm{f}(kT) = x(k, \underline{p}) = \underline{m}^T(k) \cdot \underline{p} = \begin{bmatrix} 1 & kT & \dots & (kT)^n \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

and for N measurements it yields:

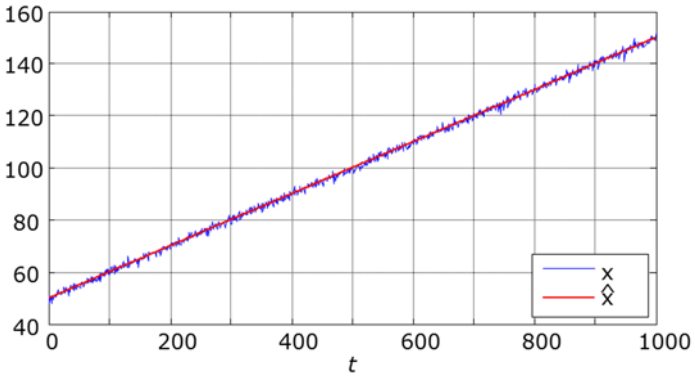
$$\underline{M} = \begin{bmatrix} 1 & 1T & \dots & (1T)^n \\ 1 & 2T & \dots & (2T)^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & NT & \dots & (NT)^n \end{bmatrix}$$

Example

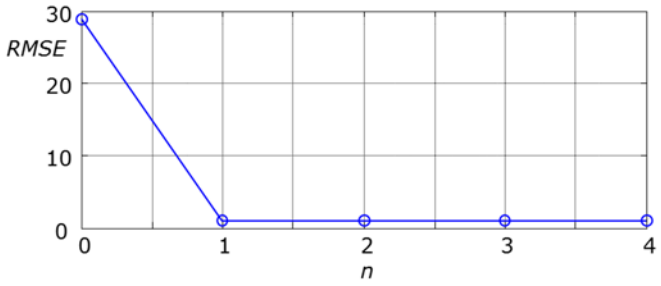
A disturbed linear function $x(k)$ with k=1,…,1000 and T=1 is captured (see diagram below). A linear polynomial approach (n=1) is identified, e.g.:

$$\underline{\hat{p}} = \begin{bmatrix} \hat{a}_0 \\ \hat{a}_1 \end{bmatrix} = \begin{bmatrix} 50.034 \\ 0.099 \end{bmatrix}.$$

In the following the model output $\hat{x}(k, \underline{\hat{p}}) = \underline{m}^T(k) \cdot \underline{\hat{p}}$ is shown.



Which model order *n* should be chosen for the model approach? In many applications it is not easy to answer this question. One suggestion is to analyze the RMSE as a function of the model order *n*. The following example shows that increments of the model order beyond *n*=1 will not reduce the RMSE:



Periodic approach

To characterize periodic signals the following model approach can be used:

$$\mathrm{f}(kT) = \sum_{i=1}^{n} \left[a_i \cos(\omega_i kT) + b_i \sin(\omega_i kT) \right]$$

The frequency ω_i has to be known. It is also a linear model approach, which can be segmented into:

$$\mathrm{f}(kT) = x(k, \underline{p}) = \underline{m}^T(k) \cdot \underline{p} = \left[\cos(\omega_1 kT) \quad \dots \quad \cos(\omega_n kT) \;\vdots\; \sin(\omega_1 kT) \quad \dots \quad \sin(\omega_n kT) \right] \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_n \\ \hdashline b_1 \\ \vdots \\ b_n \end{bmatrix}$$

For all measurements it is:

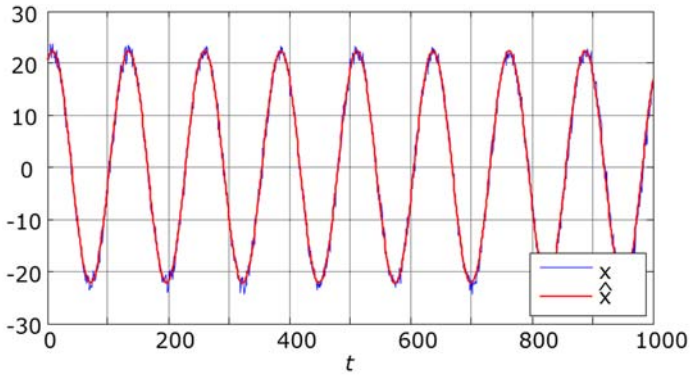
$$\underline{M} = \left[\begin{array}{ccc:ccc} \cos(\omega_1 1T) & \dots & \cos(\omega_n 1T) & \sin(\omega_1 1T) & \dots & \sin(\omega_n 1T) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \cos(\omega_1 NT) & \dots & \cos(\omega_n NT) & \sin(\omega_1 NT) & \dots & \sin(\omega_n NT) \end{array} \right]$$

Example

A disturbed periodic signal $x(k)$ with k=1,…,1000 and T=1 is captured (see diagram below). An oscillation with ω_1 = 0.05 is identified and the following parameters are identified:

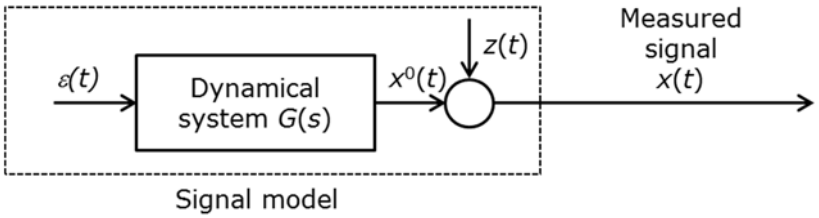
$$\underline{\hat{p}} = \begin{bmatrix} \hat{a}_1 \\ \hat{b}_1 \end{bmatrix} = \begin{bmatrix} 19.99 \\ 9.91 \end{bmatrix}.$$

In the following also the model output $\hat{x}(k, \underline{\hat{p}}) = \underline{m}^T(k) \cdot \underline{\hat{p}}$ is shown.



6.3 Stochastic signal models

Similar to section 6.2 a signal source is proposed that generates the stochastic signal:



The signal $\varepsilon(t)$ is a normal distributed, zero-mean, ergodic, and white noise process that is used to excite the dynamical system $G(s)$. The ideal signal is then disturbed by $z(t)$ before it can be measured as signal $x(t)$.

For the following modeling we will again use time discrete measurements in points $t = kT$ with k=1,…,N and sample time T.

Assessment

It is necessary to introduce a different error compared to the deterministic model approaches, because a direct comparison to parallel models is not possible. The stochastic model approach will be considered as optimal, if the model is able to predict the current signal in the best possible way using the past measurements $x(k-1)$, …, $x(k-n)$. The difference of real signal $x(k)$ and its prediction $\tilde{\hat{x}}(k, \underline{\hat{p}})$ will be called *prediction error*

$$e_P(k, \underline{\hat{p}}) = x(k) - \tilde{\hat{x}}(k, \underline{\hat{p}}).$$

As objective function we define again:

$$J(\underline{\hat{p}}) := \sum_{k=1}^{N} e_P^2(k, \underline{\hat{p}}) = \underline{e}_P^T(\underline{\hat{p}}) \cdot \underline{e}_P(\underline{\hat{p}}) \text{ with } \underline{e}_P(\underline{\hat{p}}) = \begin{bmatrix} e_P(1, \underline{\hat{p}}) \\ \vdots \\ e_P(N, \underline{\hat{p}}) \end{bmatrix}$$

Optimization

A direct solution of the optimization problem will be possible, if the prediction can be formulated as a linear function:

$$\tilde{x}(t, \underline{\hat{p}}) = \underline{m}^T(t) \cdot \underline{\hat{p}}$$

For all *N* measurements

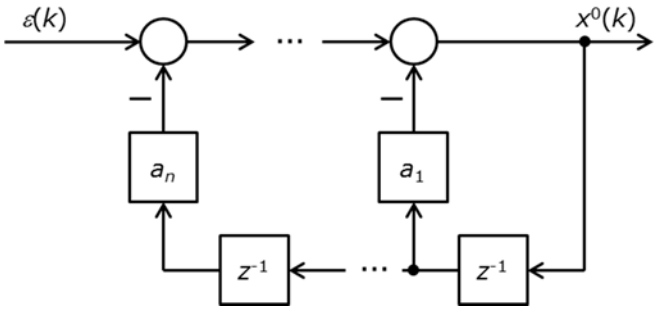
$$\underbrace{\begin{bmatrix} e_P(1) \\ \vdots \\ e_P(N) \end{bmatrix}}_{\underline{e}_P} = \underbrace{\begin{bmatrix} x(1) \\ \vdots \\ x(N) \end{bmatrix}}_{\underline{x}} - \underbrace{\begin{bmatrix} \underline{c}^T(1T) \\ \vdots \\ \underline{c}^T(NT) \end{bmatrix}}_{\underline{M}} \cdot \underline{\hat{p}}$$

The direct solution is again:

$$\underline{\hat{p}} = \left(\underline{M}^T \cdot \underline{M}\right)^{-1} \cdot \underline{M}^T \cdot \underline{x}$$

Autoregressive (AR) model

As dynamical model approach an *autoregressive (AR) model* is used, which is shown in the following block diagram:



Here, the operator z^{-1} stores the value at the input for one sample time step. Out of the block diagram we get the relation:

$$x^0(k) = -a_1 x^0(k-1) - \ldots - a_n x^0(k-n) + \varepsilon(k)$$

This can be used as a model in order to predict the current signal value in time step *k* based on the past signal values from *k*–1 back to *k*–*n*. Because the exciting process $\varepsilon(k)$ is zero-mean the best prediction is:

$$\tilde{x}(k) = -a_1 x(k-1) - \ldots - a_n x(k-n)$$

This is a linear combination of past measurements. Therefore, it can be converted into a linear optimization problem. With

$$\tilde{x}(k,\underline{\hat{p}}) = \underline{m}^T(k)\cdot\underline{\hat{p}} = \begin{bmatrix} -x_1(k-1) & \dots & -x(k-n) \end{bmatrix}\cdot\begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

we obtain matrix $\underline{M}$ for the a sequence of measurements $k=N_0,\dots,N$:

$$\underline{M} = \begin{bmatrix} -x(N_0-1) & \dots & -x(N_0-n) \\ \vdots & \ddots & \vdots \\ -x(N-1) & \dots & -x(N-n) \end{bmatrix}$$

Note: The index of the first measurement N_0 is chosen such that negative indices can be avoided in matrix $\underline{\Gamma}$.

Example

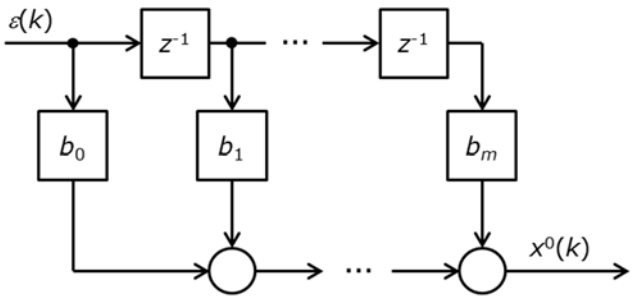
For an (undisturbed) stochastic process $x(k)$ with $k=1,\dots,1000$ the following Matlab code is used in order to identify the parameters of a second order AR model:

```
n = 2;                  % Order of AR model
N = length(x);          % Index of last measurement used
N0 = n+1;               % Index of first measurement used

% Build matrix M row by row:
M = zeros(N,n);     % Memory allocation
for k=N0:N
    M(k,:) = -(x(k-1:-1:k-n))';
end

% Solution for unknown parameters:
p = M(N0:N,:)\x(N0:end)
x_tilde = M*p;
```

The model parameters a_1 and a_2 can be identified correctly (see lecture). The standard deviation of the prediction error e_P is 1.017 in this case. This is almost equal to the standard deviation of the exciting process of σ_ε = 1.0. If it was larger, parts of the system dynamics would not be characterized sufficiently by the AR model used.

Moving Average (MA) model

As an alternative the dynamical system could be realized by the model approach of a *moving average (MA) model*:



From the block diagram we get the equation:

$$x^0(k) = b_0\,\varepsilon(k) + b_1\,\varepsilon(k-1) + \dots + b_m\,\varepsilon(k-m)$$

The drawback of this approach is that it is not possible to formulate the prediction $\tilde{x}(k)$ as a linear function of the model parameters b_i. For this reason, the MA model is not pursued for signal model parameter identification.

Combined ARMA model

The AR and the MA model can be superposed into an ARMA model.

7 Identification of discrete-time systems

An unknown dynamical system will be described as a discrete-time system. At first sight, this description is not that familiar, but it will have benefits later on, when we will implement corresponding algorithms. The system can be characterized be its *difference equation*

$$a_0\, y(k) + a_1 y(k-1) + \ldots + a_n y(k-n) = b_0 u(k) + \ldots + b_m u(k-m),$$

where k is the discrete time counter, or by the corresponding *transfer function*

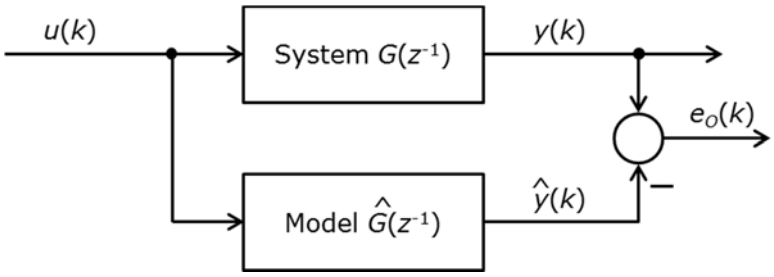
$$G(z^{-1}) = \frac{b_0 + b_1 z^{-1} + \ldots b_m z^{-m}}{a_0 + a_1 z^{-1} + \ldots a_n z^{-n}} = \frac{B(z^{-1})}{A(z^{-1})},$$

Where z^{-1} is the *unit delay operator* that delays the signal for one sample time period. In order to identify the system, it will be excited by the discrete-time input $u(k)$, and we measure the system's output $y(k)$. From that data a model of the transfer function

$$\hat{G}(z^{-1}) = \frac{\hat{b}_0 + \hat{b}_1 z^{-1} + \ldots \hat{b}_m z^{-m}}{\hat{a}_0 + \hat{a}_1 z^{-1} + \ldots \hat{a}_n z^{-n}} = \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})}$$

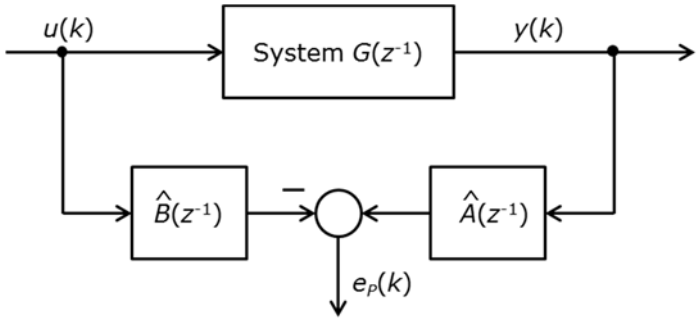
with its coefficients $\hat{a}_i$ and $\hat{b}_i$ has to be identified. The transfer function can be divided by $\hat{a}_0$ without loss of generality, such that $\hat{a}_0 = 1$ yields and the remaining (m+n+1) unknown parameters can be summarized into the parameter vector $\underline{\hat{p}} = \left[\hat{b}_0 \ldots \hat{b}_m \,\vdots\, \hat{a}_1 \ldots \hat{a}_n\right]^T$. Hence, the problem has to be solved to determine $\underline{\hat{p}}$ from the data.

A first approach would be to use the output error $e_A(k)$ as in chapter 4 as the difference between measurement $y(k)$ and model output $\hat{y}(k)$:



But, if we try to determine $\hat{y}(k)$ as a function of the measured values $u(k)$, $y(k)$ and the model $\hat{G}(z^{-1})$, this will be a nonlinear function of the parameters $\underline{\hat{p}}$. obtain receive a nonlinear optimization problem, which cannot be solved directly.

At that point it is more promising to introduce a modified error: By multiplying the output error by $\hat{A}(z^{-1})$ a new error $E_P(z) = \hat{A}(z^{-1})E_O(z)$ is defined. Within the block diagram $\hat{A}(z^{-1})$ can be shifted into both pathways from $y(k)$ and $\hat{y}(k)$, respectively, and we will come up with the following structure.



From the block diagram we get:

$$e_P = y\,\hat{A}(z^{-1}) - u\,\hat{B}(z^{-1})$$

Hence, error $e_P(k)$ is a linear combination of the input with its past values (weighted by $\hat{B}(z^{-1})$) and of the output with its past values (weighted by $\hat{A}(z^{-1})$). This alternative error definition is a linear function of the parameter vector $\underline{\hat{p}}$.

Error $e_P(k)$ can be interpreted as a *prediction error*: We can write the prediction error in time domain

$$e_P(k) = 1\,y(k) + \hat{a}_1 y(k-1) + \ldots + \hat{a}_n y(k-n) - \hat{b}_0 u(k) - \ldots - \hat{b}_m u(k-m)$$

using vectors:

$$e_G(k,\underline{\hat{p}}) = y(k) - \underbrace{\left[u(k)\ldots u(k-m) \;\vdots\; -y(k-1)\ldots -y(k-n)\right]}_{\underline{m}^T(k)} \cdot \underbrace{\begin{bmatrix}\hat{b}_0\\ \vdots\\ \hat{b}_m\\ \hat{a}_1\\ \vdots\\ \hat{a}_n\end{bmatrix}}_{\underline{\hat{p}}} = y(k) - \underbrace{\underline{m}^T(k)\cdot\underline{\hat{p}}}_{\tilde{y}(k,\underline{\hat{p}})}$$

Therein, the signal $\tilde{y}(k,\underline{\hat{p}})$ (which does not appear in the preceding block diagram) can be interpreted as a model-based prediction for the output signal $y(k)$. Based on known inputs and on the past outputs $y(k-1),\ldots,y(k-n)$ a prediction is made for the current output $y(k)$. Hence, $e_P(k,\underline{\hat{p}}) = y(k) - \tilde{y}(k,\underline{\hat{p}})$ is the error of a one-step model prediction.

We can write down this equation for several measurements in discrete time points $k=N_0,\ldots,N$ and summarize this row by row such that the parameter vector $\underline{\hat{p}}$ can be expanded to the right hand side:

$$\underbrace{\begin{bmatrix}e_P(N_0,\underline{p})\\ \vdots\\ e_P(N,\underline{p})\end{bmatrix}}_{\underline{e}_P(\underline{\hat{p}})} = \underbrace{\begin{bmatrix}y(N_0)\\ \vdots\\ y(N)\end{bmatrix}}_{\underline{y}} - \underbrace{\left[\begin{array}{ccc:ccc} u(N_0) & \ldots & u(N_0-m) & -y(N_0-1) & \ldots & -y(N_0-n)\\ \vdots & & \vdots & \vdots & & \vdots\\ u(N) & \ldots & u(N-m) & -y(N-1) & \ldots & -y(N-n)\end{array}\right]}_{\underline{M}} \cdot \underbrace{\begin{bmatrix}\hat{b}_0\\ \vdots\\ \hat{b}_m\\ \hat{a}_1\\ \vdots\\ \hat{a}_n\end{bmatrix}}_{\underline{\hat{p}}}$$

Note: By starting with the initial time point $N_0 = \max(m,n)+1$, we have a sufficient number of past values of u and y available, such that only positive time steps are used.

The optimal solution for $\underline{\hat{p}}$ minimizing the objective function

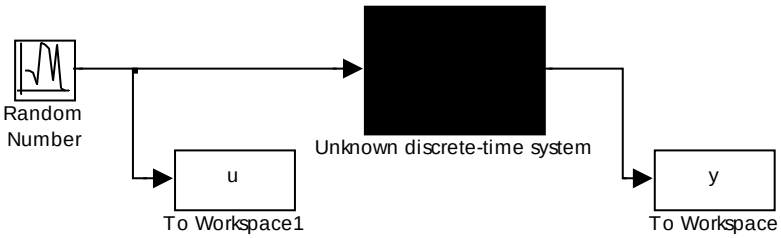
$$J(\underline{\hat{p}}) = \sum_{k=1}^{N} \left(y(k) - \tilde{y}(k, \underline{\hat{p}})\right)^2 = \underline{e}_G^T(\underline{\hat{p}}) \cdot \underline{e}_G(\underline{\hat{p}})$$

is again:

$$\underline{\hat{p}} = \left(\underline{M}^T \underline{M}\right)^{-1} \underline{M}^T \underline{y}$$

Example: Discrete-time system identification

An unknown discrete time system (black box in the Simulink block diagram) is excited by a random input u. A total number of $N = 10000$ samples of input u and (undisturbed!) output y are stored to the workspace.



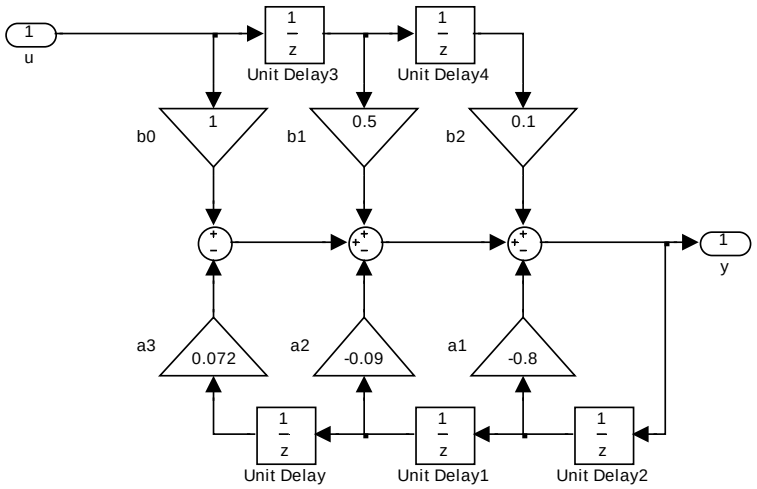
The following Matlab algorithm will determine the parameter vector $\underline{\hat{p}}$:

```
% Define the order of the transfer function:
m = 2;
n = 4;

% Number of measurements:
N = length(y);

% First measurement:
N0 = max(m,n)+1;

% Memory allocation:
M = zeros(N-N0+1,m+1+n);

% Build matrix M row by row:
for k=N0:N % Zeitschritte
    M(k-N0+1,:) = [ u(k:-1:k-m)' -y(k-1:-1:k-n)' ];
end

% Solution for the parameter vector:
p_hat = M\y(N0:end)
```

As result we obtain:

```
p_hat =
    1.0000
    0.5000
    0.1000
   -0.8000
   -0.0900
    0.0720
```

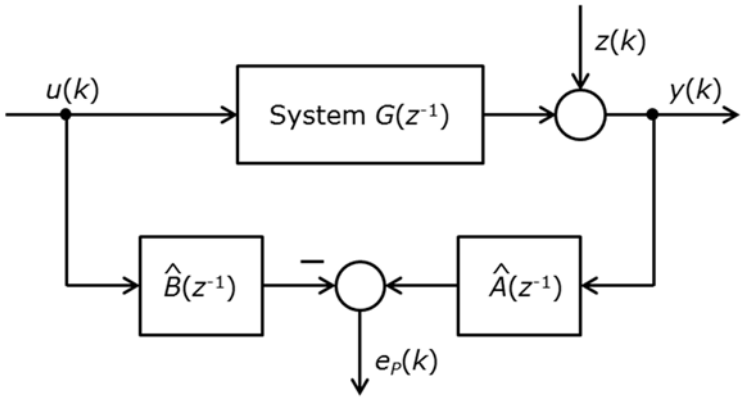
The block diagram shows the unkown process (previously shown as black box):



The model parameters have been identified correctly!

Stochastic case

In many cases the measurement *y* is disturbed. In order to include this in a model, an additive distortion signal *z* (zero-mean, white, normal distributed) is introduced:



The process of parameter identification in the deterministic case is called *parameter estimation* in the stochastic case.

Parameter estimation is called *(asymptotically) unbiased*, if the estimation of the parameters $\underline{\hat{p}}$ converges towards the real parameters $\underline{p}$, if the number of measurements *N* goes to infinity:

$$\lim_{N\to\infty} \underline{\hat{p}} = \underline{p}$$

It can be shown that (unfortunately!) the estimation based on the predictive error e_P is *not* unbiased! This means that we have to make sure that the disturbance *z* is small enough, otherwise we have to expect systematical errors in $\underline{\hat{p}}$. Methods of *instrumentation variables* have been developed that solve this problem by providing an undisturbed output $\tilde{y}$.

8 Recursive parameter estimation

Up to know, we have assumed that all measurements $\underline{y}$ and all inputs $\underline{u}$ are available in order to determine the parameter vector $\underline{\hat{p}}$. This calculation is performed independently from the process operation. Therefore, this way of parameter identification can be considered as an offline method. In contrast, we would call a method *online*, if it calculates a current parameter vector $\underline{\hat{p}}(k)$ during the runtime of the process.

For the linear optimization problem it would be possible to update vector $\underline{y}$ and matrix $\underline{M}$ in each time step *k*, and to solve $\underline{\hat{p}} = \left(\underline{M}^T \underline{M}\right)^{-1} \underline{M}^T \underline{y}$ for the parameter vector. But, in each time step the complete system of equations would have to be solved, which is numerically quite expensive. A *recursive solution* would be much more efficient, which determines $\underline{\hat{p}}(k)$ assuming that $\underline{\hat{p}}(k-1)$ is known. This should be derived in the following.

Recursive algorithm

Initial condition in k=0: Define

$$\underline{\hat{p}}(0),\ \underline{P}(0)$$

If the initial parameter vector $\underline{\hat{p}}(0)$ is unknown, the initial matrix $\underline{P}(0)$ should be chosen as a "large" positive definite matrix, e.g. $\underline{P}(0) = 10^{10} \cdot \underline{I}$.

Recursion from time step k to k+1:

1. Step: $$\underline{K}(k+1) = \frac{\underline{P}(k)\,\underline{c}(k+1)}{1 + \underline{c}^T(k+1)\underline{P}(k)\,\underline{c}(k+1)}$$

2. Step: $$\underline{\hat{p}}(k+1) = \underline{\hat{p}}(k) + \underline{K}(k+1) \cdot \left[\underline{y}(k+1) - \underline{m}^T(k+1)\,\underline{\hat{p}}(k)\right]$$

3. Step: $$\underline{P}(k+1) = \left[\underline{I} - \underline{K}(k+1)\underline{c}^T(k+1)\right] \cdot \underline{P}(k)$$

This will minimize the objective function $J(\underline{\hat{p}}) = \underline{e}_y^T(\underline{\hat{p}}) \cdot \underline{e}_y(\underline{\hat{p}})$.

Recursive algorithm with fading memory

In case of time-varying processes, it is appropriate to introduce time-depending weighing factors. A weighing factor with exponential forgetting will discount the weights for past measurements. Introducing a forgetting factor $0 < \lambda < 1$ (e.g. $\lambda = 0.99$) the object function can be modified:

$$J(\underline{\hat{p}}) = \underline{e}_y^T(\underline{\hat{p}}) \cdot \underline{Q} \cdot \underline{e}_y(\underline{\hat{p}}) \quad \text{with} \quad \underline{Q} = \text{diag}(\lambda^{N-1},\ \ldots\ \lambda^2,\ \lambda,\ 1)$$

1. Step: $$\underline{K}(k+1) = \frac{\underline{P}(k)\,\underline{c}(k+1)}{\lambda + \underline{c}^T(k+1)\underline{P}(k)\,\underline{c}(k+1)}$$

2. Step: $$\underline{\hat{p}}(k+1) = \underline{\hat{p}}(k) + \underline{K}(k+1) \cdot \left[\underline{y}(k+1) - \underline{m}^T(k+1)\,\underline{\hat{p}}(k)\right]$$

3. Step: $$\underline{P}(k+1) = \frac{1}{\lambda}\left[\underline{I} - \underline{K}(k+1)\underline{c}^T(k+1)\right] \cdot \underline{P}(k)$$