Introduction to Intelligent Decision Support Systems

Igor Škrjanc

Faculty of Electrical Engineering

PPT lecture slides, February 2019

Table of Contents

1. Introduction to Intelligent Decision Support Systems
2. The importance of nonlinear system identification
   - Linear or nonlinear problem
   - Prediction
   - Simulation
   - Optimization
   - Analysis
   - Control
   - Fault detection
3. Identification steps
   - Selecting model inputs.
   - Selecting the form of excitation signals.
   - Selecting model architecture.
   - Selecting the type of model dynamics.
   - Selecting a model order.
   - Selecting the structure and complexity of the model.
   - Estimation of model parameters.
   - Validation of the model.
   - Influence of method parameters
4. Theoretical, experimental and combined models
5. Terminology

Definition of decision-making
Decision-making is a type of action where on the basis of certain experience and knowledge we make decisions through which we influence the operation of the observed system.

Knowledge and experience are most often written in the form of models.

The importance of nonlinear system identification

Model based decision-making
The models are used in the analysis of systems, in forecasting and simulating the behaviour of systems in the future. In engineering we use them for the design of new processes and the analysis of existing ones and for the design of control systems, optimization, monitoring, detection and diagnostics of faults.

Linear or nonlinear problem

Nonlinear models
The construction of nonlinear models makes sense where the system operates in a wider working area. Before developing a nonlinear model, we first need to find out if this is really needed. Therefore, we first build a linear model. If this is not enough, only then do we start the development of a nonlinear model.

Examples of the models' usage

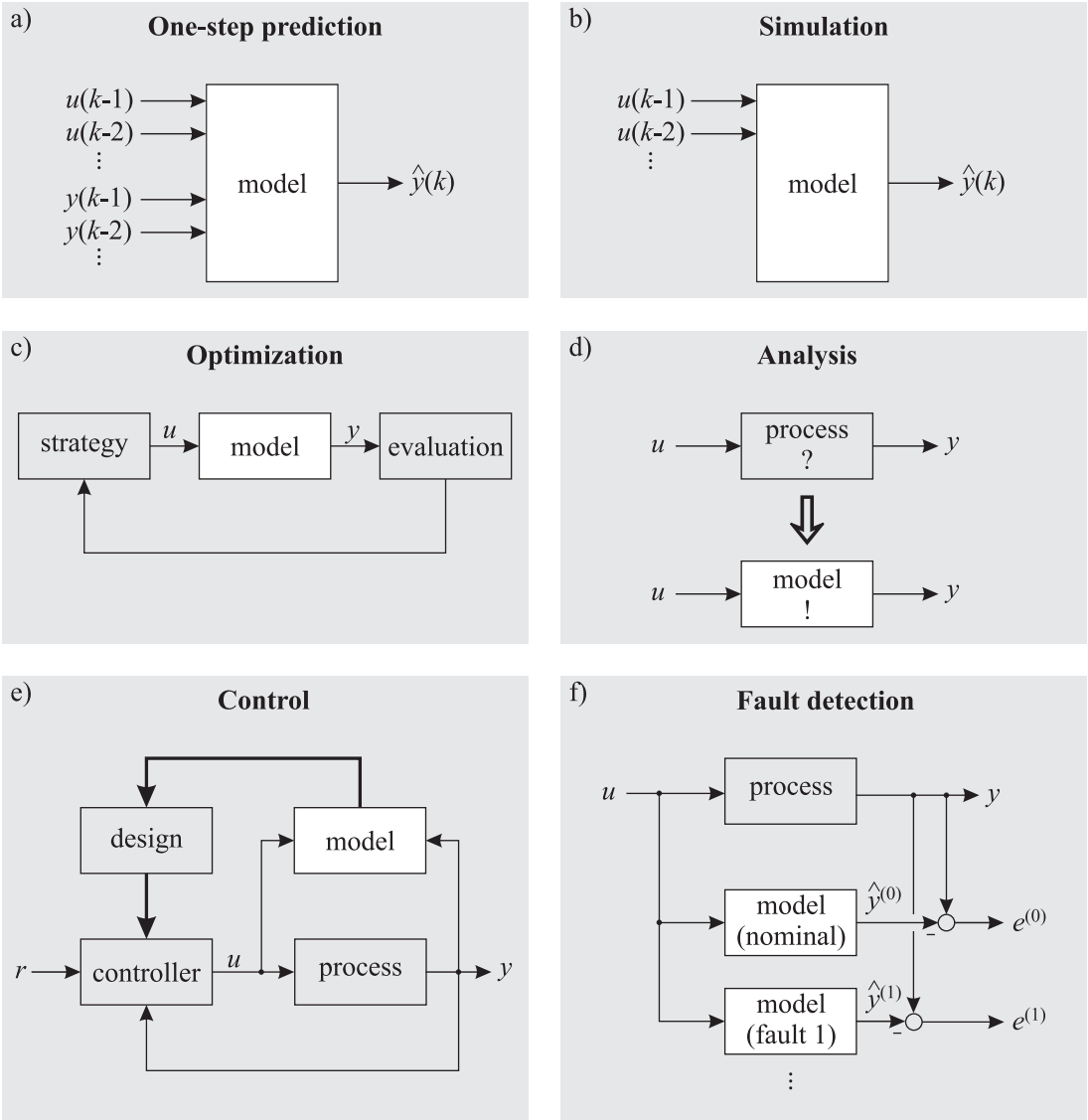


Figure 1: The use of models for a) prediction, b) simulation, c) optimization, d) analysis, e) control and f) fault detection

Prediction

Predicting the output of the process for several steps in the future is called **multi-step prediction** (*multi-step* ahead or h*-step ahead prediction*). The number of steps h in the future is called **prediction horizon** Here, two alternative prediction methods can be used:

- the model can be constructed in such a way that it directly predicts h steps in the future. In such a model, we must additionally integrate the input signals in the future $u(k), u(k+1), \ldots, u(k+h-1)$. The disadvantage is that the dimension of the problem grows with the prediction horizon h and that we can only predict the value at the horizon h, although we often also need a prediction for $1, 2, \ldots, h$ steps.

Examples of the models' usage

- The model in Figure a) can be used for a one-step prediction. In the next step, the same model is used for prediction by replacing $k \rightarrow k+1$ and using a prediction in the previous step $\hat{y}(k)$. This is repeated h times, to reach h-th horizon. This method accumulates the error and contains a feedback loop which can be unstable.

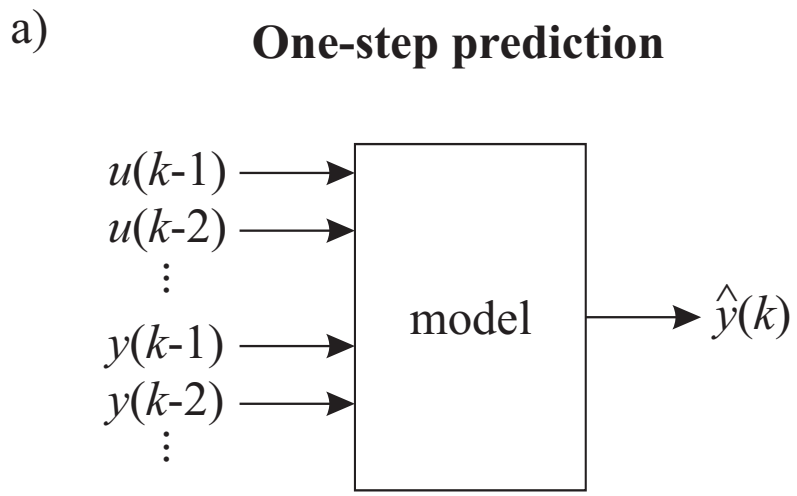


Figure 2: The use of models for a) prediction, b) simulation, c) optimization, d) analysis, e) control and f) fault detection

Simulation

- Frequent use in case of optimization, control and fault detection, Figure b).
- The use as a soft sensor (software-based sensor), where the model replaces the physical sensor.
- The simulation is a feedback structure, which additionally complicates modelling and identification and requires additional vigilance in terms of model stability.

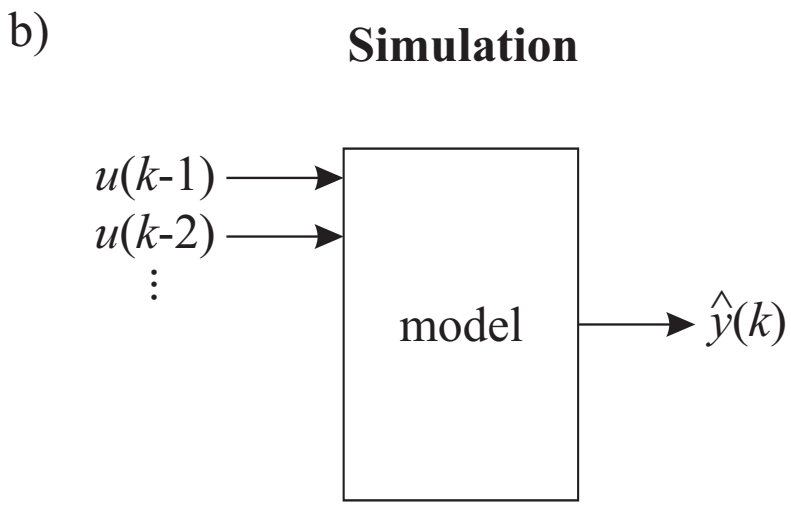


Figure 3: The use of models for a) prediction, b) simulation, c) optimization, d) analysis, e) control and f) fault detection

Optimization

- Figure c) shows the use of a prediction or simulation model in the case of optimization of a certain criterion function, which enables the search for the optimum operating point or optimal process input.

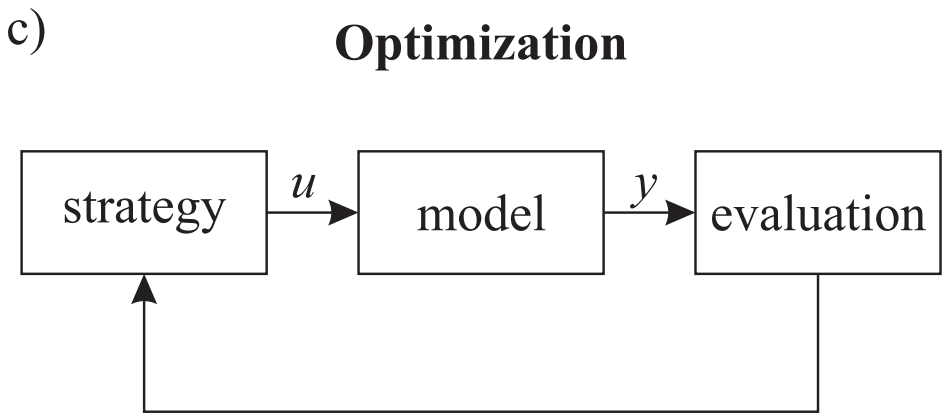


Figure 4: The use of models for a) prediction, b) simulation, c) optimization, d) analysis, e) control and f) fault detection

Analysis

- Figure d) shows the use of the model in getting to know and understanding the operation of the process, for various parameters and different process inputs.

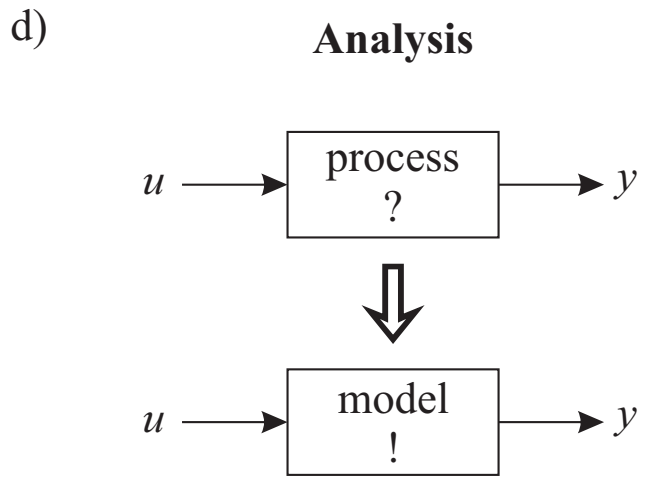


Figure 5: The use of models for a) prediction, b) simulation, c) optimization, d) analysis, e) control and f) fault detection

Control

- Figure e) indicates the use of the model for the purpose of control design. Modern control systems are based on a process model that allows the implementation of comprehensive information about the process and even its uncertainties.
- Identification for control purposes explores how to most easily identify the model which will suffice for the purposes of control design.

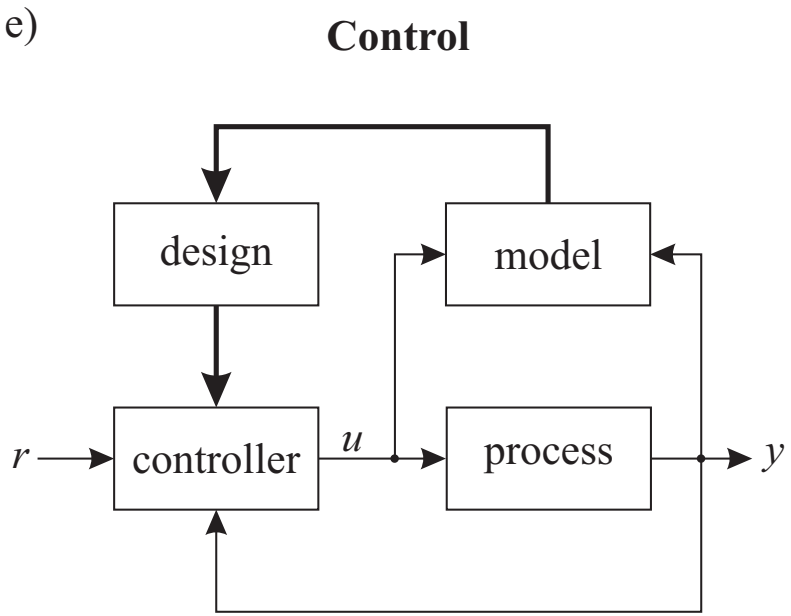


Figure 6: The use of models for a) prediction, b) simulation, c) optimization, d) analysis, e) control and f) fault detection

Control

- Linear models: the model that is most accurate in the mid-frequency range around the cutoff frequency of the closed-loop system due to the stability of the closed loop
- Setpoint tracking: the accuracy of the model in the frequency range of the reference signal, which is usually in the low range.
- Regulatory control: is solved with the introduction of an integration law that eliminates disturbances at the input and output of the process.
- In nonlinear cases, the most important is the static characteristic.

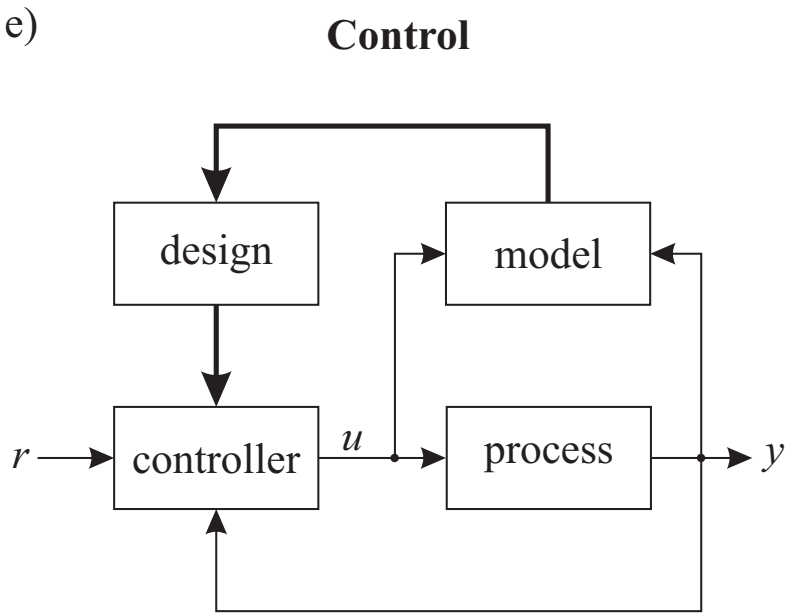


Figure 7: The use of models for a) prediction, b) simulation, c) optimization, d) analysis, e) control and f) fault detection

Fault detection

- Figure f) shows the strategies for fault detection. In this case, we build a model that describes the operation of the process under normal circumstances (nominal model), and fault models (error $1, 2, \ldots$). By comparing the model outputs for a nominal example ($\hat{y}^{(0)}$), the outputs for fault models ($\hat{y}^{(i)}$ for fault i) and model output y, we can detect the fault i. We compare if any of the errors between the output and the the output of the fault model $e^{(i)}$ is for $i > 0$ smaller than $e^{(0)}$. Comparisons are usually performed on filtered signals of errors to eliminate the impact of errors due to modelling and noise and to avoid the wrong alarms.

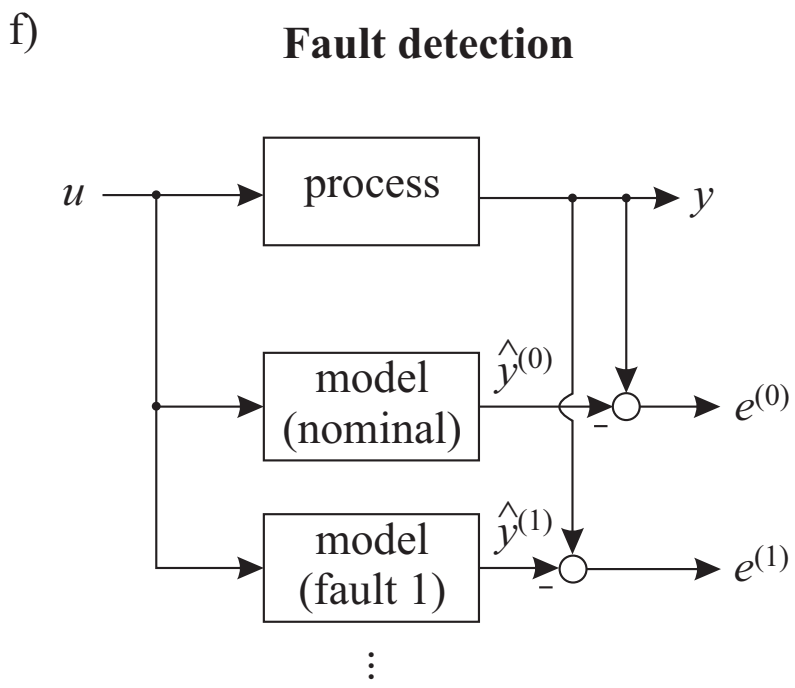


Figure 8: The use of models for a) prediction, b) simulation, c) optimization, d) analysis, e) control and f) fault detection

Identification steps

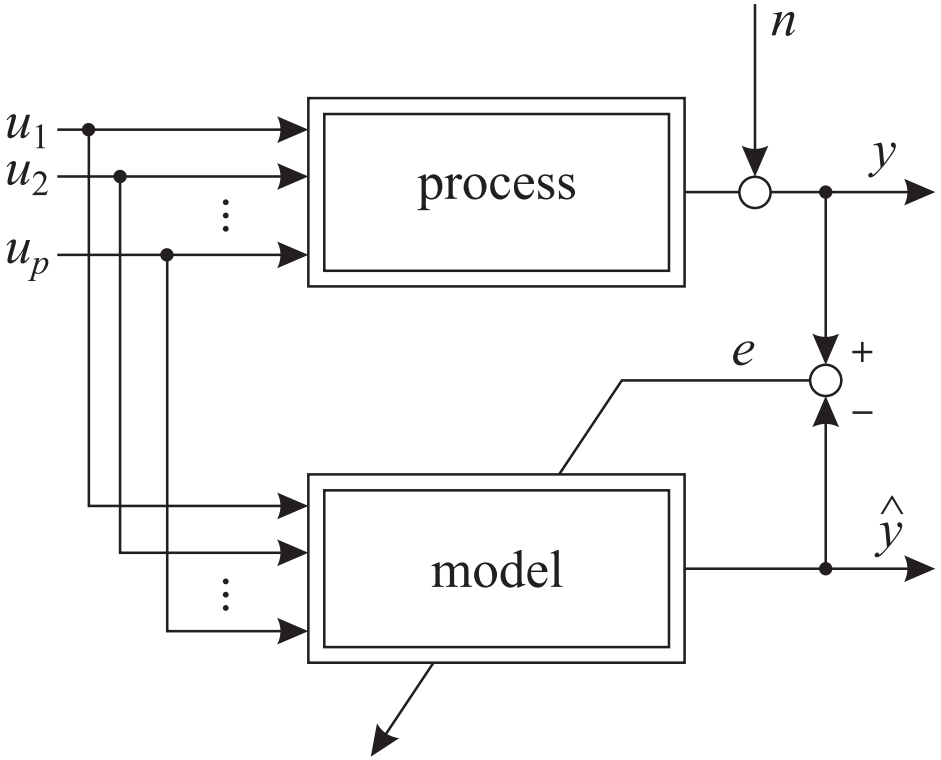


Figure 9: System identification: Adaptation of the model parameters so that the model describes the behavior of the process in the best possible way. Process and model with the same inputs $u = [u_1\ u_2\ \cdots\ u_p]^{\mathrm{T}}$, and outputs y and $\hat{y}$ are compared on the basis of the error e, which is used for the adaptation of the model.

Identification steps

Identification steps

- selecting model inputs.
- selecting the form of excitation signals.
- selecting model architecture.
- selecting the type of model dynamics.
- selecting model order.
- selecting the structure and complexity of the model.
- estimation of model parameters.
- evaluation of the model.

Identification steps

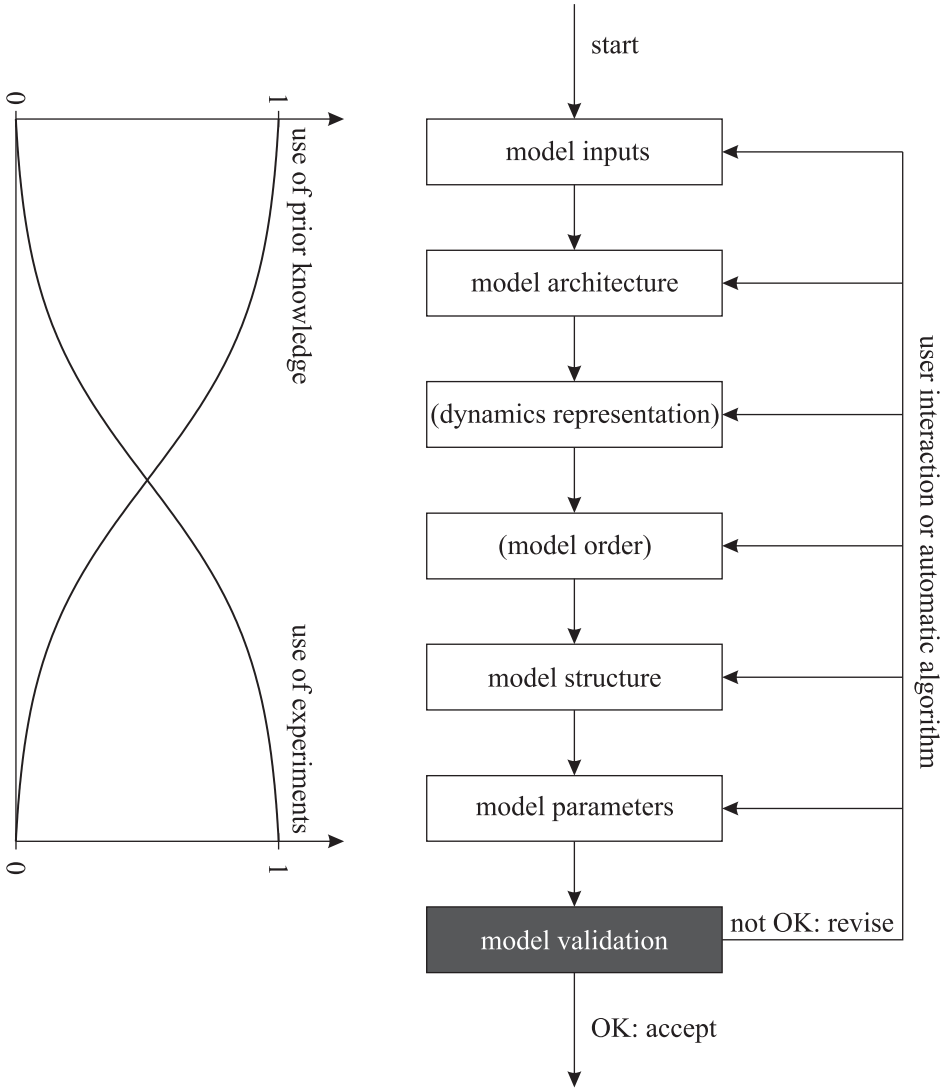


Figure 10: The identification procedure and the associated closed-loop structure (brackets define the steps which are required only in the case of dynamic systems).

Selecting model inputs.

Step 1: Selecting model inputs offers a choice between four strategies:

- *Using all inputs:* leads to models of higher dimensions with a higher number of parameters. We can use it along with regularization, which prevents the pre-parametrization and poor conditionality of the problem (singular matrices) by adding the criteria in the criterion function.
- *Examination of all input combinations:* Even though this strategy leads to the best solution, in practice it is not useful, because the number of trials increases combinatorially with the number of potential inputs.
- *Unsupervised input selection:* A very useful choice of inputs in this case is based on Principal component analysis (PCA), which enables the removal of irrelevant inputs in a relatively simple way. The main disadvantage of the PCA method is that the selection is performed on the data of the input space, without output data. This can mean that in individual cases we do not take into account inputs that can be very important for the quality of the model.
- *Supervised inputs selection:* The inputs are selected so as to obtain the highest possible accuracy of the model. For linear models we use correlation analysis, in the case of nonlinear models this represents a complex optimization problem that can be solved with, for example, evolutionary algorithms (EAs).

Selecting the form of excitation signals

Step 2: Selecting the form of excitation signals requires prior knowledge of the process and the purpose of using the model. In the case of the models that we get based on identification (*black box modelling*), measurements are the most important and the only source of information. Behaviour of the process that is not covered by the measured data can not be identified because we did not detect it. That of course limits the quality of the model.

Step 3: Selecting Model Architecture

- *Problem domain:* Classification (sorting), approximation (regression, modelling) of the system.
- *Usage:* Model can be used for simulation purposes, optimization, control, fault detection, etc.
- *Dimension problem:* The number of relevant inputs (and outputs) limits the choice of architecture.
- *Accessibility and quality of measurements:* In case of insufficient and very noisy data it is difficult to build a good local model.
- *Model development limits:* Time, material, etc.
- *Memory capacity constraints:* There are still some memory capacity constraints in certain areas (car industry), but these restrictions disappear quickly.
- *Off-line and on-line learning:* All architectures are suitable for off-line learning, but not for on-line learning. The on-line learning is still a matter of research.
- *User's experience:* User's experience has a great influence on the selection of methods and architectures of models.
- *Accessibility of tools:* Accessibility of tools greatly influences the selection of certain architectures of models.
- *User's acceptance:* The main thing when adopting new methods in modelling and identification is the understanding of basic methods and interpretation of models.

Selecting the type of model dynamics

Step 4: Selecting the type of model dynamics is most dependent on the use of the model. If we use it for one-step-ahead prediction, then we can use the NARX or NARMAX structure.

Step 5: Selecting a model order is usually done by combining the knowledge of the process and experiment. Choosing the higher order results in a higher complexity of the model.

Step 6: Selecting the structure and complexity of the model is much more demanding than parameter optimization. It can be performed automatically if we use methods such as the orthogonal least squares method.

Step 7: Estimation of model parameters is the simplest for automation. Usually it is solved by using a linear or nonlinear optimization technique. Linear techniques usually operate completely automatically, while nonlinear techniques require certain interaction with the user. According to the chosen technique of optimization, we need to select an initial estimation of parameters and certain setup parameters of the method.

Step 8: Validation of the model serves as an estimation of all the previous steps to see whether we have carried them out appropriately and whether we have obtained a satisfactory model.
The first step in validation is to test the model on the data we have used to train the model. We also call this method verification. If verification does not give relevant results, we do not continue with further steps. The reason may be that the number of learning data is too small or that the model's structure is inadequate. If the verification is satisfied we contiue with the second step, testing on new set of data.

Influence of method parameters

Method parameters

- Method parameters are parameters that are not calculated by the method automatically, but are set by the user. Such typical parameters are the initial values of model parameters which we identify, learning constants, number of clusters, the number of neurons and the number of layers in neuron networks, parameters that finish the algorithm.
- These parameters are usually set manually during simulation tests.

Influence of method parameters

The desired effects of the method parameters are as follows:

- **Separated influence of method parameters.** The problem with setting arises when we can not individually observe the influence of an individual parameter, but the influence of an individual parameter changes when changing another parameter.
- **Small number of method parameters.**
- *Method parameters must have* **a significant influence on the operation of the algorithm**.
- **Easy choice of preset method parameter values.**
- **Acceptable and simple sensitivity characteristic of the method parameters.**

Theoretical, experimental and combined models

Modelling methods can be divided into three different approaches:

- **Theoretical modelling** is based on the complete development of a model based on equilibrium equations, i.e. physical equations, chemical, biological, econometric, etc. Such models are not based on the measured data which are measured during the process. Model parameters have physical significance.
- **Experimental modelling** is based solely on measured data. Both the structure and the model parameters are estimated from the measured data. Usually we do not try to introduce knowledge about the process, or very little. Model parameters do not have direct physical interpretation.
- **Combined modelling** represents a compromise between theoretical and experimental modelling. Usually the structure of the model is what is determined based on the knowledge of the process, and the parameters are evaluated on the basis of the measured data.

Theoretical, experimental and combined models

	White box	Gray box	Black box
Information sources	first principles insights	qualitative knowledge (rules) some insights + some data	experiments data
Features	good extrapolation good understanding high reliability scalable		short development time little domain expertise req. can be used in addition also for not understood proc.
Drawbacks	time consuming detailed domain expertise req. knowledge restricts accuracy only for well understood proc.		no reliable extrapolation not scalable data restricts accuracy little understanding
Application areas	planning, construction, design rather simple processes		only for existing processes rather complex processes

Figure 11: Theoretical, experimental and combined models

Theoretical, experimental and combined models. In combined models, we can encounter very different combinations among theoretical and experimental modelling.

Terminology

The terminology we use for the field of system identification and optimization is classical:

- **Process:** We use it as a synonym for an industrial process or a system that is being explored. It can be a static or dynamic process.
- **Linear parameters:** Parameters that act linearly.
- **Nonlinear parameters:** Parameters that act nonlinearly.
- **Learning:** Optimization of a structure or model parameters to achieve optimum value of a specific criterion function. If the emphasis is on optimization of the structure, then we usually talk about *learning*, but if the focus is on searching for parameters, then we talk about *estimation*.
- **Generalization:** Evaluation of the model output for the input vector which is not included in the learning set. Generalization can be *interpolation*, if the input vector is within the area covered by input vectors from the learning set, or *extrapolation*, if the input vector lies outside this area.

Introduction to Optimization

Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for the 2nd Lecture, February 2019

Table of Contents

1. Introduction to Optimization
2. An Overview of Optimization Methods
3. Criterion Functions for Supervised Learning
   - Maximum Likelihood Method
4. Criteria Functions in Unsupervised Learning

Introduction to Optimization

We can distinguish three different approaches to optimization. They differ according to the information we need to identify the model:

- **supervised learning**,
- **reinforcement learning**,
- **unsupervised learning**.

Introduction to Optimization

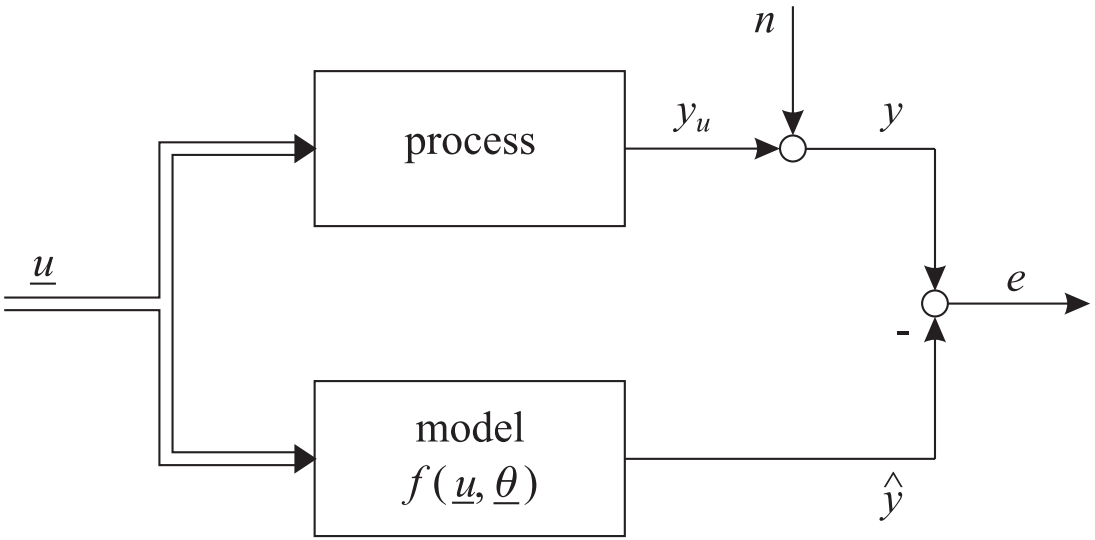
- **Supervised learning** searches for the relations in the set of input-output data pairs. For each input vector we know the output value. The goal of supervised learning is in minimizing the selected error rate between output of the process and the output of the model to obtain the best possible model.

- In the case of **reinforcement learning** we do not know the output of the process for each input vector. A typical example of this type of learning are games where we can not determine the adequacy of each move.

- In the case of **unsupervised learning** we consider only the input data. The main purpose of unsupervised methods is to search for similarities within the data set and their grouping into clusters.

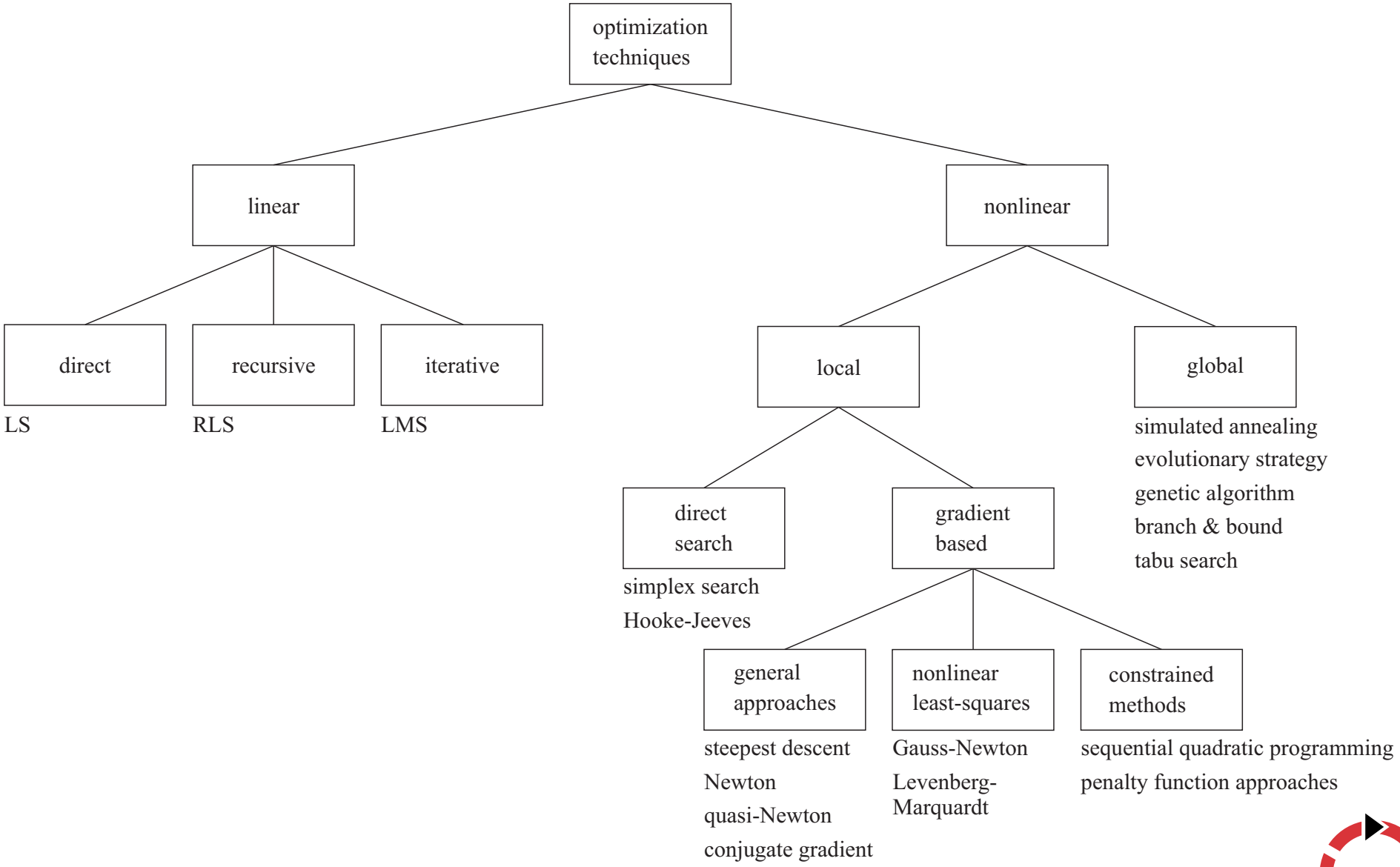
Introduction to Optimization

- A model that can be described by the function $f(\cdot)$ maps the input vector $\underline{u}$ into the scalar output of the model y.
- The model is parameterized with the n parameters that can be written into the vector of parameters $\underline{\theta}$ so that $\hat{y} = f(\underline{u}, \underline{\theta})$.
- The goal of the optimization is to find such parameters $\underline{\theta}$ that the approximation of the output of the model $\hat{y}$ best describes the measured output of the process y that is corrupted by noise n.
- Optimization is the search for the solution point in n -dimensional space of model parameters $\underline{\theta}$.



Slika 1: Process and model

An Overview of Optimization Methods



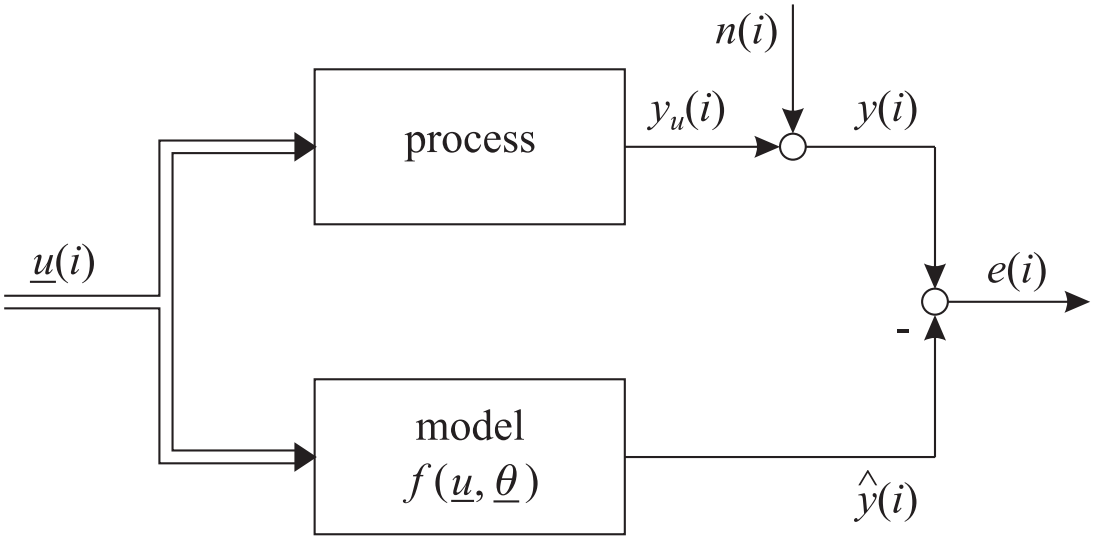
Slika 2: Overview of linear and nonlinear optimization methods

Criterion functions for supervised learning

We optimize the error $e(i)$, which is defined as the difference between the measured outputs of the model $y(i)$ and the output of the model $\hat{y}(i)$ for the whole learning data set $i = 1, \ldots, N$, where N represents the number of all measurements. Since normally the measured output of the model $y(i)$ is corrupted by the noise $n(i)$, we do not know the true output of the process $y_u(i)$.

The first **criterion function is the sum of squared errors** (*least squares, LS)*)

$$I(\underline{\theta}) = \sum_{i=1}^{N} e^2(i) \qquad \text{where is} \quad e(i) = y(i) - \hat{y}(i)\,. \tag{1}$$



Slika 3: Process and model for sample i

Criterion functions for supervised learning

The criterion function of the sum of squared errors can be extended to a form where each contribution is weighted by the q_i weight and we get **the method of the sum of weighted squared errors**,

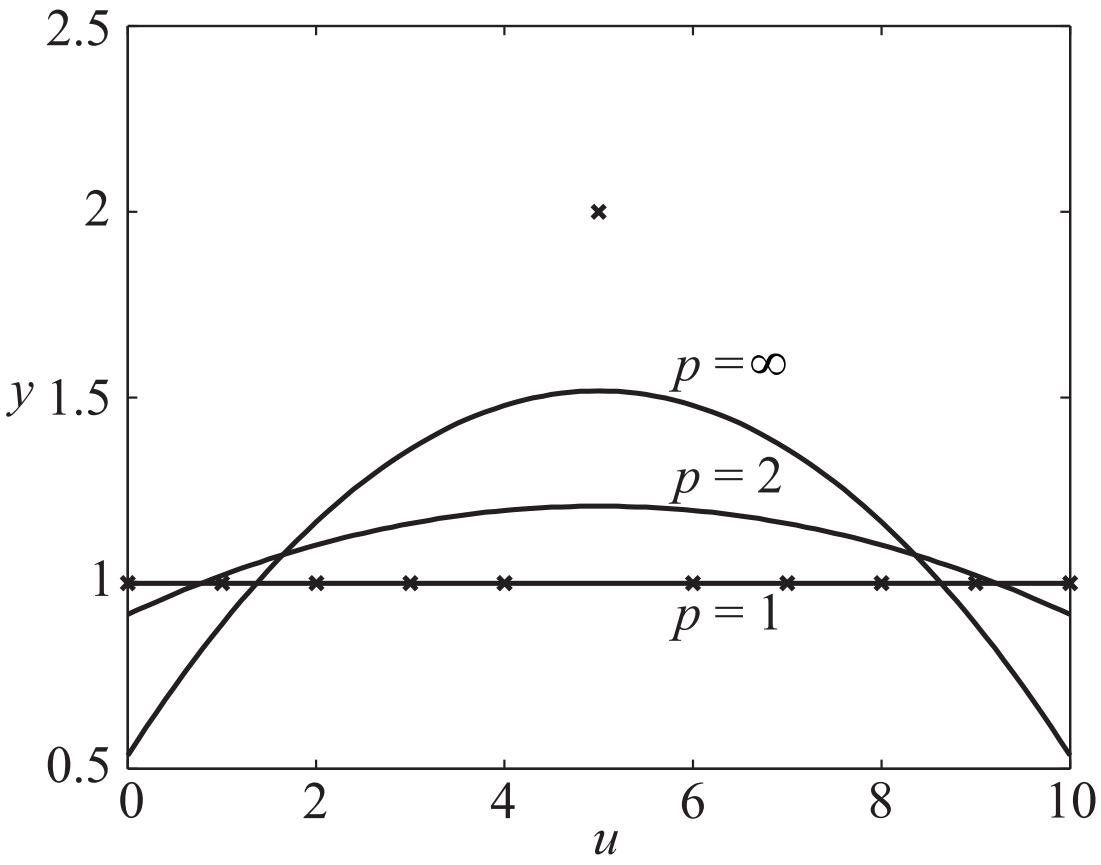
$$I(\underline{\theta}) = \sum_{i=1}^{N} q_i \, e^2(i) \, . \tag{2}$$

The method is also called the *weighted least squares (WLS) method* .

Criterion functions for supervised learning

The generalization of this method is a criterion function of the following form:

$$I(\underline{\theta}) = \left(\sum_{i=1}^{N} q_i \, \|e(i)\|^p \right)^{1/p} . \tag{3}$$



Slika 4: Adjustment of the second-order polynomial to 11 input-output pairs for the criterion function (3) at $p = 1, 2,$ and ∞. The higher order p leads to higher sensitivity to exceptions (5,2)

Maximum Likelihood Method

- None of the previous methods included assumptions about the noise n. We can show that the method of least squares implicitly assumes the uncorrelated noise $n(i)$ (i.e. $E\{n(i)n(j)\} = 0$ for $i \neq j$, if $E\{n(i)\} = 0$) with Gaussian distribution, mean value equal to zero and a variance equal to σ^2. In this way, the error $e(i)$ has the same distribution as noise $n(i)$ and probability density function (pdf) of the random variable of the model's error $e(i)$ is:

$$p\left(e(i)\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} e^2(i)\right) . \tag{4}$$

- Since we assume that the errors are independent of each other, the maximum likelihood function is equal to the product of all density functions for each sample:

$$L\left(e(1), e(2), \ldots, e(N)\right) = p\left(e(1)\right) \cdot p(e(2)) \cdot \ \ldots \ \cdot p(e(N)) \tag{5}$$

or

$$L\left(e(1), e(2), \ldots, e(N)\right) = \left(2\pi\sigma^2\right)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{N} e(i)^2\right) \tag{6}$$

Maximum Likelihood Method

- Optimal parameters are those that maximize the likelihood function (5). By using the logarithm, we transform the problem again into the search of a minimum and we get

$$I(\underline{\theta}) = -\ln\left(p(e(1))\right) - \ln\left(p(e(2))\right) \ldots - \ln\left(p(e(N))\right) \tag{7}$$

  or otherwise

$$I(\underline{\theta}) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N} e(i)^2 \tag{8}$$

- We can easily see that for a noise with Gaussian distribution and a mean value equal to zero, it results in the criterion function of the weighted least squares.

Criteria Functions in Unsupervised Learning

- The criterion functions dealt with so far are useless with unsupervised learning because we do not have an output measurement $y(i)$, i.e. we do not have the input-output pair of measurements. Therefore in the criterion function only the inputs $\underline{u}(i)$ can be considered.
- Criteria adhere to the relations with the surroundings. In the input hyperspace, we search for distances between hyperpoints, we can search for hyperlines, hyperspheres, hyperellipsoids, etc.
- Normally a criteria function is the distance of the measured input vector to the specified hyper objects in the space, in the simplest case, e.g. distance between the point and the center of the swarm or to the line that determines the swarm.

Linear Optimization

Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for 3rd Lecture, February 2019

Table of Contents

1. Linear optimization
2. Least squares method - LS
   - Explanation of the least squares method
     - Least squares method from the correlation perspective
     - Conditionality of Regression Matrix
   - Covariance matrix of estimated parameters
   - Confidence interval
   - Orthogonal regressors
   - Regularization
   - Weighted least squares method - WLS
3. Recursive least squares method - RLS
   - Reducing computational complexity
   - Identification of time-varying processes
4. Selection of regressors
   - Methods for selecting regressors

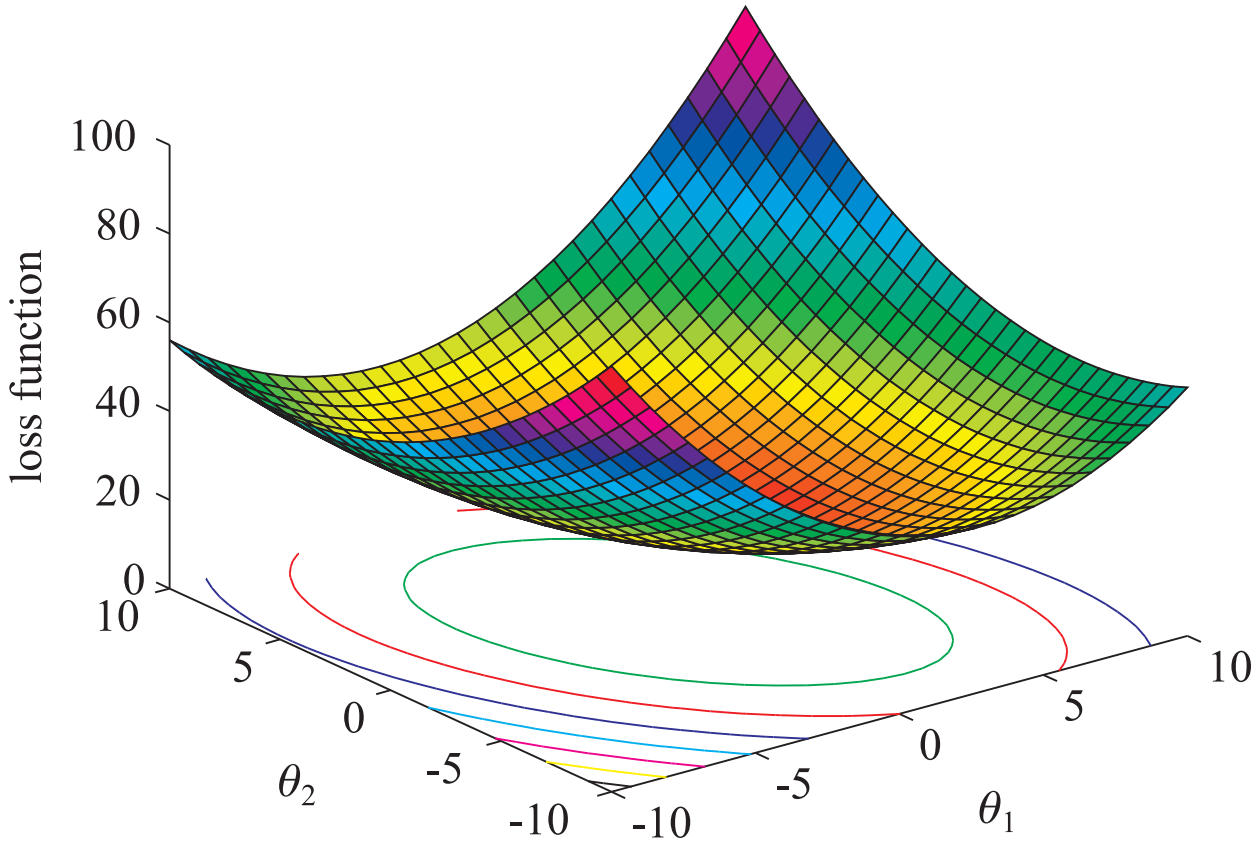
Linear optimization

Optimization is linear if the gradient of the cost function $I(\underline{\theta})$ is linear in the parameters.

Linear optimization has the following important properties:

- existence of an optimal solution, which is a global optimum;
- the surface of the criterion function is a hyper-parabola in the form $\frac{1}{2}\,\underline{\theta}^{\mathrm{T}}\underline{H}\,\underline{\theta} + \underline{h}^{\mathrm{T}}\underline{\theta} + h_0$ with n-dimensional vector of parameters $\underline{\theta}$, $n \times n$-dimensional matrix $\underline{H}$, n-dimensional vector $\underline{h}$, and the scalar h_0;
- each step can be calculated analytically;
- there are a lot of numerically stable and fast algorithms;
- recursive forms are possible;
- it can be implemented online.

Linear optimization



Slika 1: Surface of the criterion function in the case of a linear optimization problem with two parameters

Linear Optimization

Let's assume the following structure of the model output $\hat{y}$, i.e. its linear dependence on n of parameters θ_i:

$$\hat{y} = \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n = \sum_{i=1}^{n} \theta_i x_i \qquad \text{at} \quad x_i = g_i(\underline{u}) \tag{1}$$

The variables x_i are called *regressors* or *independent variables*, the parameters θ_i are called *regression coefficients*, y is called *dependent variable*, and the whole problem is called *linear regression*. The number of parameters is equal to the number of regressors.

Least squares method (LS)

$i = 1, \ldots, N$ input-output pairs $\{\underline{u}(i), y(i)\}$,

$$\underline{e} = \begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(N) \end{bmatrix} \quad \underline{y} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} \quad \underline{\hat{y}} = \begin{bmatrix} \hat{y}(1) \\ \hat{y}(2) \\ \vdots \\ \hat{y}(N) \end{bmatrix} \quad \underline{n} = \begin{bmatrix} n(1) \\ n(2) \\ \vdots \\ n(N) \end{bmatrix} \tag{2}$$

$$\underline{X} = \begin{bmatrix} x_1(1) & x_2(1) & \cdots & x_n(1) \\ x_1(2) & x_2(2) & \cdots & x_n(2) \\ \vdots & \vdots & & \vdots \\ x_1(N) & x_2(N) & \cdots & x_n(N) \end{bmatrix} \quad \underline{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \tag{3}$$

Least squares method - LS

Columns in the *regression matrix* $\underline{X}$ are single *regression vectors*

$$\underline{x}_i = \begin{bmatrix} x_i(1) \\ x_i(2) \\ \vdots \\ x_i(N) \end{bmatrix} \quad \text{for } i = 1, \ldots, n \tag{4}$$

Regression matrix can be written

$$\underline{X} = [\underline{x}_1 \; \underline{x}_2 \; \cdots \; \underline{x}_n] \tag{5}$$

Example: The least squares method for polynomial form

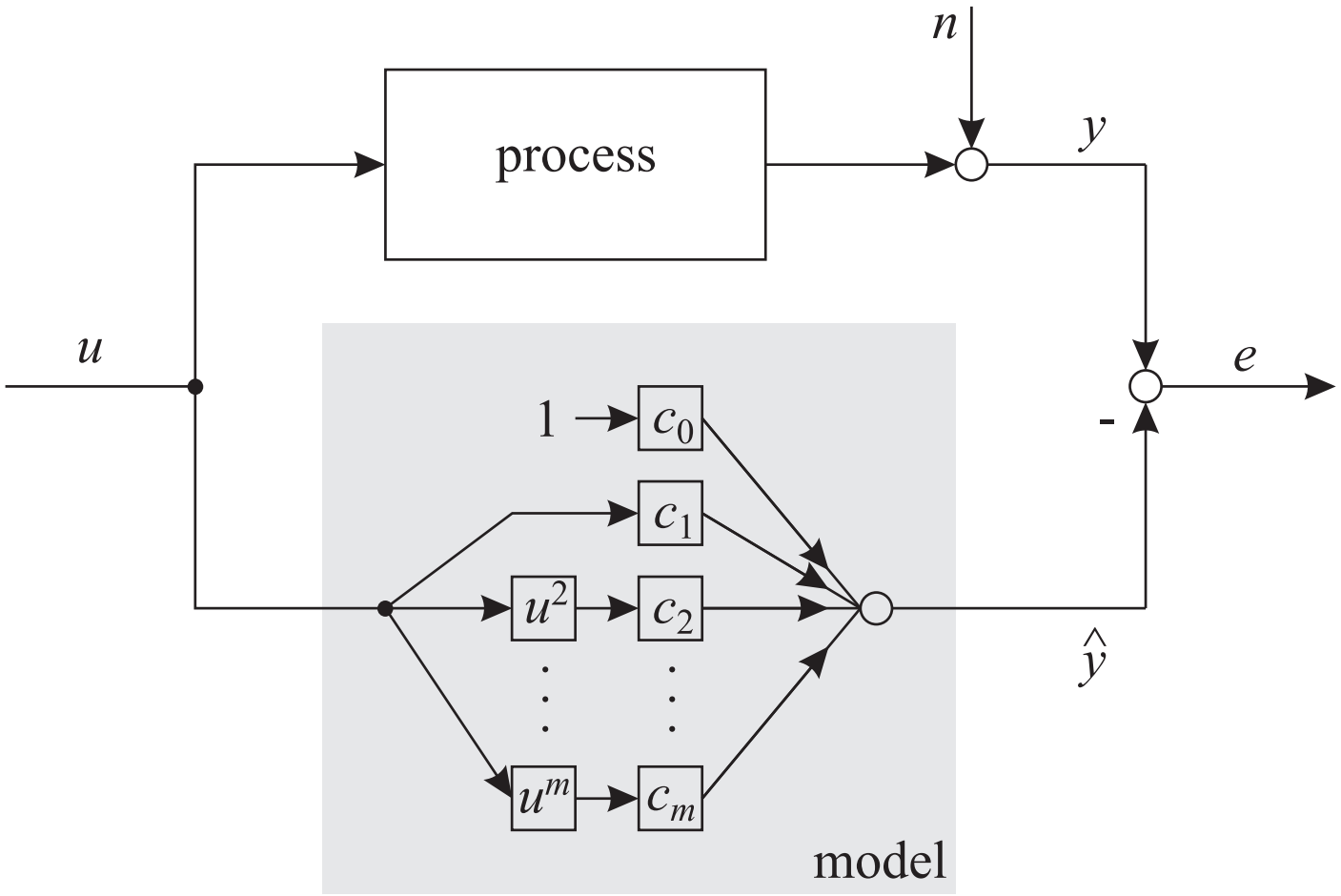
The structure of our model is a m-order polynomial with which we will approximate the data. The output of the model $\hat{y}$ is

$$\hat{y} = c_0 + c_1\, u + c_2\, u^2 + \ldots + c_m\, u^m = \sum_{i=0}^{m} c_i\, u^i \tag{6}$$

We get the error $e(k) = y(k) - \hat{y}(k)$, which is equal to

$$e(k) = y(k) - c_0 - c_1\, u - c_2\, u^2 - \ldots - c_m\, u^m \tag{7}$$

Example



Slika 2: Polynomial model $\hat{y} = c_0 + c_1 u + c_2 u^2 + \ldots + c_m u^m$

Example

The regression matrix $\underline{X}$ for N measurements and the vector of model parameters $\underline{\theta}$ are:

$$\underline{X} = \begin{bmatrix} 1 & u(1) & u^2(1) & \cdots & u^m(1) \\ 1 & u(2) & u^2(2) & \cdots & u^m(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & u(N) & u^2(N) & \cdots & u^m(N) \end{bmatrix} \qquad \underline{\theta} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} \tag{8}$$

Example

Let's look at an example of model identification in the form of a time-discrete linear dynamic process. A model for a one-step-ahead prediction of the output of the process can be in the case of the m-th order dynamics written as

$$\hat{y}(k) = b_1 u(k-1) + \ldots + b_m u(k-m) \; - a_1 y(k-1) - \ldots - a_m y(k-m) \tag{9}$$

or otherwise

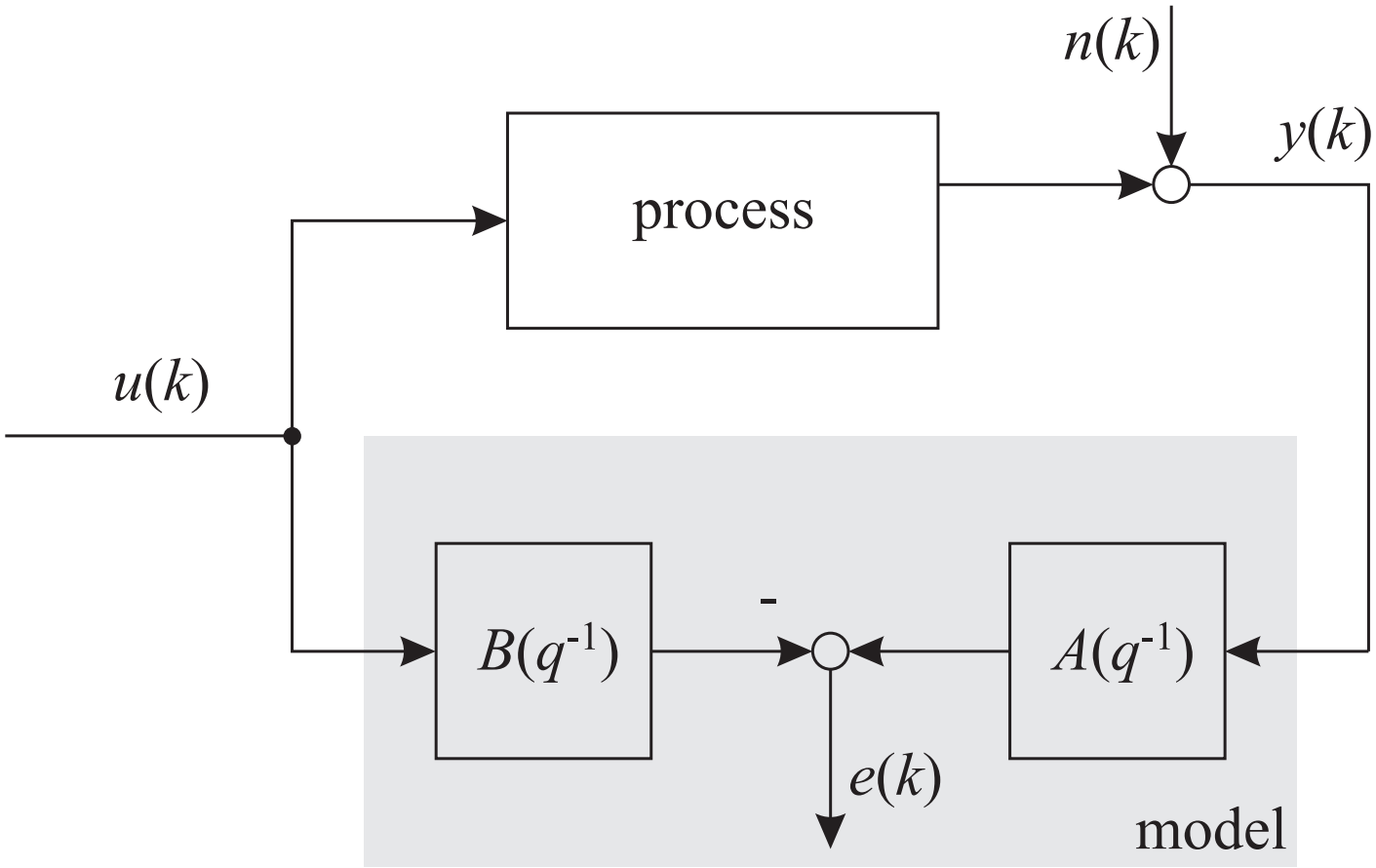
$$\hat{y}(k) = [u(k-1) \; \ldots \; u(k-m) \; - y(k-1) - \; \ldots \; - y(k-m)] \, [b_1 \; \cdots \; b_m \;\; a_1 \; \cdots \; a_m]^{\mathrm{T}} \tag{10}$$

In this case, certain regressors are dependent on the output of the process ($y(k-i)$, $i = 1, \ldots, m$). This implies a violation of the assumption that the regressors depend only on the inputs.

Example

The error can be written $e(k) = y(k) - \hat{y}(k)$ and we get

$$e(k) = y(k) + a_1 y(k-1) \ldots + a_m y(k-m) - b_1 u(k-1) - \ldots - b_m u(k-m) \tag{11}$$



Slika 3: IIR filter for one-step-ahead prediction $B(q^1) = b_1 q^{-1} + b_2 q^{-2} + \ldots + b_m q^{-m}$ and $A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_m q^{-m}$

Example

The regression matrix $\underline{X}$ at N measurement samples and vector of parameters $\underline{\theta}$ are now

$$\underline{X} = \begin{bmatrix} u(m) & \cdots & u(1) & -y(m) & \cdots & -y(1) \\ u(m+1) & \cdots & u(2) & -y(m+1) & \cdots & -y(2) \\ \vdots & & \vdots & \vdots & & \vdots \\ u(N-1) & \cdots & u(N-m) & -y(N-1) & \cdots & -y(N-m) \end{bmatrix} \tag{12}$$

$$\underline{\theta} = [b_1 \ \cdots \ b_m \ \ a_1 \ \cdots \ a_m]^{\mathrm{T}} \tag{13}$$

The regression matrix has the dimensions $(N-m) \times n$.

Explanation of the least squares method

The output of the model can be written in vector form $\underline{\hat{y}} = \underline{X}\,\underline{\theta}$. If we enter this in the least squares method, we get:

$$I(\underline{\theta}) = \frac{1}{2}\,\underline{e}^{\mathrm{T}}\underline{e} \longrightarrow \min_{\underline{\theta}} \qquad \text{pri} \quad \underline{e} = \underline{y} - \underline{\hat{y}} = \underline{y} - \underline{X}\,\underline{\theta} \tag{14}$$

The criterion function is equal to

$$I(\underline{\theta}) = \frac{1}{2}\,\left(\underline{y}^T\underline{y} - \underline{y}^T\,\underline{X}\,\underline{\theta} - \underline{\theta}^T\,\underline{X}^T\,\underline{y} + \underline{\theta}^T\,\underline{X}^T\,\underline{X}\,\underline{\theta}\right) \tag{15}$$

Explanation of the least squares method

The criterion function is a parabolic function if it is presented in the parameter space of the vector $\underline{\theta}$:

$$I(\underline{\theta}) = \frac{1}{2}\,\underline{\theta}^{\mathrm{T}}\underline{H}\,\underline{\theta} + \underline{h}^{\mathrm{T}}\underline{\theta} + h_0 \tag{16}$$

with the square term

$$\underline{H} = \underline{X}^{\mathrm{T}}\underline{X} \tag{17}$$

and the linear term

$$\underline{h} = -\underline{X}^{\mathrm{T}}\underline{y} \tag{18}$$

because it is considered that $\underline{X}^{\mathrm{T}}\underline{y} = \underline{y}^{T}\,\underline{X}$ and the constant

$$h_0 = \frac{1}{2}\underline{y}^{\mathrm{T}}\underline{y} \tag{19}$$

Explanation of the least squares method

A square term is called the $\underline{H}$ Hessian matrix, i.e. second derivative of the criteria function.
In the optimum, the derivative $I(\underline{\theta})$ with respect to the vector of parameters $\underline{\theta}$ is zero. From this it follows that for an optimum the error $\underline{e}$ is orthogonal to all regressors $\underline{x}_i$ (columns of the matrix $\underline{X}$):

$$\frac{\partial I(\underline{\theta})}{\partial \underline{\theta}} = \underline{H\theta} + \underline{h} \tag{20}$$

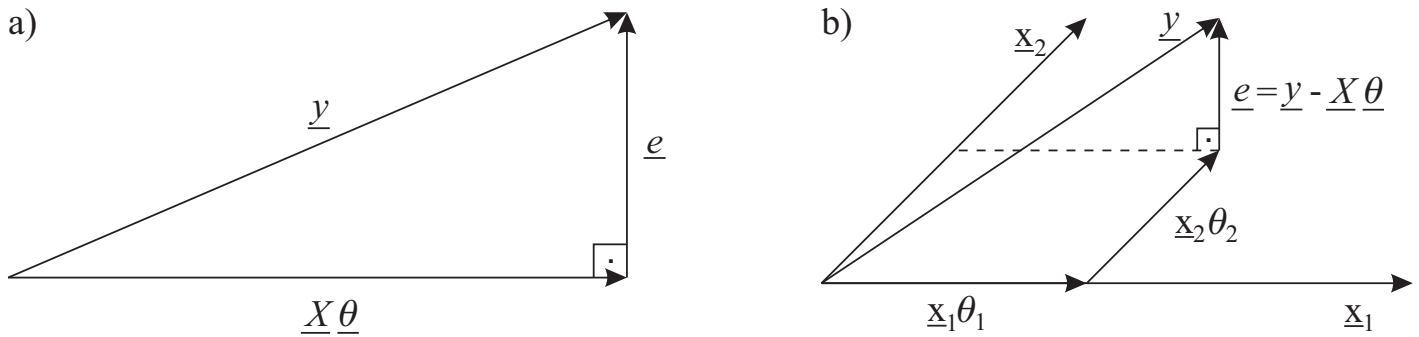
$$\frac{\partial I(\underline{\theta})}{\partial \underline{\theta}} = -\underline{X}^{\mathrm{T}}\left(\underline{y} - \underline{X\theta}\right) = -\underline{X}^{\mathrm{T}}\underline{e} = \underline{0} \tag{21}$$

We get a solution using the least squares method

$$\hat{\underline{\theta}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,. \tag{22}$$

Explanation of the least squares method

The figure shows the orthogonality between the error and the regressors.



Slika 4: Optimum of the criterion function means that the vector $\underline{X}\,\underline{\theta}$ is closest to the vector $\underline{y}$. The orthogonal vector $\underline{e}$ is the smallest and orthogonal to all the regressors of matrix $\underline{X}$: a) projection for the n-regression problem, b) a problem with two regressors

The error $\underline{e}$ between the measured output of the process $\underline{y}$ and the output of the model $\hat{\underline{y}} = \underline{X}\,\hat{\underline{\theta}}$ is called *residual*. In practical cases, we try to determine the quality of the model through the residual. The residual which has the characteristics of the white noise indicates good modelling because a useful piece of information is described with the model, and the remainder is mutually unrelated, which is the property of the white noise.

Least squares method from the correlation perspective

The least squares method can also be represented from the correlation perspective. If we calculate the correlation matrix $\underline{X}^{\mathrm{T}}\underline{X}$

$$\underline{X}^{\mathrm{T}}\underline{X} = N \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} x_1^2(i) & \frac{1}{N}\sum_{i=1}^{N} x_1(i)x_2(i) & \cdots & \frac{1}{N}\sum_{i=1}^{N} x_1(i)x_n(i) \\ \frac{1}{N}\sum_{i=1}^{N} x_2(i)x_1(i) & \frac{1}{N}\sum_{i=1}^{N} x_2^2(i) & \cdots & \frac{1}{N}\sum_{i=1}^{N} x_2(i)x_n(i) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{N}\sum_{i=1}^{N} x_n(i)x_1(i) & \frac{1}{N}\sum_{i=1}^{N} x_n(i)x_2(i) & \cdots & \frac{1}{N}\sum_{i=1}^{N} x_n^2(i) \end{bmatrix} \tag{23}$$

and we calculate $\underline{X}^{\mathrm{T}}\underline{y}$

$$\underline{X}^{\mathrm{T}}\underline{y} = N \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} x_1(i)y(i) \\ \frac{1}{N}\sum_{i=1}^{N} x_2(i)y(i) \\ \vdots \\ \frac{1}{N}\sum_{i=1}^{N} x_n(i)y(i) \end{bmatrix} \tag{24}$$

Least squares method from the correlation perspective

then the estimation of model parameters can be written down as

$$\underline{\hat{\theta}} = \mathrm{corr}\{\underline{x}, \underline{x}\}^{-1} \cdot \mathrm{corr}\{\underline{x}, y\}\,, \tag{25}$$

where $\mathrm{corr}\{\underline{x}, \underline{x}\}$ is an autocorrelation function corresponding to auto and cross correlations of all combinations of regressors x_i and x_j,

$$\mathrm{corr}\{\underline{x}, \underline{x}\} = \begin{bmatrix} \mathrm{corr}\{x_1, x_1\} & \mathrm{corr}\{x_1, x_2\} & \cdots & \mathrm{corr}\{x_1, x_n\} \\ \mathrm{corr}\{x_2, x_1\} & \mathrm{corr}\{x_2, x_2\} & \cdots & \mathrm{corr}\{x_2, x_n\} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{corr}\{x_n, x_1\} & \mathrm{corr}\{x_n, x_2\} & \cdots & \mathrm{corr}\{x_n, x_n\} \end{bmatrix} \tag{26}$$

and $\mathrm{corr}\{\underline{x}, y\}$ represents the cross-correlation vector of all correlations of regressors x_i and the output y,

$$\mathrm{corr}\{\underline{x}, y\} = \begin{bmatrix} \mathrm{corr}\{x_1, y\} \\ \mathrm{corr}\{x_2, y\} \\ \vdots \\ \mathrm{corr}\{x_n, y\} \end{bmatrix} \tag{27}$$

Conditionality of Regression Matrix

In solving the problem of the least squares we name the term $\left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}$ also *pseudo inverse* of the matrix $\underline{X}$, and sometimes we write it as $\underline{X}^{+}$. If $\underline{X}$ has full rank n, which requires at least as many measurements as there are unknown parameters ($N \geq n$), the matrix must have at least as many rows as there are columns. This condition is not sufficient, since the matrix columns must be linearly independent.
To calculate the inverse of the matrix $\underline{X}^{\mathrm{T}}\underline{X}$, which actually represents the Hessian matrix of the criterion function, the degree of its conditionality is decisive.

$$\underline{H} = \frac{\partial^2 I(\underline{\theta})}{\partial \underline{\theta}^2} = \underline{X}^{\mathrm{T}}\underline{X} \tag{28}$$

Conditionality of Regression Matrix

Only in the case of good conditionality of the Hessian matrix do we get a good estimate of the parameters of the model. Conditionality of the matrix χ is defined as the quotient between the maximum and the minimum eigenvalue of the matrix $\underline{H}$. Since the matrix $\underline{H} = \underline{X}^{\mathrm{T}}\underline{X}$ is a symmetric and positive semi-definite matrix, all eigenvalues are real and non-negative. If $\underline{H}$ is non-singular, then all eigenvalues are strictly positive (if the matrix has a full rank):

$$\chi = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{29}$$

Covariance matrix of estimated parameters

The model parameter can be estimated with varying accuracy. The accuracy of the parameters is expressed by the covariance matrix of estimated parameters:

$$\mathrm{cov}\{\underline{\hat{\theta}}\} = \mathrm{E}\left\{\left(\underline{\hat{\theta}} - \mathrm{E}\{\underline{\hat{\theta}}\}\right)\left(\underline{\hat{\theta}} - \mathrm{E}\{\underline{\hat{\theta}}\}\right)^{\mathrm{T}}\right\} \tag{30}$$

where

$$\underline{\hat{\theta}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y} \quad \text{in} \quad \mathrm{E}\{\underline{\hat{\theta}}\} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\mathrm{E}\{\underline{y}\} \tag{31}$$

when $\mathrm{E}\{\underline{X}\} = \underline{X}$, i.e., $\underline{X}$ is deterministic.
That is why we get

$$\underline{\hat{\theta}} - \mathrm{E}\{\underline{\hat{\theta}}\} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\left(\underline{y} - \mathrm{E}\{\underline{y}\}\right) \tag{32}$$

Covariance matrix of estimated parameters

Since it is considered that $\underline{y} - \mathrm{E}\{\underline{y}\} = \underline{y} - \mathrm{E}\{\underline{y}_u + \underline{n}_y\} = \underline{y} - \underline{y}_u = \underline{n}_y$, the parameter estimates depend on the noise properties. We get

$$\begin{aligned} \mathrm{cov}\{\hat{\underline{\theta}}\} &= \mathrm{E}\left\{\left(\left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{n}_y\right)\left(\left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{n}_y\right)^{\mathrm{T}}\right\} \\ &= \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\mathrm{E}\{\underline{n}_y\,\underline{n}_y^{\mathrm{T}}\}\underline{X}\left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1} \end{aligned} \tag{33}$$

because the Hessian matrix is symmetric: $\left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{\mathrm{T}} = \underline{X}^{\mathrm{T}}\underline{X}$. If n_y is the white noise with the variance σ^2, then $\mathrm{E}\{\underline{n}_y\,\underline{n}_y{}^{\mathrm{T}}\} = \sigma^2\underline{I}$ ($\mathrm{E}\{n_y(i)n_y(i)\} = \sigma^2$ and $\mathrm{E}\{n_y(i)n_y(j)\} = 0$ for $i \neq j$. We get the final covariance matrix of estimated parameters

$$\mathrm{cov}\{\hat{\underline{\theta}}\} = \sigma^2\left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1} = \sigma^2\underline{H}^{-1} \tag{34}$$

Covariance matrix of estimated parameters

$$\hat{\sigma}^2 = \frac{\underline{e}^{\mathrm{T}}\underline{e}}{N-n} \tag{35}$$

Confidence interval

$$
\begin{aligned}
\mathrm{cov}\{\underline{\hat{y}}\} &= \mathrm{E}\left\{\left(\underline{\hat{y}} - \mathrm{E}\{\underline{\hat{y}}\}\right)\left(\underline{\hat{y}} - \mathrm{E}\{\underline{\hat{y}}\}\right)^{\mathrm{T}}\right\} \\
&= E\left\{\left(\underline{X}\left(\underline{\hat{\theta}} - \mathrm{E}\{\underline{\hat{\theta}}\}\right)\right)\left(\underline{X}\left(\underline{\hat{\theta}} - \mathrm{E}\{\underline{\hat{\theta}}\}\right)\right)^{\mathrm{T}}\right\} \\
&= \underline{X}\,\mathrm{E}\left\{\left(\underline{\hat{\theta}} - \mathrm{E}\{\underline{\hat{\theta}}\}\right)\left(\underline{\hat{\theta}} - \mathrm{E}\{\underline{\hat{\theta}}\}\right)^{\mathrm{T}}\right\}\underline{X}^{\mathrm{T}}
\end{aligned}
\tag{36}
$$

So we get the covariance matrix of the model output $\underline{\hat{y}}$

$$
\mathrm{cov}\{\underline{\hat{y}}\} = \underline{X}\,\mathrm{cov}\{\underline{\hat{\theta}}\}\,\underline{X}^{\mathrm{T}} = \hat{\sigma}^2\underline{X}\;\underline{H}^{-1}\,\underline{X}^{\mathrm{T}} \tag{37}
$$

Confidence interval

The tolerance band $\hat{\underline{y}}$ is defined as the positive and negative offset of the standard deviation (or even a multiple of standard deviation) from the estimated output of the model:

$$\hat{\underline{y}} \pm \sqrt{\mathrm{diag}\left(\mathrm{cov}\{\hat{\underline{y}}\}\right)} \tag{38}$$

If we want to determine the so called confidence interval of the output signal of the model, then we must ensure that the output with a certain probability lies within the prescribed interval with the known distribution function of the noise. If we assume the Gaussian distribution, then the interval of one σ determines 68.2 % probability, and interval of 2σ means 95.4 % probability that the output lies within the interval.

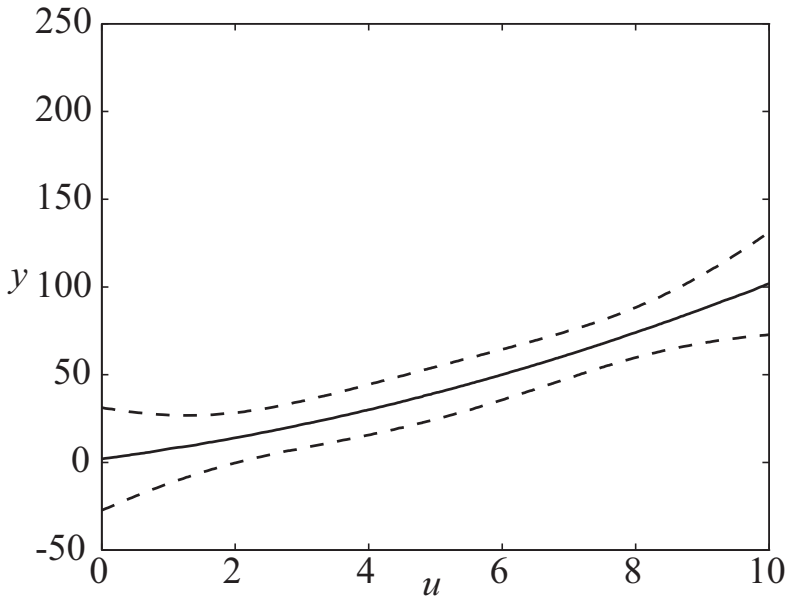
Example: Tolerance band and missing data

The effect of the missing data on the tolerance band will be presented on the example of the function

$$y = 0.5u^2 + 5u + 2 \tag{39}$$

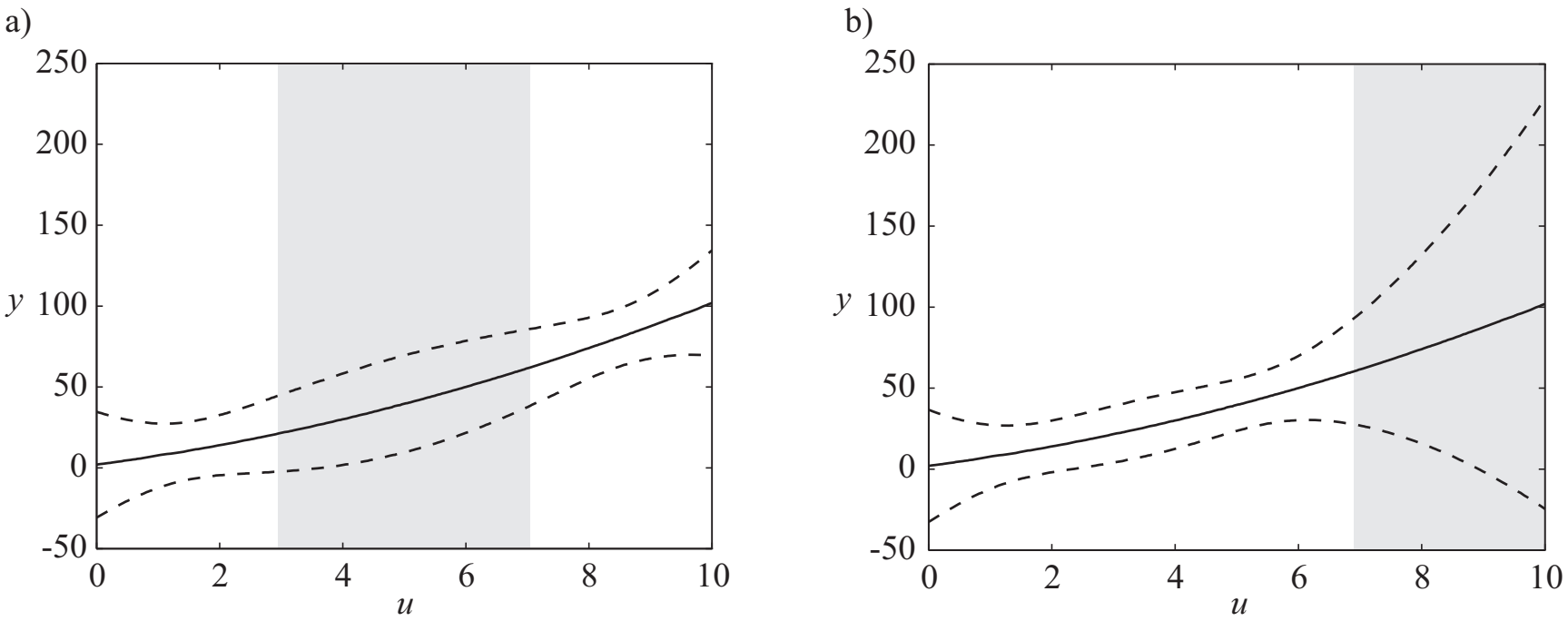
which we will approximate with the second-order polynomial on the basis of a learning set that contains 1000 data which are evenly distributed in the interval [0, 10]. The figures show three different examples:

Example: Tolerance band and missing data



Slika 5: Approximation of the function $y = 0.5u^2 + 5u + 2$ with second-order polynomial. The learning data is evenly distributed on the interval [0, 10]. The dashed line denotes a tolerance band. The width of the tolerance band increases at the edges of the learning interval. This is due to missing data to the left of point 0 and right of point 10.

Example: Tolerance band and missing data



Slika 6: a) Missing data on interval (3, 7). b) Missing data on interval (7, 10). The error of the estimated output of the model increases in the area where there are no learning data. We find that the tolerance band in example (b) is much greater than in example (a), although more data is missing in example (a). The reason for this is that in example (b) at $u > 7$ we deal with an extrapolation problem

Orthogonal regressors

A very important special example of the least squares method is obtained if the regressors are orthogonal among themselves, i.e., $\underline{x}_i^{\mathrm{T}}\underline{x}_j = 0$ for $i \neq j$. In this case, Hessian matrix $\underline{H}$ is

$$\underline{H} = \underline{X}^{\mathrm{T}}\underline{X} = \text{diag}\left(\underline{x}_1^{\mathrm{T}}\underline{x}_1,\ \underline{x}_2^{\mathrm{T}}\underline{x}_2,\ \cdots,\ \underline{x}_n^{\mathrm{T}}\underline{x}_n\right) \tag{40}$$

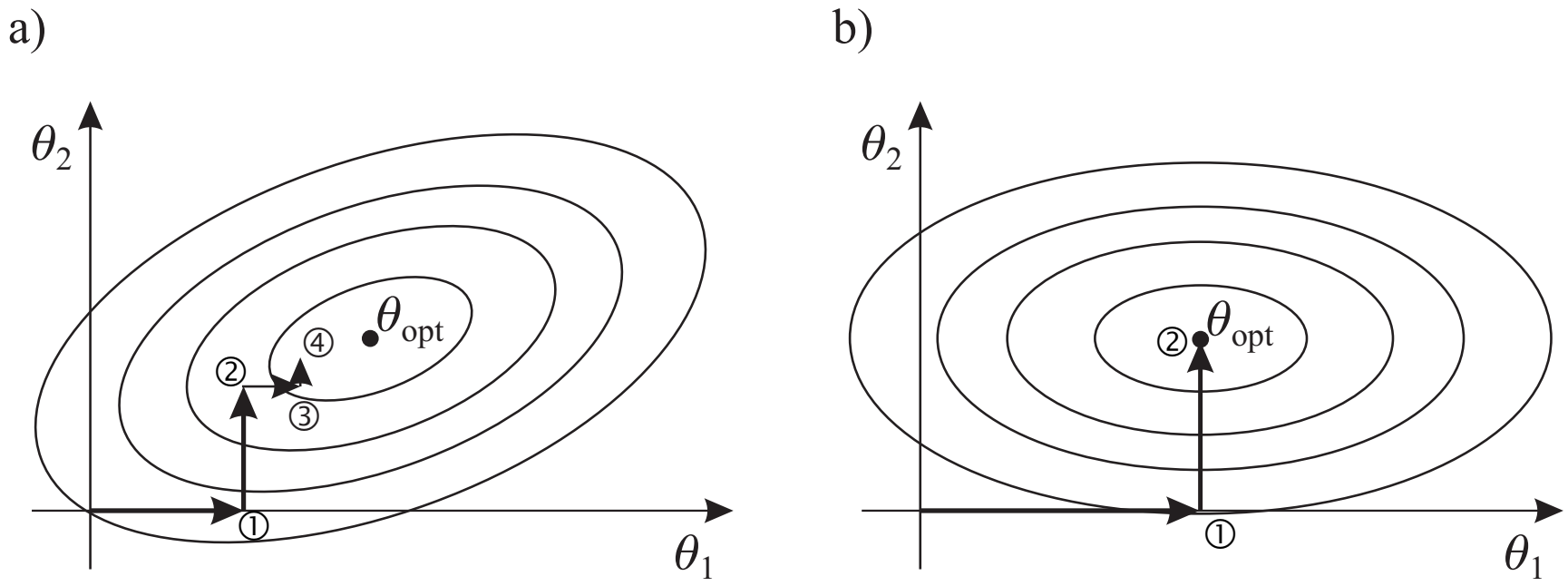
then the inverse becomes:

$$\underline{H}^{-1} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1} = \text{diag}\left(\frac{1}{\underline{x}_1^{\mathrm{T}}\underline{x}_1},\ \frac{1}{\underline{x}_2^{\mathrm{T}}\underline{x}_2},\ \cdots,\ \frac{1}{\underline{x}_n^{\mathrm{T}}\underline{x}_n}\right) \tag{41}$$

The optimal model parameters are then:

$$\hat{\underline{\theta}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y} = \text{diag}\left(\frac{1}{\underline{x}_1^{\mathrm{T}}\underline{x}_1},\ \frac{1}{\underline{x}_2^{\mathrm{T}}\underline{x}_2},\ \cdots,\ \frac{1}{\underline{x}_n^{\mathrm{T}}\underline{x}_n}\right)\underline{X}^{\mathrm{T}}\underline{y} \tag{42}$$

Orthogonal regressors



Slika 7: Contour diagram for the criteria functions of a) non-orthogonal and b) orthogonal regressors. If the regressors are orthogonal we can calculate the minimum of the function by calculating optimizations for individual parameters. This significantly reduces the complexity of the problem. In the case of non-orthogonal regressors, we must repeat several steps of optimization for individual parameters. The more the regressors are orthogonal, the conditionality of the Hessian matrix is better and optimization convergence is faster.

Orthogonal regressors

Each parameter θ_i from the vector $\underline{\theta}$ can be specified separately

$$\hat{\theta}_i = \frac{\underline{x}_i^{\mathrm{T}} \underline{y}}{\underline{x}_i^{\mathrm{T}} \underline{x}_i} \tag{43}$$

The optimal parameter $\hat{\theta}_i$ depends only on the regressor $\underline{x}_i$ and the measured output $\underline{y}$. There should be no correlation between other regressors in this case. For the case of non-orthogonal regressors, we can say that with the inverse of the Hessian matrix $\underline{H}^{-1}$ we decorrelate the correlated regressors.

Regularization

Methods where in the criterion function we take into account also the values of the model parameters are called *regularization methods*. In this way, of course, we also indirectly influence the variance of parameters:

$$I(\underline{\theta}, \alpha) = \frac{1}{2}\left(\underline{e}^{\mathrm{T}}\underline{e} + \alpha|\underline{\theta}|^2\right) \longrightarrow \min_{\underline{\theta}} \tag{44}$$

The background of adding an term $\alpha|\underline{\theta}|^2$ is easy. Parameters that do not have a major influence on the solution of the equation strive for smaller values, because this reduces the value of the criterion function. In this way, only the parameters that are indispensable remain, since their influence on the output signal is more important than the contribution in the second term of the criterion function.
Regularized least squares method is written:

$$\hat{\underline{\theta}} = \left(\underline{X}^{\mathrm{T}}\underline{X} + \alpha\underline{I}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y} \tag{45}$$

Example: Regularization and conditionality of the matrix

The conditionality of the Hessian matrix

$$\underline{H} = \begin{bmatrix} 100 & 0 \\ 0 & 0.01 \end{bmatrix}$$

is equal to $\chi = 10000$. For $\alpha = 1$ the conditionality of the modified Hessian matrix is

$$\underline{H}_{\text{reg}} = \begin{bmatrix} 100 & 0 \\ 0 & 0.01 \end{bmatrix} + 1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 101 & 0 \\ 0 & 1.01 \end{bmatrix}$$

only $\chi_{\text{reg}} \approx 100$.

Example: Regularization and contour diagram

This example illustrates the effect of regularization on the form of the criterion function. In the example, the following criterion function is selected:

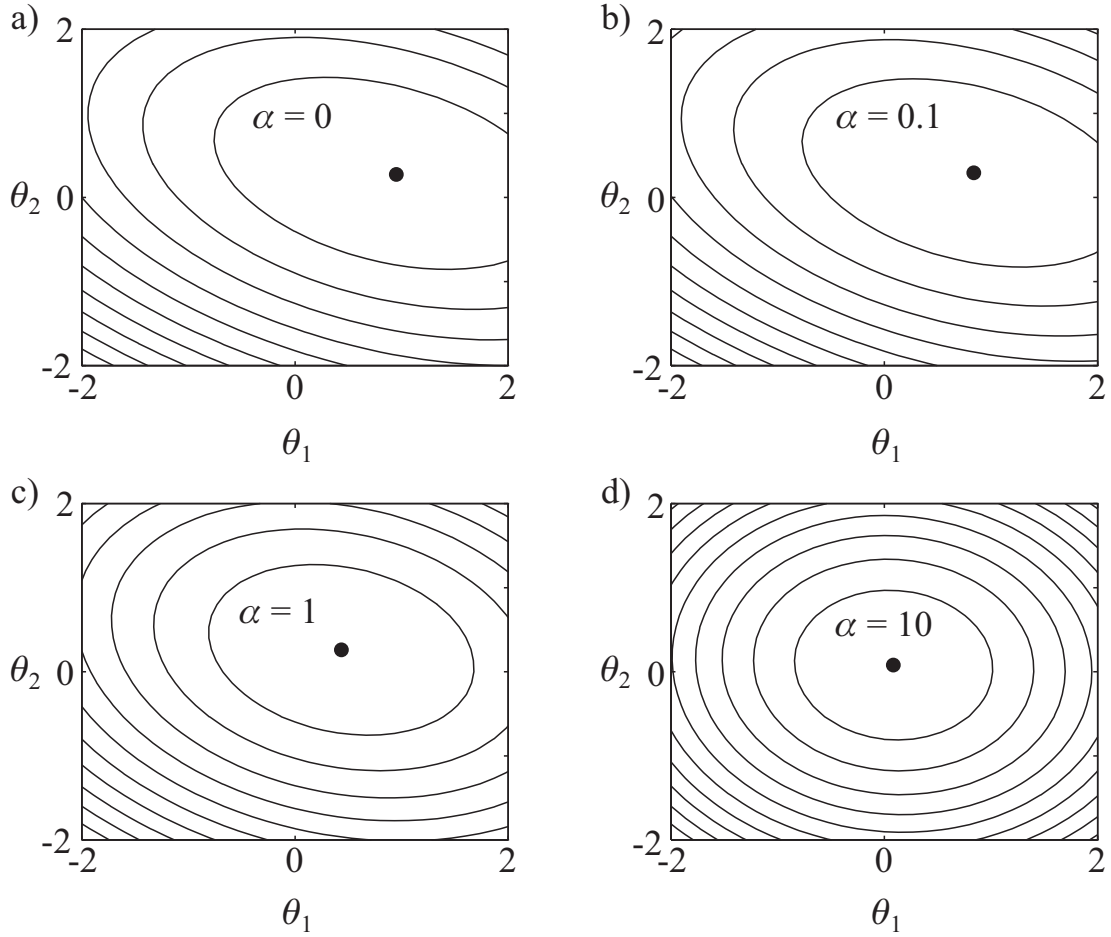
$$I(\underline{\theta}) = (\theta_1 - 1)^2 + 2\,(\theta_2 - 0.5)^2 + \theta_1\theta_2 \tag{46}$$

The contour diagram is shown in the figure. The minimum is at $\underline{\theta} \approx [0.86\ 0.29]^{\mathrm{T}}$. Regularization with parameter α gives the following criterion function:

$$I(\underline{\theta}, \alpha) = (\theta_1 - 1)^2 + 2\,(\theta_2 - 0.5)^2 + \theta_1\theta_2 + \alpha\,(\theta_1^2 + \theta_2^2) \tag{47}$$

Example: Regularization and contour diagram

The figures show the effect of the regularization parameter α on the shape of the contour diagram. If α increases, the minimum of the criterion function approaches the value $[0\ 0]$ and the contour diagram goes from the ellipse to the circle. This means that the Hessian matrix is getting better conditioned.



Slika 8: Contour curves of the criterion function for a different regularization parameter α. The higher value of the regularization parameter leads to more rounded contours of the graph, where the minimum is near the origin of the coordinate system.

Weighted least squares method (WLS)

The least squares method can be expanded into the weighted least squares method. In this case, the criterion function can be written as:

$$I(\underline{\theta}) = \frac{1}{2}\,\underline{e}^{\mathrm{T}}\underline{Q}\,\underline{e} \tag{48}$$

where $\underline{Q}$ is the weight matrix. Most often the weight matrix has diagonal shape:

$$\underline{Q} = \begin{bmatrix} Q_{11} & 0 & \cdots & 0 \\ 0 & Q_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_{NN} \end{bmatrix} \tag{49}$$

Weighted least squares method (WLS)

$$\underline{\hat{\theta}} = \left(\underline{X}^{\mathrm{T}}\underline{Q}\,\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{Q}\,\underline{y} \tag{50}$$

The weighted least squares method is used wherever the samples and associated errors have different meaning or significance for the estimation of parameters. It is very important in the estimation of the parameters of fuzzy models in Takagi-Sugeno form. It can also be used when different tolerance and accuracy of the model is required in different areas. In this case we select large weights in the area of high accuracy and smaller ones in the area of greater tolerance.

Recursive least squares method (RLS)

So far, we have assumed that all the data $\{\underline{u}(i), y(i)\}$ that form the matrix $\underline{X}$ and the vector $\underline{y}$ for the execution of methods are available from the beginning and are processed all at the same time. When, for various reasons, we require that the method should work in real time, then we have to implement the method in *recursive form (RLS)*, which calculates the change of model parameters with regard to the previous value $\hat{\underline{\theta}}$ every time when new measurements appear. The method requires equidistant time shifts for parameter adaptation.

The correction vector depends on the measurements of the regressor

$$\underline{x}(k) = [x_1(k)x_2(k)\ldots x_n(k)]^{\mathrm{T}} \tag{51}$$

and the measured output of the process $y(k)$.

Recursive least squares method - RLS

In this notation, the regressor $\underline{x}(k)$ is a n-dimensional vector which contains measurements of all input signals that occur in regression in the k-th measurement. This means k-th row in the matrix $\underline{X}$.

The recursive least squares method can be written:

$$\begin{aligned} \underline{\hat{\theta}}(k) &= \underline{\hat{\theta}}(k-1) + \underline{P}(k)\underline{x}(k)e(k) \\ &\quad \text{at} \quad \underline{P}^{-1}(k) = \underline{P}^{-1}(k-1) + \underline{x}(k)\underline{x}^{\mathrm{T}}(k) \end{aligned} \tag{52}$$

where $\underline{P}^{-1}(k) = \underline{X}^{\mathrm{T}}(k)\underline{X}(k)$ is the approximation of the Hessian matrix $\underline{\hat{H}}$ and therefore $\underline{P}(k)$ is proportional to covariance matrix of estimated parameters.

The single-step prediction error $e(k) = y(k) - \underline{x}^{\mathrm{T}}(k)\underline{\hat{\theta}}(k-1)$ is the error between the measured output of the process $y(k)$ and single-step prediction of the output $\hat{y}(k|k-1) = \underline{x}^{\mathrm{T}}(k)\underline{\hat{\theta}}(k-1)$, where the vector of parameters is estimated from all old measurements.

Reducing computational complexity

By introducing the inverse matrix theorem we get a simplified RLS algorithm:

$$\underline{\hat{\theta}}(k) = \underline{\hat{\theta}}(k-1) + \underline{\gamma}(k)e(k)\,, \quad e(k) = y(k) - \underline{x}^{\mathrm{T}}(k)\underline{\hat{\theta}}(k-1) \tag{53}$$

$$\underline{\gamma}(k) = \frac{1}{\underline{x}^{\mathrm{T}}(k)\underline{P}(k-1)\underline{x}(k)+1}\,\underline{P}(k-1)\underline{x}(k) \tag{54}$$

$$\underline{P}(k) = \left(\underline{I} - \underline{\gamma}(k)\,\underline{x}^{\mathrm{T}}(k)\right)\underline{P}(k-1) \tag{55}$$

The recursive method of weighted least squares (RWLS), where the weighting of the data $\underline{x}(k)$ is defined with $q(k)$ becomes:

$$\underline{\hat{\theta}}(k) = \underline{\hat{\theta}}(k-1) + \underline{\gamma}(k)e(k)\,, \quad e(k) = y(k) - \underline{x}^{\mathrm{T}}(k)\underline{\hat{\theta}}(k-1) \tag{56}$$

$$\underline{\gamma}(k) = \frac{1}{\underline{x}^{\mathrm{T}}(k)\underline{P}(k-1)\underline{x}(k)+1/q(k)}\,\underline{P}(k-1)\underline{x}(k) \tag{57}$$

$$\underline{P}(k) = \left(\underline{I} - \underline{\gamma}(k)\,\underline{x}^{\mathrm{T}}(k)\right)\underline{P}(k-1) \tag{58}$$

Identification of time-varying processes

RLS generally converts to optimal parameters. In the case of a large number of samples ($k \to \infty$), the convergence degree decreases because $\underline{P}$ goes to zero. This is not problematic if it is a process that is time-invariat. If it is a time-varying process, then this presents a problem. In this case, it is often necessary to evaluate time-varying parameters, which can be achieved if we make sure that $\underline{P}$ does not become too small. This can be achieved in many ways. One of them is the introduction of the forgetting factor $\lambda \leq 1$. The data that was captured j samples ago is weighted with the value λ^j. In this way we introduce exponential forgetting in the RLS algorithm

$$\underline{\hat{\theta}}(k) = \underline{\hat{\theta}}(k-1) + \underline{\gamma}(k)e(k)\,, \quad e(k) = y(k) - \underline{x}^{\mathrm{T}}(k)\underline{\hat{\theta}}(k-1) \tag{59}$$

$$\underline{\gamma}(k) = \frac{1}{\underline{x}^{\mathrm{T}}(k)\underline{P}(k-1)\underline{x}(k) + \lambda}\,\underline{P}(k-1)\underline{x}(k) \tag{60}$$

$$\underline{P}(k) = \frac{1}{\lambda}\left(\underline{I} - \underline{\gamma}(k)\,\underline{x}^{\mathrm{T}}(k)\right)\underline{P}(k-1) \tag{61}$$

Identification of time-varying processes

In the case of weighted least squares with forgetting, where the data vector $\underline{x}(k)$ is weighted with the weight of $q(k)$, we get:

$$\hat{\underline{\theta}}(k) = \hat{\underline{\theta}}(k-1) + \underline{\gamma}(k)e(k)\,, \quad e(k) = y(k) - \underline{x}^{\mathrm{T}}(k)\hat{\underline{\theta}}(k-1) \tag{62}$$

$$\underline{\gamma}(k) = \frac{1}{\underline{x}^{\mathrm{T}}(k)\underline{P}(k-1)\underline{x}(k) + \lambda/q(k)}\,\underline{P}(k-1)\underline{x}(k) \tag{63}$$

$$\underline{P}(k) = \frac{1}{\lambda}\left(\underline{I} - \underline{\gamma}(k)\,\underline{x}^{\mathrm{T}}(k)\right)\underline{P}(k-1) \tag{64}$$

The forgetting factor λ has a usual value at an interval between 0.9 and 1. When choosing $\lambda = 1$ we get RLS without forgetting, and all data are weighted, no matter how far they are in the past. In the case of $\lambda = 0.9$, the new data is 3, 8, 24, and 68 times more weighted than data which are 10, 20, 30 and 40 samples in the past. The setting of λ is a compromise between the high robustness for the disturbance (large λ) and the rapid tracking of the changes in the estimated process parameters (small λ). The system can adjust the value of the forgetting factor according to the excitation of the process. In case of high excitation, the new information is large and factor λ can be reduced, otherwise we increase it. In the case of RLS it is sensible to observe the trace of the $\underline{P}$ matrix and react appropriately if its value approaches zero. We can reset $\underline{P}$ matrix to the initial value, or by changing the forgetting factor we can prevent that the trace of $\underline{P}$ matrix falls below a certain value.

Selection of regressors

The measured output of the process $\underline{y}$ can be represented as a prediction of the model $\underline{\hat{y}} = \underline{X}\,\underline{\hat{\theta}}$ and the error prediction $\underline{e}$

$$\underline{y} = \underline{X}\,\underline{\hat{\theta}} + \underline{e} \tag{65}$$

or in an extended form

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} x_1(1) & x_2(1) & \cdots & x_n(1) \\ x_1(2) & x_2(2) & \cdots & x_n(2) \\ \vdots & \vdots & & \vdots \\ x_1(N) & x_2(N) & \cdots & x_n(N) \end{bmatrix} \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \vdots \\ \hat{\theta}_n \end{bmatrix} + \begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(N) \end{bmatrix} \tag{66}$$

Usually, the goal in modelling is the search for a group of regressors that approximates $\underline{y}$ to a certain accuracy with the least number of regressors. This means that from the set of all possible regressors we have to choose the most influential ones. The problem also occurs while searching the process order or the number of delays in the case of linear dynamic processes and the order of polynomial or base functions in the modelling of nonlinear processes. Therefore, the methods for selection of regressors are very important. We will limit ourselves to methods where the models are linear in the parameters.

Methods for selecting regressors

The simplest solution would be to test all possible combinations of regressors, which are $2^n - 1$,

- forward selection,
- reverse selection,
- gradual selection.

Forward selection:

- First for each regressor of n possible regressors we calculate the optimal parameters
- We select the one that gives the best possible approximation of the output $\underline{y}$. We designate this regressor and the associated parameter with $\underline{x}_A$ and $\hat{\theta}_A$.
- Then, part of the output $\underline{y}$, which we do not manage to approximate is written as $\underline{y}_A = \underline{y} - \underline{x}_A \hat{\theta}_A$.
- In continuation, we repeat the same for all other non-selected $n-1$ regressors on the remainder $\underline{y}_A$.
- The second selected regressor is written as $\underline{x}_B$ and its corresponding parameter as $\hat{\theta}_B$.
- Then we calculate the remainder of the output $\underline{y}_B = \underline{y}_A - \underline{x}_B \hat{\theta}_B$, which we must approximate with other regressors.
- This procedure is repeated until we get n_s of the regressors that approximate the output so that the residual is within certain limits.

Methods for selecting regressors

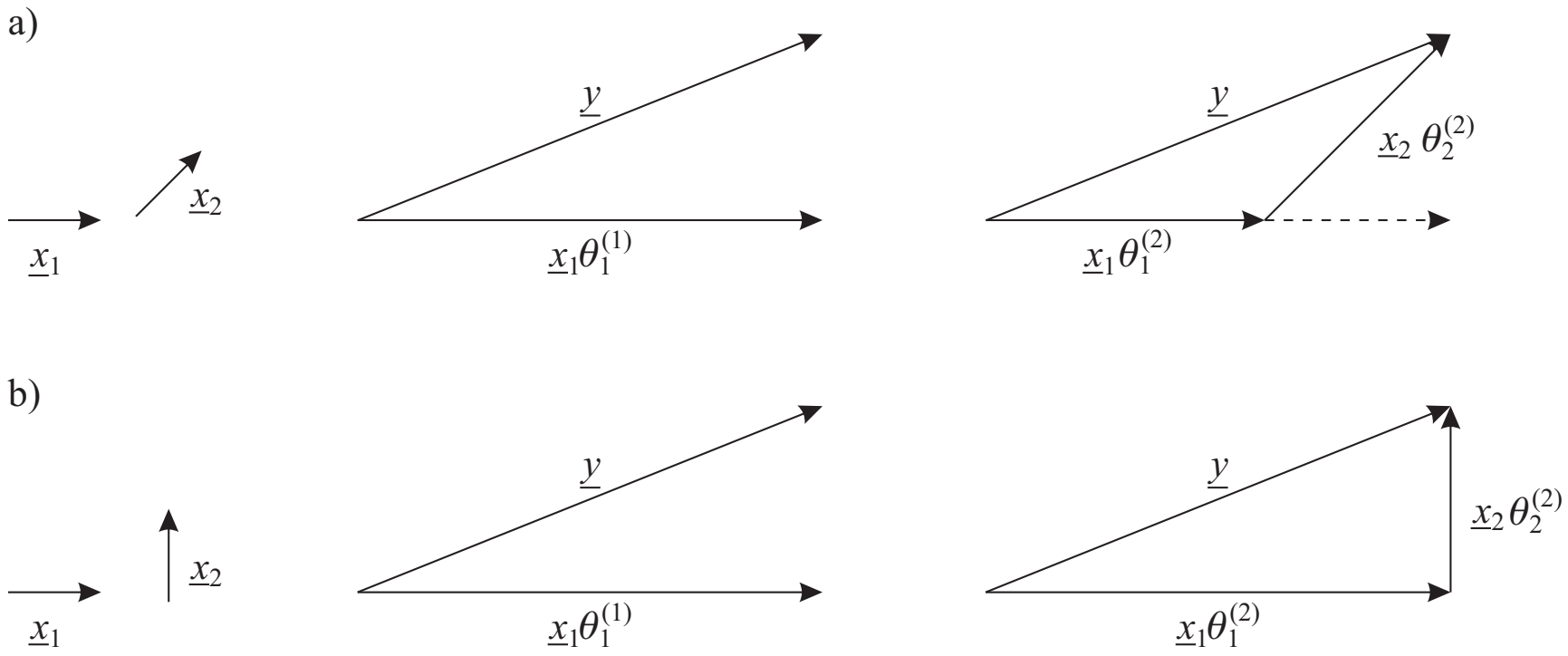
Iterative use of the correlation method.

- The parameters of the selected regressors are selected in each sequence individually, while for the correct solution it is necessary to optimize all the parameters simultaneously and iteratively calculate the remainder when approximating.

Using the Gram-Schmidt orthogonalization.

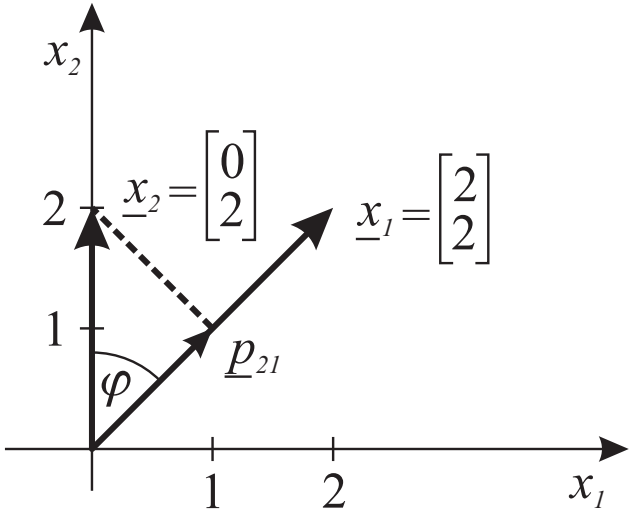
- The vectors of the regressors are orthogonalized in such a way that they are orthogonal to the previously selected regressor vector.
- The vectors of the regressors are *orthogonalized* with the **Gram-Schmidt** method.
- Since the normal number of regressor vectors n is large and the number of selected regressors n_s is small, it would be inefficient to orthogonalize the entire matrix $\underline{X}$.
- An effective procedure is such that we first look for the most influential regressor and then the rest of $n-1$ unsuitable regressors are orthogonalized according to the selected one.
- In the second step one of the $n-1$ regressors that best describes the output signal is selected, and then the rest of $n-2$ non-selected regressors are again orthogonalized according to the selected one, and so on.
- Since the regressors are orthogonal, the choice of parameters that belong to them is simple.

Methods for selecting regressors



Slika 9: Non-orthogonal and orthogonal regressors $\underline{x}_1$ and $\underline{x}_2$: a) If regressors are non-orthogonal, optimal parameters depend on the correlation between the regressors. This means that when choosing a new regressor, we need to optimize all parameters again. b) If regressors are orthogonal, then optimal parameters are independent of each other and individual parameters can be calculated independently

Methods for selecting regressors: Gram-Schmidt orthogonalization



$$\underline{x_1}^T \cdot \underline{x}_2 = |\underline{x}_1| \cdot |\underline{x}_2| \cdot \cos\varphi \tag{67}$$

$$\left|\underline{p}_{21}\right| = \cos\varphi \cdot |\underline{x}_2| = \frac{\underline{x}_1^T \cdot \underline{x}_2}{|\underline{x}_1|} \tag{68}$$

$$\underline{p}_{21} = \left|\underline{p}_{21}\right| \cdot \frac{\underline{x}_1}{|\underline{x}_1|} \tag{69}$$

$$\underline{p}_{21} = \frac{\underline{x}_1^T \cdot \underline{x}_2}{|\underline{x}_1|^2} \cdot \underline{x}_1 \tag{70}$$

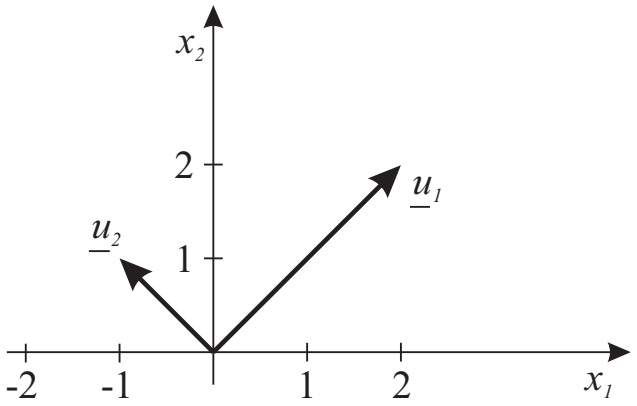
Methods for selecting regressors: Gram-Schmidt orthogonalization

$$\underline{u}_1 = \underline{x}_1 \tag{71}$$

$$\underline{u}_2 = \underline{x}_2 - \underline{p}_{21} = \underline{x}_2 - \alpha \cdot \underline{x}_1 \tag{72}$$

$$\underline{u}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \tag{73}$$

$$\underline{u}_2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix} - \frac{\begin{bmatrix} 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 2 \end{bmatrix}^T}{\begin{bmatrix} 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} 2 & 2 \end{bmatrix}^T} \cdot \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{74}$$



- Example: Matlab

Local nonlinear optimization

Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for lectures, March 2016

Table of contents

1. Local nonlinear optimization
2. Optimization with a data set and by individual data
3. Initializing parameters
4. Direct optimum search algorithms
   - Finite differential method
5. Gradient optimization methods
   - Maximum gradient method
   - Newton's method
   - Quasi-Newton method
   - Conjugate gradient method
6. Nonlinear least squares method
   - Gauss-Newton method
   - Levenberg-Marquardt method

Local nonlinear optimization

Optimization is linear if the gradient of the criterion function $I(\underline{\theta})$ is linear in parameters.

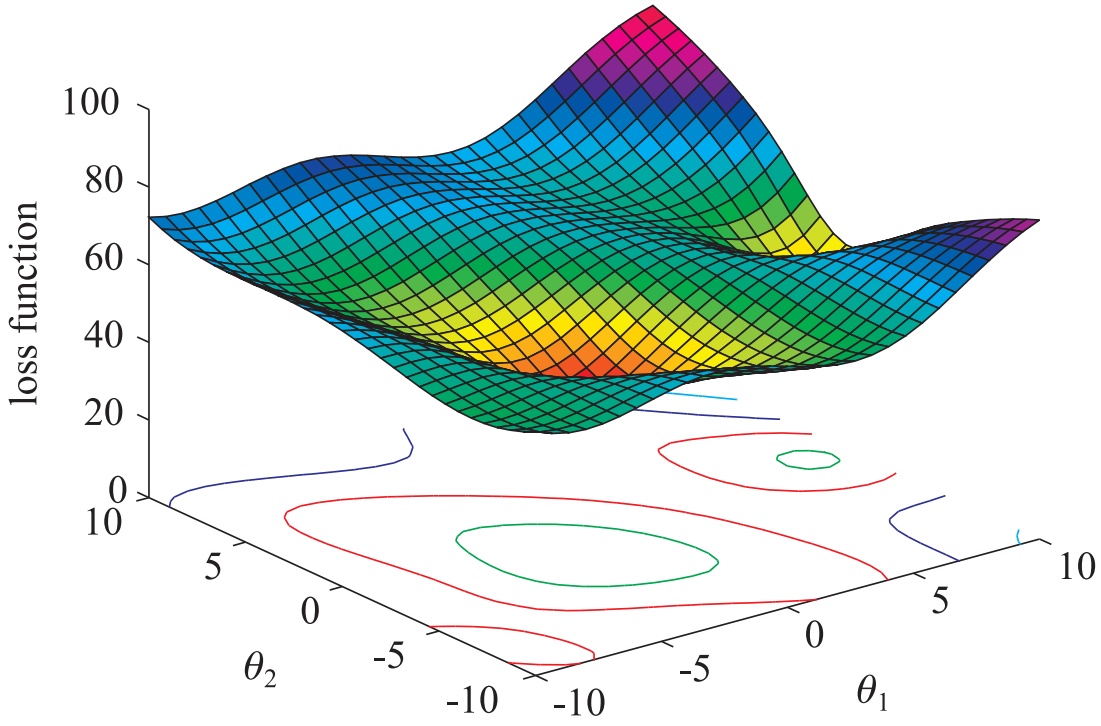
- If the gradient of the criterion function $I(\underline{\theta})$ is nonlinear in parameters $\underline{\theta}$, we need to use nonlinear optimization technique when searching for optimal parameters $\underline{\theta}_{opt}$
- In this case the parameters are called *nonlinear parameters*.
- Nonlinear parameters are weights in hidden layers of neural networks, positions and width of membership functions in fuzzy systems. Even in the context of linear systems we can encounter problems with nonlinear parameters that are related to the minimization of the output error of the model.

Nonlinear optimization problems typically have the following properties:

- there is a number of local optimums (see Fig. 1),
- hyper-surface area in the vicinity of a local optimum can be approximated (the use of second-order Taylor series) with a hyper-parabola in the form $\underline{\theta}^{\mathrm{T}}\underline{A}\,\underline{\theta} + \underline{b}^{\mathrm{T}}\underline{\theta} + c$,
- analytical solutions do not exist,
- the solution can be found only iteratively,
- on-line optimization is practically unfeasible.

Local nonlinear optimization

- Methods of local nonlinear optimization.
- Here we mean optimization where we select the initial search point and we only explore the surroundings of this point.
- In this way we find the closest local optimum.
- Exploration of the wider problem space can be expanded by starting an algorithm from a variety of starting points (*m*ulti-start technique) and finally choosing the best solution.



Slika 1: Criterion function for nonlinear optimization problem with multiple local minima

Example: Rosenbrock function

We will look at the most basic methods of local nonlinear optimization in the case of Rosenbrock function.

- The function $I(\underline{\theta})$ is shown in Fig. 2 and is defined by two parameters θ_1 and θ_2

$$I(\underline{\theta}) \;=\; 100\,(\theta_2 - \theta_1^2)^2 + (1-\theta_1)^2\,. \tag{1}$$

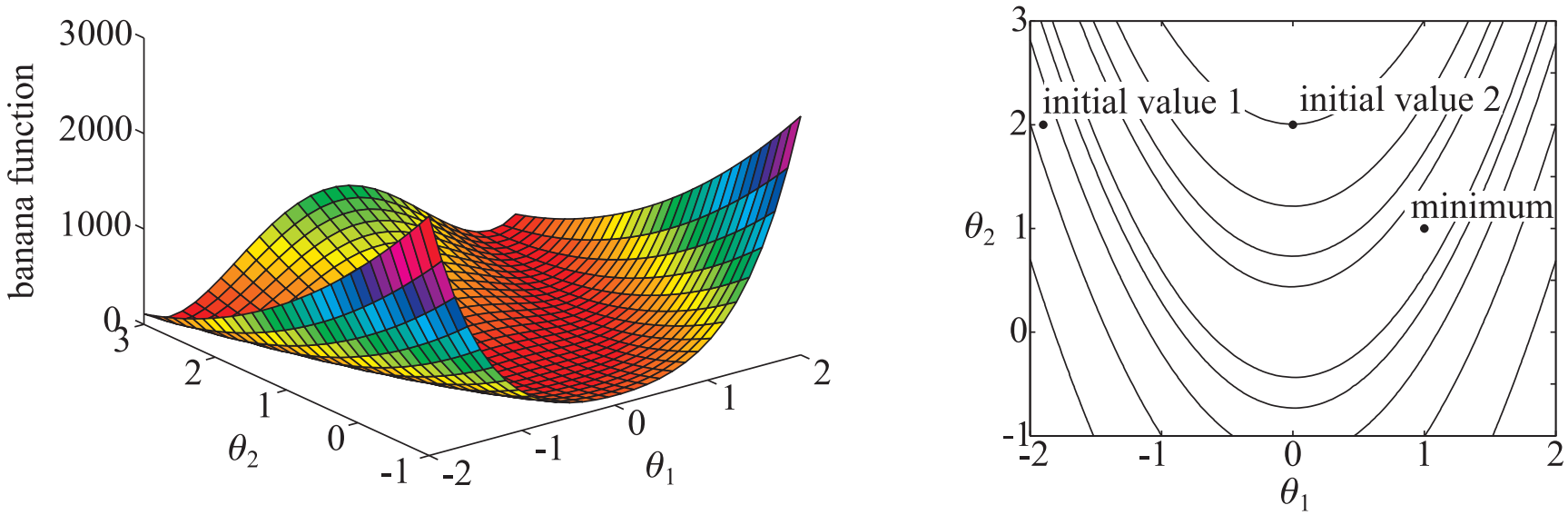
- The function has a minimum of $I(\underline{\theta}_{opt}) = 0$ at the value of optimization parameters $\underline{\theta}_{opt} = [1.0\ 1.0]^{\mathrm{T}}$.
- Figure 2b shows its contour diagram and two initial values of parameters at $[-1.9\ 2.0]^{\mathrm{T}}$ and $[0.0\ 2.0]^{\mathrm{T}}$ that we use for testing optimization methods.
- The goal of the optimization is to find the minimum of the function with the parameter accuracy of at least 0.001.
- For all gradient methods we need a gradient of the criterion function that we optimize. In our example, this is:

$$\underline{g}(\underline{\theta}) \;=\; \begin{bmatrix} \frac{\partial I(\underline{\theta})}{\partial \theta_1} \\ \frac{\partial I(\underline{\theta})}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} 400\theta_1\,(\theta_1^2 - \theta_2) + 2\,(\theta_1 - 1) \\ 200\,(\theta_2 - \theta_1^2) \end{bmatrix}, \tag{2}$$

Example: Rosenbrock function

- and the Hessian matrix, which we need in the case of Newton's method

$$\underline{H}(\underline{\theta}) = \begin{bmatrix} \frac{\partial^2 I(\underline{\theta})}{\partial^2 \theta_1} & \frac{\partial^2 I(\underline{\theta})}{\partial \theta_1 \, \partial \theta_2} \\ \frac{\partial^2 I(\underline{\theta})}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 I(\underline{\theta})}{\partial^2 \theta_2} \end{bmatrix} = \begin{bmatrix} 1200\,\theta_1^2 - 400\,\theta_2 + 2 & -400\,\theta_1 \\ -400\,\theta_1 & 200 \end{bmatrix}. \tag{3}$$



Slika 2: a) Rosenbrock's function. b) The contour diagram of the Rosenbrock function with a minimum at $[1.0\ 1.0]^{\mathrm{T}}$ and two starting points $[-1.9\ 2.0]^{\mathrm{T}}$ and $[0.0\ 2.0]^{\mathrm{T}}$, which we use when optimizing

Optimization with a data set and by individual data

- In each iteration k, we calculate a new parameter value with respect to the previous parameter values and the previous value of the criterion function and its derivatives

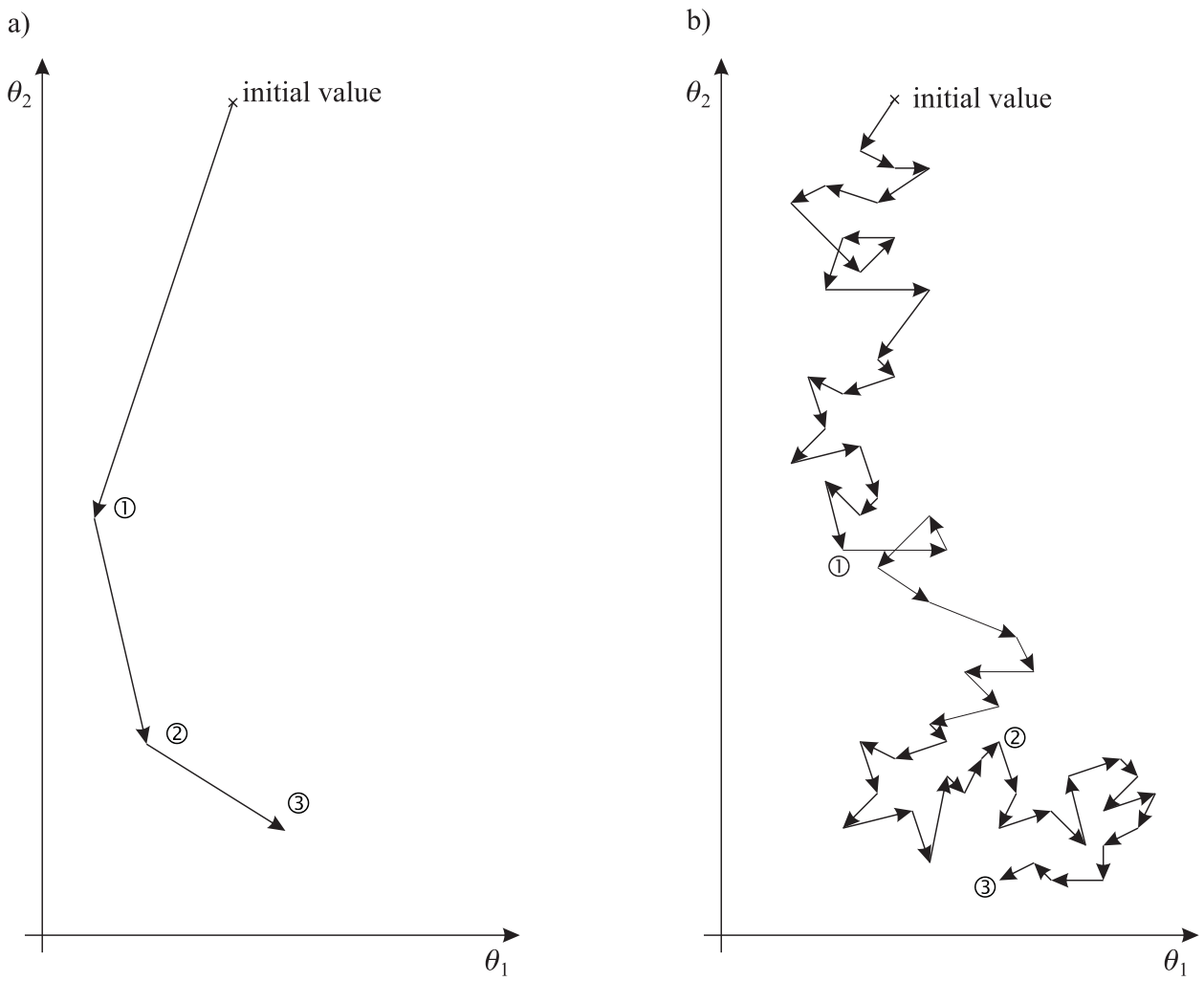
$$\underline{\theta}_k = f\left(\underline{\theta}_j,\, I(\underline{\theta}_j),\, \frac{\partial}{\partial \underline{\theta}_j} I(\underline{\theta}_j),\, \ldots\right) \quad \text{at } j = k-1,\, k-2,\, \ldots,\, 0. \tag{4}$$

- Usually when optimizing (4), we only perform the previous step at $j = k-1$, which takes into account that calculation $\underline{\theta}_{k-1}$ already includes information about past values ($j < k-1$)

$$\underline{\theta}_k = f\left(\underline{\theta}_{k-1},\, I(\underline{\theta}_{k-1}),\, \frac{\partial}{\partial \underline{\theta}_{k-1}} I(\underline{\theta}_{k-1}),\, \ldots\right). \tag{5}$$

- This approach is called the batch adaptation, because we need to use all the data from the learning set for each adaptation of the parameters. This presents a large computational burden. In such a case the solution is the division of the learning set into S subsets and the successive use (5) on all learning subsets; see Fig. 3.

Optimization with a data set and by individual data



Slika 3: Schematic representation of three iterations in the case of nonlinear optimization a) on a data set and b) on individual data. On a data set, we make one major adaptation of parameters for each iteration, in the case of adaptation to individual samples of parameters we adjust the parameters for each new datum. The overall average direction is similar to the optimization on the data set. Both processes can be converged into different solutions at very different times

Optimization with a data set and by individual data

- Extreme realization is an example when we choose $S = N$ of the subsets (for N data in a learning data set), which means that there is one datum in each subset. Then, the iteration in Eq.(5) is simplified (we assume a quadratic criterion function)

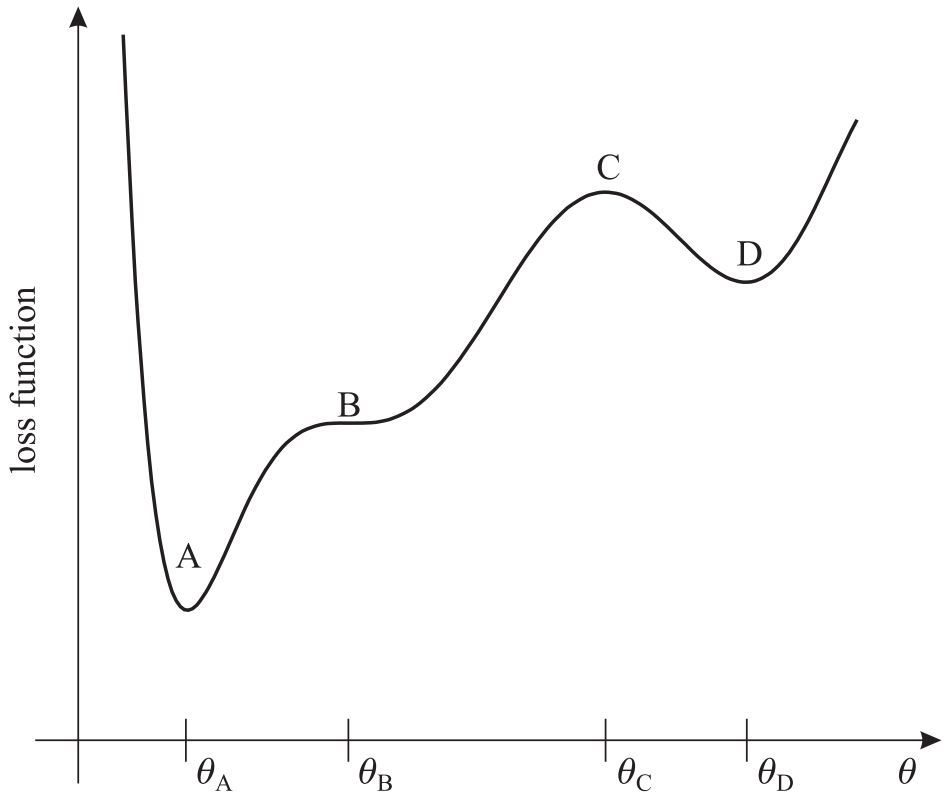
$$\underline{\theta}_k = f\left(\underline{\theta}_{k-1},\, e^2(\underline{\theta}_{k-1}),\, \frac{\partial}{\partial \underline{\theta}_{k-1}} e^2(\underline{\theta}_{k-1}),\, \ldots\right) . \tag{6}$$

  This method is called *optimization by individual data* or *current learning*.

- There is also a difference between **iterative algorithm** and **recursive optimization algorithm** .
- **Recursive algorithms** can be used in a concurrent mode (on-line) and represent a precise (exact) solution to the problem. They are only adapted to the sequential arrival of data samples, i.e. *data stream*. With multiple run through the same database a recursive algorithm does not obtain any information.
- In the case of **iterative algorithm**, multiple run through the data are required, and only in this way can we ensure convergence. Therefore, this method is also slower.
- Although the iterative algorithm can be run in the mode of individual data and the simultaneous principle, it usually yields worse results, slowly converges and is nonrobust to the data dimensions.

Initializing parameters

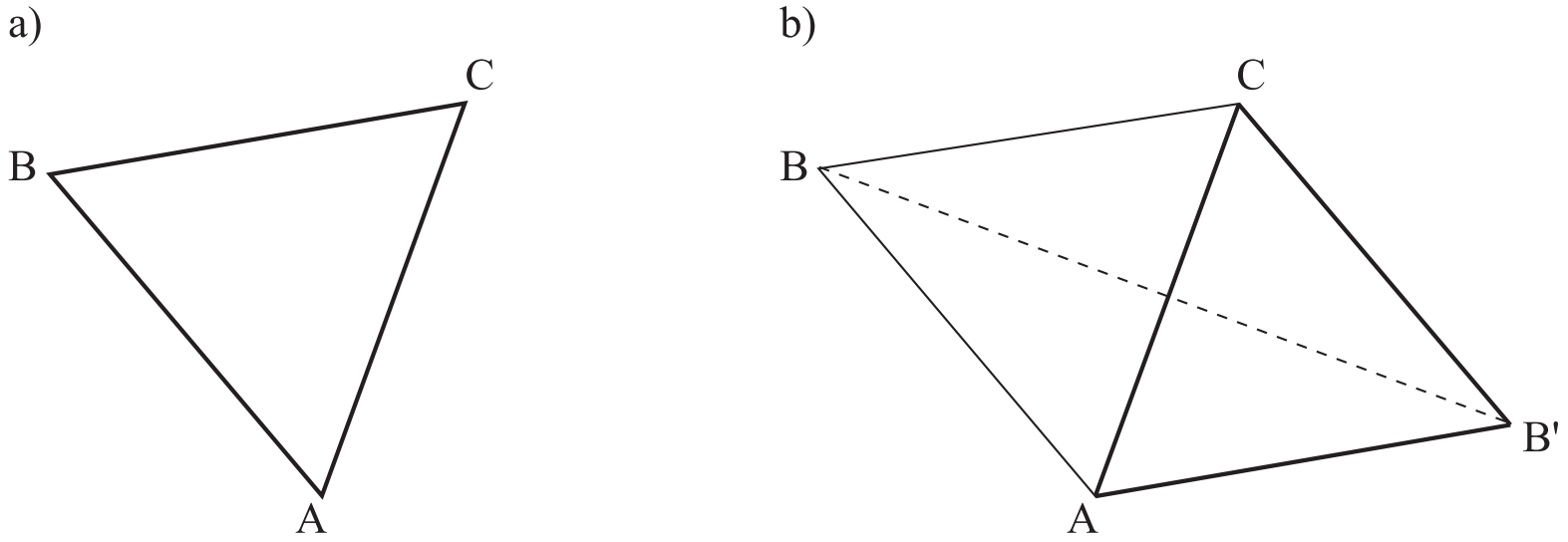
- Fig. 4 shows an example of a criterion function for one parameter θ.
- A solution converging to a global minimum at θ_A is obtained only in the case of an initial parameter that is less than θ_C.
- In the case of initialization at $\theta \geq \theta_C$, optimization converges to a local minimum at θ_D.
- Inflection points, as in the case of point B, usually do not cause convergence problems. The algorithm practically never converges to the solution in the inflection point θ_B.



Slika 4: Criterion function for nonlinear optimization problem with one parameter

Direct optimum search

- Direct optimum search algorithms are based on the calculation of the criterion function in the $n+1$ equidistant points of the parameter space, where n is the number of parameters.
- A point of the maximum value of a criterion function is mapped over a shape formed by $n+1$ points and a new simplex is defined.
- This is repeated until the result converges to a particular vicinity.
- The method is direct because we do not need derivatives. The advantages of direct methods are simplicity and comprehensibility. The methods that belong to this class are simplex method and the Hooke-Jeeves method, which are more accurately described in the literature.



Slika 5: a) Simplex (equidistant triangle) in a three-dimensional parameter space. b) A vertex with the highest criterion function (in this case B) is flipped (in this case in B') to create a new simplex

Finite differential method

- For all subsequent methods to be considered we need a gradient, i.e. the first derivative of the criterion function for calculating the search direction $\underline{p}$.
- In many cases, the gradient can be analytically calculated, as in the case of the Rosenbrock function.
- To calculate the gradient of the criterion function in the case of the sum of squared errors, it is also necessary to calculate the first derivative of the output of the model $\hat{y}$ with respect to the parameters $\underline{\theta}$:

$$\begin{aligned} \underline{g} &= \frac{\partial I(\underline{\theta})}{\partial \underline{\theta}} = \frac{\partial \frac{1}{2} \sum_{i=1}^{N} e^2(i)}{\partial \underline{\theta}} = \frac{1}{2} \sum_{i=1}^{N} \frac{\partial e^2(i)}{\partial \underline{\theta}} = \sum_{i=1}^{N} e(i) \frac{\partial e(i)}{\partial \underline{\theta}} \\ &= -\sum_{i=1}^{N} e(i) \frac{\partial \hat{y}(i)}{\partial \underline{\theta}} , \end{aligned} \tag{7}$$

where the errors $e(i) = y(i) - \hat{y}(i)$ are differences between the measured output $y(i)$ and the output of the model $\hat{y}(i)$, which is the function of parameters $\underline{\theta}$, and N is the number of samples from the learning set.

Finite differential method

- The component of the gradient i for the value of parameter $\underline{\theta}$ can be approximated by a difference quotient

$$g_i(\underline{\theta}) \approx \frac{I(\underline{\theta} + \Delta\theta_i) - I(\underline{\theta})}{\Delta\theta_i}, \tag{8}$$

  where $\Delta\theta_i$ is a small change in the direction of i-th component.
- We can also approximate the Hessian matrix using finite differentials. Column for i-th component of matrix $\underline{H}$, which is labelled by $\underline{h}_i$, can be calculated

$$\underline{h}_i(\underline{\theta}) \approx \frac{\underline{g}(\underline{\theta} + \Delta\theta_i) - \underline{g}(\underline{\theta})}{\Delta\theta_i}, \tag{9}$$

- Practically, this is only feasible if the gradient is given analytically, otherwise the computational complexity is too big.

Gradient optimization methods

- The principle of all gradient methods is based on changing the vector of parameters $\underline{\theta}_{k-1}$ in proportion to the adjustment step η_{k-1} in the direction $\underline{p}_{k-1}$, which is the direction of the gradient $\underline{g}_{k-1}$ that is rotated and scaled by the $\underline{R}_{k-1}$ matrix:

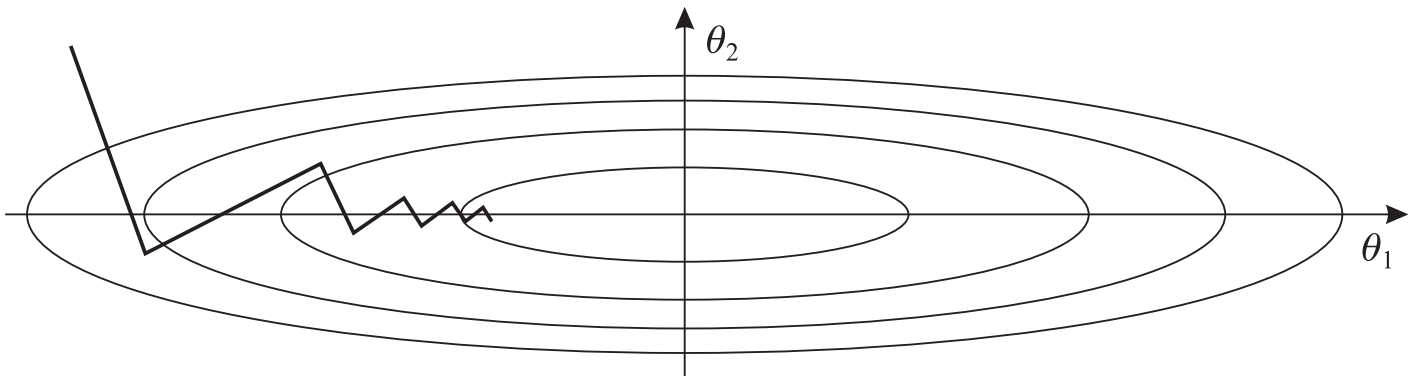
$$\underline{\theta}_k = \underline{\theta}_{k-1} - \eta_{k-1}\,\underline{p}_{k-1} \qquad \text{pri} \quad \underline{p}_{k-1} = \underline{R}_{k-1}\,\underline{g}_{k-1}\,. \tag{10}$$

Method of steepest descent

- is the simplest version of gradient methods (10) where the rotational matrix $\underline{R}$ is equal to identity matrix $\underline{I}$:

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \eta_{k-1}\, \underline{g}_{k-1}\,. \tag{11}$$

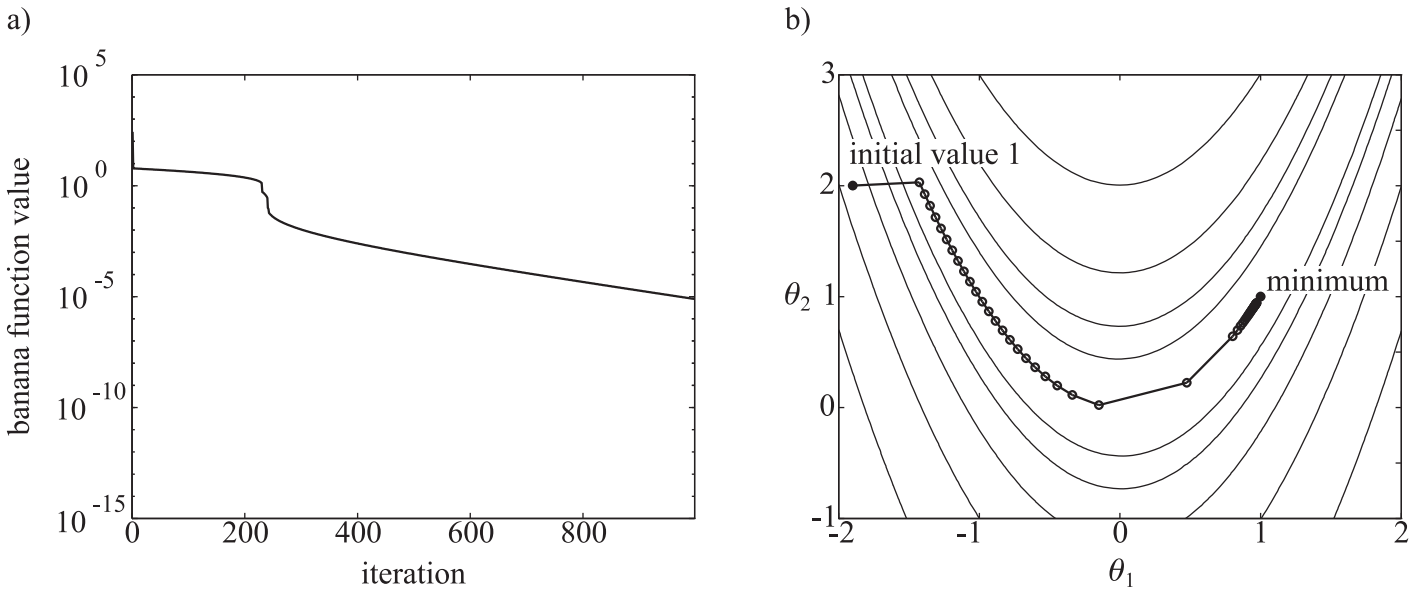
- This means that the direction of the search for the optimum is opposite from the direction of the gradient.



Slika 6: "Zig-zag behavior": Typical behavior of the maximum gradient method for a criterion function where the Hessian matrix has very different eigenvalues

Example: Maximum gradient method for the function minimization

- In this case, the Rosenbrock function was minimized by the technique of the maximum gradient.
- Fig. 7 shows the speed of convergence. Circles in the figure show every tenth iteration due to poor convergence.
- It is usually also true that the maximum gradient method needs significantly more iterations than any of the methods that compute also the second derivative.



Slika 7: Maximum gradient method for minimization of Rosenbrock's function. The speed of convergence a) with respect to iterations, b) in a parameter space (circles display every tenth iteration)

Newton's method

In **Newton's method** the rotational matrix $\underline{R}$ in (10) is chosen as the value of the inverse of the Hessian matrix $\underline{H}_{k-1}^{-1}$ at the vector of parameters $\underline{\theta}_{k-1}$

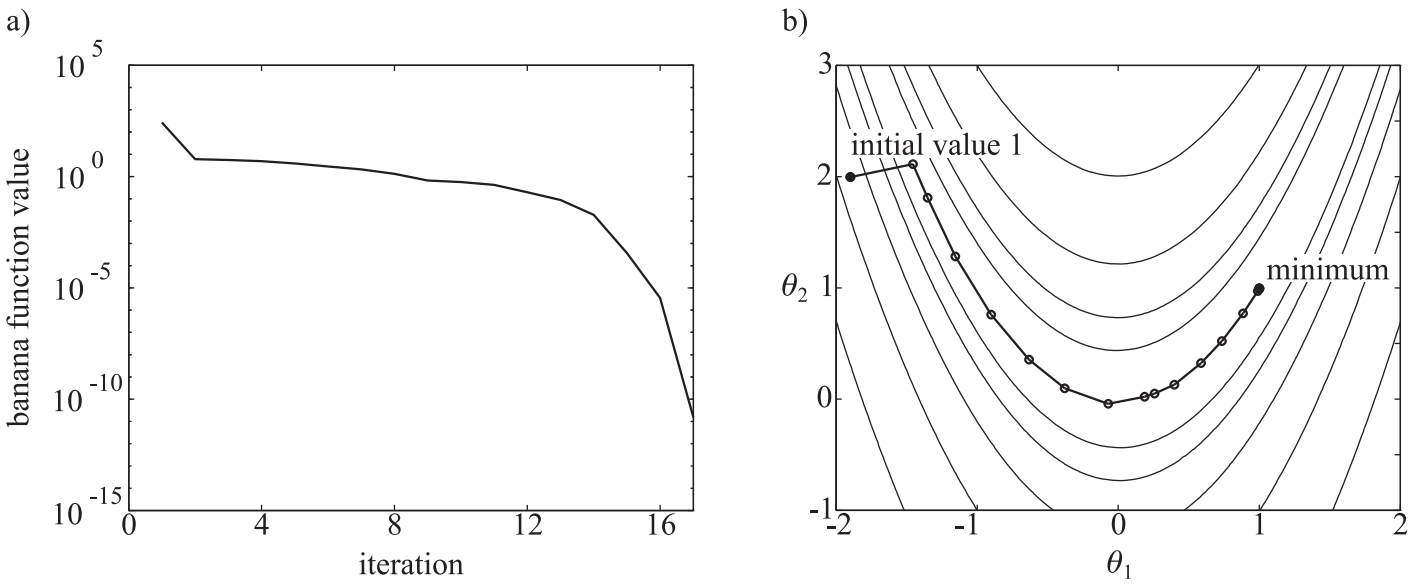
$$\underline{\theta}_k = \underline{\theta}_{k-1} - \eta_{k-1}\, \underline{H}_{k-1}^{-1} \underline{g}_{k-1} \,. \tag{12}$$

- Newton's method is usually used for the numerical calculation of the zeros of the function in the form $y = f(x)$.
- Iteration is in the form $x_k = x_{k-1} - f(x_{k-1}) \,/\, f'(x_{k-1})$.
- If we search for the minimum of the function $f(x)$, then we need to find the zero of the first derivative of the function with the iteration $x_k = x_{k-1} - f'(x_{k-1}) \,/\, f''(x_{k-1})$.
- This notation is the simplification of the equation (12) for one parameter and $\eta = 1$.

Newton's method has the following characteristics:

- requires the calculation of the second derivative,
- cubic computational complexity ($\mathcal{O}(n^3)$) because matrix inverse is required,
- quadratic complexity in the memory space, because we store the Hessian matrix,
- the fastest convergence among nonlinear optimizations,
- provides a single iteration for linear optimization problems,
- most suitable for small problems (up to ten parameters).

Newton's method



Slika 8: Newton's minimization of Rosenbrock's function a) with respect to iterations, b) in the parameter space (circles denote the evaluation of the function)

Quasi-Newton method

The idea of the **quasi-Newton method** is in the calculation of the Hessian matrix or its inverse (12) with its approximation

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \eta_{k-1}\, \underline{\hat{H}}_{k-1}^{-1} \underline{g}_{k-1}\,, \tag{13}$$

$$\text{at} \quad \underline{\hat{H}}_k = \underline{\hat{H}}_{k-1} + \underline{Q}_{k-1} \quad \text{or} \quad \underline{\hat{H}}_k^{-1} = \underline{\hat{H}}_{k-1}^{-1} + \underline{\widetilde{Q}}_{k-1}\,. \tag{14}$$

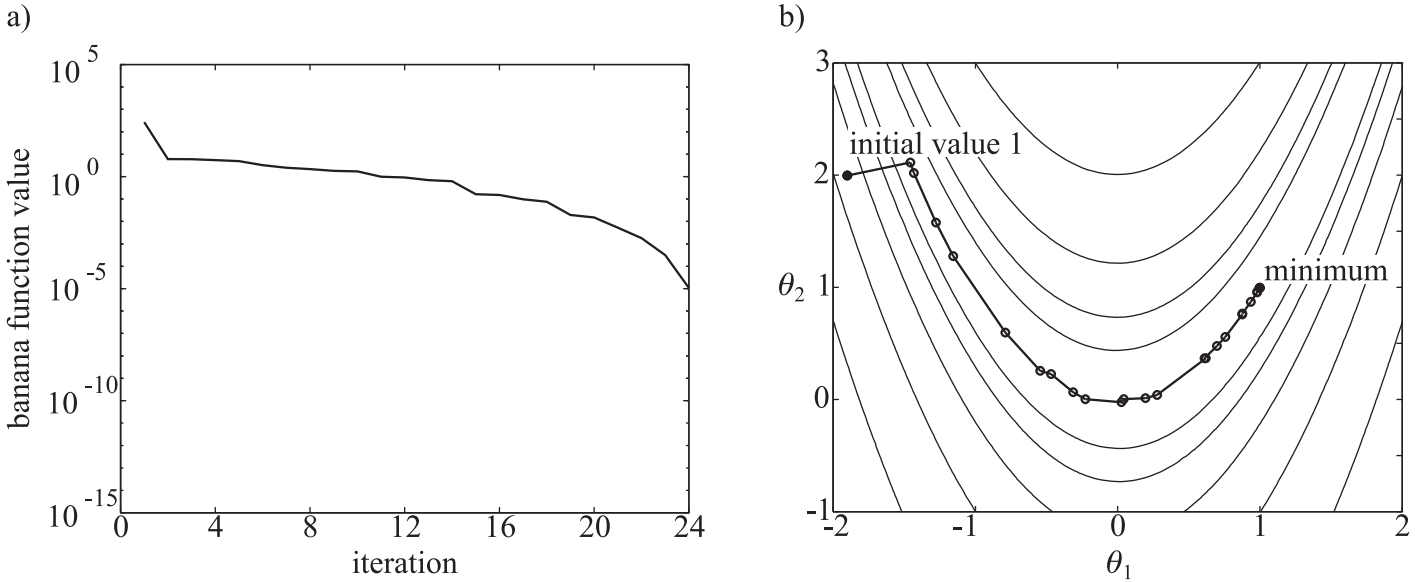
$$\underline{\hat{H}}_k^{-1} = \left(\underline{I} - \frac{\Delta\underline{\theta}_{k-1}\, \Delta\underline{g}_{k-1}^{\mathrm{T}}}{\Delta\underline{\theta}_{k-1}^{\mathrm{T}}\, \Delta\underline{g}_{k-1}}\right) \underline{\hat{H}}_{k-1}^{-1} \left(\underline{I} - \frac{\Delta\underline{\theta}_{k-1}\, \Delta\underline{g}_{k-1}^{\mathrm{T}}}{\Delta\underline{\theta}_{k-1}^{\mathrm{T}}\, \Delta\underline{g}_{k-1}}\right)^{\mathrm{T}} + \frac{\Delta\underline{\theta}_{k-1}\, \Delta\underline{\theta}_{k-1}^{\mathrm{T}}}{\Delta\underline{\theta}_{k-1}^{\mathrm{T}}\, \Delta\underline{g}_{k-1}}\,, \tag{15}$$

where $\Delta\underline{\theta}_{k-1} = \underline{\theta}_k - \underline{\theta}_{k-1}$ and $\Delta\underline{g}_{k-1} = \underline{g}_k - \underline{g}_{k-1}$.

Example: Quasi-Newton method for function minimization

In this case, we are searching for the minimum of Rosenbrock function using the BFGS quasi-Newton method.

- The method uses only the information about the gradient and calculates the inverse Hessian matrix with respect to (15).
- The number of calculations of the function and the number of iterations is 79 and 24 in the case of the first initial value, and in the case of the second initial value it is 55 and 17.
- In Fig. 9 we see a convergence from the first initial value.
- The method converges more slowly than the classic Newton's method since we do not directly apply the Hessian matrix. Of course, the quasi-Newton method is incomparably faster than the finite differential methods.



Slika 9: Quasi-Newton method in search of the minimum of Rosenbrock's function. Convergence speed a) with respect to iterations, b) in a parameter space

Conjugate gradient method

- The method is written with the following equations

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \eta_{k-1}\,\underline{p}_{k-1} \tag{16}$$

$$\text{at} \quad \underline{p}_{k-1} = \underline{g}_{k-1} - \beta_{k-1}\,\underline{p}_{k-2} \tag{17}$$

  with the scalar β_{k-1}, which distinguishes different methods of conjugate gradients.
- The most common method is Fletcher-Reeves , where

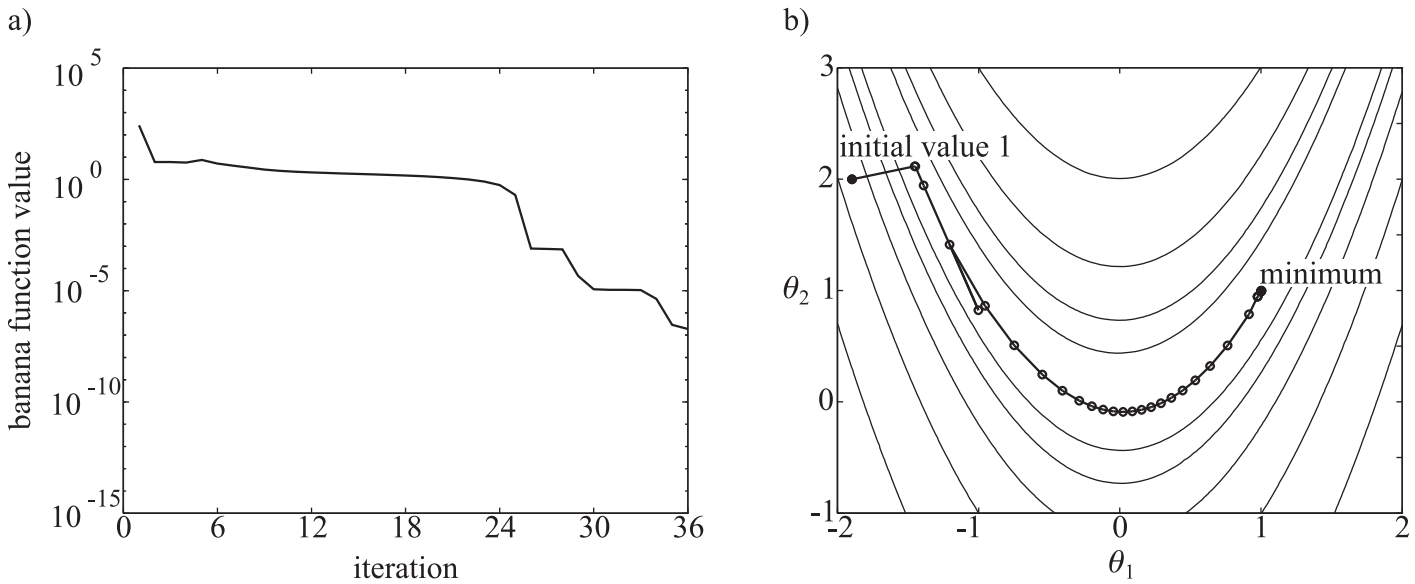
$$\beta_{k-1} = \frac{\underline{g}_{k-1}^{\mathrm{T}}\underline{g}_{k-1}}{\underline{g}_{k-2}^{\mathrm{T}}\underline{g}_{k-2}} \tag{18}$$

- The β scalar represents the accumulation of knowledge from previous iterations. In this way, the method of conjugate gradients can be seen as a compromise between the gradient and quasi-Newton method.

Conjugate gradient method

The main properties of **conjugate gradient method** are the following:

- it is not required that the second derivative should be calculated,
- linear computational complexity,
- linear requirements for the memory space,
- fast convergence,
- needs the most n iterations in the case of a linear optimization problem,
- best suited for major problems (more than 1000 parameters).



Slika 10: Conjugate gradient method for minimization of Rosenbrock function a) with respect to iterations, b) in a parameter space

Nonlinear least squares method

- Assume the function $y(u, \underline{\theta})$ that is nonlinear in the parameters.
- The value of the function approximation at the value of the parameters vector $\underline{\theta}$ and at the value of the input variable u can be written as $\hat{y}(u, \underline{\theta})$.
- The vector of parameters has the dimension of $m \times 1$ and is written as $\underline{\theta}_k = \left[\theta_k^1, ..., \theta_k^m\right]^T$.
- At the time point i where the input variable is equal to u_i the value of the function can be approximated by the first term of Taylor series

$$\hat{y}(u_i, \underline{\theta}_k) = \hat{y}(u_i, \underline{\theta}_{k-1}) + \sum_{j=1}^{m} \frac{\partial \hat{y}(u_i, \underline{\theta}_{k-1})}{\partial \theta_j} (\theta_k^j - \theta_{k-1}^j) \ , \tag{19}$$

  where θ_{k-1}^j and θ_k^j are values of the j-th parameter in the vector $\underline{\theta}_{k-1}$ and $\underline{\theta}_k$.
- The value of the partial derivative according to the j-th parameter and at the input variable u_i can be simplified written with the Jacobi matrix coefficient as follows

$$J_{ij} = \frac{\partial \hat{y}(u_i, \underline{\theta}_{k-1})}{\partial \theta_j} \ , \tag{20}$$

  the difference in the parameters is defined as $\Delta\theta_{k-1}^j = \theta_k^j - \theta_{k-1}^j$.
- Considering this, the approximation of the function can be written in a more compact form

$$\hat{y}(u_i, \underline{\theta}_k) = \hat{y}(u_i, \underline{\theta}_{k-1}) + \underline{J}_i(u_i, \underline{\theta}_{k-1}) \underline{\Delta\theta}_{k-1} \ , \tag{21}$$

  where $\underline{J}_i$ is the row vector of Jacobian coefficients $\underline{J}_i = [J_{i1}, ..., J_{im}]$ at the value of the vector of parameters $\underline{\theta}_{k-1}$.

Nonlinear least squares method

- The error between the actual output $y(u_i)$ and the approximate output $\hat{y}(u_i, \underline{\theta}_k)$ can be written as

$$e_i = y(u_i) - \hat{y}(u_i, \underline{\theta}_k),\ i = 1, ..., n\ , \tag{22}$$

where n is the number of n input-output data pairs. The above error can be extended to

$$e_i = y(u_i) - \hat{y}(u_i, \underline{\theta}_{k-1}) - \underline{J}_i(u_i, \underline{\theta}_{k-1})\underline{\Delta\theta}_{k-1},\ i = 1, ..., n\ . \tag{23}$$

- If the notation is expanded into a matrix form that takes the whole series of input-output data into account, then we get the following notation

$$\underline{e}(\underline{\theta}_{k-1}) = \underline{y} - \underline{\hat{y}}(\underline{\theta}_{k-1}) - \underline{J}(\underline{\theta}_{k-1})\underline{\Delta\theta}_{k-1}\ , \tag{24}$$

where $\underline{J}(\underline{\theta}_{k-1})$ is Jacobi matrix at the value of the vector of parameter $\underline{\theta}_{k-1}$, which is equal to $\underline{J}(\underline{\theta}_{k-1}) = [\underline{J}_1^T(\underline{\theta}_{k-1}), \cdots, \underline{J}_n^T(\underline{\theta}_{k-1})]^T$.

- By changing the vector of parameters $\underline{\theta}$ we search for the minimum of the function

$$I(\underline{\theta}_{k-1}) = \underline{e}^T(\underline{\theta}_{k-1})\underline{e}(\underline{\theta}_{k-1})\ , \tag{25}$$

- which gives us the result

$$\underline{\Delta\theta}_{k-1} = \left(\underline{J}^T(\underline{\theta}_{k-1})\underline{J}(\underline{\theta}_{k-1})\right)^{-1} \underline{J}^T(\underline{\theta}_{k-1}) \left(\underline{y} - \underline{\hat{y}}(\theta_{k-1})\right)\ . \tag{26}$$

Example: NLS method

Identification of the model with the exponential function structure

- $$y = A\exp(-(u-\mu)^2/(2\sigma^2)) \ . \tag{27}$$
- We have a set of n input-output data, pairs u_i and y_i.
- The partial derivatives are in this case

$$\frac{\partial y}{\partial A} = \exp(-(u-\mu)^2/(2\sigma^2)) \ , \tag{28}$$

$$\frac{\partial y}{\partial \mu} = \frac{A(u-\mu)}{\sigma^2}\exp(-(u-\mu)^2/(2\sigma^2)) \ , \tag{29}$$

$$\frac{\partial y}{\partial \sigma} = \frac{A(u-\mu)^2}{\sigma^3}\exp(-(u-\mu)^2/(2\sigma^2)) \ . \tag{30}$$

- Nonlinear parameters are arranged into the vector $\underline{\theta} = [A\,\mu\,\sigma]^T$.
- We choose the initial value of the vector of parameters $\underline{\theta}_0$ and we calculate $\underline{\hat{y}}(\underline{\theta}_0)$.
- We calculate the Jacobi matrix $\underline{J}(\theta_0)$ with the dimension $n \times m$ where we calculate all m partial derivatives at n input values u_i and the values of the vector of parameters $\underline{\theta}_0$. In our case $m = 3$.
- By using equation 26 we calculate the change of the vector of parameters and we change the vector of parameters accordingly. We repeat this process until the parameters converge.

Gauss-Newton method

- is a version of the general Newton method, which is presented in (12).
- Since the gradient is represented as $\underline{g} = \underline{J}^{\mathrm{T}}\underline{e}$ and the Hessian matrix as $\underline{H} \approx \underline{J}^{\mathrm{T}}\underline{J}$, then we get the Gauss-Newton algorithm
- $$I(\underline{\theta}) = \underline{e}^T\underline{e} \ , \tag{31}$$
- $$\underline{g} = \frac{\partial I(\underline{\theta})}{\partial \underline{\theta}} = 2\frac{\partial \underline{e}^T}{\partial \underline{\theta}}\underline{e} = 2\underline{J}^T\underline{e} \ , \tag{32}$$
- $$\underline{g}_{k-1} = 2\underline{J}_{k-1}^T\underline{e}_{k-1} \ , \tag{33}$$

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \eta_{k-1}\left(\underline{J}_{k-1}^{\mathrm{T}}\underline{J}_{k-1}\right)^{-1}\underline{J}_{k-1}^{\mathrm{T}}\underline{e}_{k-1} \ . \tag{34}$$

Levenberg-Marquardt method

- is an extension of the Gauss-Newton method.
- The idea is in the regularization modification of the equation (34)

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \eta_{k-1} \left(\underline{J}_{k-1}^{\mathrm{T}} \underline{J}_{k-1} + \alpha_{k-1} \underline{I}\right)^{-1} \underline{J}_{k-1}^{\mathrm{T}} \underline{e}_{k-1} \,, \tag{35}$$

where the term $\alpha_{k-1}\underline{I}$ is a regularization parameter (35) that solves the problem of the poorly conditioned matrix $\underline{J}_{k-1}^{\mathrm{T}} \underline{J}_{k-1}$.

Global nonlinear optimization

Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for the 5th lecture, March 2016

Table of contents

1. Global nonlinear optimization
2. Simulated annealing - SA
3. Particle swarm optimization - PSO
4. Evolutionary Algorithms - EA
   - Evolution Strategy - ES
   - Genetic algorithms - GA
   - Genetic programming - GP
5. Branch and bound - B & B
6. Optimization with tabu search - TS

Global nonlinear optimization

- The difference between local and global nonlinear optimization is in the dimension of the space they are investigating
- makes sense only for nonlinear optimization problems (linear problems do not have local extrema)
- nonlinear local optimization methods (initial condition, search in the direction of derivatives, the first and sometimes also the second derivative)
- with the local method we find the optimum which is closest to the initial conditions, not necessarily a global one
- even in the case of global optimization, the optimum is not always reliable

Global nonlinear optimization strategy

First:

- by repeatedly running local nonlinear optimization from different initial values of parameters
- in each run optimization converges to the nearest local optimum and the best one is chosen as the final result
- the method is parallel by nature, because we can start every run on another computer
- problem in the selection of the initial values.

Second:

- extension of local methods to search for a global optimum is adding random values to newly calculated values of parameters
- added random values slow down the convergence rate, but at the same time they allow the escape from the local optimum
- standard deviation of the added random value is reduced with the number of iterations in order to accelerate the convergence
- depending on their nature, however, such methods still search for a solution only around initial parameters.

Examples of the first strategy

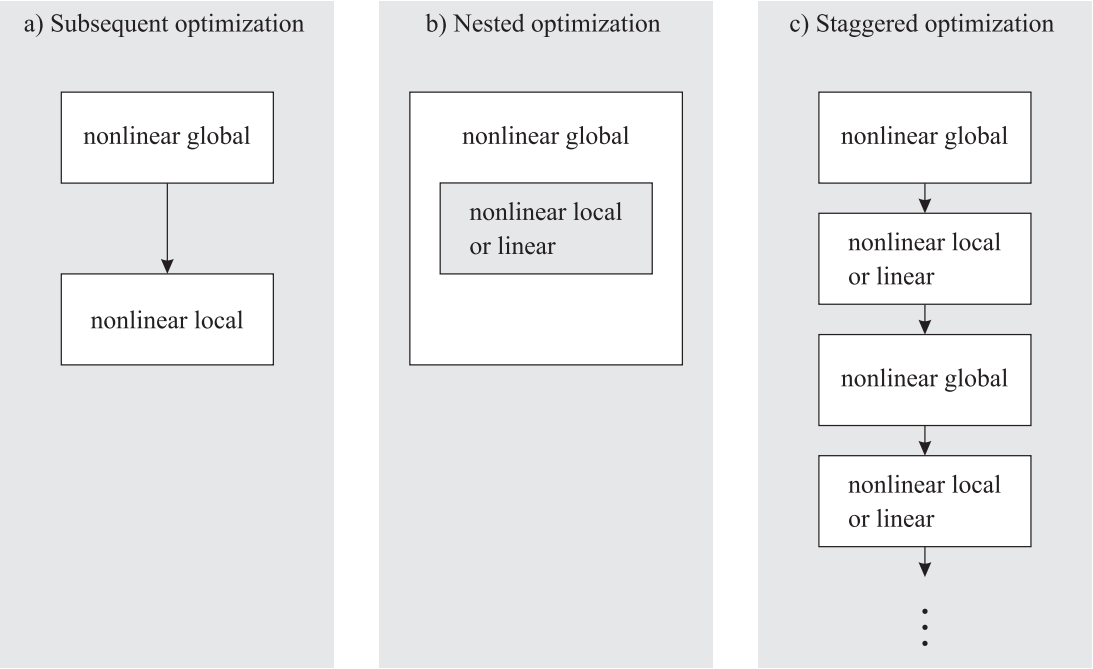


Figure 1: Combination of different optimization techniques: a) global optimization of all parameters followed by local optimization of all parameters, b) global optimization of some parameters or a structure with nested-form optimization (in each iteration), followed by local optimization of all parameters non-linear local or linear optimization of other parameters, c) repetitive global optimization with some parameters followed by nonlinear local or linear optimization of other parameters

Global nonlinear optimization

Global nonlinear techniques **are used in case** if at least one of the following criteria is met:

- we are searching for the global optimum,
- function and its derivatives are very nonlinear,
- some parameters are not real numbers, but rather integers or binary numbers, or even a mixed problem.

Dimension problem:

- the biggest problem of all global optimization methods is computational complexity
- the partitioning of the parameter space into individual intervals increases exponentially with the number of parameters n
- the problem is common to all the global optimization algorithms
- the dimension problem is solved by looking more closely at the areas that are more likely to contain the optimum

- most global nonlinear techniques contain stochastic elements
- stochastic elements allow the escape from local optimums and enable wider search
- it does not necessarily mean that the cost function monotonically decreases in relation to iterations
- Because of this, at the end of optimization we take the best solution, not the best solution at the final iteration

In general, we could say that *global methods allow us to find an appropriate area, and with local methods we can find the optimums of parameter space*.

Simulated annealing - SA

Method description:

- simulated annealing (SA) is a stochastic global optimization method
- physical analogy of a potential field and a fast or hot particle inside this field
- in general the particle moves in the direction of a smaller potential energy
- non-zero temperature, or kinetic energy, allows motion in the opposite direction, i.e. random jumps to higher potential energy and escaping from local minimum and searching for a global solution
- the particle loses energy and in this way the possibility of moving in the opposite direction of the potential field decreases
- the energy of the particle must slowly decrease in order to achieve a global optimum, otherwise the particle can be lost at the local minimum
- in the context of optimization the particle represents a point in the parameter space, and the potential field represents the criterion function.

Pseudo code for SA

1. Initialization of the initial temperature of the particle T_0, the constant change of parameters η, the number of iterations and the number of tries within one iteration.
2. Initialization of the initial value of the vector of parameters $\underline{\theta}_0$ in the parameter space.
3. Calculation of the criterion function for the initial parameter $I(\underline{\theta}_0)$.
4. **repeat**
5. new iteration $k = k + 1$
6. Calculation of a new point in the parameter space $\underline{\theta}_{\text{new}}$, which is calculated as

   $$\underline{\theta}_{new} = \underline{\theta}_{k-1} + \Delta\underline{\theta}_k$$

   and is generated with the probability density function $g(\Delta\underline{\theta}_k, T_k)$.
7. Calculation of the criterion function for a new vector of parameters (particle) $I(\underline{\theta}_{\text{new}})$.
8. We accept the new vector of parameters with the probability $h(\Delta I_k, T_k)$, where

   $$\Delta I_k = I(\underline{\theta}_{\text{new}}) - I(\underline{\theta}_{k-1})$$

   i.e., we set $\underline{\theta}_k = \underline{\theta}_{\text{new}}$ or we retain the old vector of parameters, i.e. $\underline{\theta}_k = \underline{\theta}_{k-1}$.
9. We decrease the temperature of the particle with respect to the predicted annealing T_k.
10. **until** the condition for completing the algorithm is not fulfilled.

Simulated annealing - SA

The Gaussian function for various temperatures T is shown in figure 2.

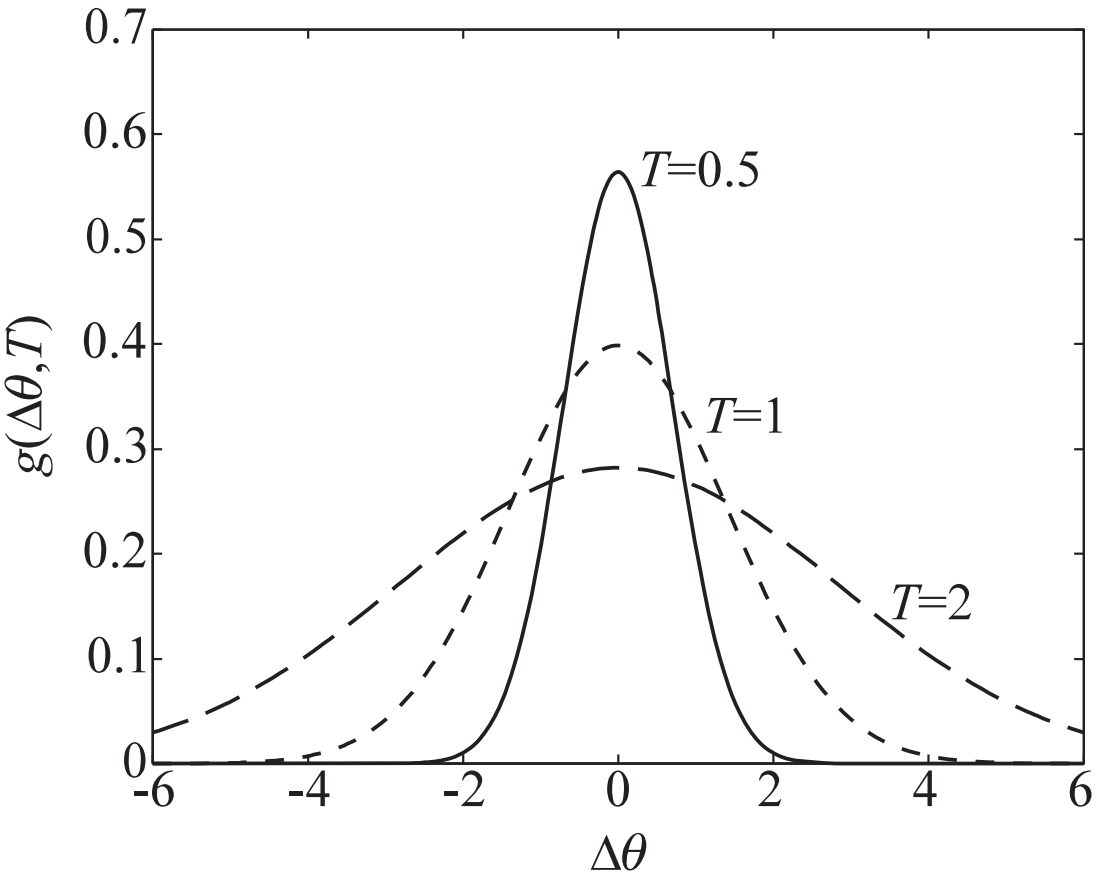


Figure 2: Gaussian probability distribution function for different annealing temperatures T. A lower temperature implies parameter increments close to zero. This means that the new parameters are close to the old ones and that we are exploring the local space.

Simulated annealing - SA

The change of each individual parameter in the case of **Gaussian distribution** $g(\Delta\theta_k^i, T_k)$ is the same as

$$\Delta\theta_k^i = \eta_i \mathcal{N}(0, T_k^2) \tag{1}$$

where $\mathcal{N}(0, T_k^2)$ is a generator of a random variable with a normal distribution.
Due to "wider tails" of the function, this leads to a wider global search. **Cauchy distribution** in the centralized form is given with the following equation

$$g(\Delta\underline{\theta}, T) = \pi^{-d/2} \frac{\Gamma(\frac{d+1}{2})}{\Gamma(\frac{1}{2})} \frac{(\det(\underline{\Sigma})^{-1/2}}{\left(1 + \Delta\underline{\theta}^{\mathrm{T}} \underline{\Sigma} \Delta\underline{\theta}\right)^{\frac{d+1}{2}}} \tag{2}$$

where Γ is a gamma function.
In the case of the dimension $d = 1$, we get a simplified notation

$$g(\Delta\underline{\theta}, T) = \frac{1}{\pi} \left(\frac{T}{T^2 + \Delta\underline{\theta}^{\mathrm{T}} \Delta\underline{\theta}} \right) \tag{3}$$

In the case of FA optimization, the annealing function has the following form:

$$T_k = \frac{T_0}{k} \,. \tag{4}$$

Simulated annealing - SA

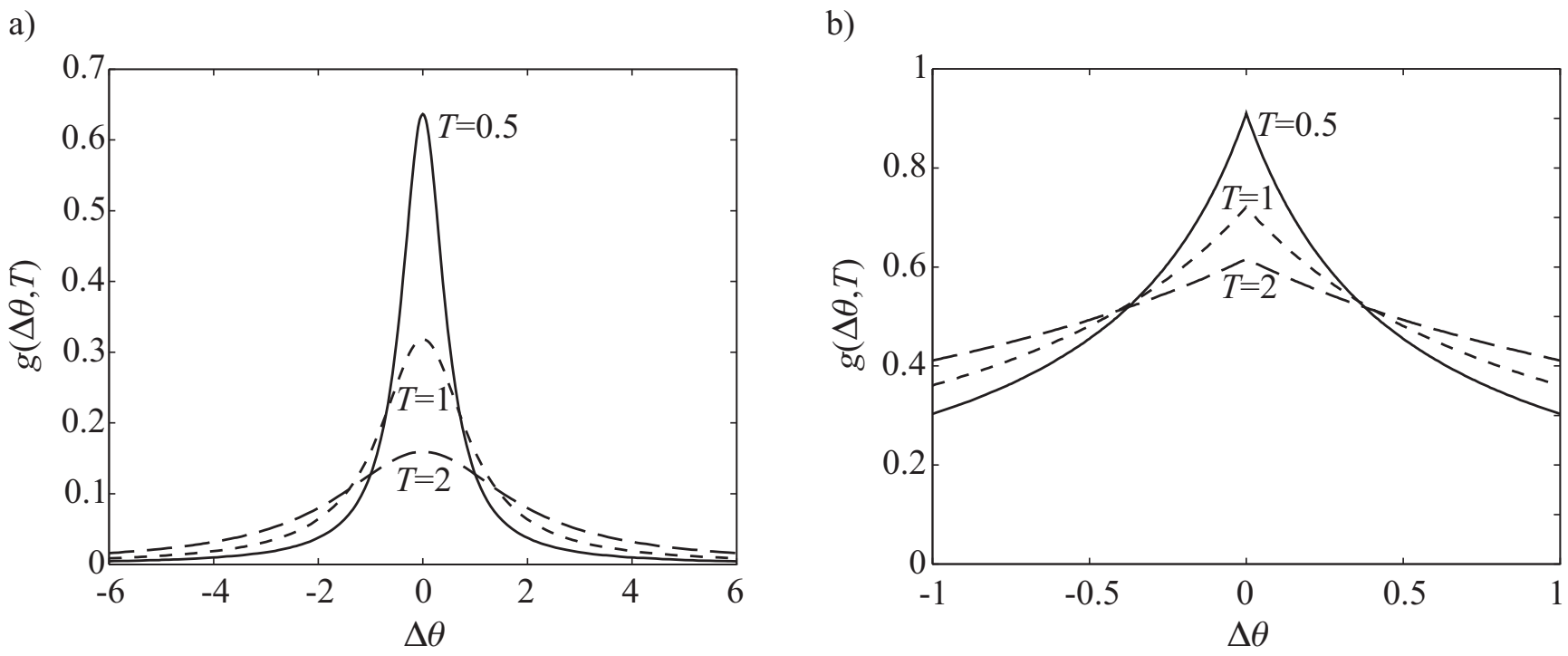


Figure 3: a) Cauchy distribution for different annealing temperatures T. Cauchy distribution has a “broader tail” than Gaussian, which is presented in figure 2 and therefore allows a better global search for the optimum. b) The probability density function for VFSR and ASA is defined on the interval $[-1, 1]$ and can be linearly transformed to an arbitrary interval [A, B]. This has a “wider tail” than the Cauchy probability density function (pdf) in figure 3, and therefore allows easier testing of local minima when looking for a global minimum

Probability of acceptance

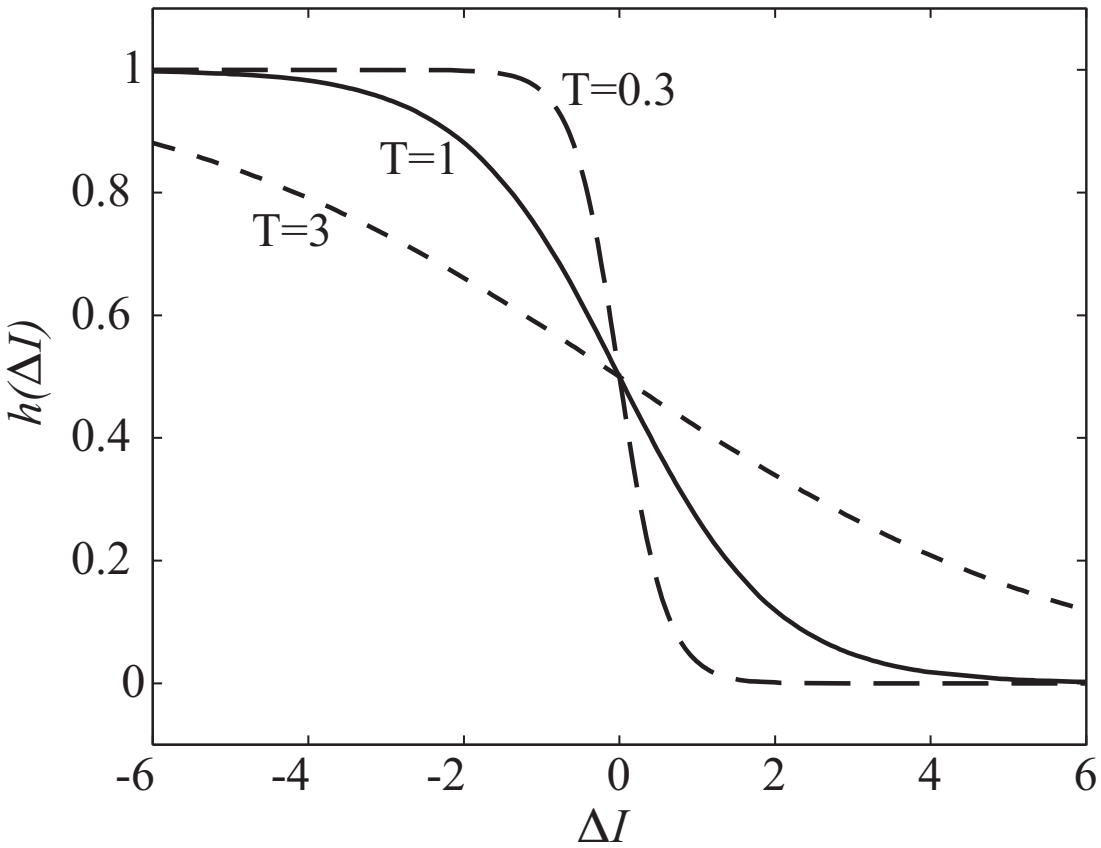


Figure 4: Probability for acceptance of a solution in the case of simulation annealing for high, medium and low temperature T. The particles that reduce the criterion function ($\Delta I < 0$) are always accepted with a probability greater than 0.5, and the particles that increase the criterion function ($\Delta I > 0$) are accepted with smaller probability. For very high temperatures ($T \to \infty$), all particles are accepted with the same probability, while at low temperatures ($T \to 0$) particles are accepted only when the criterion function decreases.

Simulated annealing - SA

The probability density function for accepting a new parameter vector

$$\begin{aligned} h(\Delta I_k, T_k) &= \frac{\exp\left(-I_k/T_k\right)}{\exp\left(-I_k/T_k\right) + \exp\left(-I_{k-1}/T_k\right)} \\ &= \frac{1}{1 + \exp\left(\Delta I_k/T_k\right)}\,, \end{aligned} \tag{5}$$

where ΔI_k represents the difference between the criterion function for new and old parameters.

The annealing function is fundamental for the convergence of the algorithm towards the global optimum

$$T_k = \frac{T_0}{\ln k}\,. \tag{6}$$

For the simplest changing of the parameters where $\underline{\Sigma}^{-1} = \sigma^2 \underline{I}$ and the variance depends on the temperature T, we get the following function

$$g(\Delta\underline{\theta}, T) = (2\pi T)^{-\frac{d}{2}} \exp\left(-\frac{\Delta\underline{\theta}^{\mathrm{T}}\Delta\underline{\theta}}{2T^2}\right)\,, \tag{7}$$

Simulated annealing - SA

Description of the algorithm

- the algorithm is initialized to the initial temperature, we select the initial values of the vector of parameters, the learning constant, the number of iterations, and the number of tries within one iteration
- the change of the vector of parameters is determined by means of a generator of the probability density function $g(\cdot)$
  - (i) the probability of small parameter changes is greater than the probability of major changes
  - (ii) large parameter changes are more likely to occur at high temperatures of particles than at low temperatures
- in this way, the algorithm begins to explore a very wide parameter space and, over time, it starts to explore a narrow area
- the probability density function for the acceptance of the solution $h(\cdot)$ has the following properties:
  - (i) acceptance is more likely if the new vector of parameters improves the criterion function,
  - (ii) the probability of acceptance of the vector of parameters with a poorer criterion function is greater at higher temperatures than at lower ones.
- we reduce the temperature of the particle according to the estimated annealing function

Simulated annealing - SA

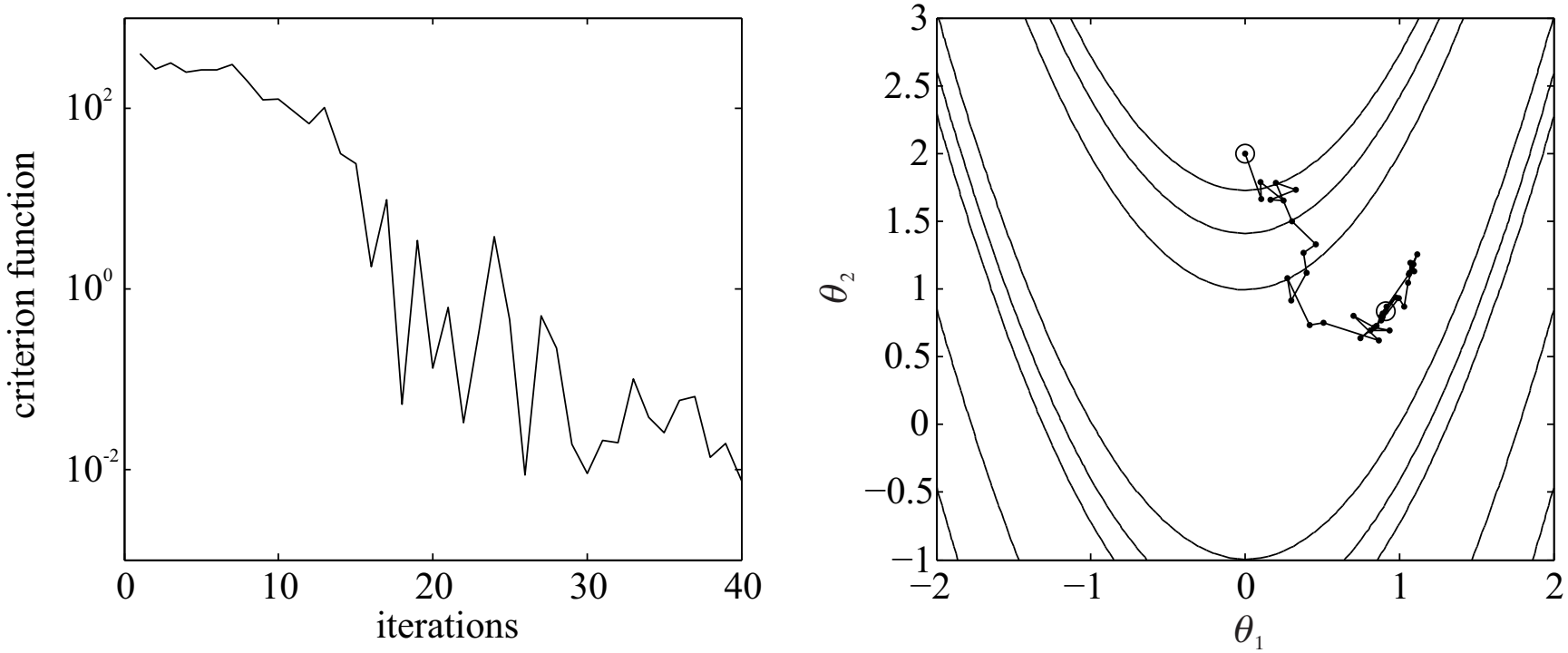


Figure 5: a) An example of a search for a minimum of Rosenbrock function at initial conditions $[0.0\ 2.0]^{\mathrm{T}}$ with respect to iterations, b) in parameter space

Simulated annealing - SA

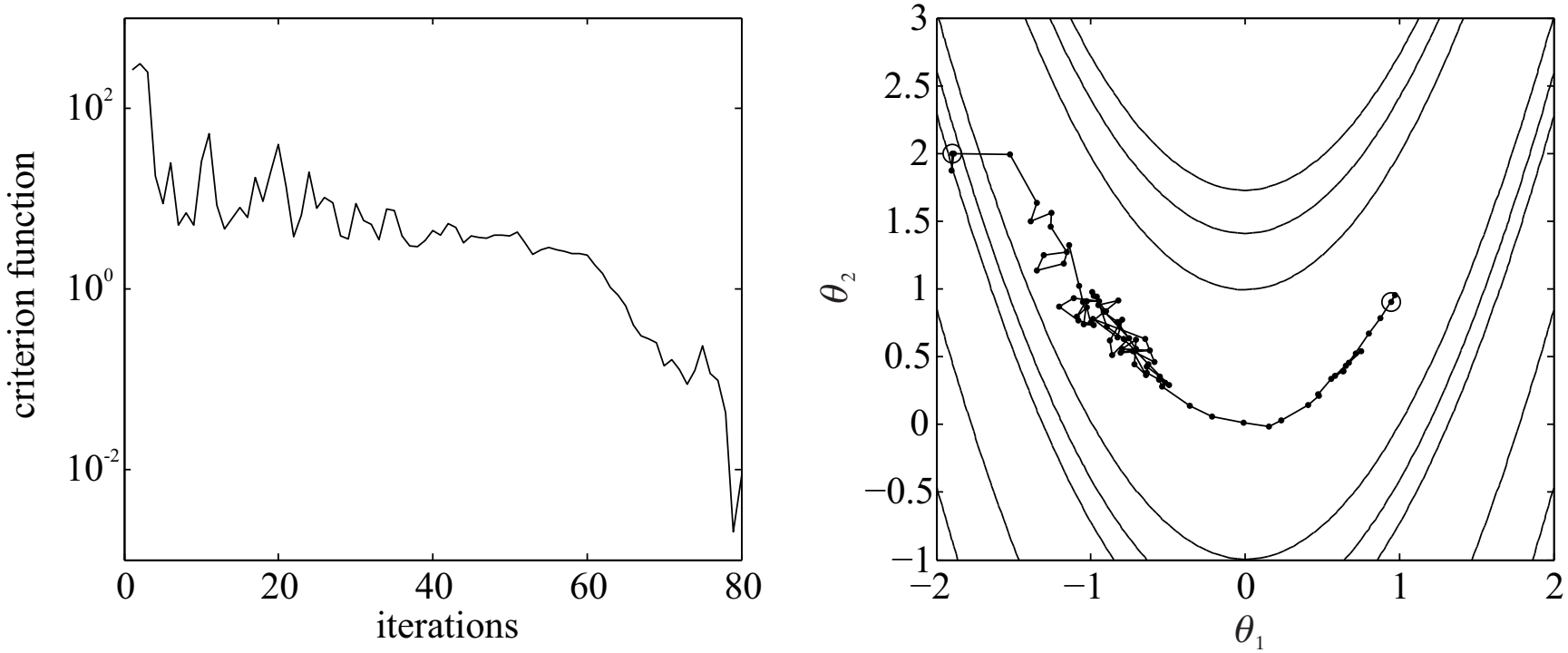


Figure 6: a) An example of a search for a minimum of Rosenbrock function at initial conditions $[-1.9\ 2.0]^{\mathrm{T}}$ with respect to iterations, b) in parameter space

Particle swarm optimization - PSO

- *Particle Swarm Optimisation - PSO* belongs to the family of stochastic optimization techniques
- it is based on the search of the optimum by using the group of n particles, which are candidates for the solution of the optimization problem
- the particles are randomly distributed over the parameter space, which in this case has the dimension d
- a particle is the point in the parameter space $\underline{\theta}_i = [\theta_{i1}, \ldots, \theta_{id}]$

- each particle has its position $\underline{\theta}_i$ and speed $\Delta\underline{\theta}_i$.
- for each particle we define the value of the criterion function, which is often called (*fitness* - F)
- the relation between the criterion function I and the fitness function is $F = -I$, if in the case of a criterion function we usually search for the minimum, then in the case of a fitness function we search for the maximum

Learning Rule

- the change of the position of the vector of parameters (the particle) depends on the value of the vector of parameters at the highest quality function for that particular particle, which is called $\underline{p}^{best}$ and the value of the vector of parameters with the highest quality function in the whole population (swarm) of particles $\underline{g}^{best}$.
- The dependence of the change in the position of the particle from $\underline{p}^{best}$ is called the cognitive component; the dependence of the change from $\underline{g}^{best}$ is called the social component and ω as the inertia parameter

$$\Delta\underline{\theta}_i(k+1) = \omega\Delta\underline{\theta}_i(k) + c_1\underline{r}_p \circ (\underline{p}_i^{best} - \underline{\theta}_i(k)) + c_2\underline{r}_g \circ (\underline{g}^{best} - \underline{\theta}_i(k)),\ i = 1, ..., n\ , \tag{8}$$

where $\circ$ is the operator of the product by individual components.

Pseudo code for particle swarm optimization (PSO)

- The initialization of the number of particles n, the inertia constant ω, the cognitive constant c_1, and the social behaviour constant c_2.
- The initialization of the initial values of the position $\underline{\theta}_i,\ i = 1, ..., n$ and the particle velocities $\Delta\underline{\theta}_i,\ i = 1, ..., n$
- The initialization $I(\underline{g}^{best}) >> 0,\ I(\underline{p}_i^{best}) >> 0$
- **repeat**
- new iteration $k = k + 1$
- The calculation of the criterion function $I(\underline{\theta}_i)$ for all particles, $i = 1, ..., n$.
- The calculation of globally the best particle $\underline{g}_{best} = argmin(I(\underline{g}^{best}), I(\underline{\theta}_1), ..., I(\underline{\theta}_n))$
- The calculation of the best value of the particle $\underline{p}_i^{best} = argmin(I(\underline{p}_i^{best}), I(\underline{\theta}_i)),\ i = 1, ..., n$.
- The calculation of the vectors of random values $\underline{r}_p$ and $\underline{r}_g$ with the dimension d.
- The calculation of the movement of particles (velocity)

$$\Delta\underline{\theta}_i = \omega\Delta\underline{\theta}_i + c_1\underline{r}_p \circ (\underline{p}_i^{best} - \underline{\theta}_i) + c_2\underline{r}_g \circ (\underline{g}^{best} - \underline{\theta}_i),\ i = 1, ..., n$$

  and the limitation of velocities of particles

$$\Delta\theta_r^i \in \left[\Delta\theta_r^{\min}, \Delta\theta_r^{\max}\right],\ r = 1, ..., d,\ i = 1, ..., n$$

- The calculation of new particle values

$$\underline{\theta}_i = \underline{\theta}_i + \Delta\underline{\theta}_i,\ i = 1, ..., n$$

- The calculation of position limits

$$\theta_r^i \in \left[\theta_r^{\min}, \theta_r^{\max}\right],\ r = 1, ..., d,\ i = 1, ..., n$$

- **until** the condition for completing the algorithm is not fulfilled.

Particle swarm optimization - PSO

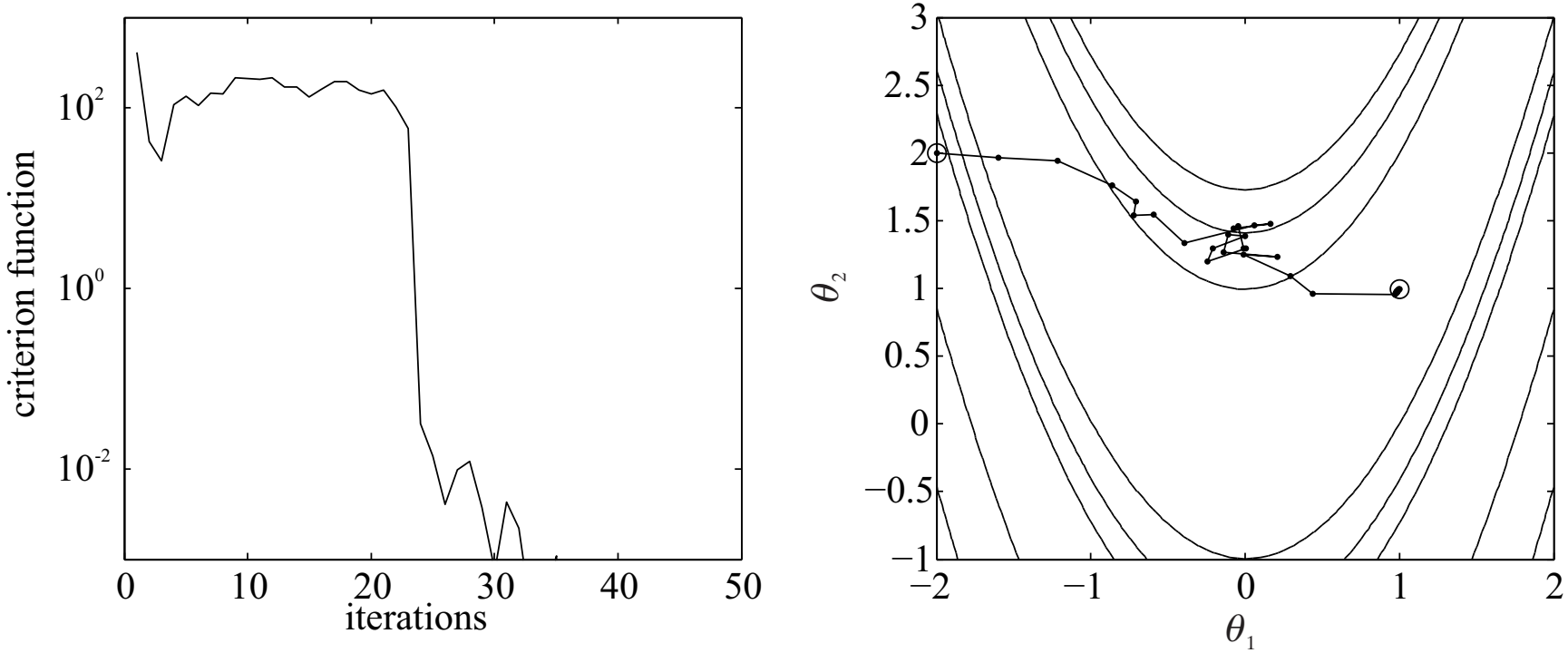


Figure 7: a) An example of the search for the minimum of Rosenbrock function with PSO at initial conditions $[-2\ 2.0]^{\mathrm{T}}$ with respect to iterations, b) in parameter space

Particle swarm optimization - PSO

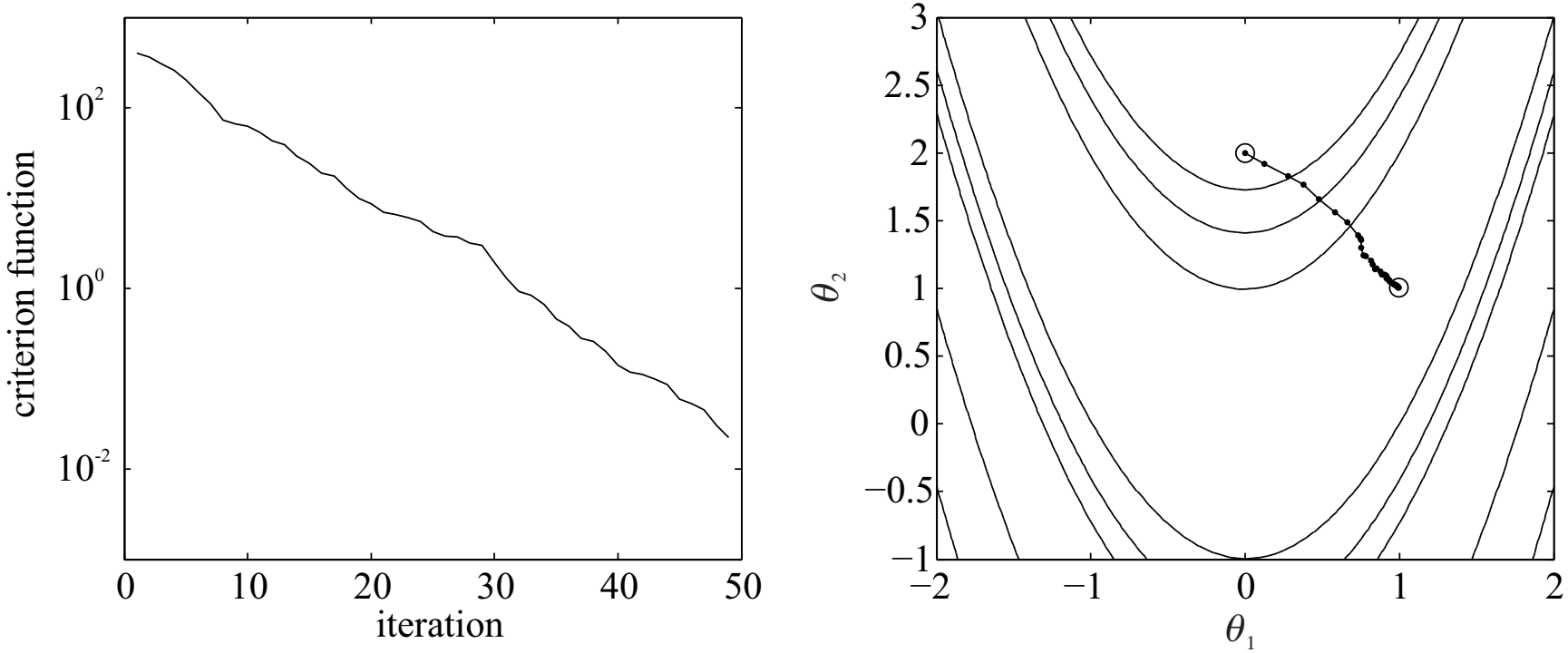


Figure 8: a) An example of the search for the minimum of Rosenbrock function with PSO at initial conditions $[0\ 2.0]^{\mathrm{T}}$ with respect to iterations, b) in parameter space

Evolutionary Algorithms - EA

Method description:

- Evolutionary algorithms belong to the family of stochastic optimization techniques
- they are based on the imitation of nature, like simulated annealing (SA) and particle swarm optimization (PSO)
- in the GA and PSO method we have a population or a swarm of individuals or particles
- population or swarm of particles means the parallel operation of the algorithm, which therefore can be greatly speeded up.

- artificial population develops through generations (i.e. iteratively).
- individuals change from generation to generation due to: **recombination**, **mutations** and other genetic operators.
- with genetic operators new individuals are generated
- for each new **individual we calculate its criterion function**, with **selection** we select better individuals, those with a lower value of the criterion function.

- The criterion function is usually called the fitness function and is an inverse of the classical cost function, i.e. $1/I(\underline{\theta})$ or $-I(\underline{\theta})$.
- selection ensures that the population develops in the direction of better individuals, which means the solving of the optimization problem.

Evolutionary Algorithms - EA

Table 1: Comparison of terminology in the field of evolutionary algorithms and classical optimization

Evolutionary Algorithm	Classical Optimization
Individual (individum)	Vector of parameters
Population	Set of vectors of parameters
Fitness function	Inverse value of cost function
Fitness	Inverse cost function
Generation	Iteration
Influence of evolution operators	Change of the vector of parameters

Pseudo code for an evolutionary algorithm - EAs

1. Random selection of the initial population of S individuals (the point in the parameter space)

$$\underline{\theta}_0 = [\underline{\theta}_{1,0}\ \underline{\theta}_{2,0}\ \cdots\ \underline{\theta}_{S,0}]$$

2. Calculation of the criterion function for the initial population, or fitness function $\mathrm{F}(\underline{\theta}_0)$
3. **repeat**
4. New iteration $k = k + 1$
5. Selection of the most suitable individuals

$$\underline{\theta}_{\text{new}} = \mathrm{Select}(\underline{\theta}_{k-1})$$

6. Genetic operations (mutation, recombination, etc.)

$$\underline{\theta}_k = \mathrm{GenOps}(\underline{\theta}_{\text{new}})$$

7. Calculation of the criterion function for the initial population, or fitness function $\mathrm{F}(\underline{\theta}_k)$
8. **until** the condition for completing the algorithm is not fulfilled

Evolutionary Algorithms - EA

Division of EA and properties:

- evolutionary algorithms differ according to the **selection** strategy and the selection of genetic operators
- the choice of strategic parameters, such as the frequency of mutation and recombination, is a matter of the planner
- or it is optimized with EA itself, which naturally again distinguishes the algorithms among themselves
- large differences are also in encoding of the parameters or structures that we optimize.

- Evolutionary algorithms can be divided into the *evolutionary strategies (ES)*, *genetic algorithms (GA)*, *genetic programming (GP)* and *evolutionary programming (EP)*.

Evolution strategy - ES

Description of basic properties

- The main feature of the evolution strategy (ES) is the treatment of optimized parameters in their original form
- real parameters are treated as real numbers, and integer parameters as integers.

- for ES **mutation** is one of the dominant genetic operators
- parameters mutate with respect to normal Gaussian distribution.
- in the simplest case all parameters change according to the same Gaussian distribution

$$\theta_i^{(\mathrm{new})} = \theta_i + \Delta\theta \,, \tag{9}$$

where $\Delta\theta$ is distributed with respect to the following probability density function:

$$p(\Delta\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{\Delta\theta^2}{\sigma^2}\right) \,. \tag{10}$$

Individual at ES

- standard deviations of parameter changes are encoded into the individual and therefore they are also the object of optimization
- with ϑ_i we label *strategic parameters*
- an individual with n parameters θ_i and additional n individual strategic parameters ϑ_i
- optimization of strategic parameters is called *self-adaptation* or *second level of learning*

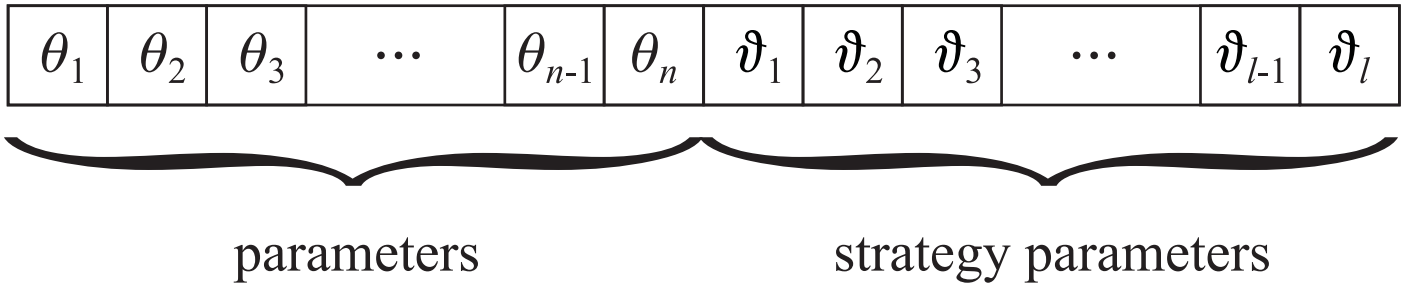


Figure 9: In the case of the ES the individual contains, in addition to the parameters that we optimize, also the strategic parameters (meta parameters) that determine the internal behaviour of the ES. They are also liable to optimization

Recombination

- beside mutation *recombination* plays an important role as well
- two principles of recombination: the first is the *method of averaging*, which calculates a given average of two parents and transfers the value to the descendant, while the other is *discrete recombination*, where the descendant takes certain parameters from one parent, and other parameters from the other. We can extend both approaches to more parents.

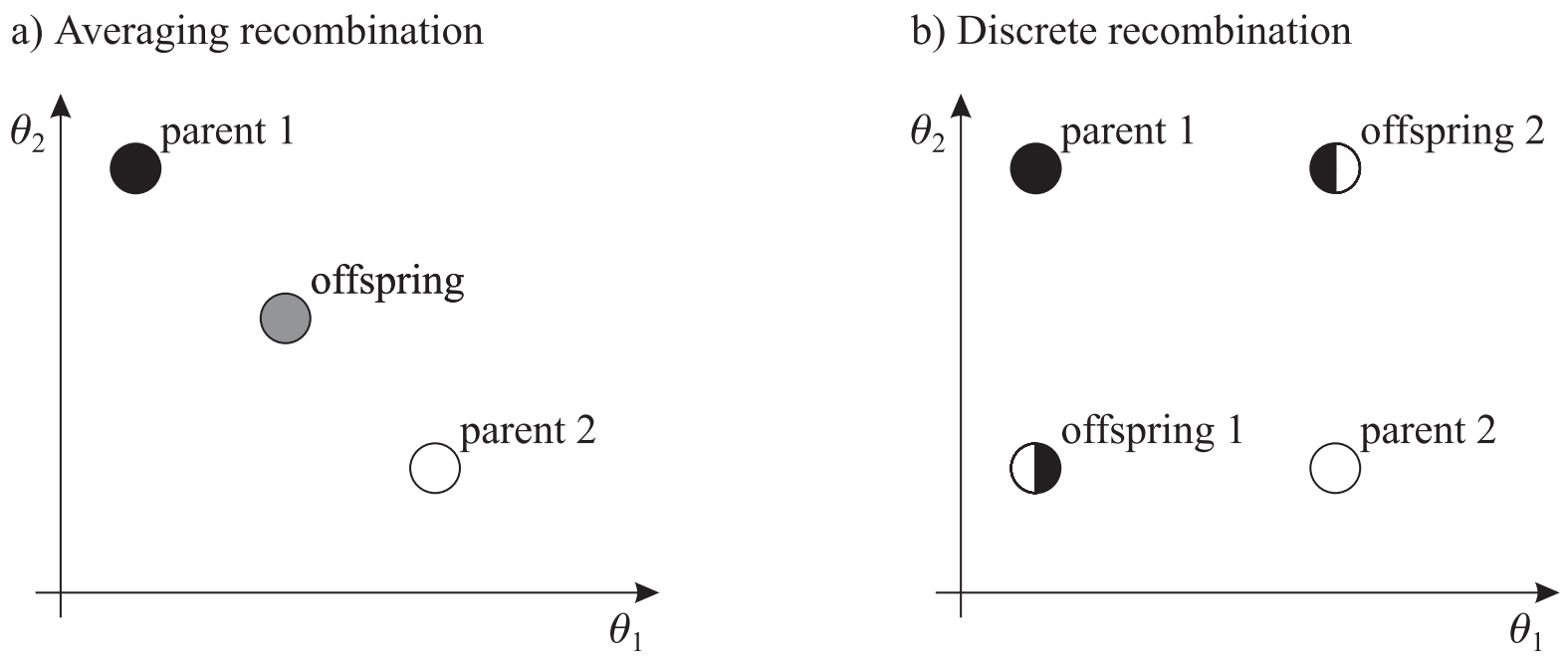


Figure 10: The recombination in the evolution strategy is performed as a) averaging or b) a discrete combination

Evolution strategy - ES

Selection

- *Selection* is completely deterministic in the case of ES
- individuals are ranked according to their fitness function and we select the best ones
- there are two different variants of deterministic selection: $(\mu + \lambda)$ - *strategy* and (μ, λ) - *strategy*
- μ is the number of parents in each generation and λ is the number of children resulting from mutations and recombinations.
- in each generation we choose only μ best individuals for new parents.
- greater values of μ lead to a more global search
- suggested relationship μ/λ is $1/7$, which represents a good compromise between local and global search.

- $(\mu + \lambda)-$ **strategy**: μ parents of new generation are chosen from both μ parents and λ children of previous generation. This means that parents compete with their children in the selection. In this way, the information about the best individual can not be lost, i.e. the criterion function is monotonous regarding the generations.
- $(\mu, \lambda)-$ **strategy**: μ parents of new generation are chosen from λ children only.

Evolution strategy - ES

Table 2: Comparison between the evolution strategy and the genetic algorithm

	Evolution Strategy	Genetic Algorithm
Encoding	Real values	Binary (by nature)
Mutations	Important	Negligible
Recombination	Important	Important
Selection	Deterministic	Random
Strategic parameters	Adaptive	Constant

Pseudo code for the evolution strategy $\mu + \lambda$ - ES

1. Initialization of parameters λ, μ and σ and the maximum number of generations k_{max}.
2. Random selection of the initial population of μ individuals (points in the parameter space) for the determination of the initial population $gen = 0$

$$\theta_i = \mathcal{N}(0, \sigma^2),\ i = 1, ..., \mu$$

3. Calculation of the criterion function $I(\theta_i),\ i = 1, ..., \mu$
4. **repeat**
5. New generation $k = k + 1$
6. Recombination of λ children $\theta_j^{new} = (\theta_a + \theta_b)/2,\ j = 1, ..., \lambda$
7. Mutation of children $\theta_j^{new} = \theta_j^{new} + \mathcal{N}(0, \sigma^2),\ j = 1, ..., \lambda$
8. Calculation of the criterion function of a group of children $I(\theta_j^{new}),\ j = 1, ..., \lambda$
9. Selection of the best μ individuals from the set of $\mu + \lambda$ individuals according to the criterion function
10. **until** $k > k_{max}$
11. Selection of the best individual as a result of the algorithm

Introduction and properties of the method

- Genetic algorithms have been developed independently and almost simultaneously with evolution strategies
- In the case of the GA method the parameters are called *genes*, which compose the *chromosome*, which is the vector of parameters
- GA writes all the different parameter types (real, integer, binary) in binary form
- The chromosome is defined as a binary notation, which is shown in figure 11.

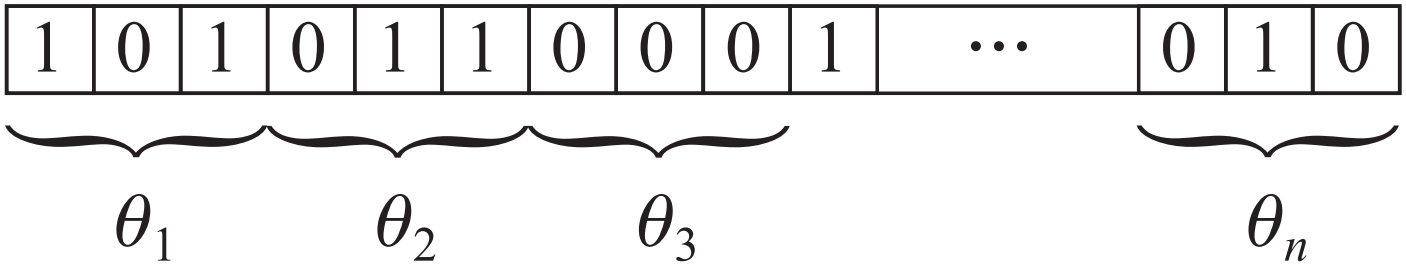


Figure 11: The chromosome contains parameters which are optimized in a binary notation

Example of binary encoding

Encoding the real value into a binary notation

- real parameter $\theta = 0.95$
- with the lower limit $\underline{\theta} = -1$ and the upper limit $\overline{\theta} = 2$
- We encode its value with a 12-bit binary notation $b = 12$
- The length of the binary notation b specifies 2^b of different levels of value and $2^b - 1$ intervals between the lower and upper values (quant).
- The smallest interval between two levels is the resolution interval and is equal to

$$\Delta\theta = \frac{\overline{\theta} - \underline{\theta}}{2^b - 1}$$

- The difference between the real parameter θ and its lower limit $\underline{\theta}$ can be written with the nearest natural number of resolution intervals $N \in \mathcal{N}$ and the remainder δ

$$N = round\left(\frac{\theta - \underline{\theta}}{\Delta\theta}\right) \tag{11}$$

$$\theta - \underline{\theta} = \Delta\theta \cdot N + \delta \tag{12}$$

The resolution interval is $\Delta\theta = 7.3260 \cdot 10^{-4}$.

- $N = 2662$ and $delta = 2 \cdot 10^{-4}$. The value δ is ignored and it is the quantization error.

Example of binary encoding

- The value of the number of resolution intervals is written in a binary form

$$2662 = 1 \cdot 2^{11} + 1 \cdot 2^{9} + 1 \cdot 2^{6} + 1 \cdot 2^{5} + 1 \cdot 2^{2} + 1 \cdot 2^{1}$$

  We get a binary notation of the real parameter $\theta_{bin} = 101001100110$

Conversion from binary to real value

- to convert a binary notation of a parameter with the known lower and upper value to the real value, we use the following equation

$$\theta = \left(\overline{\theta} - \underline{\theta}\right) \frac{\sum_{i=0}^{b-1} a_i 2^i}{2^b - 1} + \underline{\theta} \tag{13}$$

  where $a_i,\ i = 0, ..., b-1$ are values of the digits in the binary notation with increasing exponents.

- The value of the binary notation is in our case

$$\theta = \frac{(2-(-1))}{2^{12}-1}(2^{11} + 2^{9} + 2^{6} + 2^{5} + 2^{2} + 2^{1}) + (-1) = 0.9502 \tag{14}$$

Mutation

- *Mutation* in the GA method does not play such a significant role as for the ES
- Mutation easily inverts the bit in the notation. The mutation rate, which is typically about 0.001 to 0.01, determines the probability of the mutation rate for each bit of the individual chromosome notation

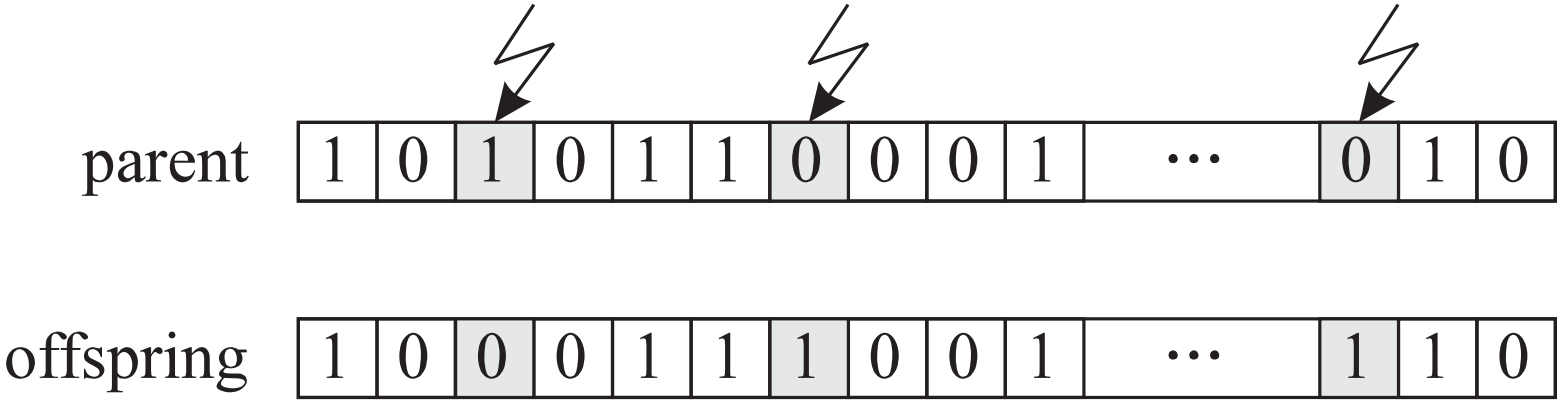


Figure 12: Mutation in genetic algorithms inverts the bit in the binary chromosome notation

Recombination

- *Recombination* is the main operator of the GA method. The binary notation (chromosome) of two (or more) parents is divided into two (or more) parts, and pieces of these chromosomes are crossed (cross)

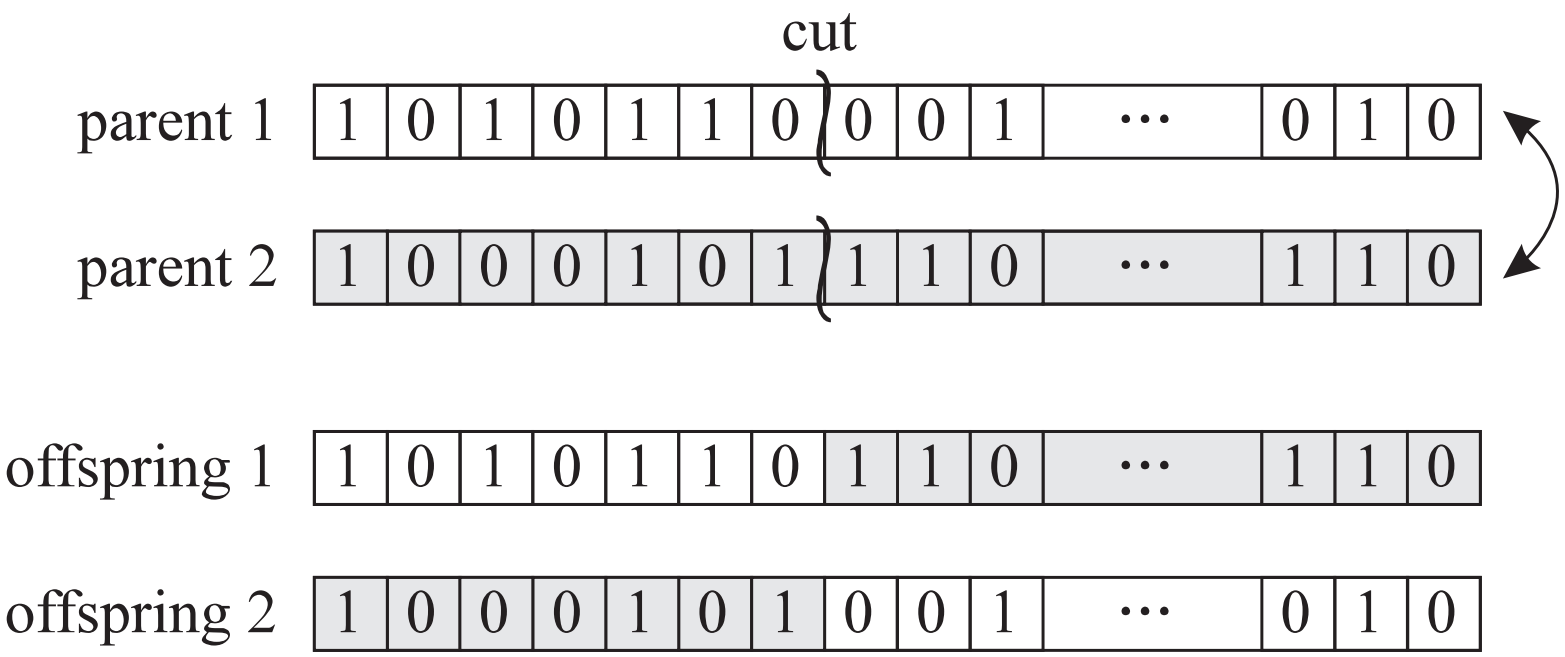


Figure 13: Recombination in genetic algorithms is called crossing because parents' code notations are split into one or more points and are recombined with one another.

Selection

- *Selection* is performed randomly in the case of the GA method
- two most common principles: **roulette wheel selection** and **tournament selection**
- the individual does not necessarily have to survive, even though it has a very good criterion function. This is in line with the imitation of nature.
- The most common process is selecting with the *roulette wheel* . Every individual has the possibility to survive, which is proportional to his fitness function F_i, $i = 1, ..., n$.
- The probability of choice for each individual $P_i^{(\text{select})}$ is calculated by its normalized (positive) quality value:

$$P_i^{(\text{select})} = \frac{F_i}{\sum_{j=1}^{n} F_j} \,. \tag{15}$$

  where n is the size of the population.

The fitness function must be positive: we invert the criterion function $1/I(\underline{\theta})$ or we invert the sign of the criterion function to obtain $C - I(\underline{\theta})$, where C is greater than the maximum value of the criterion function.

Roulette wheel

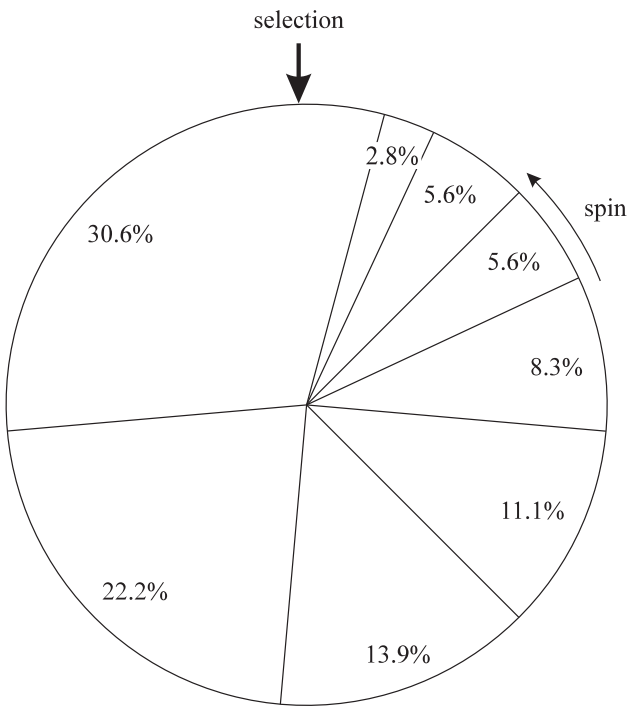


Figure 14: When selecting with a roulette wheel, each individual has a probability to survive, which is proportional to his fitness function. There are 8 individuals in the figure.

Tournament Selection

- Alternative selection method
- From the set of all possible individuals we select two, and according to their criterion functions we select the better one as one of the parents. We do the same for the other parent.

Selection pressure

- the pressure of selecting a certain individual increases with generations
- the reduction of global exploration
- the problem is solved by limiting the best values, the worst values are set to 0. In this way, the selection pressure is more evenly distributed over all generations.

Figure 15 shows that for the very high selection pressure, the diversity of the population decreases because only the best individuals survive. This leads to *premature convergence*, i.e. GA loses its global search capability.

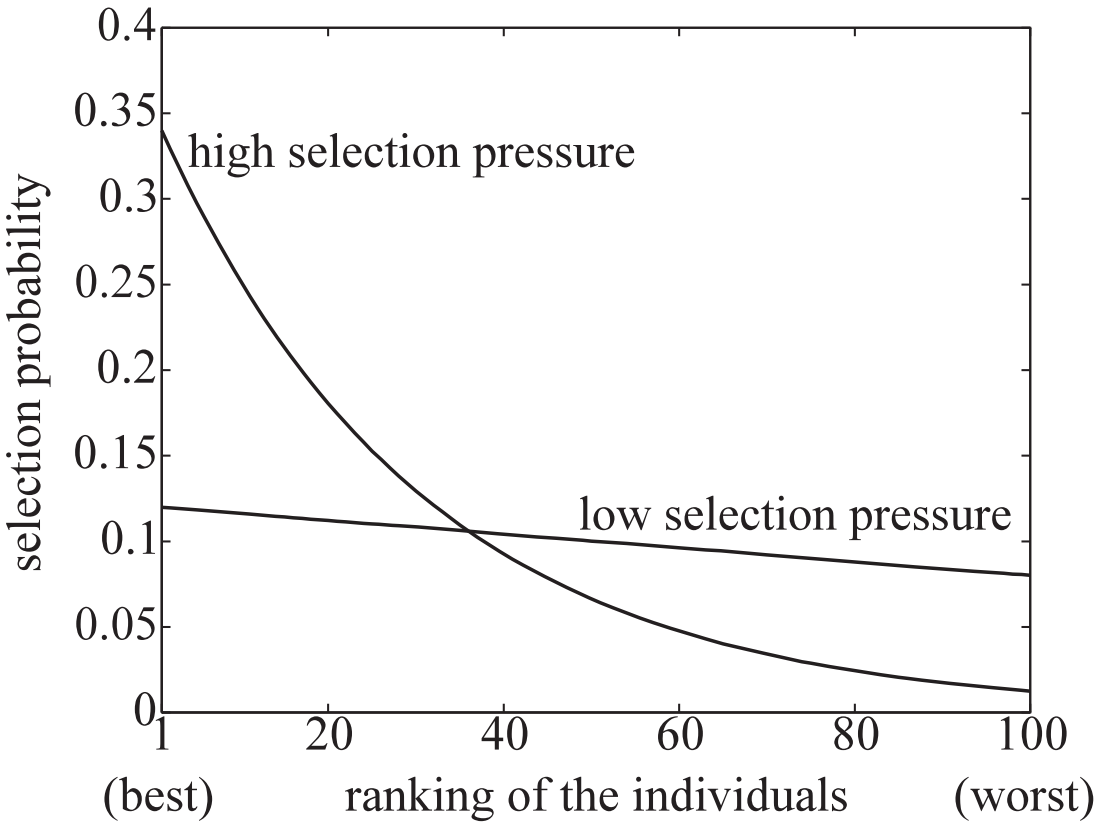


Figure 15: We select ten individuals from the set of 100. The selection pressure is great if better individuals are favoured over the poor ones. If all individuals had the same viability, then the selection pressure would be lowest. In this way, GA degenerates into random search

Method properties

- Genetic programming is very similar to genetic algorithms, but it works on tree structures rather than on binary notations.
- Trees are variable structures that allow us to write certain relations very effectively.
- Two examples (figure 16) of a logical expression and a mathematical equation are written with a tree structure.
- Tree leaves usually represent variables and constants, while other links are operators.
- The user specifies the set of variables, constants and operators to be used in the tree.

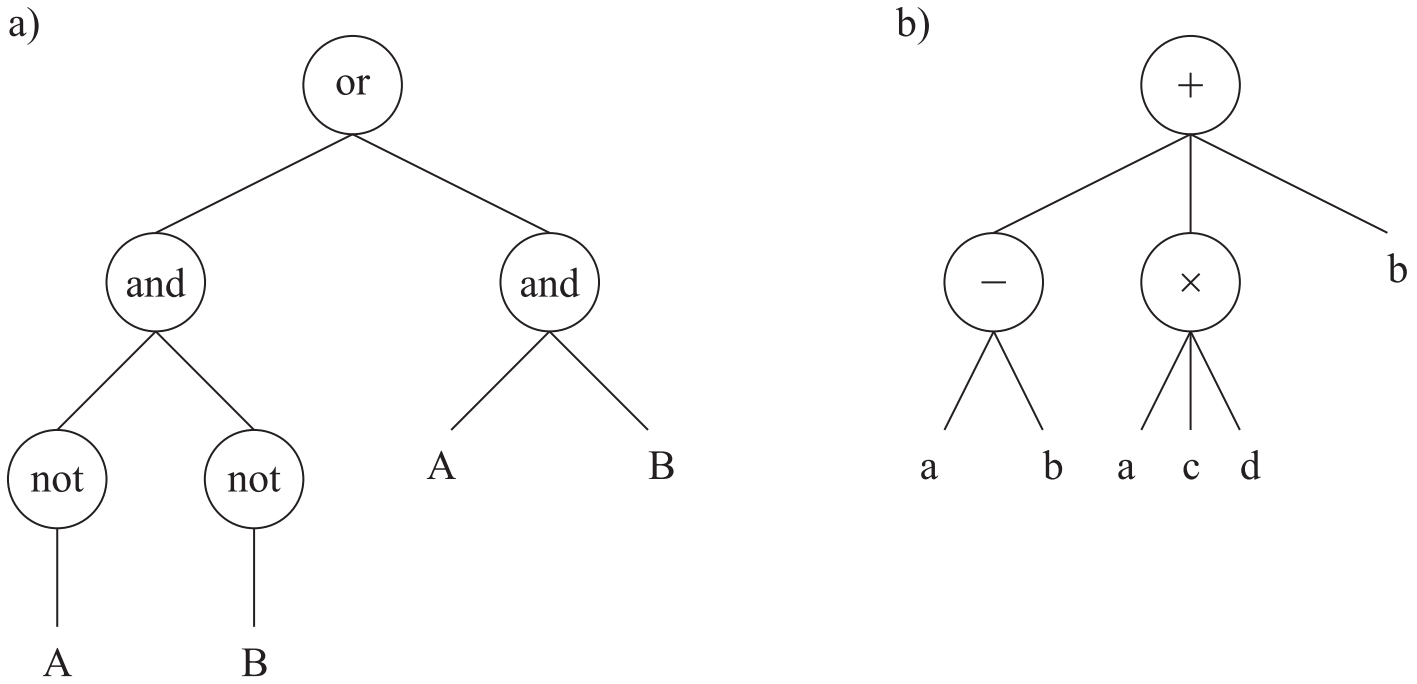


Figure 16: In the case of genetic programming, the individual is represented by a tree structure. The tree structure allows the execution of a) logical operations (here: XOR), b) mathematical equations (here we have the operation $a + acd$ realized in a redundant manner), and many other structures

Mutation and Crossing

- Mutation and crossing as we know it in the GA method need to be changed for the use in the GP method.
- implementation of the mutation, parts of the tree (variables, constants and operators) can be randomly modified, deleted or newly generated.

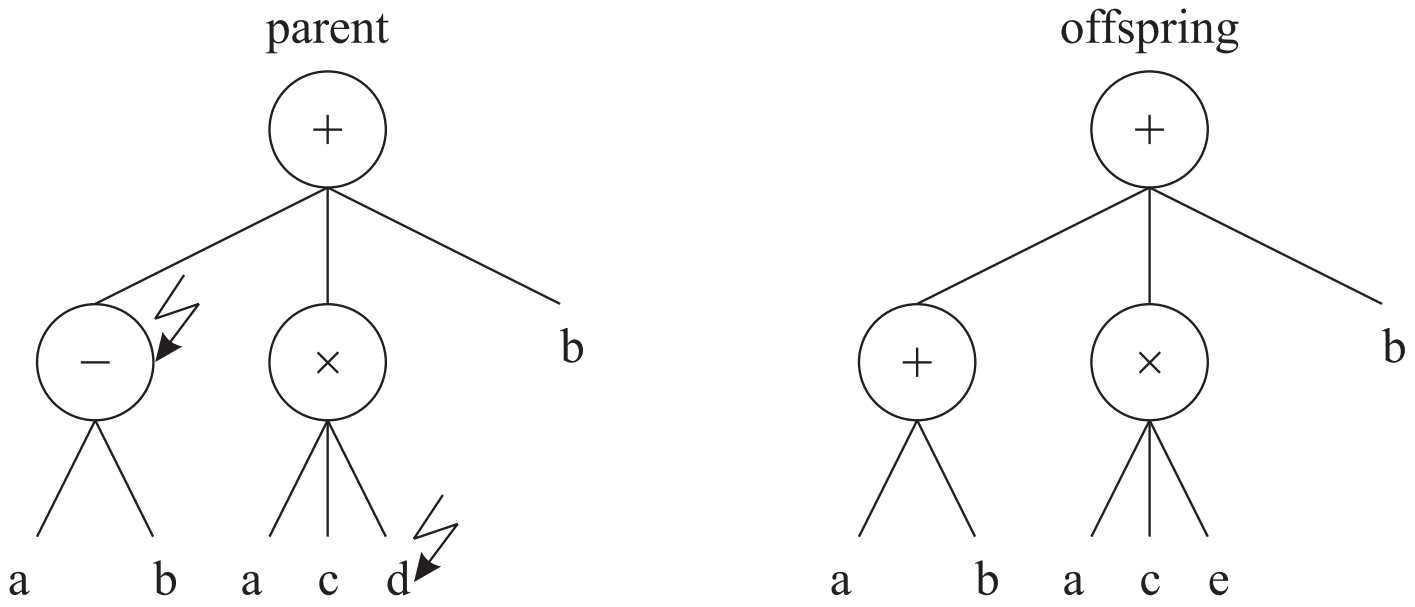


Figure 17: The mutation in genetic programming replaces parts of the tree structure with new variables and operators

Genetic programming - GP

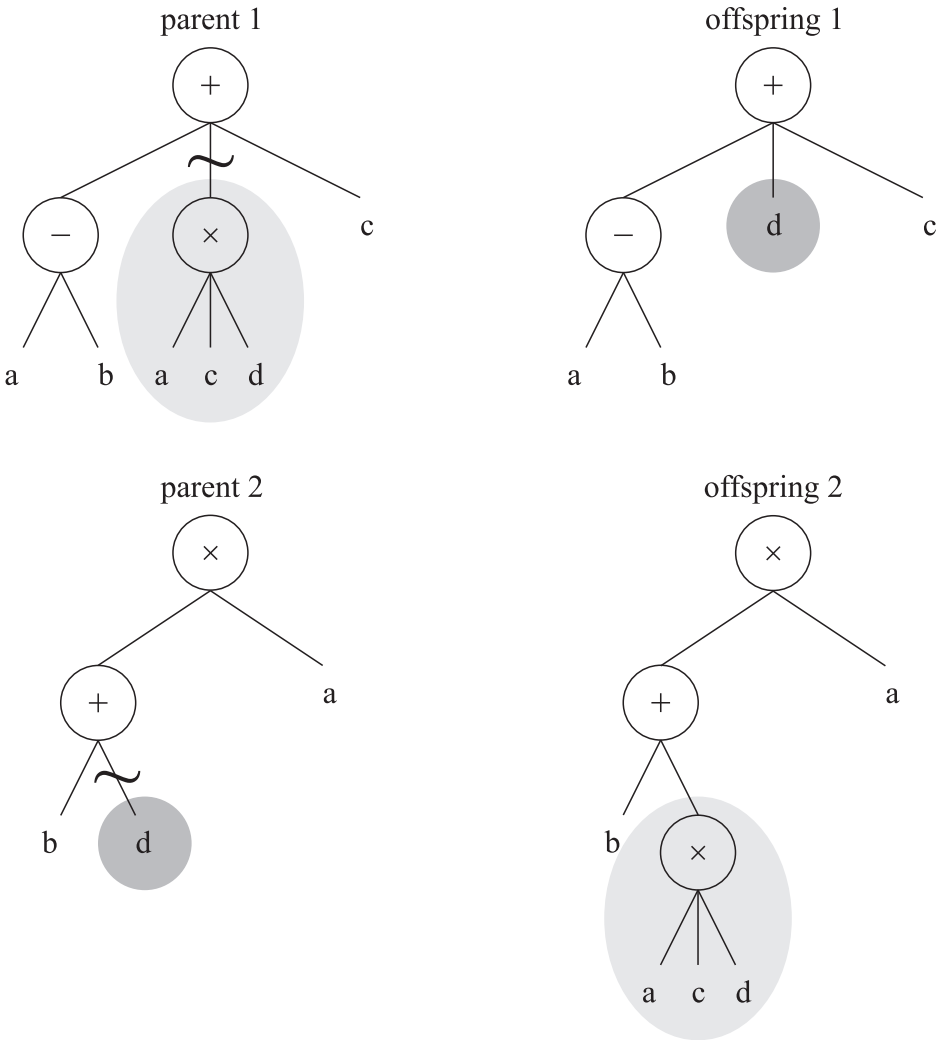


Figure 18: Recombination in genetic programming replaces tree parts among parents

Branch and bound - B & B

Introduction and method properties:

- (*Branch and bound - B & B*) is a tree method for searching for the optimum of the criterion function
- it is suitable for solving combinatorial problems
- With certain constraints it can also be used for optimizing parameters. In this case, the parameters must be discrete or we discretize them to a certain length of the binary notation.
- The idea behind B & B method is in the construction of a tree that contains all possible combinations of discrete parameters (see figure 19), and in exploring the tree.
- The method, based on the calculation of the criterion function in each node of the tree, determines a further path and eliminates the parts of the tree that do not lead in the optimal direction.

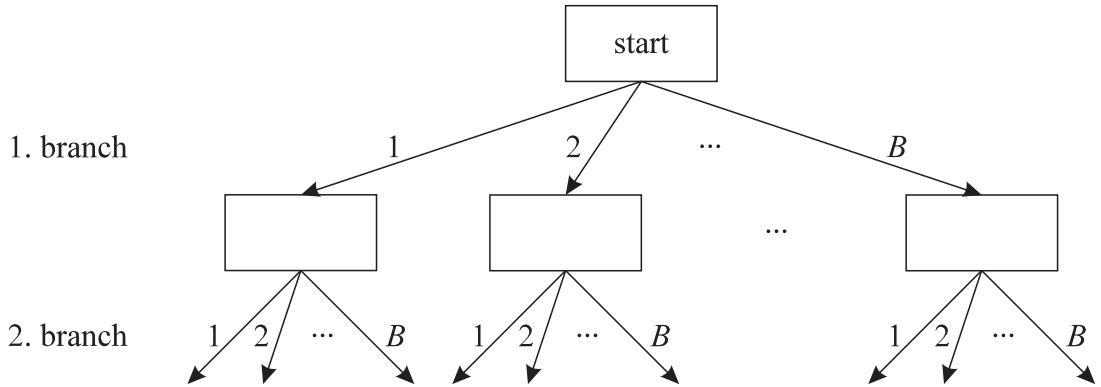


Figure 19: The illustration of the B & B method in the case of three parameters, each having B discrete values

Branch and bound - B & B

Main features:

- information is needed on the lower boundary of the criterion function in the given branch, from the treated node on
- we use this information to eliminate the part of the tree (*pruning*) for which we know that results in a worse criterion function than the current optimal.
- we can also easily enter other constraints that can be checked on each node and eliminate part of the tree if the restrictions are violated.

- The problem of extending B & B to optimizing **real-value parameters** is that we do not know the impact of individual parameters on the criterion function.
- An example of the optimization of three parameters that are discretized on B values. In this case the tree structure has 3 levels and B^3 end-nodes (leaves). This structure represents the whole model.
- Criterion function can not be calculated by first or second branch. Values can not be calculated because the last parameter appears only after the second branch. However, this parameter can have a very important influence on the criterion function. In this way, it is not possible to *shrink the tree* and the method becomes ineffective.

Example of the optimization of product creation with B & B optimization

- The product is processed in four stages (1, 2, 3, and 4).
- There are four devices available (A, B, C, and D) which need different time to perform a certain phase.
- Each device can only be used for one phase throughout the entire manufacturing process. The times of individual devices for a particular phase are displayed in table 3.
- To start B & B optimization, it is necessary to set the lower limit of production time $LB_0 = 2 + 3 + 1 + 4 = 10$, which is not necessarily feasible.
- From the starting point ($LB_0 = 10$) we fork the tree into 4 branches, one branch for each device.
- When device A is foreseen for first phase, we first eliminate the first column and the first row in the table. We check the minimum sum in the other three columns and rows having in mind that it is not necessarily feasible and we get $LB_1 = 9 + 3 + 1 + 4 = 17$.
- We do the same for the second column and device B. We get $LB_2 = 2 + 3 + 1 + 4 = 10$
- for device C we get $LB_3 = 7 + 4 + 5 + 4 = 20$,
- for device D we get $LB_4 = 8 + 3 + 1 + 6 = 18$.
- The smallest lower limit is $LB_2 = 10$ and hence we continue branching in this node.
- Phase 1 with device B, and for the second phase we create 3 branches from this node, the first one for device A, the second one for device C, and the third one for device D. The first branch has $LB_5 = 2 + 6 + 1 + 4 = 13$, the second branch has $LB_6 = 2 + 3 + 5 + 4 = 14$ and the third branch has $LB_7 = 2 + 7 + 1 + 7 = 17$.
- The best solution $LB_5 = 13$, which goes through device B to device A.
- From the LB_5 node we continue to two branches. The first branch has selected device C for phase 3, and device D for phase 4 .
- We get $LB_8 = 2 + 6 + 1 + 4 = 13$ and the second branch has the opposite choice, resulting in $LB_9 = 2 + 6 + 8 + 9 = 25$. The optimal solution: $B \rightarrow A \rightarrow C \rightarrow D$.

Branch and bound - B & B

Table 3: Processing times for different processing stages and different devices

	Device A	Device B	Device C	Device D
Phase 1	9	2	7	8
Phase 2	6	4	3	7
Phase 3	5	8	1	8
Phase 4	7	6	9	4

Optimization with tabu search - TS

Idea and features:

- the method is very useful in solving combinatorial problems and operational problems (travelling salesman problem, finding the best paths, optimal schedules, etc.).
- combining successful local optimizations into a global optimization strategy
- the simplest version is the multiple sequential startup of local optimization methods from a variety of randomly selected initial values, where the TS algorithm remembers the already examined points and then avoids them (the already explored points are tabu).
- In terms of global space exploration, it is essential to allow the algorithm to make a step towards a worse solution, because only in this way it can avoid local optimums.

Unsupervised learning methods

Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for the 6th lecture, November 2016

Table of contents

1. Unsupervised learning methods
2. Principal component analysis
   - Matrix inversion with collinear data and regression based on PCA
     - Example of regression using PCA
3. Clustering techniques
   - Similarity measures
4. Clustering algorithms
   - k-means clustering
   - Fuzzy c-means clustering - FCM
     - An example of FCM algorithm
     - Example of an inner matrix norm with an inverse variance diagonal
   - Gustafson-Kessel clustering method - GK
   - Fuzzy Maximum Likelihood Estimates method - FMLE

Introduction to unsupervised learning

The most important types of learning: supervised, unsupervised and reinforcement learning.

- **Supervised learning**. We are looking for mapping, i.e. function f in a given set of functions based on data sets $\underline{u} \in U,\ y \in Y$, which maps the input data into the output data based on function $f:\ \underline{u} \rightarrow y$.

- **Unsupervised learning**. Unsupervised learning differs from the supervised learning by the fact that in the case of unsupervised learning we do not have a defined apriori output, but all data are treated as input and we are trying to find the relations between these data.

- **Reinforcement learning.** This type of learning is based on information about the final outcome of the observed experiment. A typical example of this kind of learning are games (chess, cards), where it is not possible to evaluate each move separately. Through the final result of the game (win, defeat, draw) we make conclusions about the quality of the chosen strategy.

Unsupervised learning

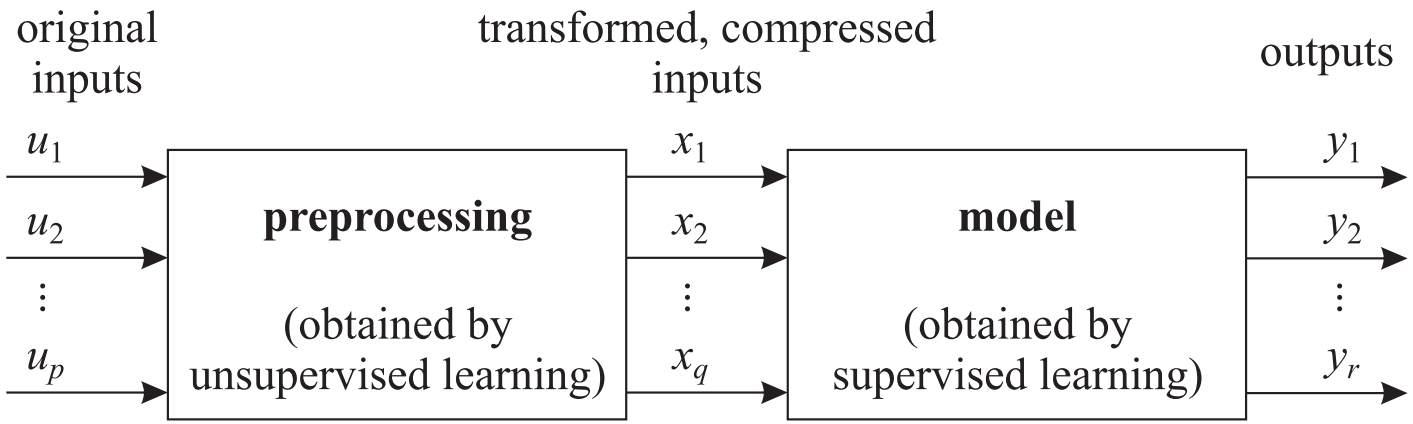


Figure 1: Data preprocessing is usually performed with unsupervised learning techniques. Original variables $u_1, u_2, \ldots, u_p$ are first pre-processed to obtain their transformations $x_1, x_2, \ldots, x_q$, where the number of transformers q is usually smaller than the number of original variables p, since processing usually also means compressing the input space and reducing the complexity of the model.

Example of using unsupervised learning

An example of using unsupervised learning for a simple classification example.

- partitioning of data in the two-dimensional space of variables u_1-u_2.
- variables u_1 and u_2 represent two properties on the basis of which we can distinguish different classes which are presented by integer values.
- value of the class represents the model output y
- Let's assume that we have a problem similar to the XOR operation. Thus the group of data (clusters) up on the left and down on the right belong to class 1, the other two belong to class 0.

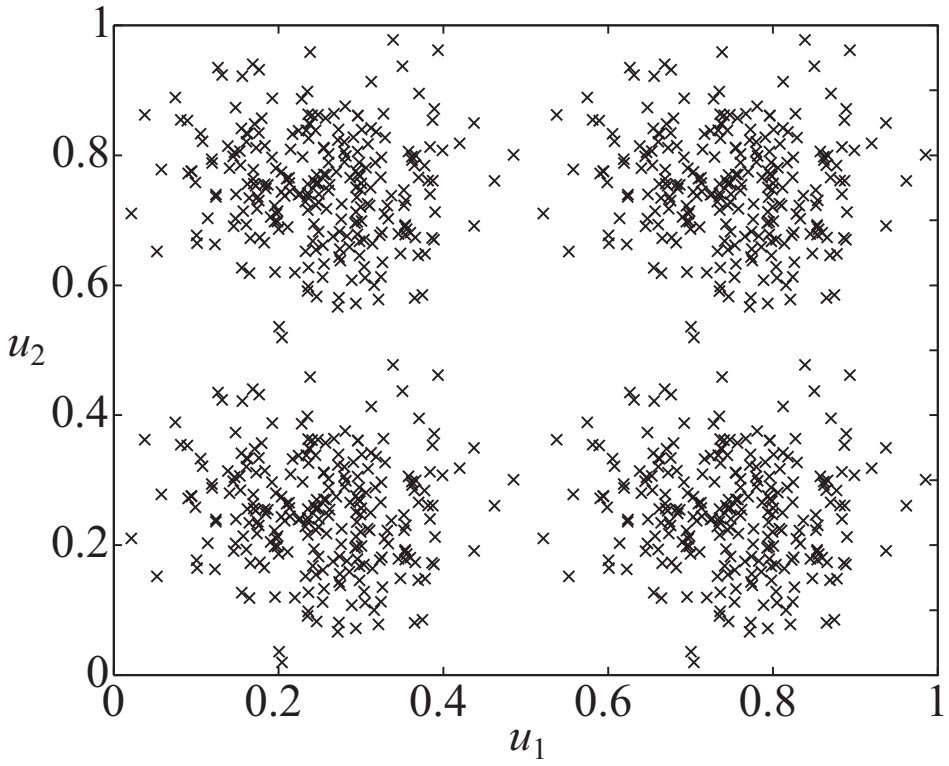


Figure 2: Four clusters in a two-dimensional space. Without the information on an appropriate output, we can distinguish four groups of data, i.e. clusters, by means of unsupervised learning.

Example of using unsupervised learning

- In supervised learning, each learning sample would include the variables u_1 and u_2 and the corresponding value of the output y, i.e. 1 or 0 (*labeled variables*).
- In the case of unsupervised learning, we do not know the output, which means that the learning set consists only of input data without information on the corresponding classes.
- The best we can do with unsupervised learning is to find groups, i.e. clusters, in the input data. In this way we can find, for example, centers of individual clusters with values (0.25, 0.25), (0.75, 0.25), (0.25, 0.75) and (0.75, 0.75) in the form of circles with a radius of 0.25.
- In the second step, these clusters are transformed with supervised learning to the output values of the corresponding classes . In this way, by using unsupervised learning, we transformed the problem of mapping the entire set of input data to the mapping of 4 centers of clusters into two classes. This has greatly reduced the complexity of the second transformation.

Unsupervised learning on input-output data

- integrating the output variable without explicit optimization of the criterion function with respect to the output error (as with supervised learning)
- the method is called *product space clustering*
- we perform the clustering on the entire *product space*, which consists of the variables

$$\underline{u}_e = [u_1 \; u_2 \; \cdots \; u_m \;\; y]^{\mathrm{T}} \, . \tag{1}$$

- we obtain clusters with relations between input-output variables, which in another way define a model within this subspace

Principal Component Analysis

- Principal Component Analysis deals with the covariance matrix of centred and normalised data
- with transformation (rotation) of the axes of the coordinate system we obtain a new orthogonal system which results in the largest covariance of data with regard to the newly selected axes.

An example is shown in figure 3.

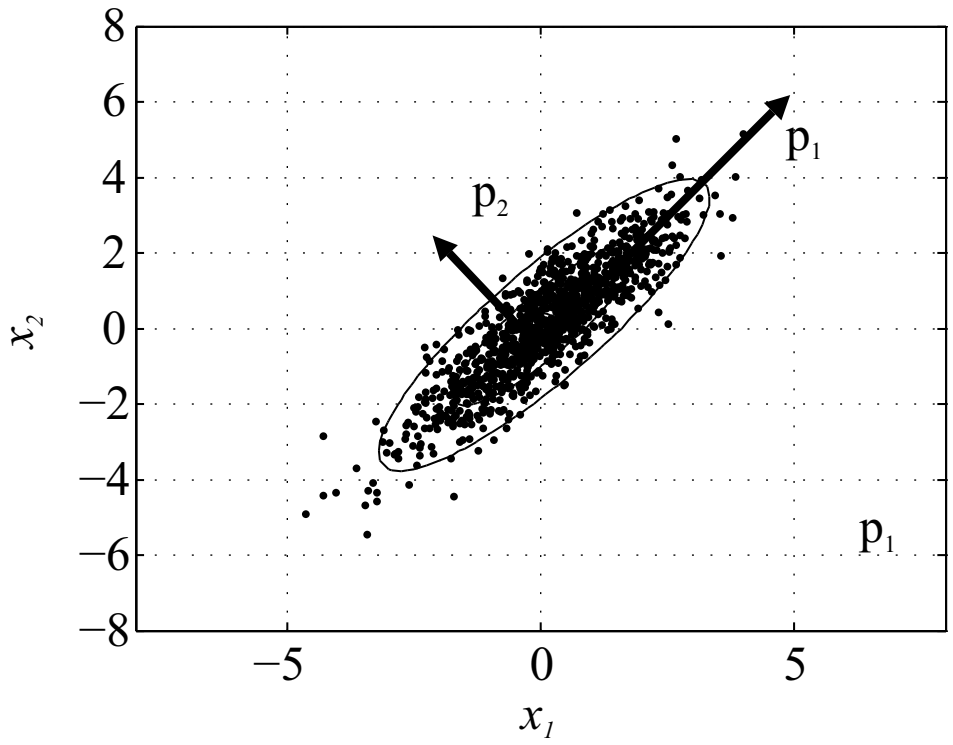


Figure 3: Analysis of data by principal component analysis (PCA) for a two-dimensional problem.

Principal component analysis

- Measurements of two random variables x_1 and x_2 in n time frames. Each pair (x_1, x_2) represents a point in two-dimensional space of the measurements.
- Measurements are written into a matrix of data $\underline{X} \in \mathbb{R}^{n \times 2}$, where each row represents the sample of measurements of both variables in certain time instant $\underline{x}_k = [x_{1k},\ x_{2k}]$.
- Variables are, for example, the consequences of two processes that are strongly correlated, which means that there is a certain linear dependence between them and that the deviations from this linear relationship are the result of noise.

Principal component analysis

- projection of the data matrix $\underline{X}$ into the set of base vectors $\underline{p}$ is written as $\underline{X}\underline{p}$.
- we search for the base vector around which the variance of the data is the largest.
- optimization of the maximum variance by limiting the length of the base vectors can be most easily solved by using the Lagrangian coefficients, where we optimize the functional:

$$J(\underline{p}) = \left(\underline{X}\underline{p}\right)^T \left(\underline{X}\underline{p}\right) + \lambda \left(1 - \underline{p}^T \underline{p}\right) \tag{2}$$

If we calculate the partial derivative $\frac{\partial J}{\partial \underline{p}}$ we get the following equation

$$\frac{\partial J}{\partial \underline{p}} = \left(\underline{X}^T \underline{X}\right) \underline{p} - \lambda \underline{p} = 0 \tag{3}$$

Principal component analysis

- directions of the maximum data variance are directions of eigenvectors $\underline{p}_i,\ i = 1, ..., m$, corresponding eigenvalues λ_i represent a variance in those directions
- eigenvectors and eigenvalues of covariance matrix $\underline{X}^T\underline{X}$ can be calculated by using decomposition of singular values of matrix $\underline{X}^T\underline{X}$ (SVD) as shown by the following equation

$$\underline{C} = \frac{1}{n-1}\underline{X}^T\underline{X} = \underline{P\Lambda P^T} = \sum_{j=1}^{m} \lambda_j \underline{p}_j \underline{p}_j^T \tag{4}$$

- $\underline{P} \in \mathbb{R}^{m \times m}$ is the matrix m of orthonormal vectors $\underline{p}_j,\ j = 1, ..., m$
- $\underline{\Lambda}$ is a diagonal matrix of eigenvalue $\lambda_j,\ j = 1, ..., m$.
- orthonormal eigenvector $\underline{p}_j$ belongs to eigenvalue λ_j
- matrix of eigenvectors $\underline{P}$ is arranged in such a way that the column vectors are ranked from left to right by their respective decreasing eigenvalues $(\lambda_1 > \lambda_2 > \cdots > \lambda_m)$.

Principal components and redundancy

- in the data matrix there can also be certain collinear variables
- in the case of collinearity we do not need all eigenvectors of covariance matrix $\underline{C}$
- s of eigenvectors that explain the main part of information (ϵ) are called *principal components*
- Criteria for choosing the number of eigenvectors is as follows

$$\frac{\sum_{j=1}^{s} \lambda_j}{\sum_{j=1}^{m} \lambda_j} \geq \epsilon \tag{5}$$

  where ϵ is the selected threshold, which is usually defined by value $\epsilon = 0.95$.
- covariance matrix can be approximated by a matrix which contains only the main components

$$\underline{C}_s = \sum_{j=1}^{s} \lambda_j \underline{p}_j \underline{p}_j^T \tag{6}$$

  where vectors $\underline{p}_j, j = 1, \ldots, s$ define the main directions of dissemination of data and are weighted with the corresponding roots of eigenvalues.

Principal component analysis

Variables of the new space

- orthogonal variables that are an inner product of measured variables and mapping matrix are called **scores**

$$\underline{T} = \underline{XP} \tag{7}$$

- $\underline{T}$ is the scoring matrix
- matrix of the principal components $\underline{P}$ is a statistical model of the process
- scores are pseudo-variables that do not have a real background, because they are linear combinations of columns of matrix $\underline{X}$.
- if we take $\underline{P}_s$ so that we only take s of the first columns or eigenvectors of matrix $\underline{P}$, we get an approximated data matrix

$$\underline{T}_s = \underline{XP}_s \tag{8}$$

$$\underline{X}_s = \underline{T}_s \underline{P}_s^T \tag{9}$$

Matrix inversion with collinear data and regression based on PCA

- regressions in the case of a singular information matrix due to data collinearity
- regression is based on the scores

$$\underline{y} = \underline{X\theta} + \underline{e}_y \tag{10}$$

$$\underline{y} = \underline{TP^T}\,\underline{\theta} + \underline{e}_y \tag{11}$$

$$\underline{y} = \underline{T\theta}_T + \underline{e}_y \tag{12}$$

where $\underline{e}_y$ is the error between the model and the measured data and $\underline{\theta}_T = \underline{P}^T\underline{\theta}$.

The parameters of the model are obtained using the least squares method, where the target function is $I(\underline{\theta}) = \frac{1}{2}\underline{e}_y^T\underline{e}_y$

Matrix inversion with collinear data and regression based on PCA

$$\begin{aligned} \frac{\partial I(\underline{\theta})}{\partial \underline{\theta}} &= \left(\underline{y} - \underline{X\theta}\right)^T \left(\underline{y} - \underline{X\theta}\right) = 0 \\ \underline{\theta} &= \left(\underline{X}^T \underline{X}\right)^{-1} \underline{X}^T \underline{y} \end{aligned} \tag{13}$$

or in case we work with scores $\underline{T}$

$$\underline{\theta} = \underline{P} \left(\underline{T}^T \underline{T}\right)^{-1} \underline{T}^T \underline{y} \tag{14}$$

$$\left(\underline{X}^T \underline{X}\right)^{-1} \approx \left(\underline{X}_s^T \underline{X}_s\right)^{-1} \tag{15}$$

$$\left(\underline{X}_s^T \underline{X}_s\right)^{-1} = \left(\underline{P}_s \underline{T}_s^T \underline{T}_s \underline{P}_s^T\right)^{-1} \tag{16}$$

Considering the relation between the eigenvalue matrix $\underline{\Lambda}_s$ and scores $\underline{T}_s$, we get

$$\left(\underline{X}_s^T \underline{X}_s\right)^{-1} = \left(\underline{P}_s \underline{\Lambda}_s \underline{P}_s^T\right)^{-1} \tag{17}$$

Matrix inversion with collinear data and regression based on PCA

The inverse of the information matrix is

$$\left(\underline{X}_s^T \underline{X}_s\right)^{-1} = \underline{P}_s \underline{\Lambda}_s^\dagger \underline{P}_s^T \tag{18}$$

where $\underline{\Lambda}_s^\dagger$ is a diagonal matrix with inverse eigenvalues of s principal vectors

$$\underline{\Lambda}_s^\dagger = \begin{bmatrix} \frac{1}{\lambda_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{\lambda_s} \end{bmatrix} \tag{19}$$

Vector of the parameters of the model can now be calculated as

$$\underline{\theta} = \underline{P}_s \underline{\Lambda}_s^\dagger \underline{P}_s^T \underline{X}_s^T \underline{y} \tag{20}$$

Example of regression using PCA

- process is modelled with linear combination

$$y = -0.58x_1 + 1.333x_3 + \mathcal{N}(0, 0.02)$$

  where a stochastic signal with a mean value of $\bar{x}$ and a variance σ_x^2 is written with $\mathcal{N}(\bar{x}, \sigma_x^2)$.
- the data matrix $\underline{X}$ consists of variables $x_1 = [1\ -1\ 0\ 1]^T$, $x_2 = [0\ 1\ -1\ 1]^T$ and linearly dependent third variable $x_3 = -2x_1 + 1.5x_2 + \mathcal{N}(0.0.01)$.
- data matrix is

$$\underline{X} = (x_1, x_2, x_3) = \begin{bmatrix} 1.0 & 0.0 & -1.9985 \\ -1.0 & 1.0 & 3.4944 \\ 0.0 & -1.0 & -1.5034 \\ 1.0 & 1.0 & -0.4958 \end{bmatrix} \tag{21}$$

- the output values of the process are (for an assumed error in measurement):

$$\underline{y} = [-3.2475\ 5.2389\ -2.0067\ -1.2417]^T$$

Example of regression using PCA

- $\underline{X}$ is poorly conditioned, so using the regular least squares method we get

$$\underline{y} = 0.8166x_1 - 1.050x_2 + 2.0335x_3$$

- singular values of the matrix $\underline{X}$ are $\lambda_1 = 21.7109,\ \lambda_2 = 3.0000$ in $\lambda_3 = 0$.
- for proper approximation of the system are required two eigenvectors
- the principal eigenvectors are $\underline{p}_1 = [-0.2971\ 0.2234\ 0.9283]^T$ and $\underline{p}_2 = [-0.6009\ \ -0.7993\ \ -0.0000]^T$ and in this way the matrix $\underline{P}_2 = \left[\underline{p}_1\ \underline{p}_2\right]$.
- by using the PCA method we get

$$y = -0.6287x_1 + 0.0362x_2 + 1.3095x_3$$

Definition of a cluster

A cluster is a group of data that are more similar to one another than they are similar to data from other clucters (Bezdek, 1976).

- clusters have different shapes
- with *similarity measure* the user must define what kind of clusters he will try to find
- the most common forms of clusters are hypercircles if we are talking about multidimensional problems. In this case, the similarity measure is the distance between the points and the center of the cluster. In this case, the cluster is defined by **center** or **prototype**
- other similarity measures can give prototypes of lines, circles or ellipses

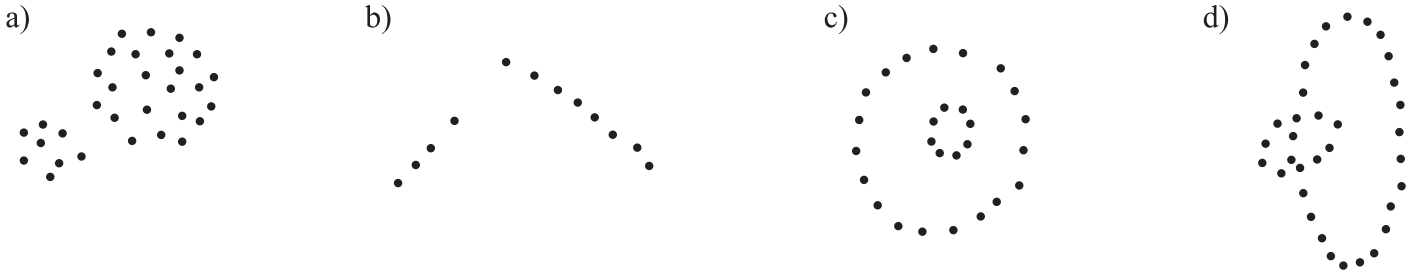


Figure 4: Examples of different shapes of clusters: a) circle, b) line, c) circumference, d) elliptic curve

Clustering techniques

Division of the clustering technique:

- depending on the type of variables for which it is appropriate (continuous, integer, binary),
- according to the similarity measures,
- hierarchical and non-hierarchical,
- specified or self-adaptive number of clusters,
- crisp or fuzzy clusters.

Similarity measures

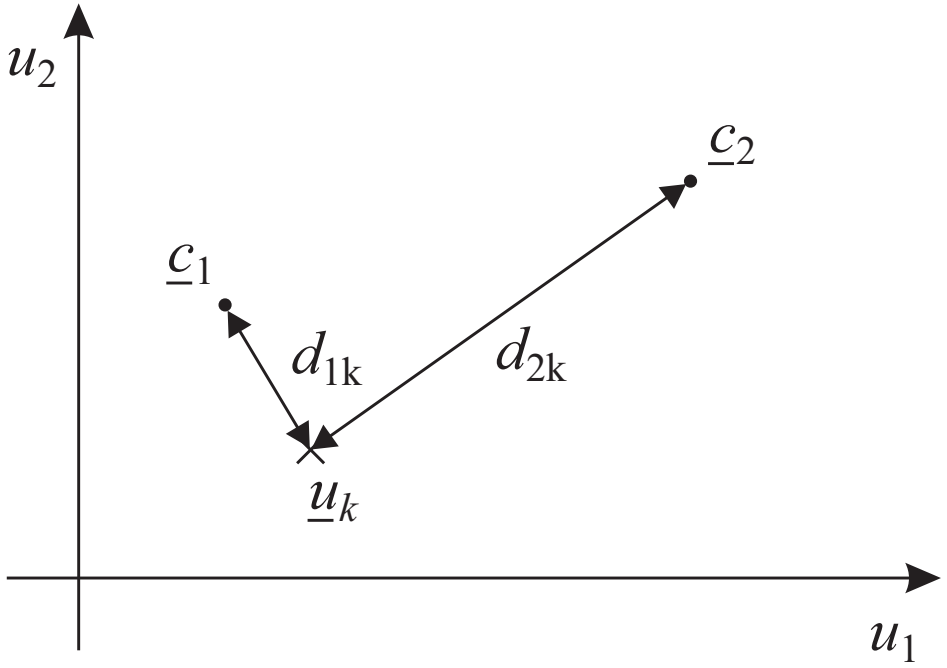


Figure 5: The distances for fuzzy clustering with c-means algorithm. Distances can be obtained with different norms.

Similarity measures

Each measure has certain specific properties. The basic norm is a generalized Minkowski norm. The Minkowski norm is written as $||\underline{x}||_p$, where $p >= 1$. In case of m dimensional vector $\underline{x}$, it is defined by the equation 22

$$||\underline{x}||_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{\frac{1}{p}}, \; p >= 1 \tag{22}$$

If $p = 2$ we get the most famous Euclidean norm (L_2) if $p = 1$ we get Manhattan or Hamming distance or norm (L_1), but if $p \to \infty$ then this is the Chebyshev distance or norm (L_∞).

Table 1: **Selected distance norms between two vectors**

	Distance norms
Minkowski	$d(\underline{x},\underline{y}) = \left(\sum_{i=1}^{m} \mid x(k) - y(k) \mid^{p}\right)^{\frac{1}{p}}\ ,\ p >= 1$
Euclidean	$d(\underline{x},\underline{y}) = \left(\sum_{k=1}^{m} (x(k) - y(k))^{2}\right)^{\frac{1}{2}}$
Generalized squared norm	$d(\underline{x},\underline{y}) = \left(\underline{x} - \underline{y}\right)^{T} \underline{A}^{-1} \left(\underline{x} - \underline{y}\right)$
Hamming	$d(\underline{x},\underline{y}) = \sum_{k=1}^{m} \mid x(k) - y(k) \mid$
Chebyshev	$d(\underline{x},\underline{y}) = \max_{k=1,\ldots,m} \mid x(k) - y(k) \mid$
Canberra	$d(\underline{x},\underline{y}) = \sum_{k=1}^{m} \frac{\lvert x(k)-y(k)\rvert}{x(k)+y(k)},\ x(k), y(k) > 0$
Angular	$d(\underline{x},\underline{y}) = \frac{\sum_{k=1}^{m} x(k)y(k)}{\sum_{k=1}^{m} x^{2}(k) \sum_{k=1}^{m} y^{2}(k)}$

K-means clustering algorithm

- *K-means* is the most common and simple method
- it is the basis for all other methods
- The name k-means, or c-means, is the abbreviation for a constant number of clusters which is predefined.

- The algorithm optimizes the following criterion function:

$$I(\underline{v}) = \sum_{i=1}^{c} \sum_{k \in \mathcal{S}_i} ||\underline{x}(k) - \underline{v}_i||^2 \longrightarrow \min_{\underline{v}_i} , \tag{23}$$

K-means clustering algorithm

- index k defines the elements of cluster $\mathcal{S}_i$
- c is the number of clusters
- $\underline{v}_i$ is the center or prototype of the cluster
- the set $\mathcal{S}_i$ contains all indexes of those patterns (among all N) belonging to cluster i, i.e. closest to the center of the cluster $\underline{v}_i$. The centers of the clusters are optimization parameters altered by the algorithm to minimize criterion function in equation 23. The criterion function contains the sum of all squared deviations between the centers of the clusters and the samples that belong to that cluster. That is why we can write it down as

$$I(\underline{v}) = \sum_{i=1}^{c} \sum_{k=1}^{N} \mu_{ik} ||\underline{x}(k) - \underline{v}_i||^2 \,, \tag{24}$$

where $\mu_{ik} = 1$ if the sample $\underline{x}(k)$ is an element of cluster i and $\mu_{ik} = 0$ if there is no element.

Algorithm: Pseudo-code for method with k-means

1. Initialization of the number of clusters c and the cluster centers $\underline{v}_i$, $i = 1, \ldots, c$. This can be done by random selection of c samples.
2. **repeat**
3. Placing samples in the clusters with the closest center.
4. Calculation of the cluster centers:

$$\underline{v}_i = \frac{\sum\limits_{k \in \mathcal{S}_i} \underline{x}(k)}{N_i}, \tag{25}$$

   where i is the index of samples belonging to i-th cluster, i.e. they are the members of $\mathcal{S}_i$. N_i is the number of elements in $\mathcal{S}_i$ ($\sum_{i=1}^{c} N_i = N$)

5. **until** $\|\underline{v}_{new} - \underline{v}\| < min_{impro}$

K-means clustering algorithm

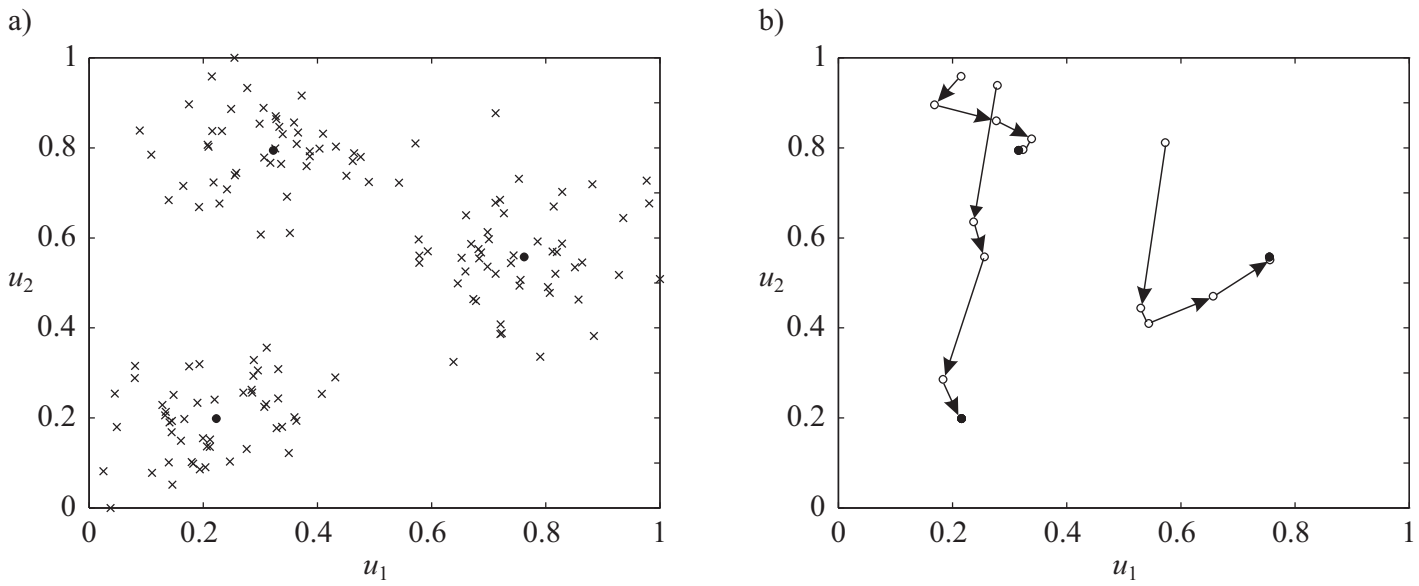


Figure 6: a) K-means clustering b) converges after 5 iterations. The black circles represent the final positions of the clustering centers.

K-means clustering algorithm

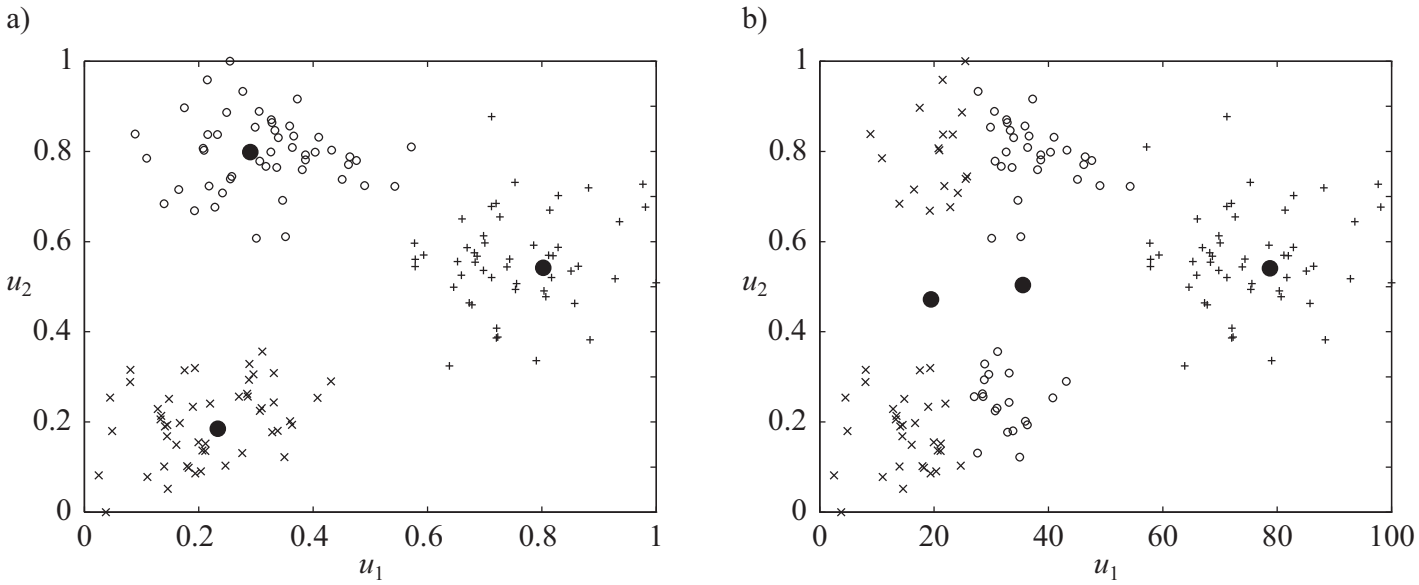


Figure 7: a) Comparison between normalised and b) non-normalised data in the case of k-means clustering. Although the cluster centers are (reasonably) appropriately distributed, figure (b) shows that samples are assigned to the clusters solely according to the deviations in dimension u_1, since the values in this dimension dominate in the distance norm

K-means clustering algorithm

An alternative to data normalization is the use of other measures of similarity that we use in equation 24. Quadratic Euclidean norm

$$d_{ik}^2 = ||\underline{x}(k) - \underline{v}_i||^2 = (\underline{x}(k) - \underline{v}_i)^{\mathrm{T}}(\underline{x}(i) - \underline{v}_i) \tag{26}$$

can be expanded into the general Mahalanobis norm

$$d_{ik,\underline{\Sigma}}^2 = ||\underline{x}(k) - \underline{v}_i||_{\underline{\Sigma}}^2 = (\underline{x}(k) - \underline{v}_i)^{\mathrm{T}}\underline{\Sigma}^{-1}(\underline{x}(k) - \underline{v}_i)\,. \tag{27}$$

The matrix norm $\underline{\Sigma}$ scales and rotates the coordinate axes. In a special case when $\underline{\Sigma} = \underline{I}$, the Mahalanobis norm is equal to the Euclidean norm. For

$$\underline{\Sigma}^{-1} = \begin{bmatrix} 1/\sigma_1^2 & 0 & \cdots & 0 \\ 0 & 1/\sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\sigma_p^2 \end{bmatrix}, \tag{28}$$

where p is the dimension of the data space, the Mahalanobis norm is equal to the Euclidean norm with normalized variables $x_l^{(\mathrm{normalized})} = x_l/\sigma_l$.

K-means clustering algorithm

Figure 8 shows the distance norms. In the case of k-means clustering, the norm is the same for the entire data space. There are methods (Gustafson-Kessel), where each cluster has its own covariance matrix and individual norm.

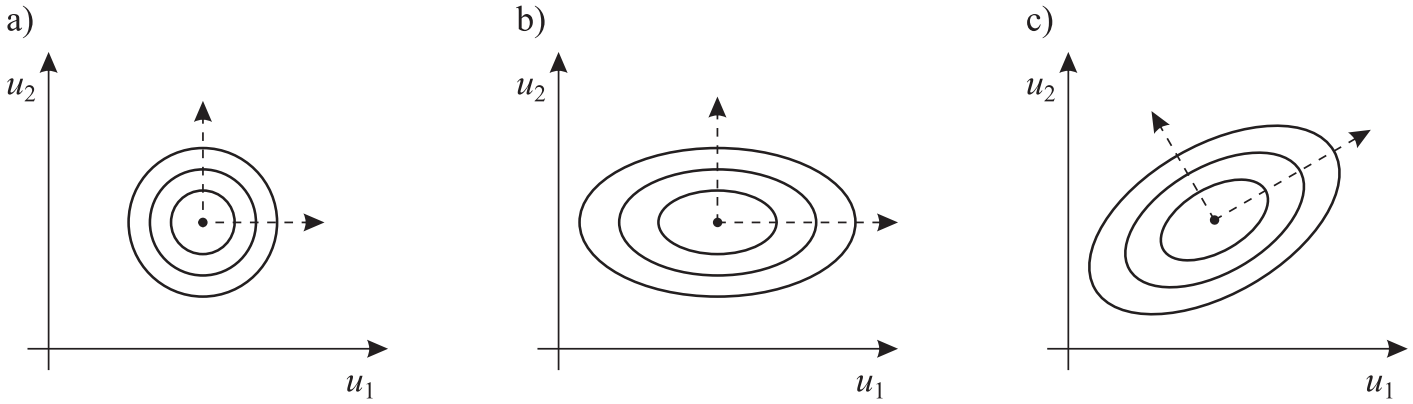


Figure 8: The lines of the same distance at different norms: a) Euclidean ($\underline{\Sigma} = \underline{I}$), b) diagonal ($\underline{\Sigma}$ = diagonal) and c) Mahalanobis norm ($\underline{\Sigma}$ = general)

The idea of fuzzy transition between clusters

- basic method of fuzzy clustering, fuzzy c-means clustering
- the data set that we obtained with measurements of m different variables, at n different time frames
- the sample of measurements is called the m-dimensional vector, which consists of measurements at individual measuring points $\underline{x}(k) = [x_1(k), \ldots, x_m(k)]^T$, $\underline{x}(k) \in \mathbb{R}^m$
- the whole data set, where we have n samples of measurements, can be written as $\underline{X} = \{\underline{x}(k) \mid k = 1, 2, \ldots, n\}$, $X \in \mathbb{R}^{n \times m}$.

$$\underline{X} = \begin{bmatrix} x_1(1) & x_2(1) & \ldots & x_m(1) \\ x_1(2) & x_2(2) & \ldots & x_m(2) \\ \vdots & \vdots & \vdots & \vdots \\ x_1(n) & x_2(n) & \ldots & x_m(n) \end{bmatrix}. \tag{29}$$

Fuzzy c-means clustering - FCM

Membership matrix has the following properties:

- membership grades are real numbers on the interval

$$\mu_{ik} \in [0,1]\,,\ 1 \leq i \leq c,\ 1 \leq k \leq n$$

- sum of all membership grades of each sample of measurements to individual fuzzy subsets is equal to one

$$\sum_{i=1}^{c} \mu_{ik} = 1,\ 1 \leq k \leq n$$

- no fuzzy set is empty and none of them contains all the data

$$0 < \sum_{k=1}^{n} \mu_{ik} < n,\ 1 \leq i \leq c$$

If matrix U satisfies all the above properties, then it belongs to a fuzzy set defined as:

$$M = \{U \in \mathbb{R}^{c \times n} \mid \mu_{ik} \in [0,1]\,, \forall i,k;\ \sum_{i=1}^{c} \mu_{ik} = 1, \forall k;\ 0 < \sum_{k=1}^{n} \mu_{ik} < n,\ \forall i\}. \tag{30}$$

The distance between the sample and the center of the cluster

- the distance is $d(\underline{x}(k), \underline{v}_i)$, where $\underline{x}(k)$ is the selected data sample and $\underline{v}_i$ is the center of the i -th fuzzy set $\underline{v}_i$, $\underline{x}(k) \in \mathbb{R}^m$.
- minimizing the criterion function (objective function)

$$I(\underline{X}, \underline{U}, \underline{V}) = \sum_{i=1}^{c} \sum_{k=1}^{n} (\mu_{ik})^{\eta} \, d^2 \left(\underline{x}(k), \underline{v}_i\right) \tag{31}$$

under the following conditions (restrictions)

$$\sum_{i=1}^{c} \mu_{ik} = 1 \quad \forall k, \tag{32}$$

- $\underline{V}$ is the matrix of vectors of the clusters' centers $\underline{v}_i$, $\underline{V} = \left[\underline{v}_1, \ldots \underline{v}_c\right]^T$
- η is a parameter that defines the fuzziness of transitions from one set to another, from completely separate sets ($\eta \to 1$) to completely fuzzy sub-sets ($\eta \to \infty$). In our processing, the default value will be $\eta = 2$. For the distance norm we select the Euclidean norm.

Fuzzy c-means clustering - FCM

Considering the limitations, the criterion function can now be written in the form of Lagrange functional, which is solved by using Picard's iteration method:

$$I(\underline{X}, \underline{U}, \underline{V}, \underline{\lambda}) = \sum_{i=1}^{c} \sum_{k=1}^{n} (\mu_{ik})^{\eta} \left(\underline{x}(k) - \underline{v}_i\right)^T \left(\underline{x}(k) - \underline{v}_i\right) + \sum_{k=1}^{n} \lambda_k \sum_{i=1}^{c} (\mu_{ik} - 1) \tag{33}$$

where $\lambda_k,\ k = 1, ..., n$ are Lagrange coefficients. The minimum of the criterion function $I(\underline{X}, \underline{U}, \underline{V}, \underline{\lambda})$ is obtained by the iterative optimization process.

Fuzzy c-means clustering - FCM

The membership values are calculated as normalized distances from the sample to the centers. The calculation is as follows:

$$\mu_{ik} = \left(d_{ik}^2 \sum_{j=1}^{c} \left(\frac{1}{d_{jk}^2} \right)^{\frac{1}{\eta-1}} \right)^{-1} \tag{34}$$

where d_{ik} is the Euclidean distance between the sample of measurements $\underline{x}(k)$ and the center of the i-th cluster $\underline{v}_i$ and is defined as follows:

$$d_{ik}^2 = \left(\underline{x}(k) - \underline{v}_i\right)^T \left(\underline{x}(k) - \underline{v}_i\right), \; 1 \leq i \leq c, \; 1 \leq k \leq n \tag{35}$$

The center of the cluster $\underline{v}_i$ is obtained as the weighted average where the weights are memberships of the sample of measurements to the i-th cluster:

$$\underline{v}_i = \frac{\sum_{k=1}^{n} \mu_{ik}^{\eta} \underline{x}(k)}{\sum_{k=1}^{n} \mu_{ik}^{\eta}}, \; i = 1, ..., c \tag{36}$$

Algorithm: pseudo-code of the fuzzy c-means clustering - FCM

1. Initialization of the number of clusters c,
2. Selection of weights η
3. Selection of tolerance threshold for termination of iteration min_{impro} and selection of the maximum number r_{max}
4. Initialization of an arithmetic matrix with random values $U^0 \in M$
5. **repeat**
6. Calculation of cluster centers: $\underline{v}_i^{(r)} = \frac{\sum_{k=1}^{n} (\mu_{ik}^{(r)})^{\eta} \underline{x}(k)}{\sum_{k=1}^{n} (\mu_{ik}^{(r)})^{\eta}}, \; i = 1, ..., c$
7. Calculation of distances: $d_{ik}^2 = \left(\underline{x}(k) - \underline{v}_i^{(r)}\right)^T \left(\underline{x}(k) - \underline{v}_i^{(r)}\right), \; i = 1, ..., c, \; k = 1, ..., n$
8. Calculation of the membership matrix: $\mu_{ik}^{(r)} = \left(d_{ik}^2 \sum_{j=1}^{c} \left(\frac{1}{d_{jk}^2} \right)^{\frac{1}{\eta - 1}} \right)^{-1}$
9. **until** $\|\underline{U}^{(r)} - \underline{U}^{(r-1)}\| < min_{impro}$ or $r >= r_{max}$

Fuzzy c-means clustering - FCM

- **Observation 1.** The algorithm can move to a singularity point if $d_{ik} = 0$ in the case of a k-th measurement and i-th cluster.
- **Observation 2.** Criteria by which we finish the algorithm may be different ($\|\underline{U}^{(r)} - \underline{U}^{(r-1)}\|_\infty < min_{impro}$, $\|\underline{V}^{(r)} - \underline{V}^{(r-1)}\|_\infty < min_{impro}$ or $\max_{i,k}\left(|\mu_{ik}^{(r)} - \mu_{ik}^{(r-1)}|\right) < min_{impro}$).
- **Observation 3.** The fuzziness weight η plays an important role in determining clusters. By selecting the limit value $\eta \to 1$ we theoretically get a classic clustering with crisp sets of elements. However, in practice, due to numerical problems, we have to select a value that is greater than one. In case when $\eta \to \infty$ we get a complete fuzzy overlapping of the space ($\mu_{ik} = \frac{1}{c},\ \forall i, k$) and the centers are equal to the mean value of an unweighted data set.
- **Observation 4.** Selecting the number of clusters c is the most important parameter. The selection can be performed by running the algorithm several times.

An example of FCM algorithm

Inner norm

- Selection of the matrix $\underline{\Sigma} = \underline{I}$, $\underline{I} \in \mathbb{R}^{m \times m}$ enables good classification of the data that have a radial shape,
- data sets that have a different shape are not optimally sorted.
- figure 9 shows a data set which has an ellipsoidal shape and is not optimally arranged
- contour diagram shows contours that connect points with memberships from 0.5 to 0.9 with step 0.1.

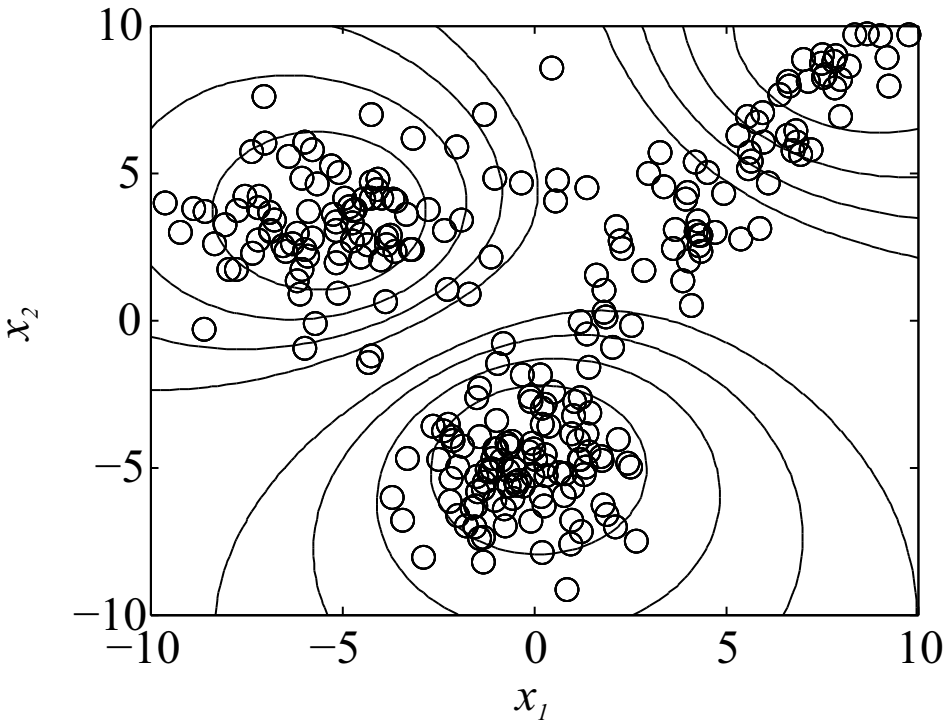


Figure 9: Example of clustering with Euclidean norm.

An example of FCM algorithm

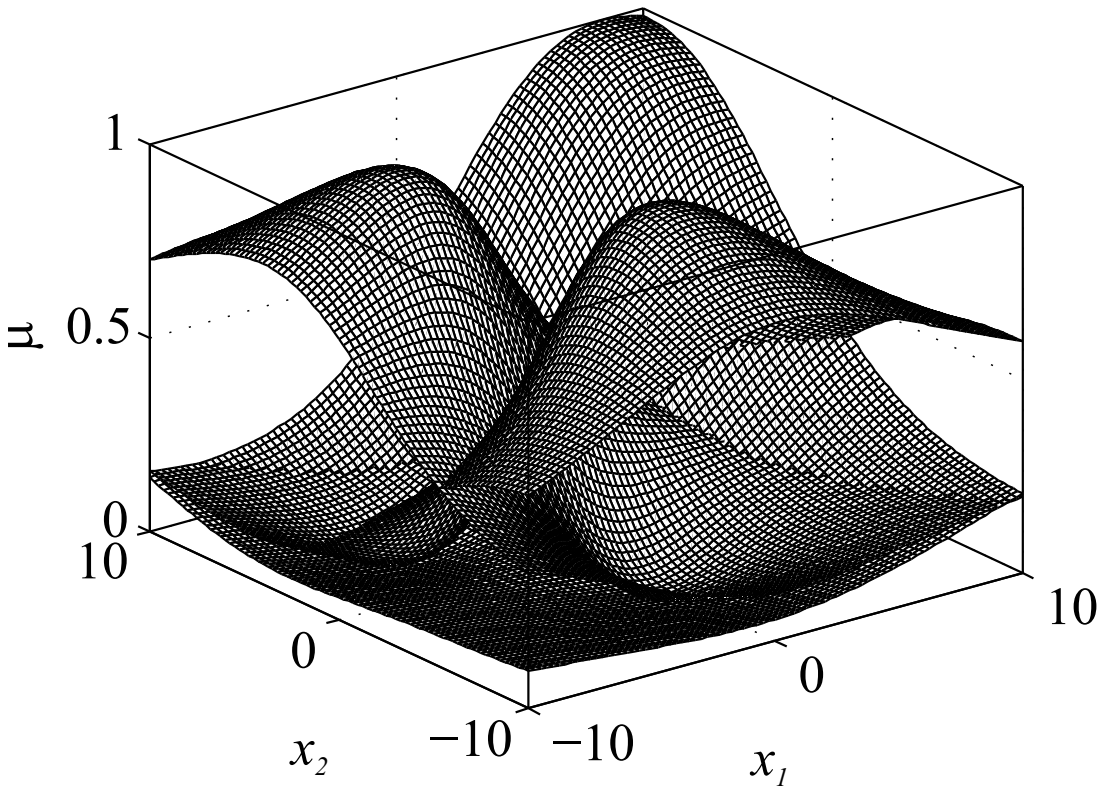


Figure 10: Membership functions in 3D space for clustering with Euclidean norm.

Example of an inner matrix norm with an inverse variance diagonal

Solving the problem of elliptic clusters

- Selecting matrix $\underline{A}^{-1}$ with diagonal elements that are inverse values of data variance in each direction

$$\underline{\Sigma}^{-1} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_m^2} \end{bmatrix} \tag{37}$$

An example of FCM algorithm

Solution to the problem of elliptic clusters with the inverse variance norm

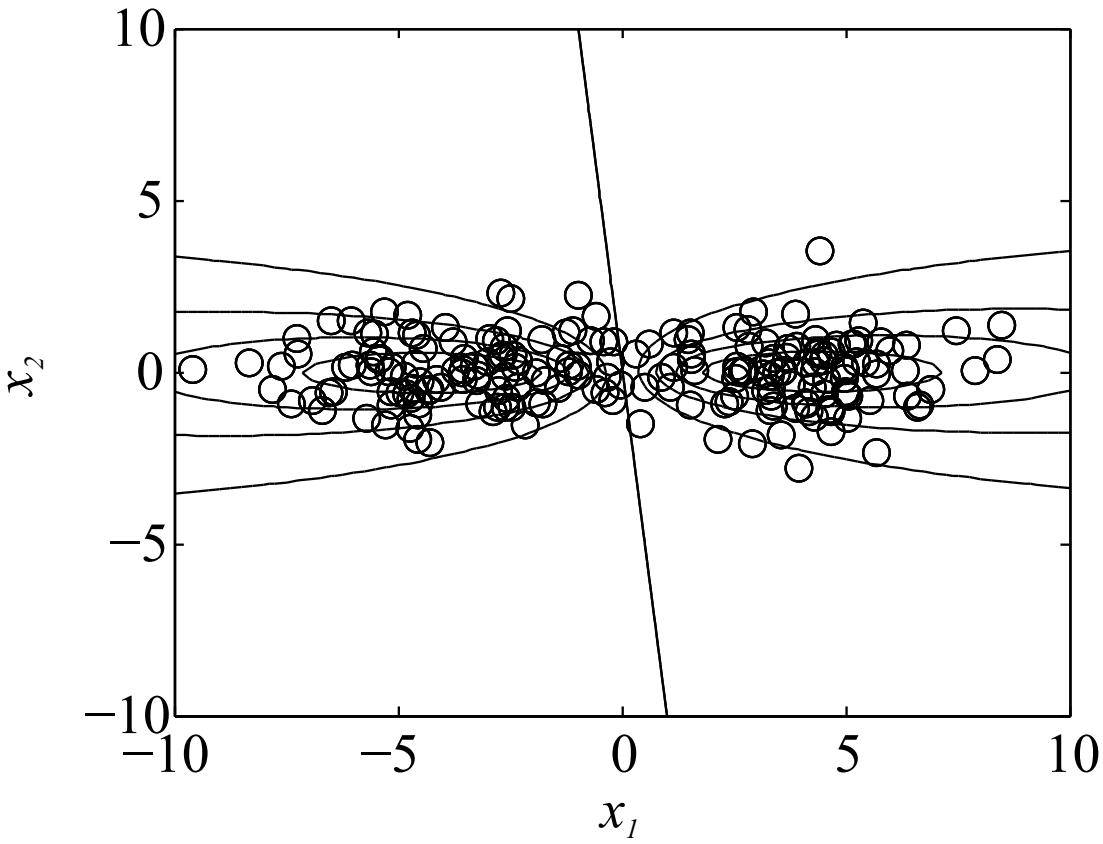


Figure 11: Example of clustering with the inverse variance norm

An example of FCM algorithm

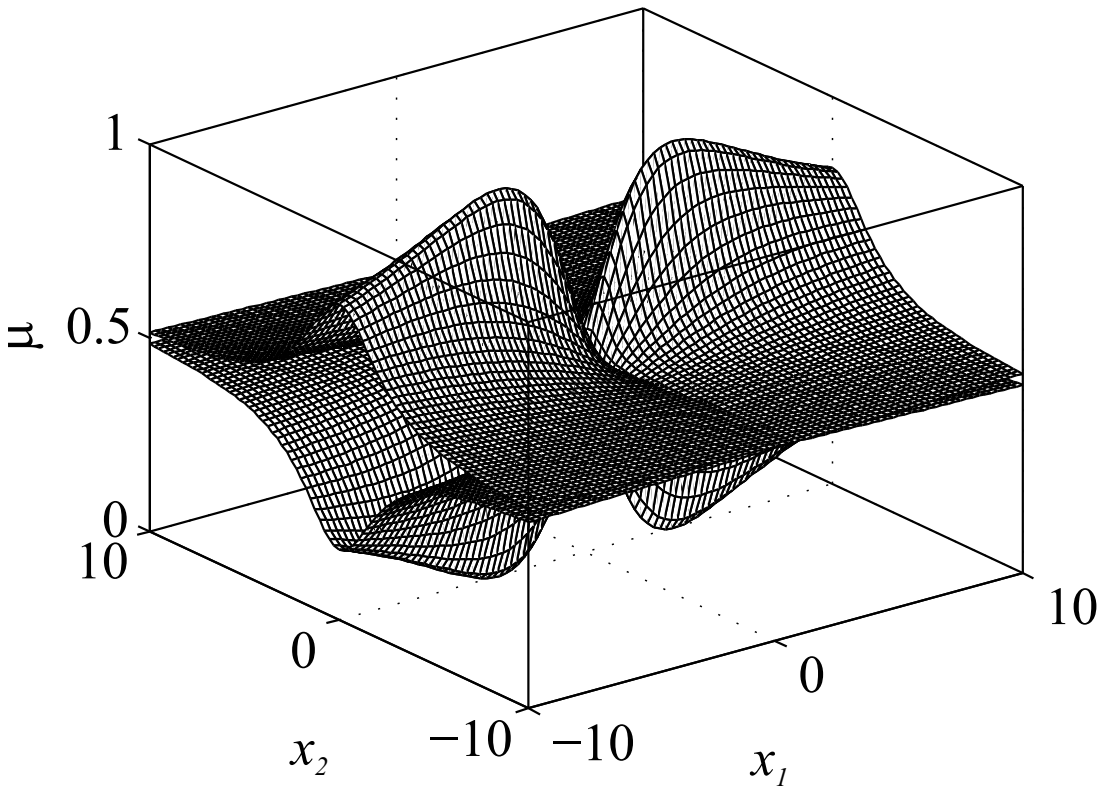


Figure 12: Membership functions in a 3D space for clustering with the inverse variance norm.

Example of an inner matrix norm with an inverse variance diagonal

Problem of different shapes of clusters

- The method shows disadvantages when there are clusters of different shapes among data, which we can see in picture 13.

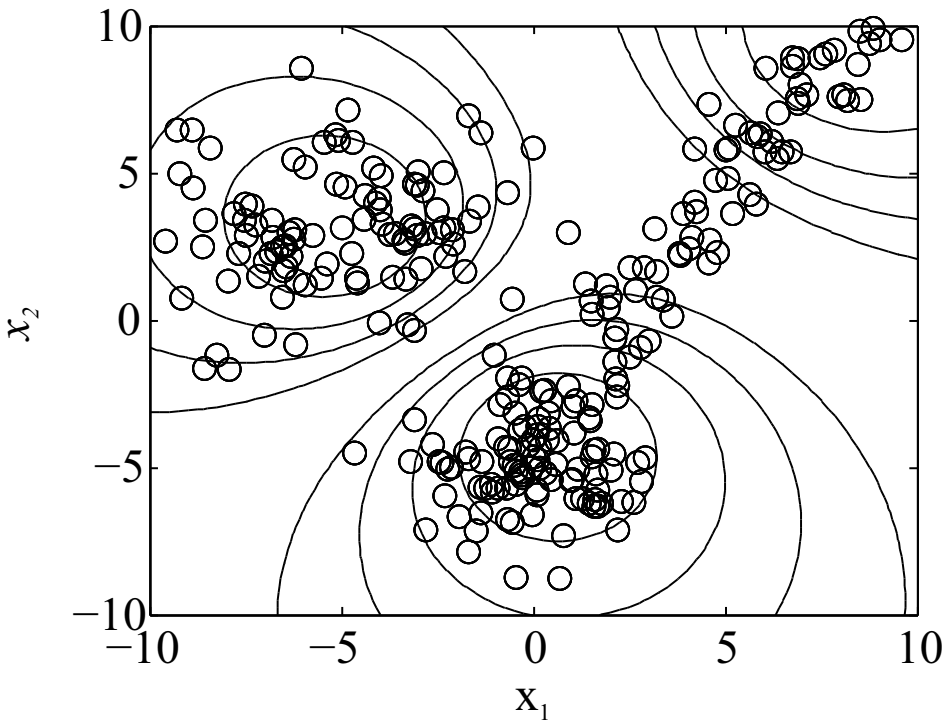


Figure 13: Example of clustering with inverse variance norm

Gustafson-Kessel clustering method - GK

- the disadvantage of the algorithm with a constant matrix of the inner product $\underline{\Sigma}$ is in its constant value, which disables the detection of different shapes of clusters in the same data set.
- the advantage is in fast convergence and robustness of the algorithm.
- The Gustafson-Kessel method introduces a variable matrix of the inner product $\underline{\Sigma}_i,\ i = 1, ..., c$.
- in the GK method each cluster has its own matrix of the inner product:

$$d_{ik}^2 = \left(\underline{x}(k) - \underline{v}_i^{(r)}\right)^T \underline{\Sigma}_i^{-1} \left(\underline{x}(k) - \underline{v}_i^{(r)}\right),\ 1 \leq i \leq c,\ 1 \leq k \leq n \tag{38}$$

Gustafson-Kessel clustering method - GK

Normalized fuzzy covariance matrix

- matrix of the inner product in calculating the distances d_{ik}.
- the fuzzy covariance matrix of the cluster i is defined as

$$\underline{\Sigma}_i = \frac{\sum_{k=1}^{n} \mu_{ik}^{\eta} \left(\underline{x}(k) - \underline{v}_i\right) \left(\underline{x}(k) - \underline{v}_i\right)^T}{\sum_{k=1}^{n} \mu_{ik}^{\eta}}, \; i = 1, \ldots, c \tag{39}$$

Gustafson-Kessel clustering method - GK

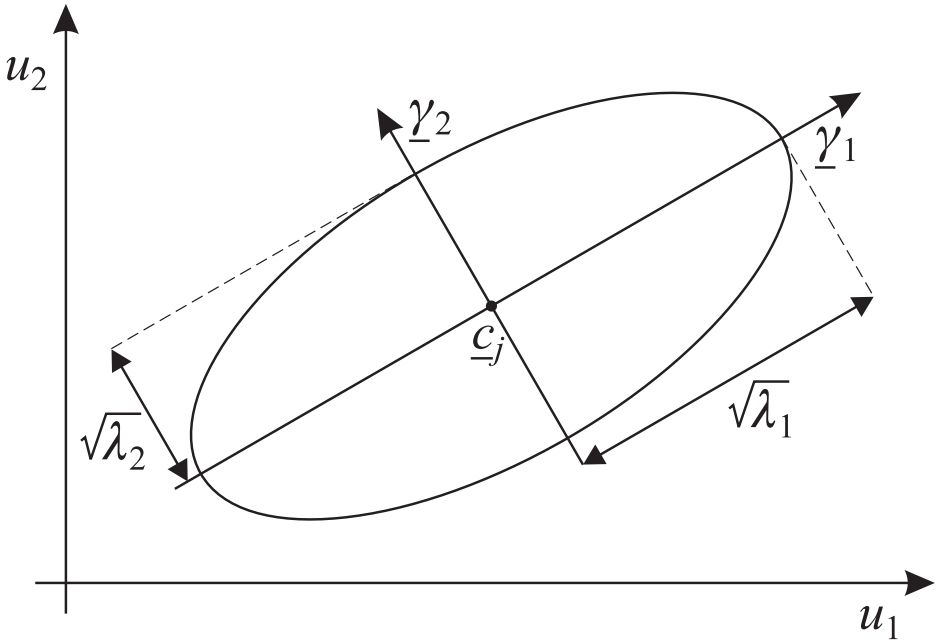


Figure 14: Interpretation of eigenvalues λ_l and eigenvectors $\underline{\gamma}_l$, $l = 1, \ldots, p$ of fuzzy covariance matrix $\underline{F}_j$. The ellipse is a contour diagram of the function $(\underline{u}(i) - \underline{c}_j)\underline{\Sigma}_j^{-1}(\underline{u}(i) - \underline{c}_j)^{\mathrm{T}} = 1$ at $\underline{\Sigma}_j$. The area is determined by the eigenvalues (hypervolume for higher dimensions), and the orientation is changed by the eigenvectors of matrix $\underline{\Sigma}_j$, which is calculated by the Gustafson-Kessel clustering algorithm

Algorithm: Pseudo-code of Gustafson-Kessel fuzzy clustering method

1. Initialization of the number of clusters c,
2. Selection of the weights η
3. Selection of the tolerance threshold to complete the iteration min_{impro} and selection of the maximum number r_{max}
4. Initialization of membership matrix with random values $U^0 \in M$
5. Selection of the initial value of the factor $\rho_i^0 = 1,\ i = 1, \ldots, c$
6. **repeat**
7. Calculation of the cluster centers: $\underline{v}_i^{(r)} = \frac{\sum_{k=1}^{n} (\mu_{ik}^{(r)})^{\eta} \underline{x}(k)}{\sum_{k=1}^{n} (\mu_{ik}^{(r)})^{\eta}},\ i = 1, \ldots, c$
8. Calculation of the fuzzy covariance matrix: $\underline{\Sigma}_i = \frac{\sum_{k=1}^{n} \left(\mu_{ik}^{(r-1)}\right)^{\eta} \left(\underline{x}(k) - \underline{v}_i\right)\left(\underline{x}(k) - \underline{v}_i\right)^T}{\sum_{k=1}^{n} \left(\mu_{ik}^{(r-1)}\right)^{\eta}},\ i = 1, \ldots, c$
9. Calculation of the matrix of the inner product : $\underline{F}_i = \left(\rho_i^{(r-1)} |\underline{\Sigma}_i|\right)^{\frac{1}{m}} \underline{\Sigma}_i^{-1},\ i = 1, \ldots, c$
   $\rho_i^{(r)} = \frac{1}{\sum_{k=1}^{n} \mu_{ik}},\ i = 1, \ldots, c$
10. Calculation of distances: $d_{ik}^2 = \left(\underline{x}(k) - \underline{v}_i^{(r)}\right)^T \underline{F}_i \left(\underline{x}(k) - \underline{v}_i^{(r)}\right),\ i = 1, \ldots, c,\ k = 1, \ldots, n$
11. Calculation of the membership matrix: $\mu_{ik}^{(r)} = \left(d_{ik}^2 \sum_{j=1}^{c} \left(\frac{1}{d_{jk}^2}\right)^{\frac{1}{\eta-1}}\right)^{-1}$
12. **until** $\|\underline{U}^{(r)} - \underline{U}^{(r-1)}\| < min_{impro}$ or $r >= r_{max}$

Gustafson-Kessel clustering method - GK

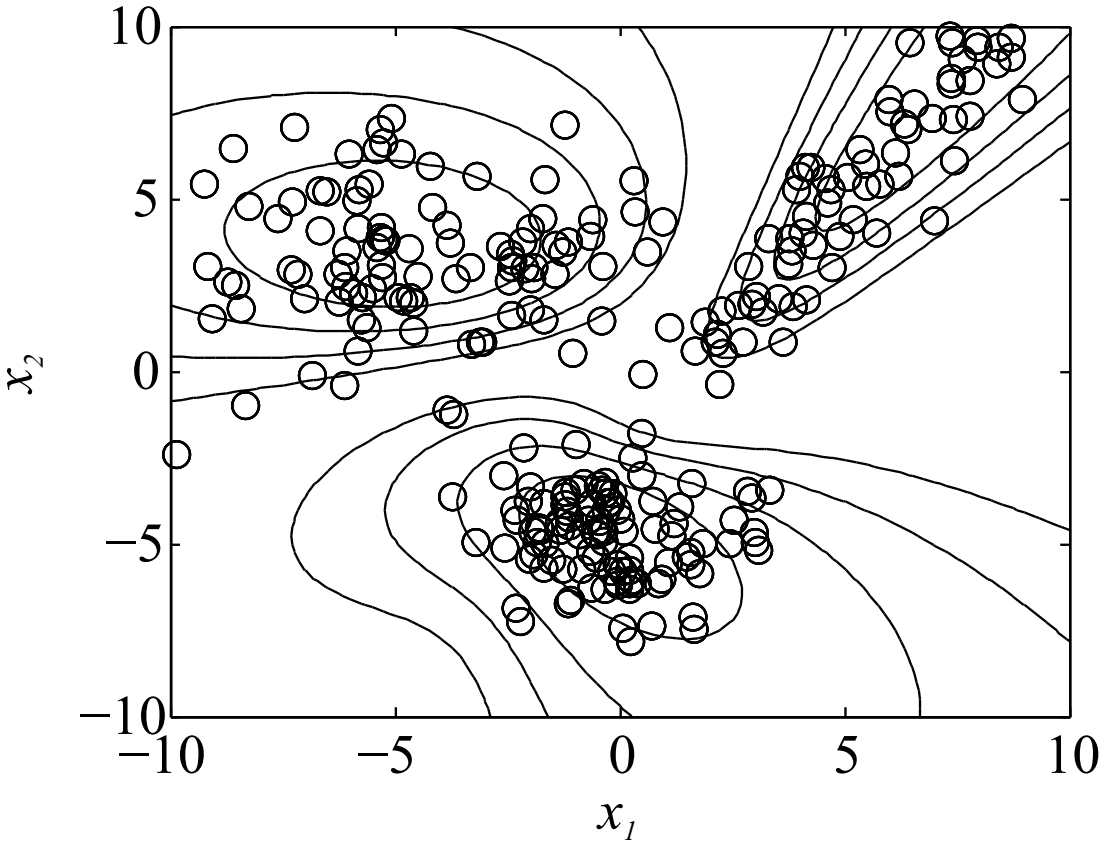


Figure 15: An example of Gustafson-Kessel fuzzy clustering method.

Gustafson-Kessel clustering method - GK

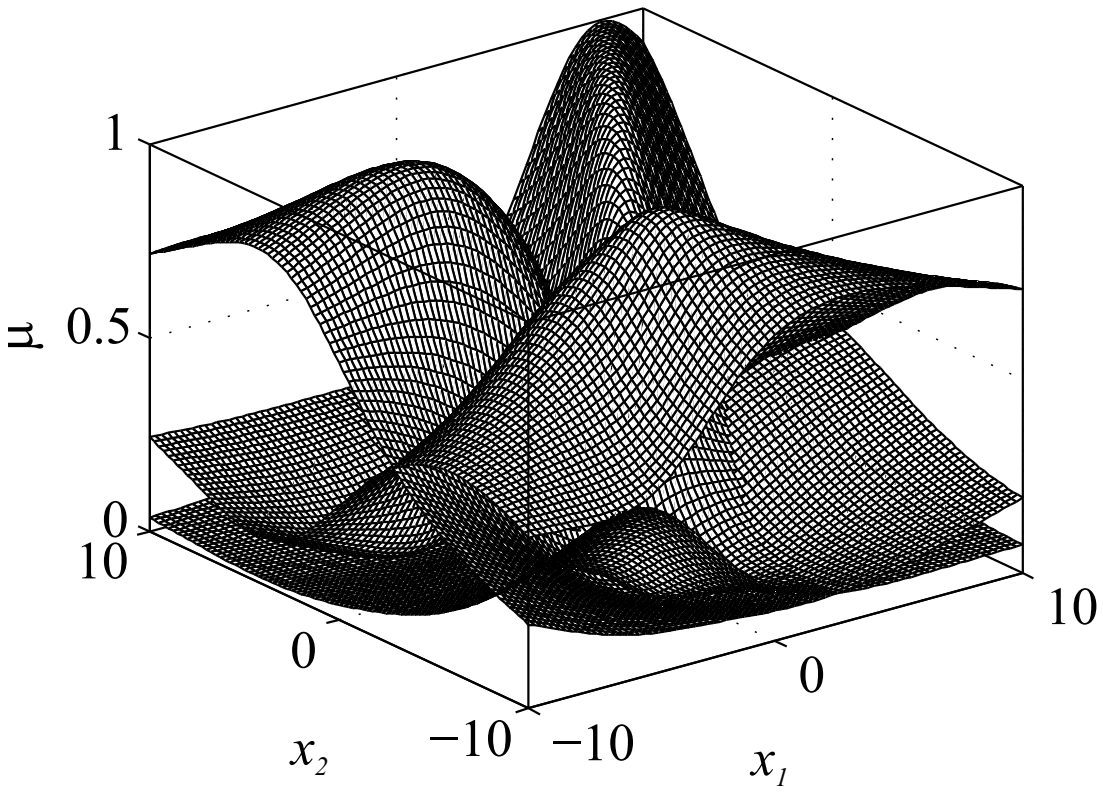


Figure 16: Membership functions in 3D space in Gustafson-Kessel clustering method.

An example of unequal cluster volumes

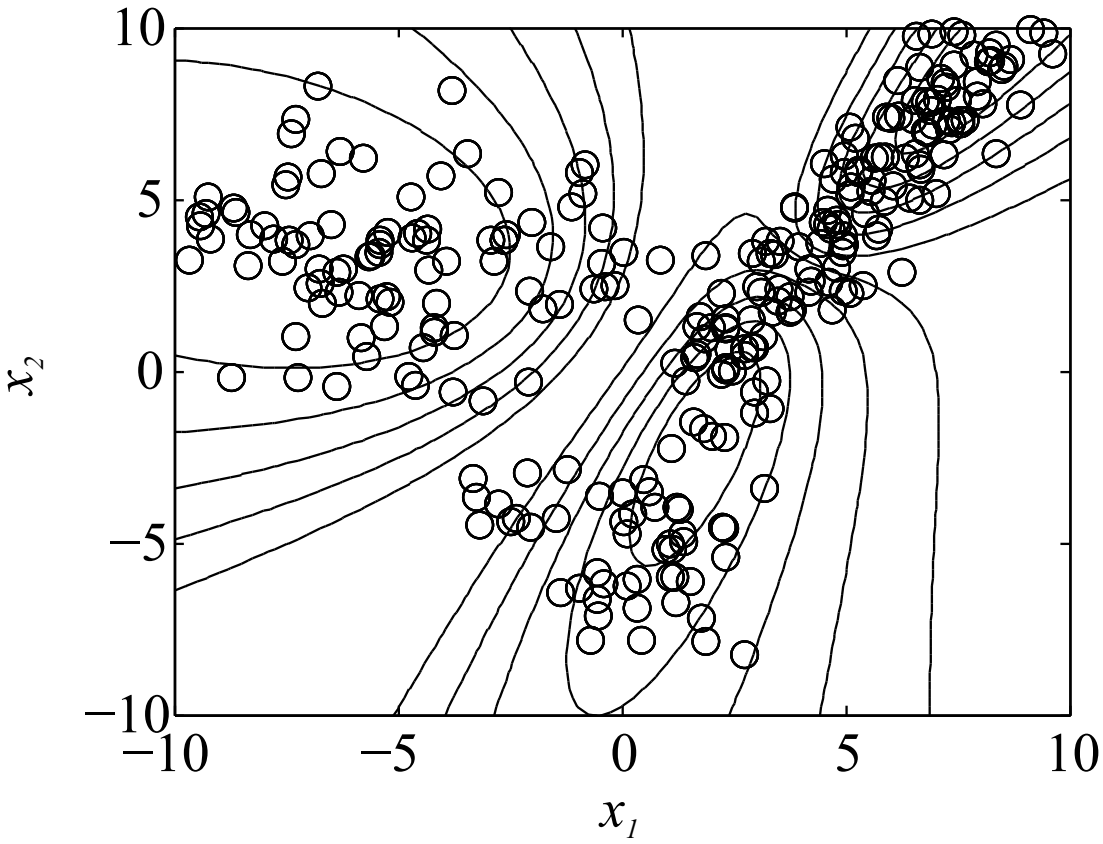


Figure 17: An example of Gustafson-Kessel fuzzy clustering **in the case of different cluster volumes**.

An example of unequal cluster volumes

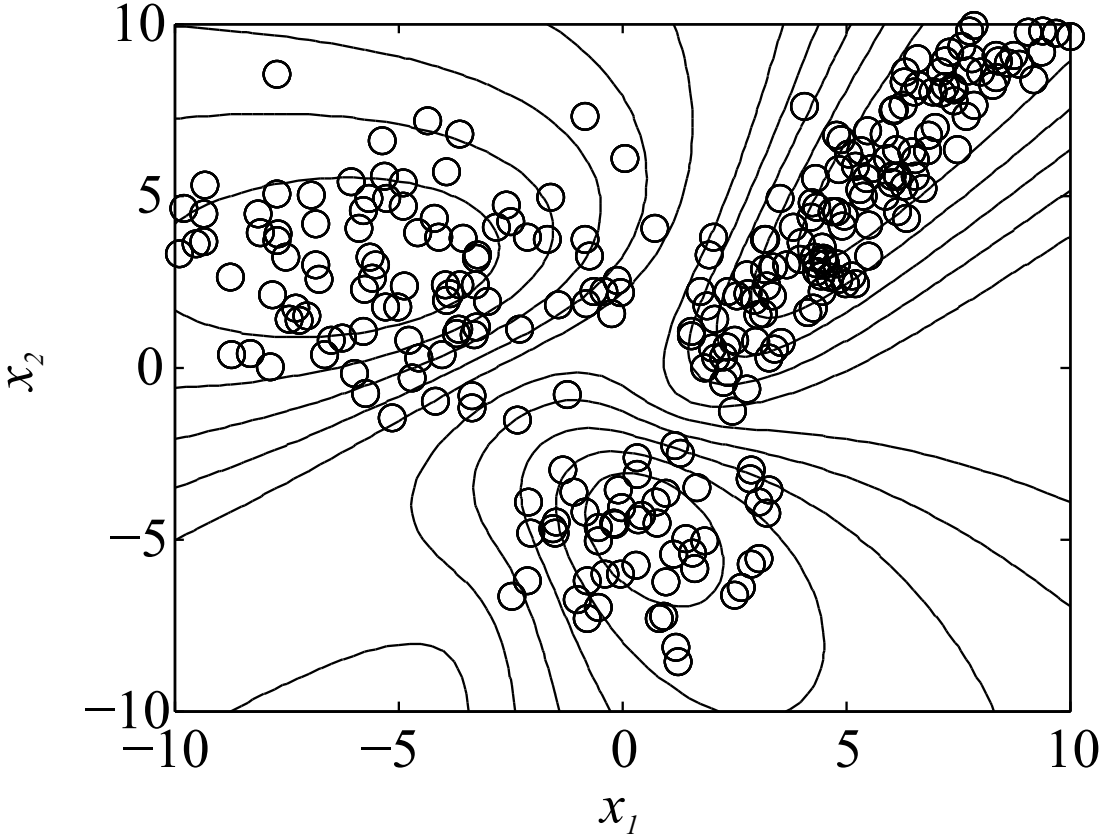


Figure 18: An example of a modified Gustafson-Kessel fuzzy clustering (normalization with the volume of the clusters) **in the case of different cluster volumes**.

Norm of the maximum likelihood

- calculation of distances using maximum likelihood function

$$d_{ik}^2 = \rho_i |\underline{\Sigma}_i|^{\frac{1}{2}} e^{\frac{(\underline{x}(k)-\underline{v}_i)^T \underline{\Sigma}_i^{-1} (\underline{x}(k)-\underline{v}_i)}{2}}, \; i = 1, ..., c, \; k = 1, ..., n \tag{40}$$

An example of maximum likelihood clustering

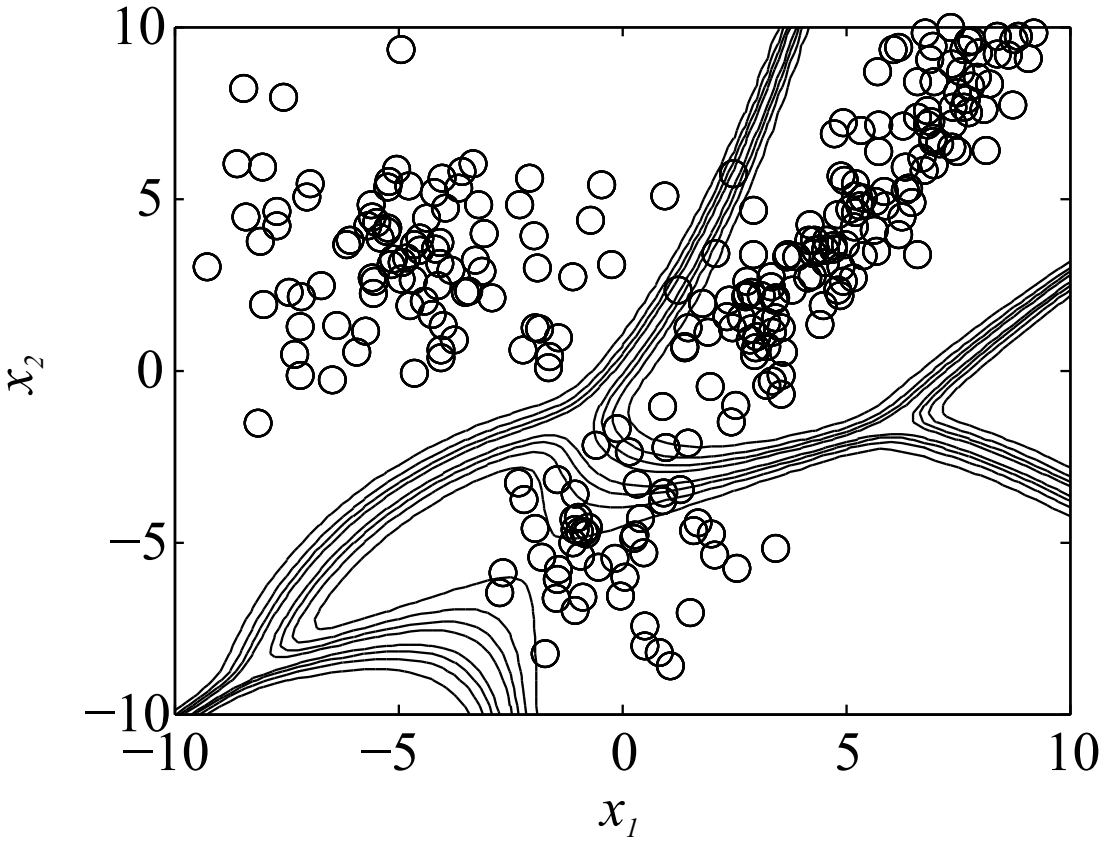


Figure 19: An example of maximum likelihood clustering **in the case of clusters of different shapes and volumes**.

An example of maximum likelihood clustering

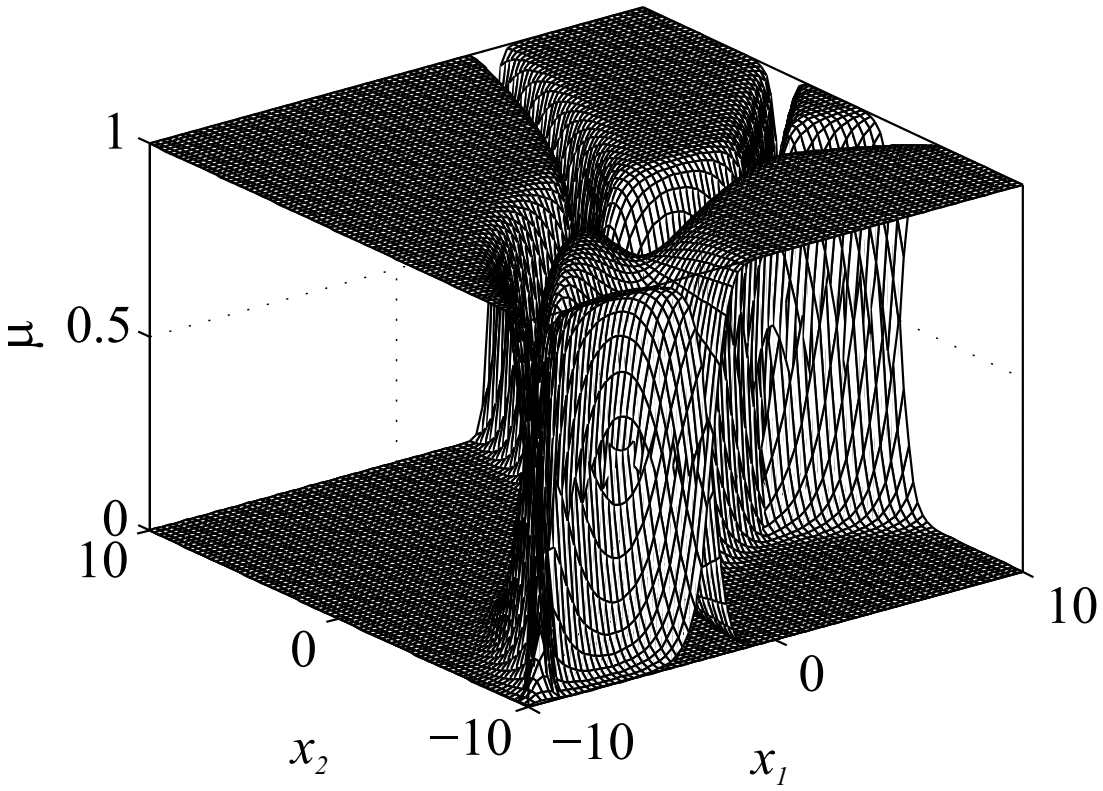


Figure 20: An example of maximum likelihood clustering **in the case of clusters of different shapes and volumes** in 3D.

An example of individual remote data

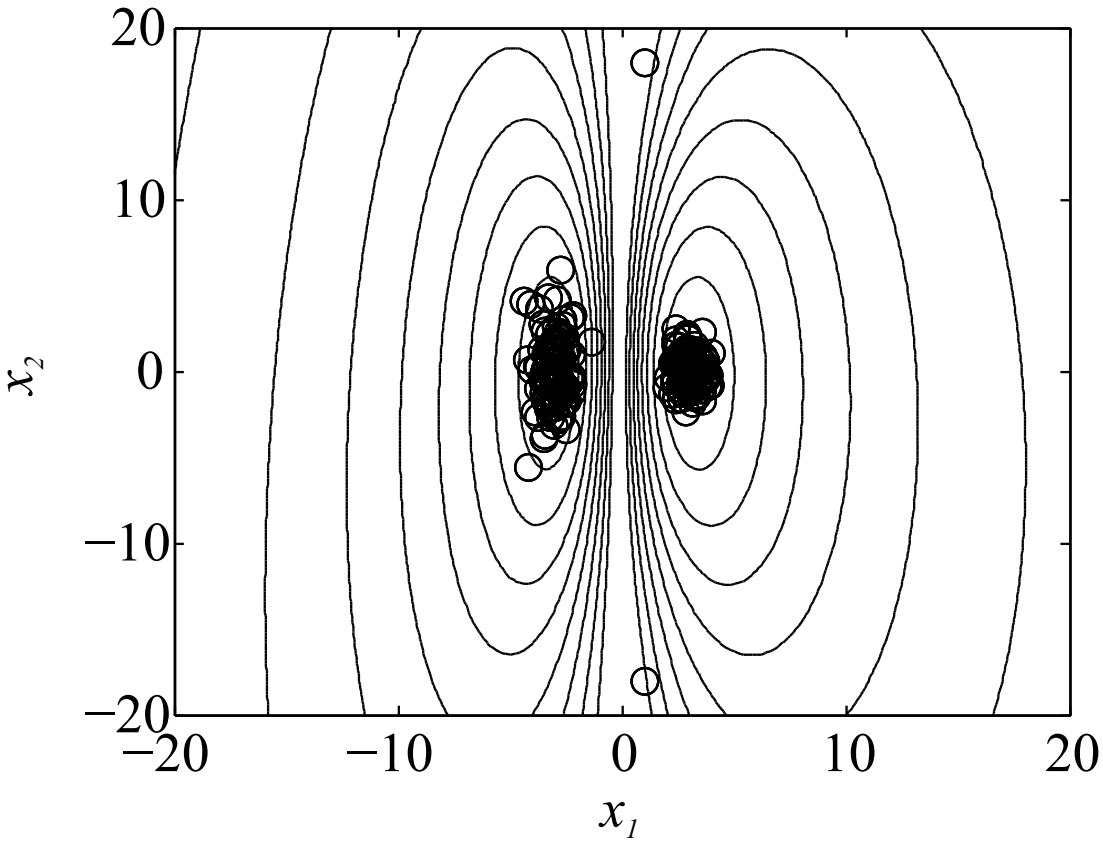


Figure 21: An example of Gustafson-Kessel fuzzy clustering in the case of individual distant data.

An example of using PCM method

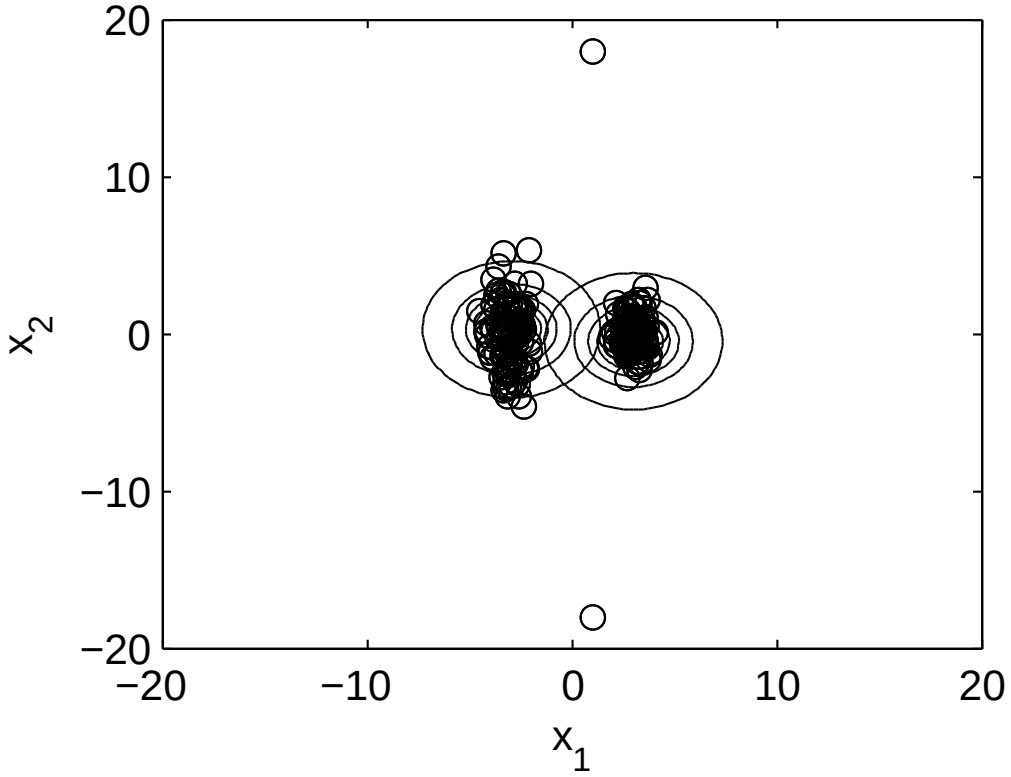


Figure 22: Fuzzy clustering according to PCM method in the case of individual distant data.

An example of using PCM method

PCM - the idea of an absolute membership degree instead of a relative degree

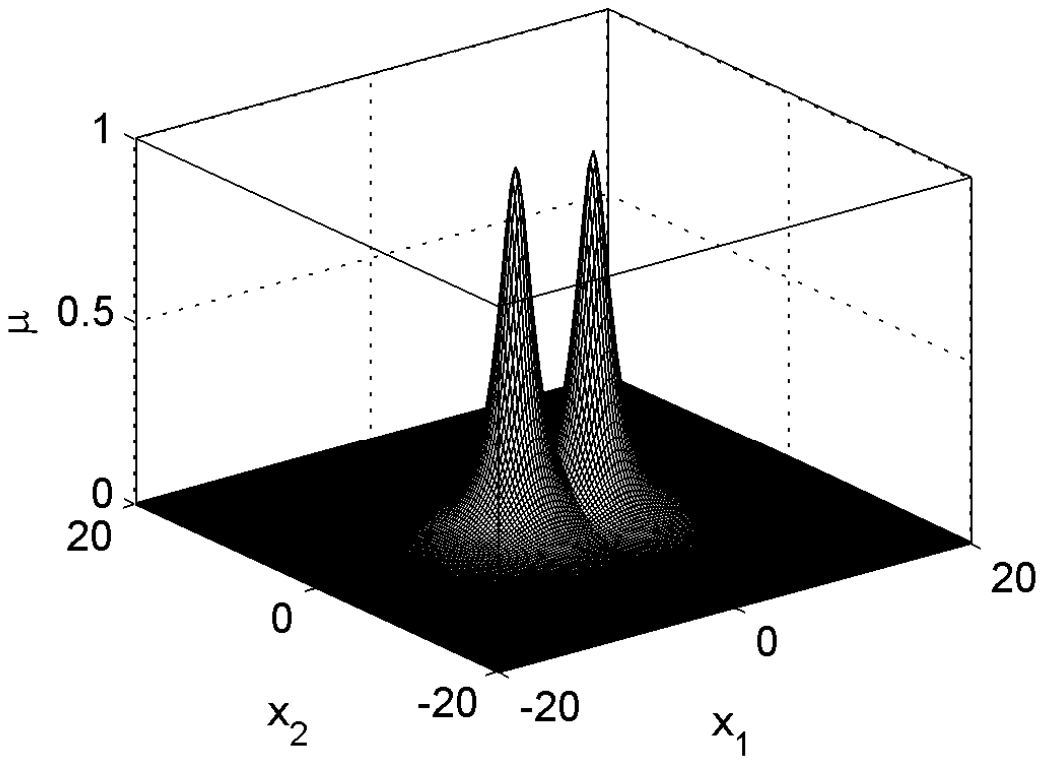


Figure 23: Membership functions in 3D space for fuzzy clustering according to PCM method in the case of individual distant data.

Possibilistic Clustring Cluster Method

1. Defining the maximum possible number of clusters c_{max}, the fuzzy parameter η, the criterion for ending the algorithm ϵ, the cluster shape parameter δ
2. Initializing the typicality for c_{max} clusters using FCM.
3. **repeat**
4. Calculating cluster centers

$$\underline{v}_i = \frac{\sum_{k=1}^{n} \mu_{ik}^{\eta} \underline{x}(k)}{\sum_{k=1}^{n} \mu_{ik}^{\eta}}, \quad i = 1, \ldots, c_{max}$$

5. Calculating the distances

$$d_{ik}^2 = (\underline{x}(k) - \underline{v}_i)^T (\underline{x}(k) - \underline{v}_i), \; i = 1, \ldots, c_{max}, \; k = 1, \ldots, n.$$

6. Calculating typicality

$$\mu_{ik} = \left(1 + \left(\frac{(\underline{x}(k) - \underline{v}_i)^T (\underline{x}(k) - \underline{v}_i)}{\delta} \right)^{\frac{1}{\eta - 1}} \right)^{-1} \tag{41}$$

7. **until** $\|\Delta U\|_{\infty} < \epsilon$

Optimizing the model complexity

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for the 7th lecture, April 2016

Table of Contents

1. Optimizing the model complexity
2. Introduction
3. Bias-variance tradeoff
   - Bias error
   - Variance error
   - Bias and variance error tradeoff
4. Model quality estimation
   - Model learning, verification and validation
   - Cross-validation

Optimizing the model complexity

Everything should be as simple as possible,
but not simpler. — Albert Einstein

Introduction

- Definition of the model complexity in terms of the number of model parameters.
- Parameters must be estimated if we want to identify the model.
- Model *complexity* also expresses its *flexibility*, which is used as a synonym.
- The model should not be too simple if we want to describe the behaviour of the process well.
- On the other hand, it should not be too complex as we would then have to evaluate a larger number of parameters.
- We also need more data to evaluate a larger number of parameters.
- There is a certain compromise, called the *optimal model complexity*.

Bias-variance tradeoff

The effect of the number of parameters on model quality

- The model's error can be divided into two parts: *bias error* and *variance error*.
- The scheme in figure 1 represents the process with the output y_u, which is disturbed by the noise n. The measured output of the process is labeled with y. The model output of the process is written with $\hat{y}$. The model was obtained on the basis of optimization (minimization) of the criterion function, which is the function of the error e, and we optimized the parameters of the model.

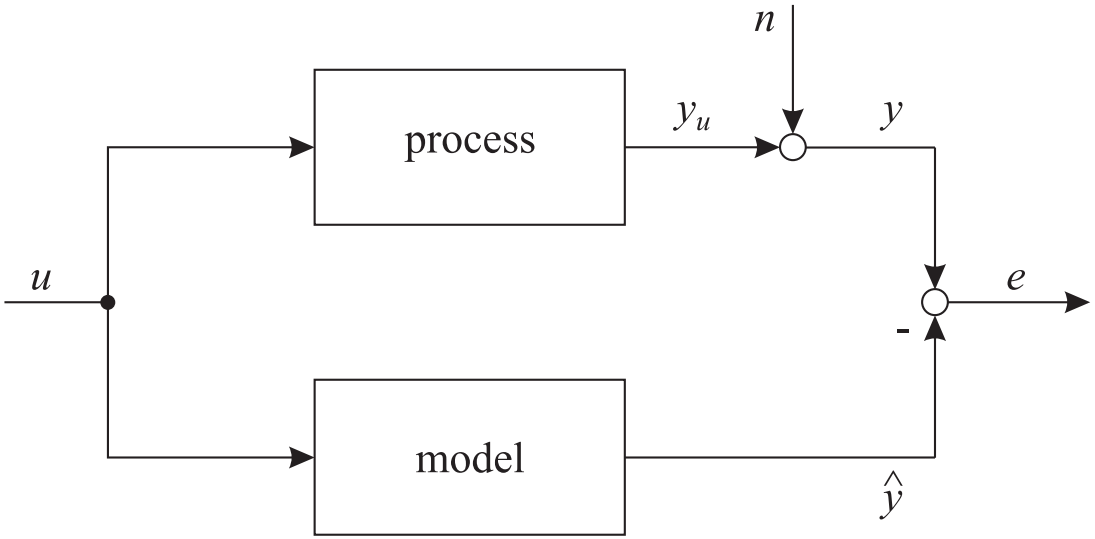


Figure 1: Process and model. The error between the output of the process and the model, e can be partitioned into one part due to noise and one part due to bias and variance.

Bias-variance tradeoff

- Since we have random variables, for the criterion function we can propose the expected value (*expectation* - E) of the squared error between the output of the process and the model.
- This is analogous to the sum of squared errors used in deterministic signals.
- The criterion function is divided into two parts:

$$\mathrm{E}\{e^2\} = \mathrm{E}\{(y-\hat{y})^2\} = \mathrm{E}\{(y_u + n - \hat{y})^2\} = \mathrm{E}\{(y_u - \hat{y})^2\} + \mathrm{E}\{n^2\}, \tag{1}$$

  where the cross term $\mathrm{E}\{(y_u - \hat{y}\cdot)\, n\}$ disappears because the noise is not correlated with the useful output of the process, nor with the output of the model.
- The first term on the right (1) represents an error between the true (immeasurable) output of the process and the output of the model.
- The second term represents a noise variance.
- Since we can not influence the noise variance with the model, only the first part is considered below. The error of the model $y_u - \hat{y}$ can be further analysed as:

$$\begin{aligned}
\mathrm{E}\{(y_u - \hat{y})^2\} &= \mathrm{E}\{(y_u - \mathrm{E}\{\hat{y}\} - (\hat{y} - \mathrm{E}\{\hat{y}\}))^2\} \\
&= \mathrm{E}\{(y_u - \mathrm{E}\{\hat{y}\})^2\} - 2\mathrm{E}\{y_u - \mathrm{E}\{\hat{y}\}\}\mathrm{E}\{\hat{y} - \mathrm{E}\{\hat{y}\}\} + \mathrm{E}\{(\hat{y} - \mathrm{E}\{\hat{y}\})^2\} \\
&= (y_u - \mathrm{E}\{\hat{y}\})^2 + \mathrm{E}\{(\hat{y} - \mathrm{E}\{\hat{y}\})^2\}
\end{aligned} \tag{2}$$

  because the expected value is $\mathrm{E}\{\hat{y} - \mathrm{E}\{\hat{y}\}\} = 0$, all the cross terms disappear and we get

$$\mathrm{E}\{(y_u - \hat{y})^2\} = (y_u - \mathrm{E}\{\hat{y}\})^2 + \mathrm{E}\{(\hat{y} - \mathrm{E}\{\hat{y}\})^2\} \tag{3}$$

- Finally, we can write down:

$$(\text{error of the model})^2 = (\text{bias error})^2 + \text{variance of the output}\,. \tag{4}$$

Bias-variance tradeoff

- The number of model parameters has a decisive influence on the above values.
- Bias and variance of the output are in conflict.
- For the optimal complexity choices, we must choose the bias and the variance of the error tradeoff.
- The difference between two data sets: *learning set* and *test set*.
- A learning set is used for learning the model, i.e. for evaluating (estimation) or optimizing the parameters. The test set is used for evaluating the quality of the model.
- Usually we use two different sets of data for this purpose.
- It is important that the model yields good results on a new set of data, on a test set (validation).

Bias error

- A bias error is part of a model error due to a limited description of the model (flexibility).
- The bias error can be reduced if a suitable structure is selected for the model.
- If, however, we are dealing with a nonlinear process, then we will in any case get a bias error because no approximation model can describe any nonlinear process with complete accuracy.
- In the case of nonlinear processes, the bias error can not be eliminated, since all the structures with which nonlinear systems are modelled are only universal approximators, because the real structures are usually not known.
- At most, we can expect that the bias becomes smaller with the increasing complexity. If we satisfy this criterion, then the approximator for a certain class of processes is a *universal approximator*.
- Bias is a systematic error of the model caused by the structure of the model. The error of the model is always greater than the bias

$$\text{bias} = y_u - \mathrm{E}\{\hat{y}\}\,, \tag{5}$$

where y_u is an immeasurable output of the process and $\hat{y}$ is the output of the model. The output of the model can be considered as a random variable due to the nature of the variables from which the model parameters were evaluated.

Bias error

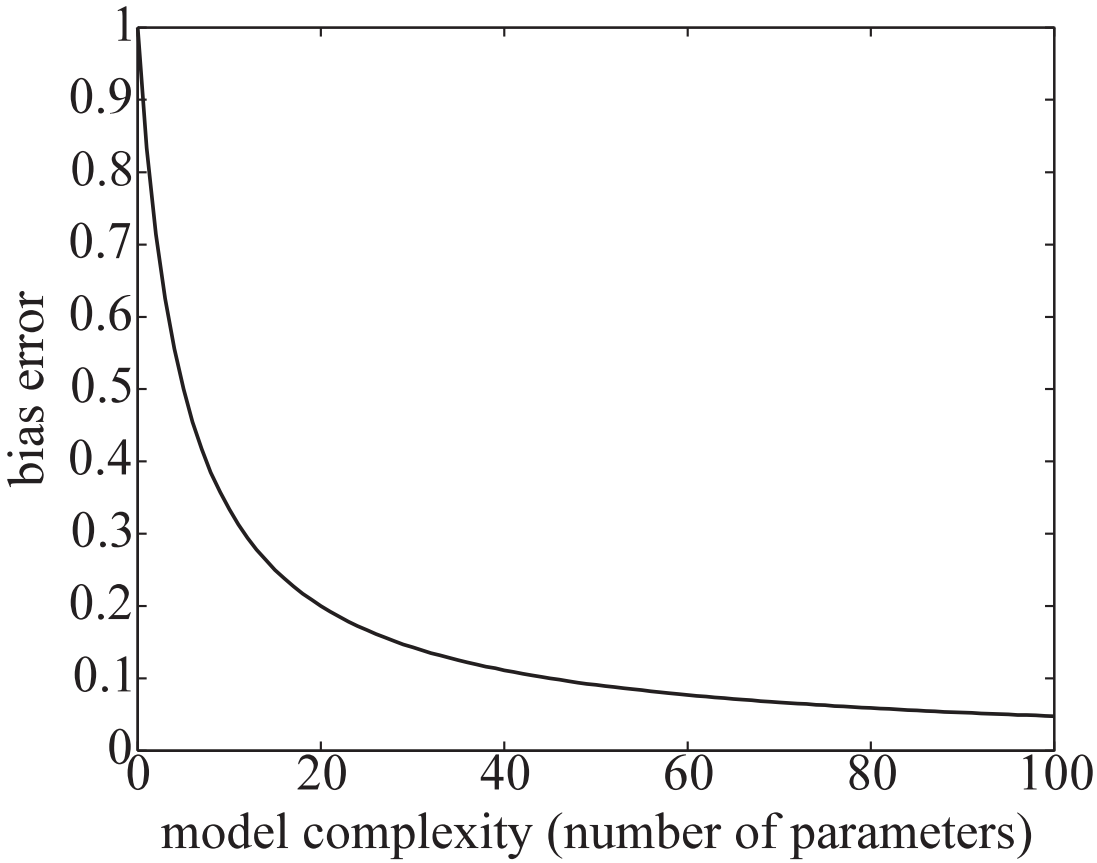


Figure 2: Typical dependency between the bias and the number of parameters of the model

Variance error

Variance error

- A variance error is an error that we get due to an error in estimating the parameters of the model. Deviation of parameters of the model from true values.
- Since in practice the parameters of the model are estimated based on a finite set of noisy data, these parameters usually differ from the true values.
- The variance error determines the size of the confidence interval.
- For an infinite set of data, the error of variance would be zero if the correct identification method were used, i.e. the estimated parameters would converge to the right ones.

Variance error

The variance error is written as

$$\text{variance error} = \mathrm{E}\{(\hat{y} - \mathrm{E}\{\hat{y}\})^2\}\,. \tag{6}$$

- Let's assume that we have the same signal several times on the input of the process.
- Depending on the nature of the noise n, we will get different output responses. If, based on these different data sets, the parameters of the model are evaluated, they are different from one another.
- Equation 6 defines a mean square deviation of these outputs of models from the mean value of the output of the model.
- In the case of a noise-free process, all models would be the same and consequently the variance error would be zero.
- In the case of a large number of learning data, the variance error increases approximately linearly with the number of the parameters of the model:

$$\begin{aligned} \text{variance error} \quad &\sim \quad \sigma^2 \frac{n}{N} \\ &\sim \quad \text{variance of the noise} \cdot \frac{\text{number of the parameters of the model}}{\text{number of training data}}\,. \end{aligned} \tag{7}$$

- The claim is approximate regardless of the type of the model. Of course, it is only valid for an infinitely large number of data in the learning set. Nevertheless, (7) is a good guide to practical design.
- More attention should be paid to the treatment when the number of learning data is low or when the model is very complex.

Variance error

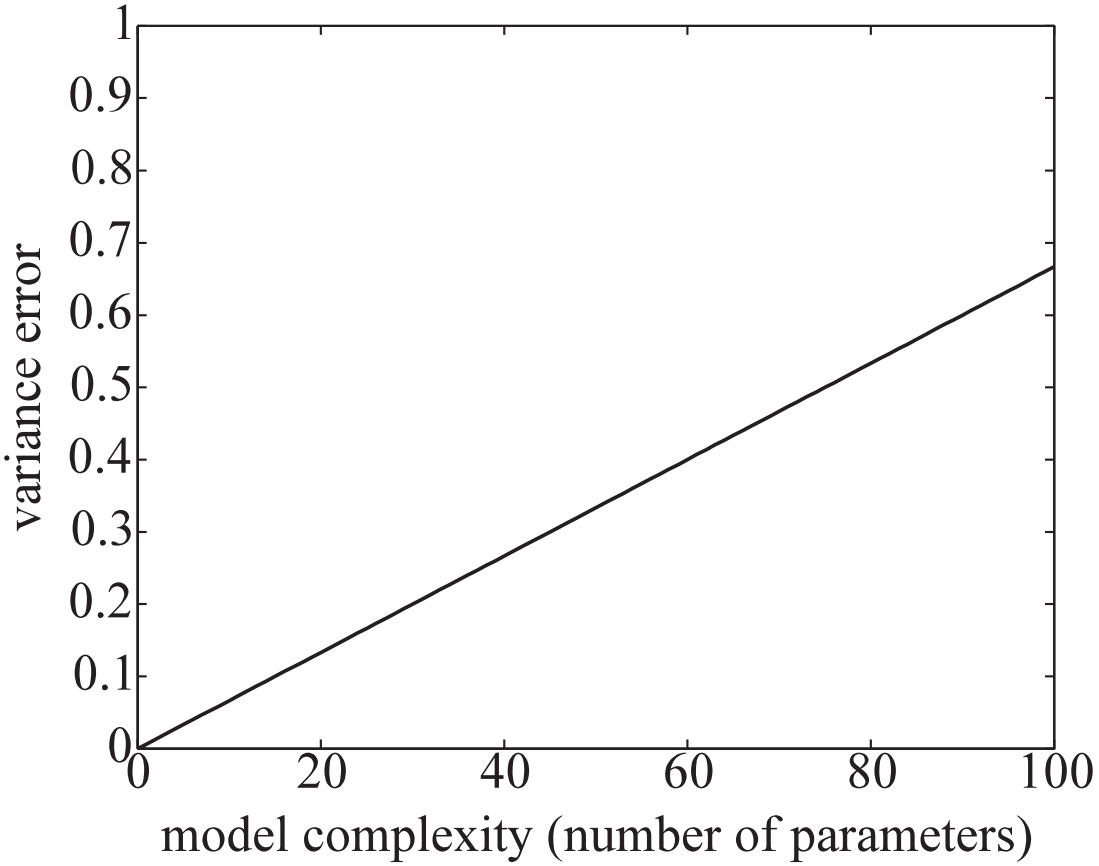


Figure 3: Typical dependency of the variance error and the number of the parameters of the model

Bias and variance error tradeoff

Influence of bias and variance error on the error of the model

- Influences in terms of model error contradict.
- The problem is called *bias-variance dilemma*.
- Figure 4b illuminates the compromise problem between the bias and the variance for the learning set, which is twice as high as in figure 4a. This leads to the variance error which is twice as low. Consequently, the optimal complexity of the model can be greater than in figure 4a.
- Intuitively we can say that a large number of learning data also allows the estimation of a greater number of parameters of the model.
- We can say that the same effect is achieved with a smaller noise variance.

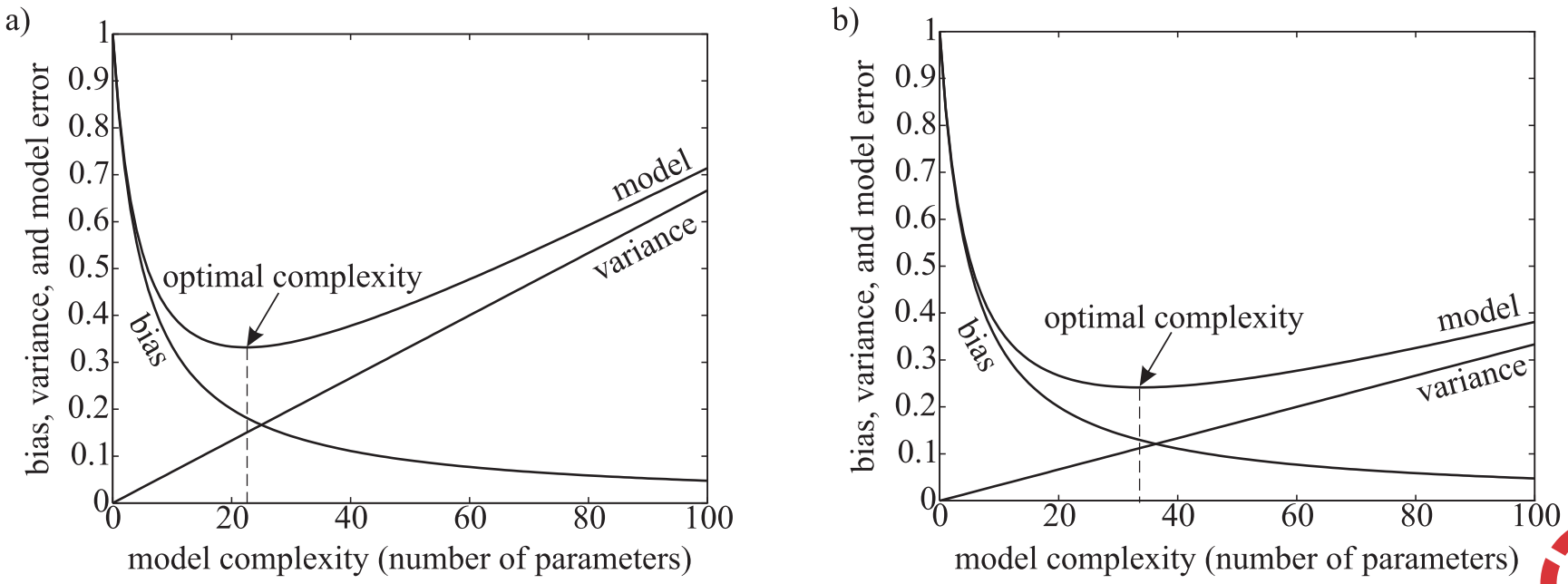


Figure 4: Bias-variance tradeoff. The error of the model consists of a bias-variance error: a) a relatively high error of variance, b) a lower error of variance due to a lesser noise variance or a larger number of learning data

Learning and test data sets

- If a learning data set was used to measure the quality of the model, the variance error would be zero. In this case, the error only depends on the bias.
- The variance error is only present if another set of data with a different noise realization is used. The reason for this is that the variance of the error is a consequence of errors of parameters of the model due to the concrete realization of the noise in the learning data.
- The error on the learning set (which is approximately equal to the bias) decreases with the complexity of the model; the error on the test set (which is equal to the bias and variance) begins to increase even when it reaches the optimal complexity point.
- If we are not aware of this effect, then we get a model that behaves well on the learning data and poorly on the test data.
- The effect is called *overfitting* (low bias and high variance) and *underfitting*, which is the result of an oversimplified structure of the model (high bias and low variance).
  Figure 5 shows the problem of underfitting and overfitting. On the left side there is a model on the learning data, and on the right side on the test data.

Bias and variance error tradeoff

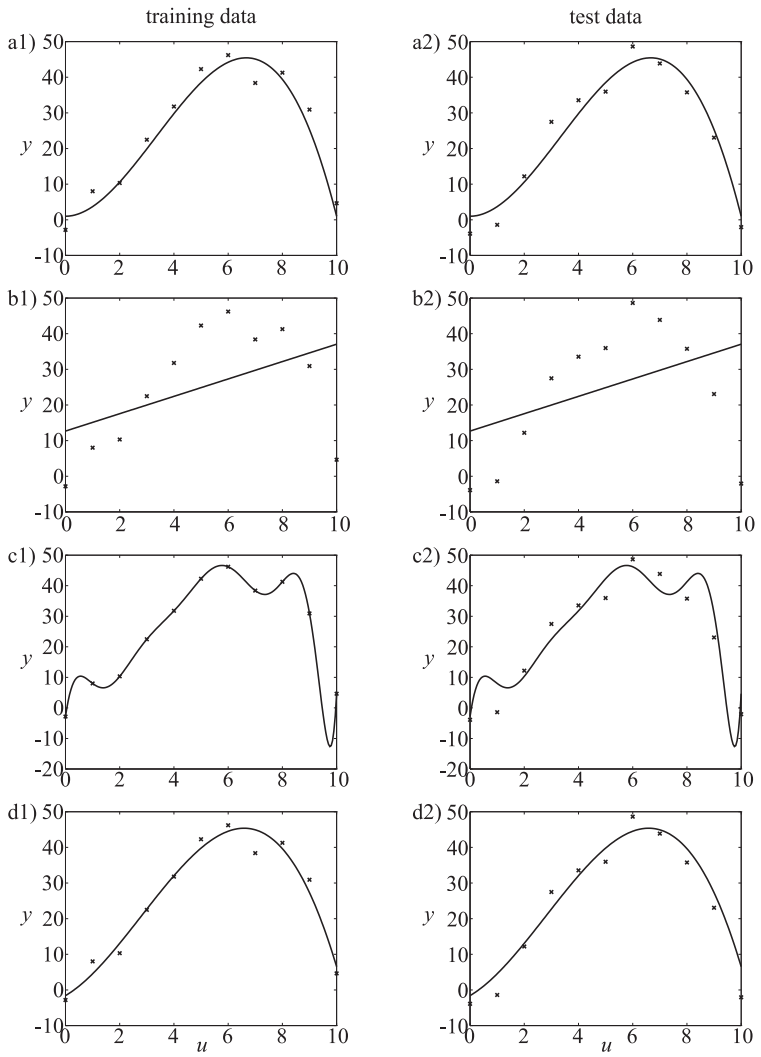


Figure 5: Underfitting and overfitting: a) original function, b) approximation with the first-order polynomial (underfitting: high bias, low variance), c) approximation with the tenth-order polynomial (overfitting zero bias: high variance), d) approximation with the fourth-order polynomial (good bias and variance, tradeoff)

Bias and variance error tradeoff

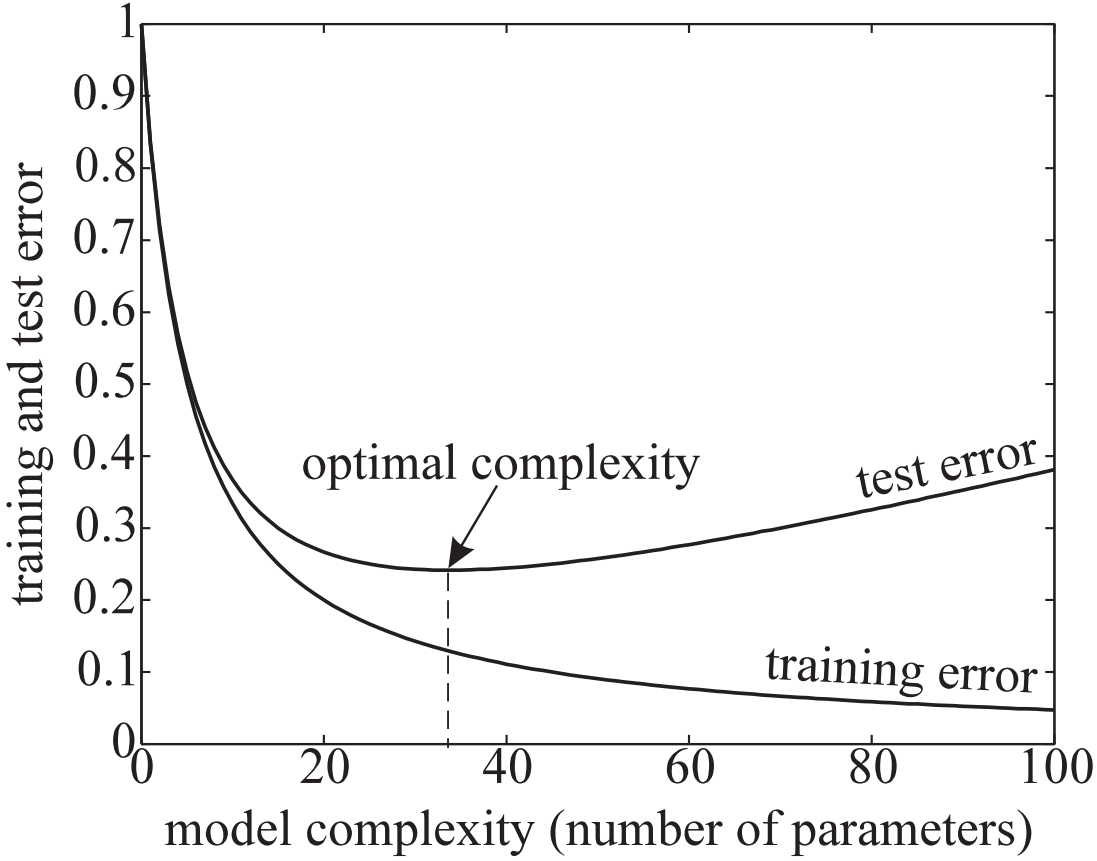


Figure 6: Learning and test error. The learning error does not contain the variance error in the error of the model, while the test error contains the complete error of the model

Model quality estimation

- The model we evaluate on learning data does not contain a variance error.
- Evaluation must be performed on test data.
- The goal is to get a model that will behave well on new, unknown data. On data that were not used for learning.
- Because of this, it is necessary to divide data into the learning set and the test set.
- If certain areas of operation contain a smaller amount of data, then we can not expect a good model for this area.
- If certain data are missing in the test set, then we can not expect a reliable estimate of the quality of the model.
- To determine the optimal complexity of the model, the following strategy (figure 7) is very appropriate: we train T models of different complexity on the learning set, we evaluate their quality on a verification data set. We select a model that has the smallest error on the verification data and is confirmed on the test data.

Cross-validation in the case of a small number of data.

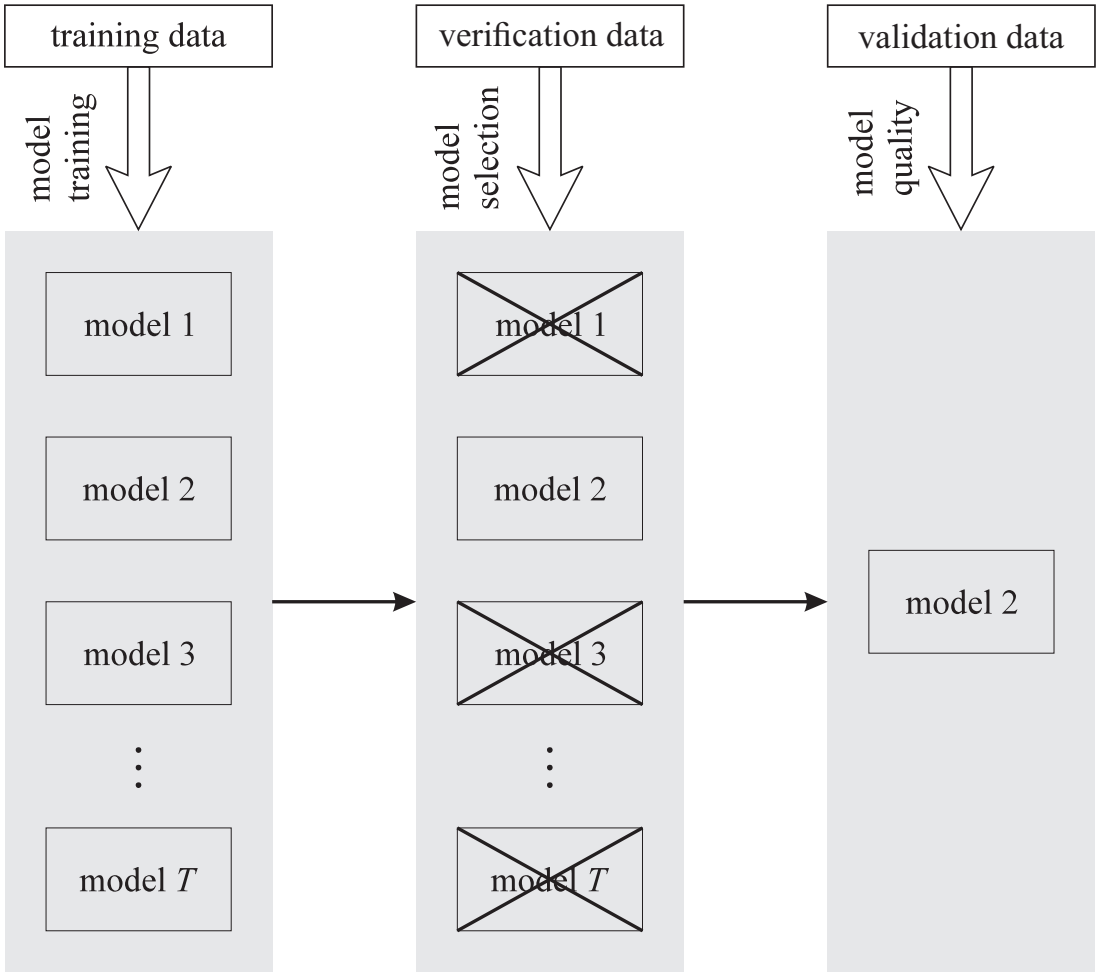


Figure 7: Learning, verification and validation. We train different models on learning data. The quality of these models is evaluated on a verification data set, where we select the best one, which is then confirmed on the validation data

Cross-validation

Cross-validation in the case of insufficient number of data

- Common situation.
- A set containing N data is divided into S subsets.
- Then $S - 1$ subsets are used for learning, and one for validation of the model.
- This procedure is repeated for all possible S combinations; see figure 8.
- In this way we can use much more original data for learning purposes.
- Since learning and testing are repeated S times with different parts of the original set, we can average all the test errors of the model in order to obtain a reliable evaluation of the quality of the model on new data.
- Cross-validation is used only to evaluate the quality of the model on new data. For the end-use, the model must be trained on the whole set of learning data to use all possible information.

Cross-validation

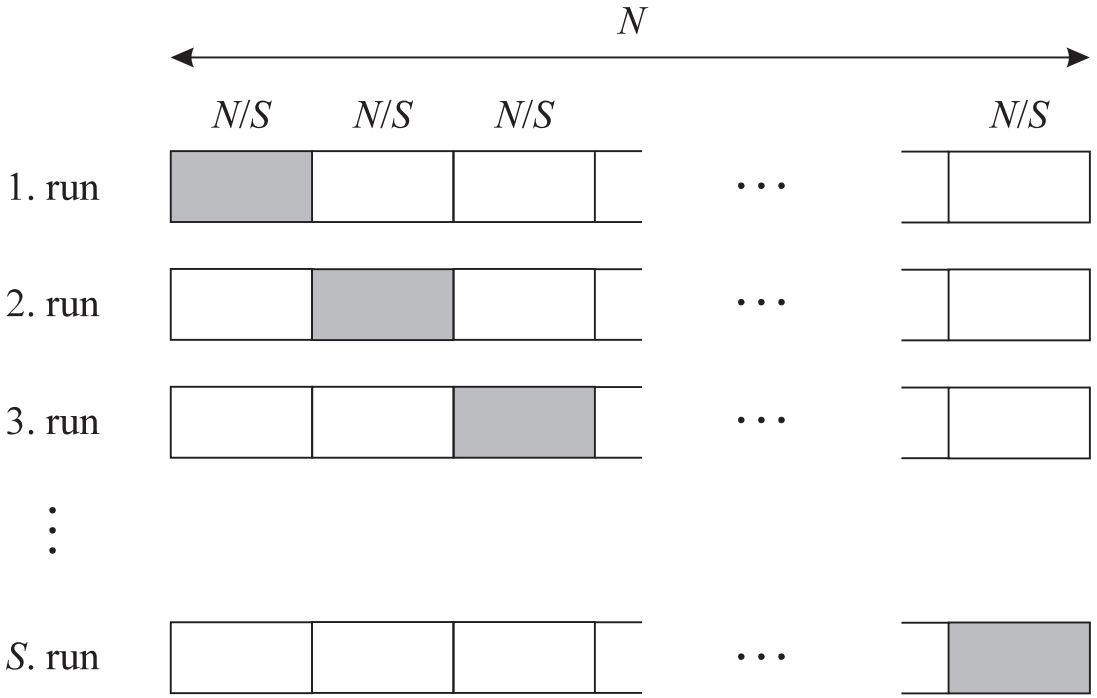


Figure 8: For cross-validation, N data are divided into S parts. In every run $S - 1$ parts are used for the learning set and the rest for verification. After all S runs we average the error of the model

Cross-validation

- A typical value for S is 10, where the number of elements in the set is the same.
- Division at $S = 10$ determines that we use $9/10$ of learning data instead of $1/2$, as is the case with the simple approach.
- The extreme situation occurs when $S = N$, which can be used in the case of an extremely small number of data. The method is called *leave-one-out*. In this case, we omit one sample and train the model at all others. We do this for all samples, N times, which requires N calculation times.

Introduction to approximation models

Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for the 8th lecture, November 2016

Table of Contents

1. Introduction to approximation models
2. Basis function models
   - Global and local basis functions
   - Linear and nonlinear parameters
3. Extended basis functions
4. Quality criterion of approximation model

Basis function models

Most of the useful ways of approximation of function $f(\cdot)$ can be written with the basis functions:

$$\hat{y} = \sum_{i=1}^{M} \theta_i^{(\mathrm{l})} \Phi_i \left(\underline{u}, \underline{\theta}_i^{(\mathrm{nl})} \right) . \tag{1}$$

- The output y is modelled (approximated) with the weighted sum of M basis functions $\Phi_i(\cdot)$.
- The basis functions are weighted by the *linear* parameters $\theta_i^{(\mathrm{l})}$ and they depend on the input vector $\underline{u}$ and the set of *nonlinear* parameters $\underline{\theta}_i^{(\mathrm{nl})}$.
- If we want to model a nonlinear function, then the basis functions must be nonlinear. Therefore, it is necessary that the parameters of these functions $\underline{\theta}_i^{(\mathrm{nl})}$ are nonlinear.

Basis function models

- The models also often include an *offset parameter* that defines the operating point.
- The appropriate linear parameter $\theta_0^{(1)}$ introduces the offset parameter

$$\hat{y} = \sum_{i=0}^{M} \theta_i^{(1)} \Phi_i \left(\underline{u}, \underline{\theta}_i^{(\mathrm{nl})} \right) \qquad \text{and} \quad \Phi_0(\cdot) = 1 \,. \tag{2}$$

- A notation with basis functions in (1) and (2) can be shown as a network in figure 1.
- In general, each node can have a different type of the basis functions $\Phi_i(\cdot)$.

Basis function models

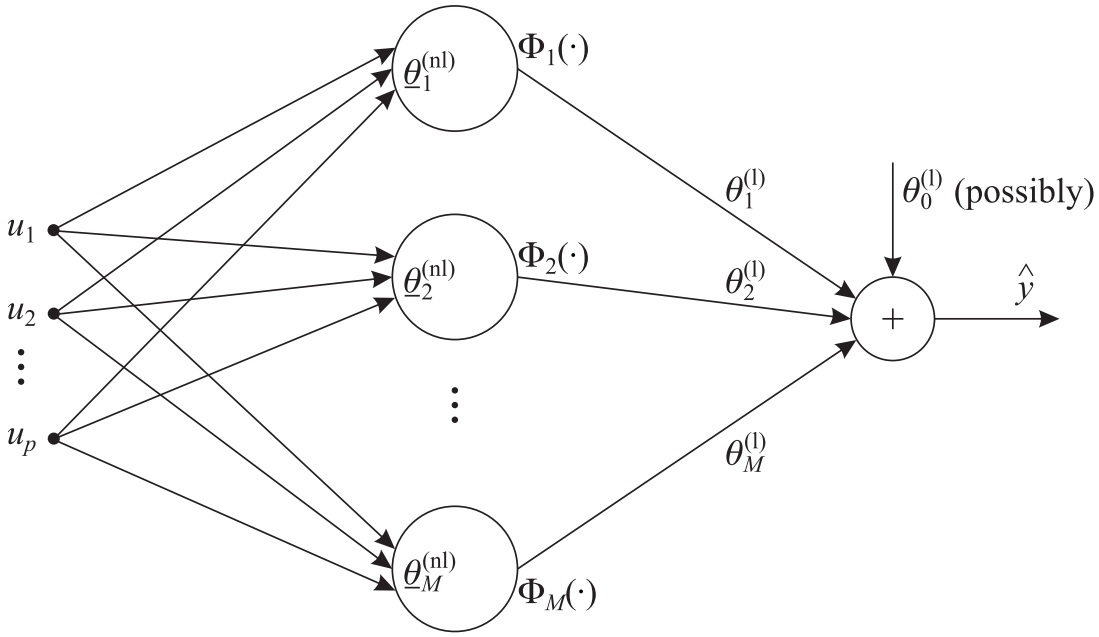


Figure 1: Network of the basis functions. Each node represents a basis function that depends on the nonlinear parameters $\underline{\theta}_i^{(nl)}$.

Global and local basis functions

- In general, the basis functions $\Phi_i(\cdot)$ can have any form.
- In fuzzy systems and neural networks, they are composed of elementary functions.
- We divide them into **global basis functions**, which contribute to the output over the entire interval of the input variable; figure 2 and **local basis functions**, which affect the output only locally; figure 3.

Global and local basis functions

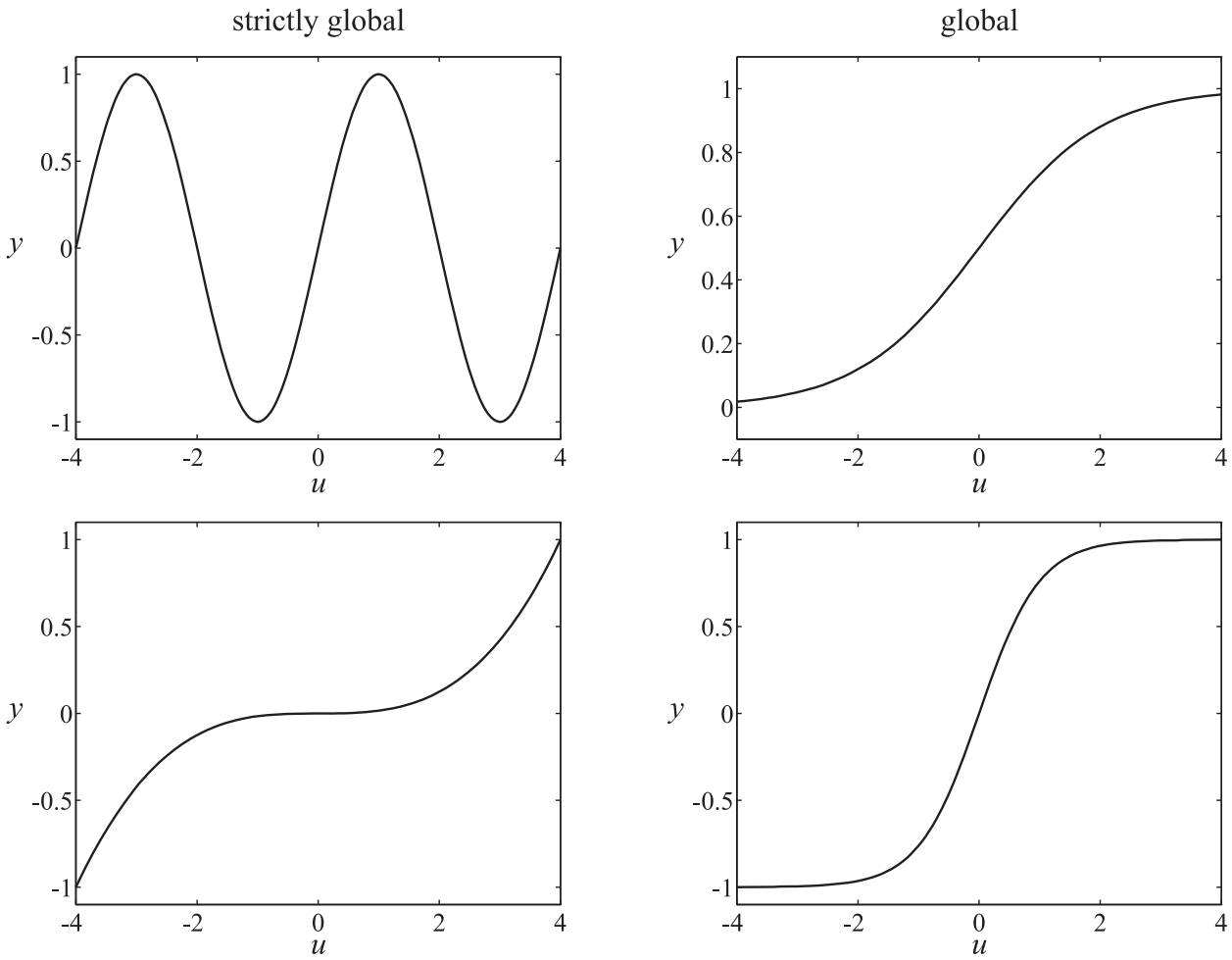


Figure 2: General global basis functions. The basis function is *strictly* global (left) if derivative of the function is global, i.e. the sine function (above) or polynomial (bottom). Otherwise, the function is not strictly global, e.g. sigmoid function (above) or tanh (below)

Global and local basis functions

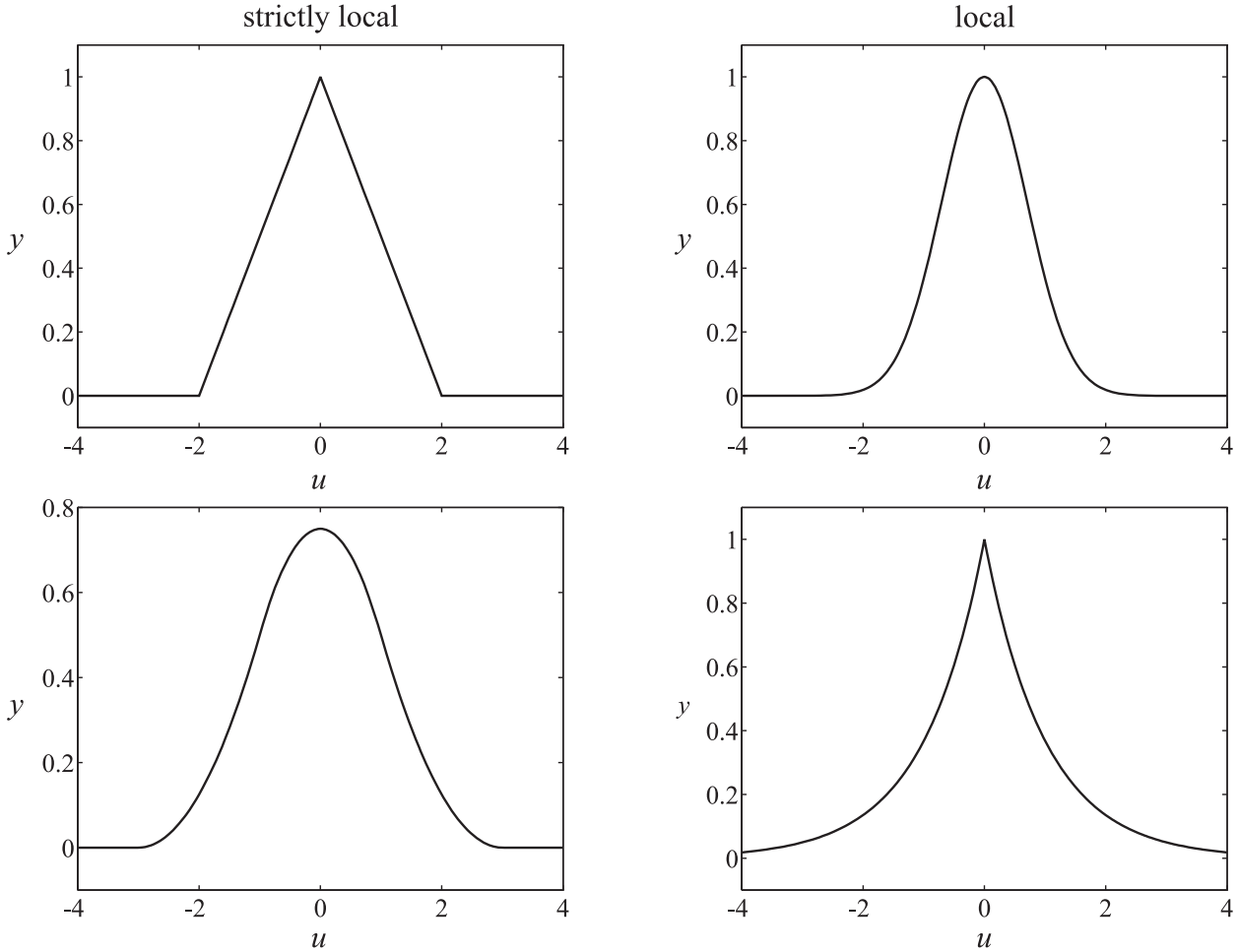


Figure 3: General local basis functions. The basis function is *strictly* local (left) if it is equal to zero outside the activation area, i.e. the triangular function (above) or the B-spline of the third-order (below). Otherwise, it is not strictly local, e.g. Gaussian function (above) or double exponential function (below)

Linear and nonlinear parameters

Independently of the model's architecture, based on basis functions we can conclude some basic properties for a wide range of models:

- The model is linear in its parameters $\theta_i^{(1)}$ if the basis functions are completely defined, i.e. nonlinear parameters are completely defined.
- Linear parameters can be estimated by any linear optimization technique, e.g. the least squares.
- Regression matrix $\underline{X}$ and vector of parameters $\underline{\theta}^{(1)}$ are equal to:

$$\underline{X} = \begin{bmatrix} 1 & \Phi_1(\underline{u}(1)) & \Phi_2(\underline{u}(1)) & \cdots & \Phi_M(\underline{u}(1)) \\ 1 & \Phi_1(\underline{u}(2)) & \Phi_2(\underline{u}(2)) & \cdots & \Phi_M(\underline{u}(2)) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \Phi_1(\underline{u}(N)) & \Phi_2(\underline{u}(N)) & \cdots & \Phi_M(\underline{u}(N)) \end{bmatrix} \quad \underline{\theta}^{(1)} = \begin{bmatrix} \theta_0^{(1)} \\ \theta_1^{(1)} \\ \theta_2^{(1)} \\ \vdots \\ \theta_M^{(1)} \end{bmatrix}, \tag{3}$$

  where N is the number of data that we need for learning. For simplicity, the dependence of the basis function on nonlinear parameters is ignored in this case.
- The first column in $\underline{X}$ and the first term in $\underline{\theta}^{(1)}$ are due to an offset parameter.

Linear and nonlinear parameters

- **Optimization of nonlinear parameters** $\underline{\theta}_i^{(\mathrm{nl})}$ is much more demanding as it requires nonlinear local or global optimization. Nonlinear parameters are usually optimized by one of the following learning techniques:
  - nonlinear optimization,
  - unsupervised learning, e.g. by clustering,
  - the use of a priori knowledge.

- Nonlinear parameters affect the basis functions. In general, they define their position in problem space, defined by the input vector $\underline{u}$, the shape of the function and its width.

- In this way, nonlinear parameters allow to set the position and shape of the basis function to adjust the model and increase its flexibility.

Extended basis functions

- The notation of the basis functions in equation 1 can be expanded into a more flexible structure by replacing linear parameters with the function (linearly parameterized) $L_i(\cdot)$:

$$\hat{y} = \sum_{i=1}^{M} L_i\left(\underline{u}, \underline{\theta}_i^{(1)}\right) \Phi_i\left(\underline{u}, \underline{\theta}_i^{(\mathrm{nl})}\right) . \tag{4}$$

- If $L_i(\cdot)$ is linearly parameterized, the parameters $\underline{\theta}_i^{(1)}$ can be estimated by linear optimization. In the simplest case when $L_i = \theta_i^{(1)}$ we get the elementary basis function.
- Another common alternative is the choice of $L_i(\cdot)$ as a linear combination of the components of the vector $\underline{u}$. So we get:

$$\hat{y} = \sum_{i=1}^{M} \left(w_{i0} + w_{i1}u_1 + w_{i2}u_2 + \ldots + w_{ip}u_p\right) \Phi_i\left(\underline{u}, \underline{\theta}_i^{(\mathrm{nl})}\right) , \tag{5}$$

where

$$\underline{\theta}_i^{(1)} = [w_{i0}\ w_{i1}\ w_{i2}\ \cdots\ w_{ip}]^{\mathrm{T}} . \tag{6}$$

- The basis functions are no longer weighted by the constant weights (parameters), but by a linear model. In general, the submodels $L_i(\cdot)$ can be arbitrarily complex.

Quality criterion of approximation model

Important quality criteria of approximation models are:

- *Interpolation:* the quality of the model within the scope of the learning set.
- *Extrapolation:* the quality of the model outside the scope of the learning set.
- *Locality:* the basis functions are local, global, strictly global, or strictly local.
- *Accuracy:* how accurate is the model with a certain number of parameters.
- *Smoothness:* smoothness of the model output.
- *Noise sensitivity:* how the noise affects the behavior of the model.
- *Parameter optimization:* how we can optimize linear and nonlinear parameters of the model.
- *Structural optimization:* how we can optimize the structure and complexity of the model.
- *Online adaptation:* how we can adapt the parameters of the model online and whether they are reliable.
- *Learning speed:* how quickly we can train the model, its structure and parameters.
- *Speed of the model calculation:* how quickly we can calculate the model output at a particular input.
- *Dimension problem:* how the model behaves when the number of input variables increases.
- *Interpretation:* can we interpret the parameters and structure of the model in terms of the functioning of the process we are modelling.
- *Incorporating a priori knowledge:* can we incorporate a priori knowledge into the model.
- *Using:* how the model's architecture is expanded.

Linear and polynomial models

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for lectures, April 2016

Table of contents

1. Introduction
2. Linear Model
3. Polynomial Model

Introduction

In this section, we will analyse classical, traditional architectures of models for describing nonlinear static processes.

- linear model that can be used if nonlinearity is weak
- polynomial model

Linear model

- Linear model can approximate nonlinear behaviour of the process when nonlinearity is weak.
- The advantage of the linear model is that it is simple and has a small number of parameters.
- Especially in the case when there is a very small number of noisy learning data, the description with the linear model can be a good choice, because in the case of a complex nonlinear model there would be a much larger variance error.
- Linear model is written as

$$\hat{y} = w_0 + w_1 u_1 + w_2 u_2 + \ldots + w_p u_p \tag{1}$$

or

$$\hat{y} = \sum_{i=0}^{p} w_i u_i \qquad \text{pri} \quad u_0 = 1\,. \tag{2}$$

Linear model

Figure 1 shows the **linear model for two input variables and one output variable**

- it is represented by a plane in a three-dimensional space.
- In the case of higher dimensions, we get a hyperplane with a DC value w_0, which is defined by $\underline{u} = \underline{0}$.
- The parameters w_i, $i > 0$ define the slope of the hyperplane in the directions of the variables u_i.
- In terms of the base formulation, the inputs u_i are the basis functions, and the coefficient w_i are linear parameters.
- This notation does not contain nonlinear parameters.

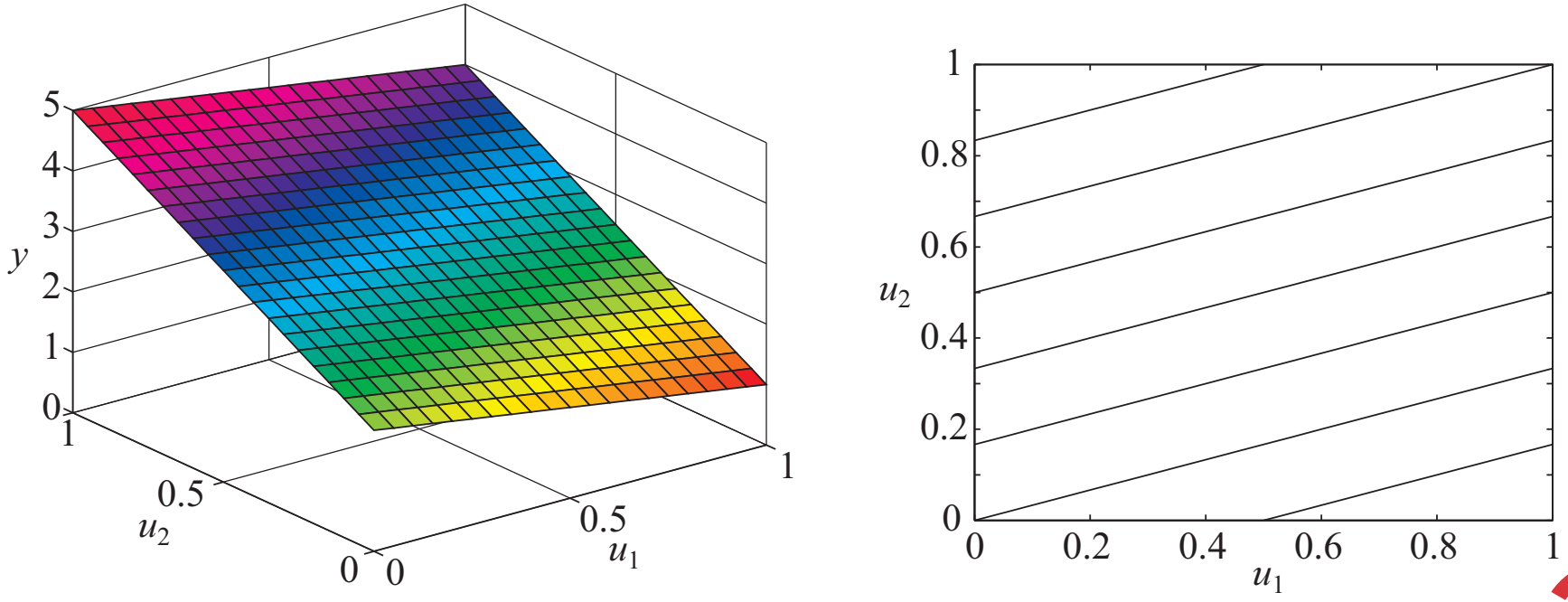


Figure 1: Linear model for two inputs and outputs. Model process with the function $y = 2 - u_1 + 3u_2$

Linear model

The main properties of the linear model are as follows:

- *Interpolation* is linear.
- *Extrapolation* is linear and therefore the gradient outside the learning area remains the same.
- *Locality* does not exist. The linear model is completely global.
- *Accuracy* depends on nonlinearity of the process. It decreases with nonlinearity.
- *Smoothness* is complete since the hyperplane gradient is constant throughout the area.
- *Noise sensitivity* is very low since we can use all the data for learning.
- *Optimization of parameters* is simple since we can use the least squares method.
- *Structural optimization* is effective with the use of the technique of selecting a structure by orthogonal least squares.
- *Real-time adaptation* is feasible with the method of recursive least squares.
- *Learning speed* is very fast. The calculation time increases with the third power depending on the number of parameters.
- *Model calculation speed* is fast because we only need p multiplications and sums.
- *Dimension problem* is small because the number of parameters increases linearly with the input dimension.
- *Interpretation* is possible through the influence of variables on the output.
- *Integrating limits* of output and parameters is possible if we use the appropriate optimization technique to optimize the parameters.
- *Incorporating a priori knowledge* is possible and requires special regression.
- *Usage* is very widespread. Linear models are an established standard.

Polynomial model

- Polynomial formulation is the simplest extension of linear models.
- By increasing the degree of the polynomial we get more flexible models.
- The polynomial of the k order and the p dimension is written as

$$\hat{y} = w_0 + \sum_{i=1}^{p} w_i u_i + \sum_{i_1=1}^{p} \sum_{i_2=i_1}^{p} w_{i_1 i_2} u_{i_1} u_{i_2} + \ldots + \sum_{i_1=1}^{p} \cdots \sum_{i_l=i_{l-1}}^{p} w_{i_1 \cdots i_l} u_{i_1} \cdots u_{i_l}. \quad (3)$$

- In each sum there is a number of parameters depending on the dimension of the problem p and the order of the terms in this sum, which is denoted by r, where $r = 0, \ldots, k$. Here, k is the highest order in the polynomial of the model. The number of combinations of the choice of r elements from the set of p elements, if we allow the elements in p to be repeated in selected combinations, is

$$m_r = \binom{p+r-1}{r}. \quad (4)$$

The number of all possible parameters in the polynomial can then be written as the sum of terms from 0-th order to k-th order,

$$M = \sum_{r=0}^{k} m_r = \binom{p-1}{0} + \binom{p}{1} + \cdots + \binom{p+k-1}{k}. \quad (5)$$

- We can show that the maximum number of parameters in k-th order polynomial of dimension p equals

$$M = \frac{(p+k)!}{p!\,k!} \quad (6)$$

Polynomial model

- The structure requires $M-1$ basis functions and M parameters. Polynomial can be expressed as

$$\hat{y} = \sum_{i=0}^{M-1} \theta_i x_i \qquad \text{za} \quad x_0 = 1\,, \tag{7}$$

- The notation is linear in the parameters θ_i and does not contain nonlinear parameters.
- In the case of three inputs the second-order polynomial equals

$$\begin{aligned} \hat{y} &= \theta_0 + \theta_1 u_1 + \theta_2 u_2 + \theta_3 u_3 \\ &+ \theta_4 u_1^2 + \theta_5 u_1 u_2 + \theta_6 u_1 u_3 + \theta_7 u_2^2 + \theta_8 u_2 u_3 + \theta_9 u_3^2\,. \end{aligned} \tag{8}$$

Polynomial model

Figure 2 shows the characteristics of this polynomial.

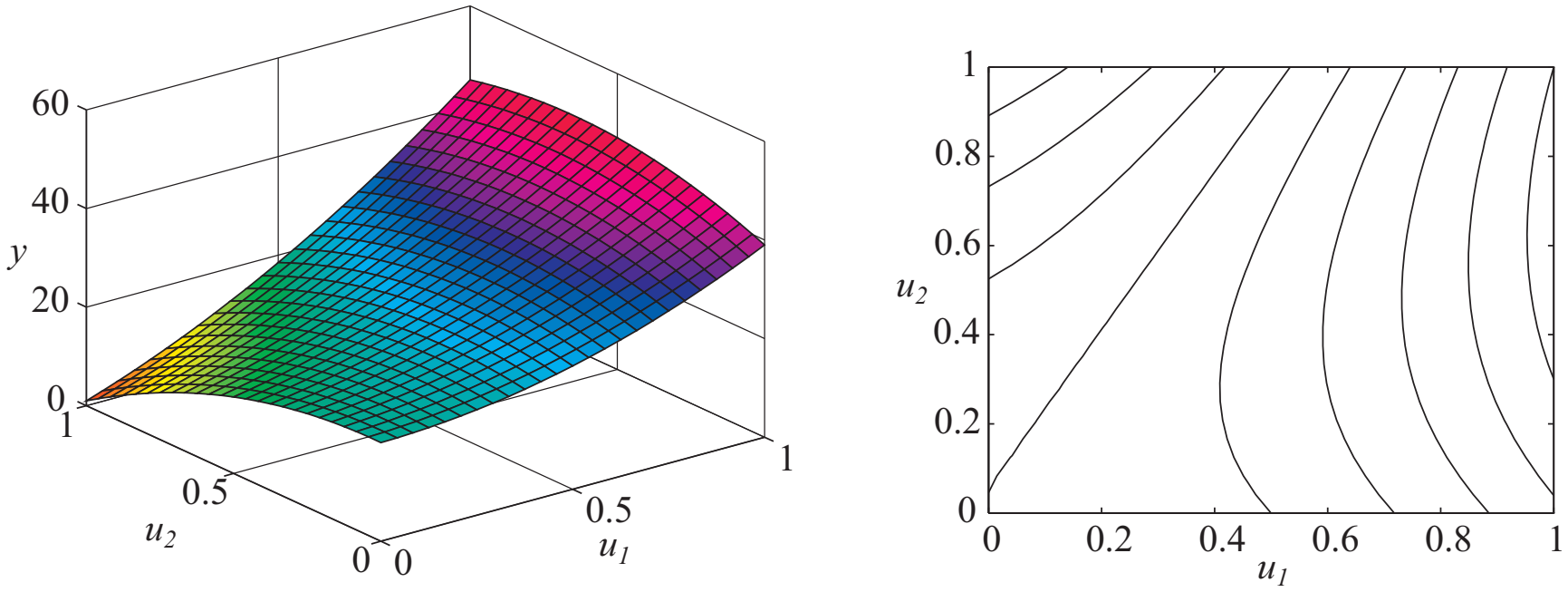


Figure 2: The second-order polynomial with two inputs and contours. Function of the model is $y = 20 + u_1 + u_2 + 15u_1^2 + 20u_1u_2 - 20u_2^2$

Polynomial model

Estimation of parameters

- Model parameters can be estimated using the least squares method.
- Regression matrix $\underline{X}$ and the vector of parameters $\underline{\theta}$ for the polynomial model of l degree and with p inputs are

$$\underline{X} = \begin{bmatrix} 1 & u_1(1) & \cdots & u_p(1) & u_1^2(1) & \cdots & u_p^l(1) \\ 1 & u_1(2) & \cdots & u_p(2) & u_1^2(2) & \cdots & u_p^l(2) \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & u_1(N) & \cdots & u_p(N) & u_1^2(N) & \cdots & u_p^l(N) \end{bmatrix} \qquad \underline{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{M-1} \end{bmatrix} \tag{9}$$

where $M-1$ is the number of basis functions with respect to equation 6.

Polynomial model

- The number of parameters and thus the complexity of the model greatly increases with the number of inputs p and with the degree of polynomial l.
- Even in the case of minor problems, it is very difficult to evaluate the parameters for the entire structure of the model.
- For this reason it is reasonable to define the structure of polynomial models.
- This leads to a reduced structure of a polynomial model in which the parameters of the model can be determined by using the method of orthogonal least squares.

- The disadvantage of the model is the tendency towards oscillatory interpolation behaviour, especially in higher-dimensional models.
- In the case of extrapolation, the problem is that polynomials tend toward $+\infty$ and $-\infty$ with large values of derivatives.
- In figure 3a and figure 3b we see the problem of extrapolation, which is higher in the case of a higher-order model.
- In most cases the extrapolation is determined by the term of highest degree.
- The extrapolation may also be non-monotonic and independent of the derivative at the boundary of the learning area.
- Although the interpolation and extrapolation properties of polynomials are poor, they are sometimes used with advantage, especially in order to write down the real structure of the process.
- Weaknesses of polynomials lead to the development of *splines*, which are locally defined polynomials, and *ortonormal polynomials*.

Polynomial model

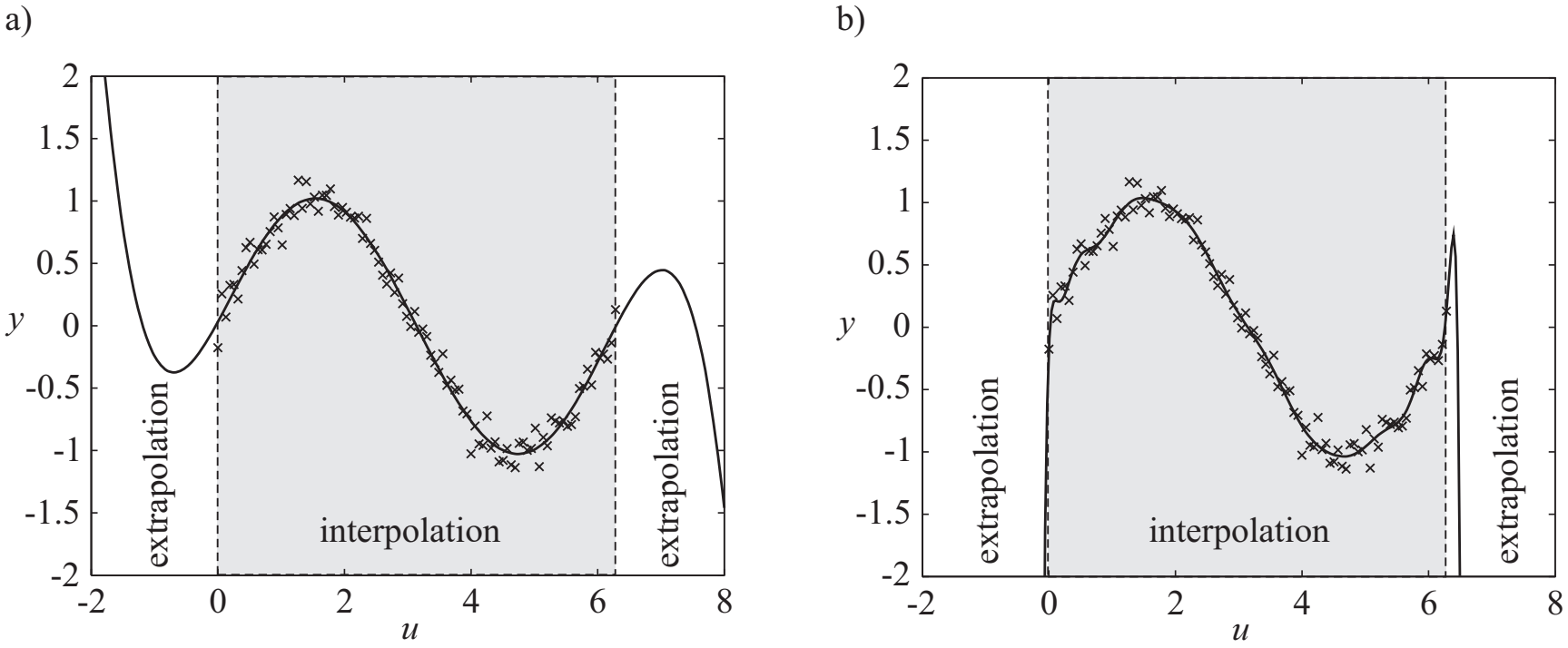


Figure 3: a) The polynomial model of 5-th degree in the approximation of the sinus. b) The polynomial of 20-th degree in the approximation of the sinus. In the figure, there are 100 noisy data marked with " $\times$ ". The broken line denotes the interval of the learning data $[0,\ 2\pi]$

Polynomial model

Polynomial properties can be combined into the following:

- *Interpolation* is non-monotonic and oscillatory for higher-order polynomials.
- *Extrapolation* has a high tendency to $+\infty$ or $-\infty$ with respect to the term of highest order.
- *Locality* does not exist. The polynomial model is completely global.
- *Accuracy* is limited, because higher order polynomials are practically inadequate.
- *Smoothness* is low. Gradient often changes the sign.
- *Noise sensitivity* is low since we can use all the data for learning.
- *Optimization of parameters* is simple since we can use the least squares method. However, the number of parameters grows rapidly with an increased degree of polynomial.
- *Structural optimization* is effective with the use of the technique of selecting a structure by orthogonal least squares. However, it can be very slow due to the large number of regressors.
- *Real-time adaptation* is feasible with the recursive least squares method, but it is very unreliable, since parameter changes in one operating point have an impact on all areas.
- *Learning speed* is fast. However, it is slowed down if structural optimization is considered.
- *Model calculation speed* is medium.
- *Dimension problem* is high since the number of parameters increases rapidly with the dimension of the problem.
- *Interpretation* is barely possible, except in cases where the structure actually describes physical equations of the process and model parameters have physical meaning.
- *Integrating limits* of output and parameters is possible if optimization technique is used to optimize parameters.
- *Incorporating a priori knowledge* is almost impossible, except in the case of the imitation of the physical process.
- *Usage* is very widespread.

Comparison between linear and polynomial models

Properties	Linear Model	Polynomial Model
Interpolation	+	−
Extrapolation	+	−−
Locality	−−	−−
Accuracy	−−	0
Smoothness	++	−
Noise sensitivity	++	+
Optimization of parameters	++	++
Structural optimization	++	0
Real-time adaptation	+	−
Learning speed	++	+
Model calculation speed	++	0
Dimension problem	++	−
Interpretation	+	0
Integrating limits	0	−
Incorporating a priori knowledge	0	−
Usage	++	+

$++/-$ = very good / very bad.

Neural networks

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for lectures, April 2016

Table of contents

1. Neural networks
2. Neural network construction
   - Projection construction
   - Radial construction
3. Multilayer perceptron (MLP)
   - MLP neuron
   - Network structure
   - Backpropagation training
   - MLP training
     - Nonlinear optimization of MLP network
   - Simulation example
   - MLP network properties
   - Multiple hidden layers networks
4. Summary

Neural networks

Neural networks and their approximation properties:

- a large number of simple units,
- strong parallel architecture,
- strongly connected units,
- robustness with respect to unit errors,
- learning based on data.

Neural networks

Table 1: **Terminology of neural networks** compared to the terminology of classical system identification

Table 1: Terminology of neural networks and classical system identification

Neural networks	System identification and statistics
Mapping or approximation	Regression
Neural network	Model
Neuron	Base function
Weight	Parameter
Bias or threshold	Offset
Input layer	Vector of inputs
Error	Residual
Learning or training	Estimation or optimization
Generalization	Interpolation or extrapolation
Overparametrization or overtraining	High variance error
Underparametrization or undertraining	High bias error
Error interval	Confidence interval
Online learning	Online adaptation
Learning	Adaptation

Neural networks

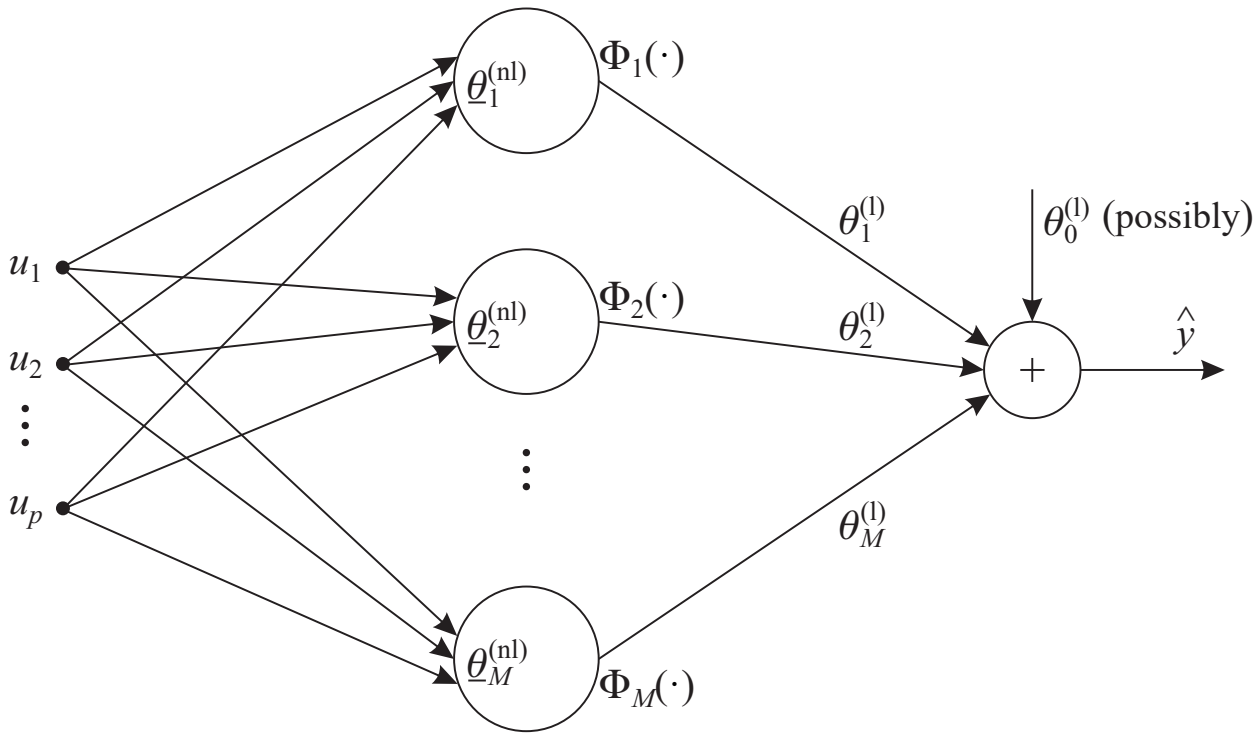


Figure 1: The neural network is a network of basis functions where all basis functions are of the same type

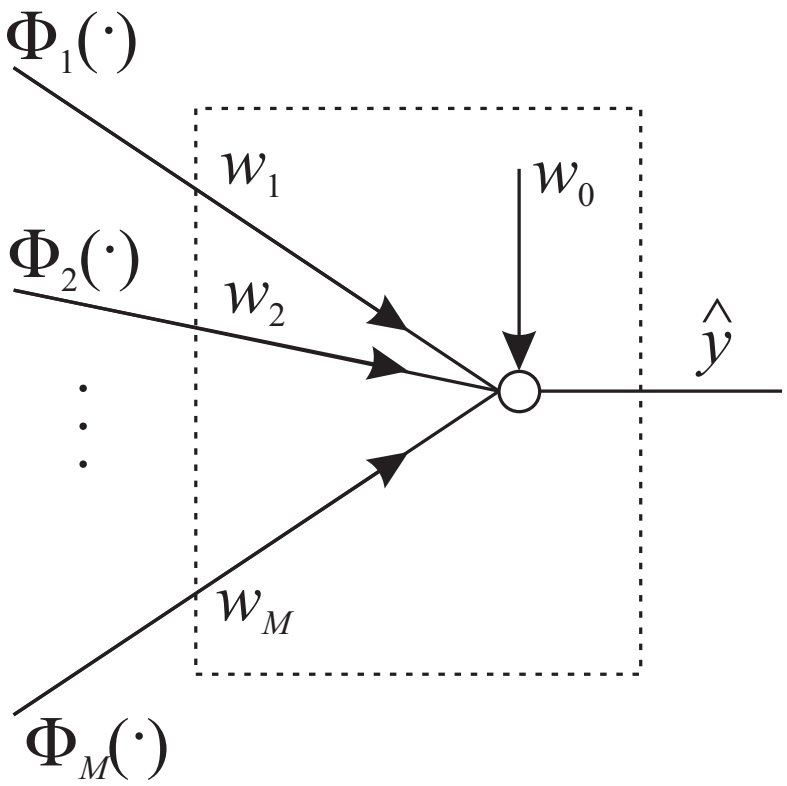


Figure 2: Output neuron performs a linear combination of hidden neurons in most network cases

Neural network construction

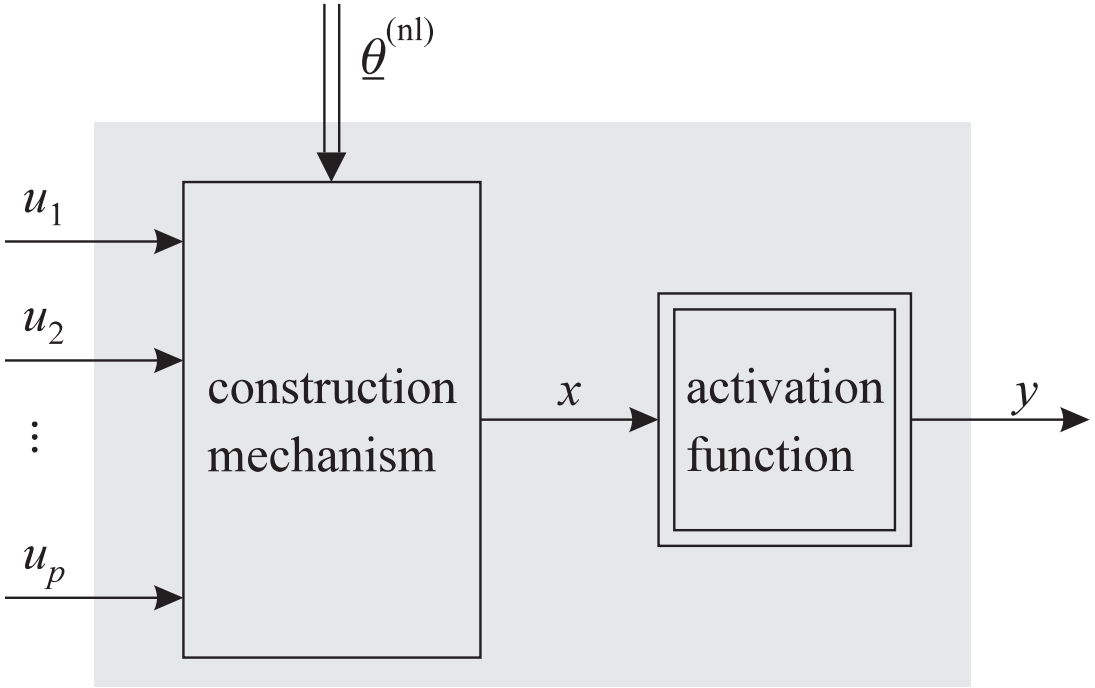


Figure 3: With operation of construction mechanism, we map the input vector $\underline{u}$ into the scalar input x using nonlinear activation functions and their parameters. The nonlinear activation function $g(x)$ maps the scalar x into the output y. The grey block generates a basis function $\Phi(\cdot)$

Projection network construction

- The principle of the projection construction was named after the scalar product between the input vector and the weight vector.
- The scalar product is geometrically interpreted as the projection of the input vector to the weight vector.
- This construction principle is used in the multilayer perceptron network (MLPs).
- An activation function operates on the scalar x obtained by projection of the input vector into a nonlinear space through nonlinear coefficients.
- This is done with a scalar product or a vector projection.

$$x = \underline{\theta}^{(\mathrm{nl})}\underline{\tilde{u}} = \theta_0^{(\mathrm{nl})}u_0 + \theta_1^{(\mathrm{nl})}u_1 + \ldots + \theta_p^{(\mathrm{nl})}u_p \tag{1}$$

where $\underline{\tilde{u}}$ is an extended input vector

$$\underline{\tilde{u}} = \begin{bmatrix} 1 \\ \underline{u} \end{bmatrix} = \begin{bmatrix} 1 \\ u_1 \\ \vdots \\ u_p \end{bmatrix} \tag{2}$$

Projection construction

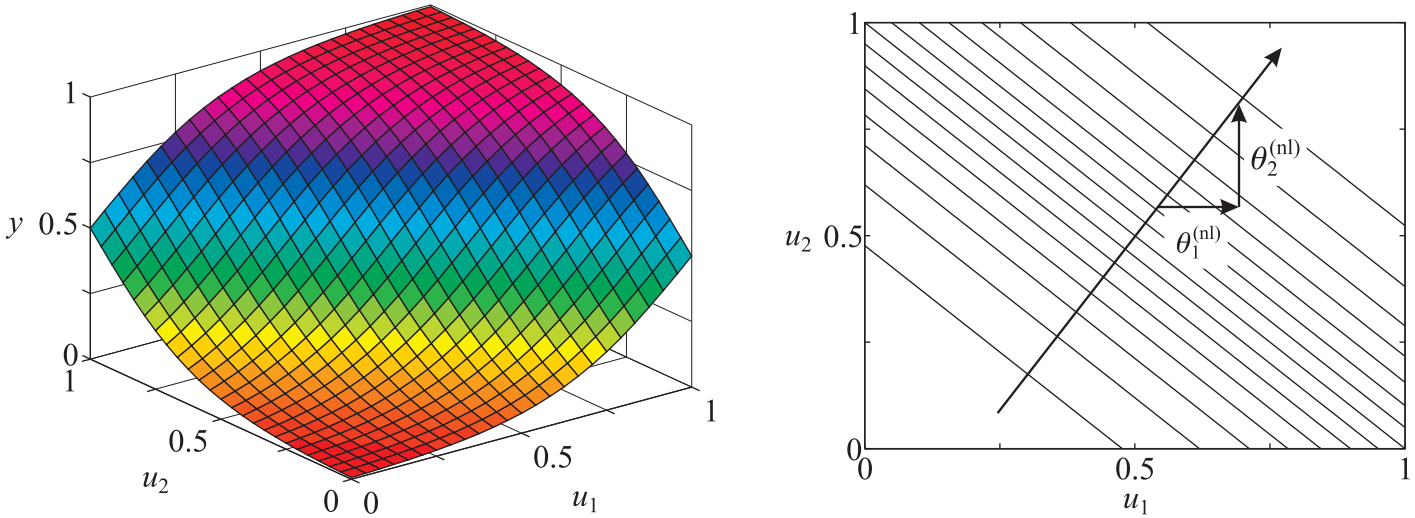


Figure 4: In the case of a projection construction, the value of the activation functions changes only in the direction of the vector of parameters (left). The contour diagram of the activation function shows the direction of nonlinearity (right). This direction and size are defined by nonlinear parameters in connection with the input vector.

Radial network construction

- Radial network construction is used in Gaussian neural networks.
- The scalar x is calculated as the difference between the input vector and the center of the activation function:

$$\begin{aligned} x &= ||\underline{u} - \underline{c}|| = \sqrt{(\underline{u} - \underline{c})^{\mathrm{T}}(\underline{u} - \underline{c})} \\ &= \sqrt{(u_1 - c_1)^2 + (u_2 - c_2)^2 + \ldots + (u_p - c_p)^2}\,, \end{aligned} \tag{3}$$

Radial construction

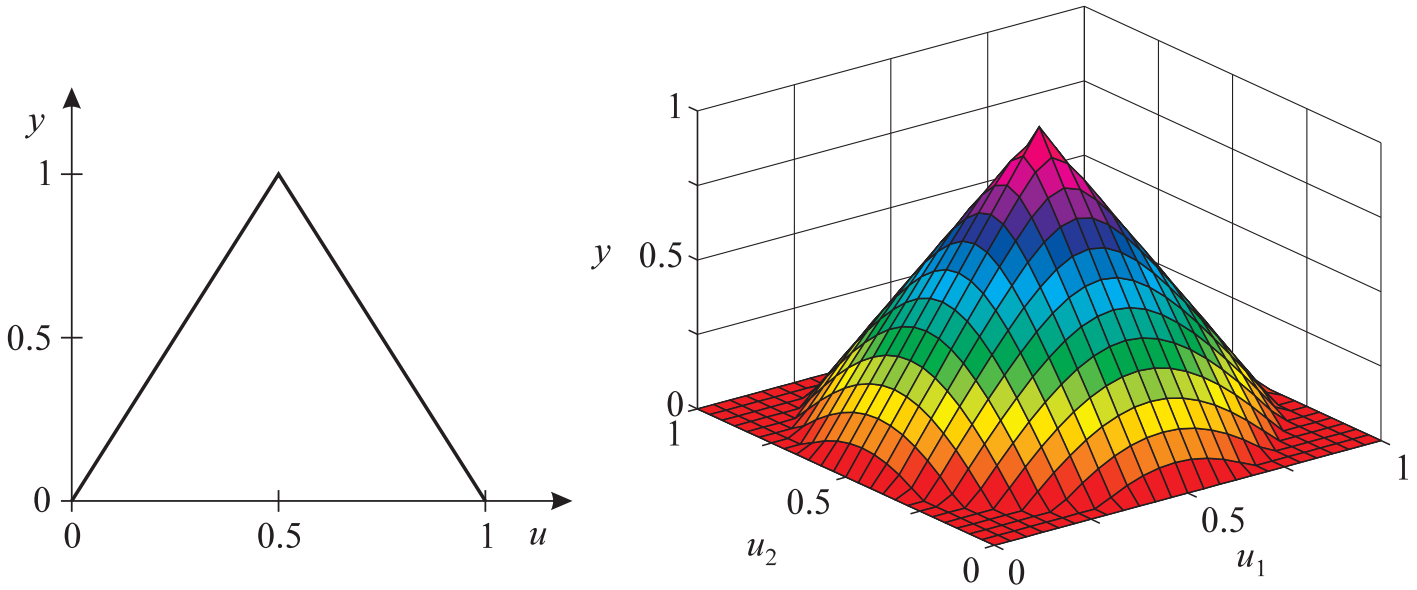


Figure 5: One and two-dimensional Gaussian or radial triangular activation function

Radial construction

- Mahalanobis norm for distance calculation

$$x = ||\underline{u} - \underline{c}||_{\underline{\Sigma}}^{-1} = \sqrt{(\underline{u} - \underline{c})^{\mathrm{T}} \underline{\Sigma}^{-1} (\underline{u} - \underline{c})} \tag{4}$$

- Parameters of the matrix norm $\underline{\Sigma}^{-1}$ are additional nonlinear parameters of the activation function.
- The matrix norm $\underline{\Sigma}^{-1}$ scales and rotates the axes; its inverse $\underline{\Sigma}$ represents a covariance matrix.
- Mahalanobis norm is equal to Euclidean norms with scaled input data $u_l^{(\mathrm{scaled})} = u_l/\sigma_l$.
- Figure 6 shows different distance norms.

Radial construction

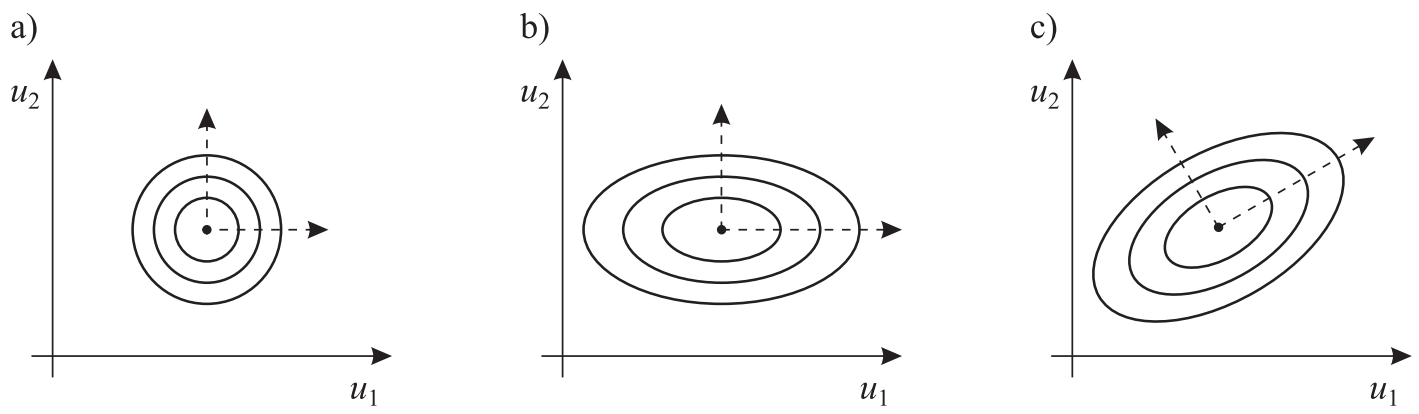


Figure 6: Lines of equal distances for different distance norms: a) Euclidean ($\underline{\Sigma}^{-1} = \underline{I}$), b) diagonal ($\underline{\Sigma}^{-1}$ = diagonal), and c) Mahalanobis norm ($\underline{\Sigma}^{-1}$ = general, positive symmetric matrix)

Multilayer perceptron (MLP)

- Multilayer perceptron (MLP) is the most well known and useful neural network architecture.
- Often it is also a synonym for a neural network. The reason for this lies in the very popular book by Rumelhart, Hinton and Williams.
- In it it has been shown that it is possible to overcome the functional limitations of a simple perceptron by adding layers to the network.

- Operation of MLP and its structure.
- A network learning algorithm, *backpropagation algorithm*.
- Overview of some modifications to the standard network architecture.

MLP neuron

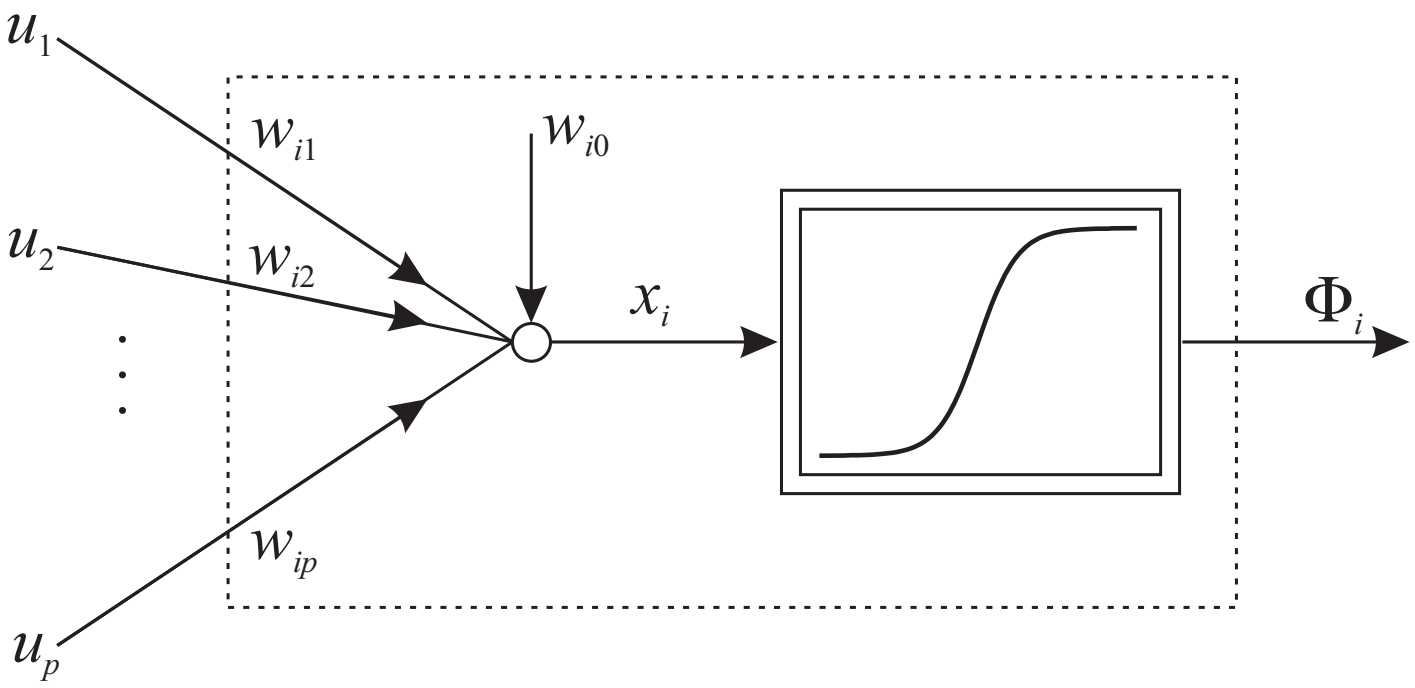


Figure 7: Perceptron: i-th hidden neuron MLP

MLP neuron

- Figure 7 shows a hidden neuron in a multilayered perceptron.
- Such a neuron is called *perceptron* .
- The functionality of this neuron can be divided into two parts. First, a projection construction is used for the projection of the input vector $\underline{u} = [u_1 \; u_2 \; \cdots \; u_p]^{\mathrm{T}}$ on the weight vector (scalar product, as a projection). In the second phase, through the nonlinear activation function, we map the input to the resulting output, $g(x)$. Usually, the activation function is of such a form that it has a limitation.
- One of such important functions is the **sigmoid function**

$$g(x) = \mathrm{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{5}$$

and **hyperbolic tangent**

$$g(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \, . \tag{6}$$

MLP neuron

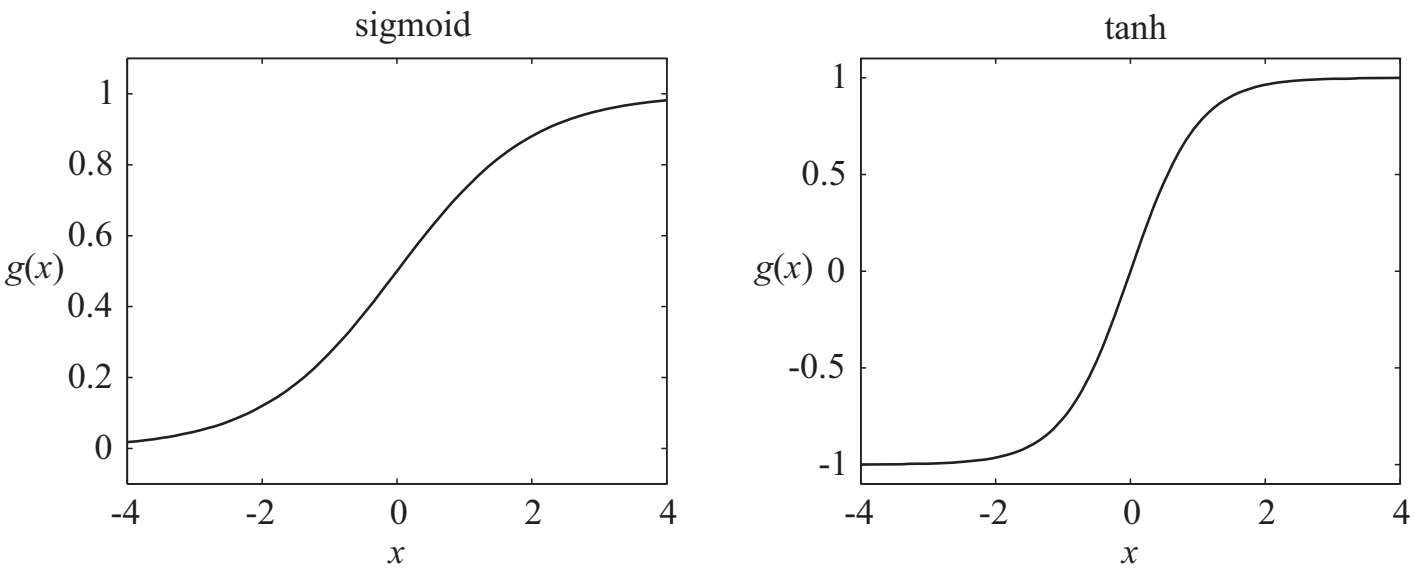


Figure 8: Typical perceptron activation function

Activation functions. An important feature is that the function itself appears in the function's derivative

- **sigmoid function**

$$\frac{d\,\Phi_i}{dx} = \frac{d\,\mathrm{sigmoid}(x)}{dx} = \frac{\exp(-x)}{(1+\exp(-x))^2} = \frac{1+\exp(-x)-1}{(1+\exp(-x))^2}$$
$$= \frac{1}{1+\exp(-x)} - \frac{1}{(1+\exp(-x))^2} = \Phi_i - \Phi_i^2 = \Phi_i(1-\Phi_i)\,. \tag{7}$$

- **tanh function**

$$\begin{aligned}\frac{d\,\Phi_i}{dx} &= \frac{d\,\tanh(x)}{dx} = \frac{1}{\cosh^2(x)} = \frac{\cosh^2(x)-\sinh^2(x)}{\cosh^2(x)} \\ &= 1-\tanh^2(x) = 1-\Phi_i^2\,. \end{aligned} \tag{8}$$

MLP nevron

$$\underline{\theta}_i^{(\mathrm{nl})} = [w_{i0}\ w_{i1}\ w_{i2}\ \cdots\ w_{ip}]^{\mathrm{T}}\,. \tag{9}$$

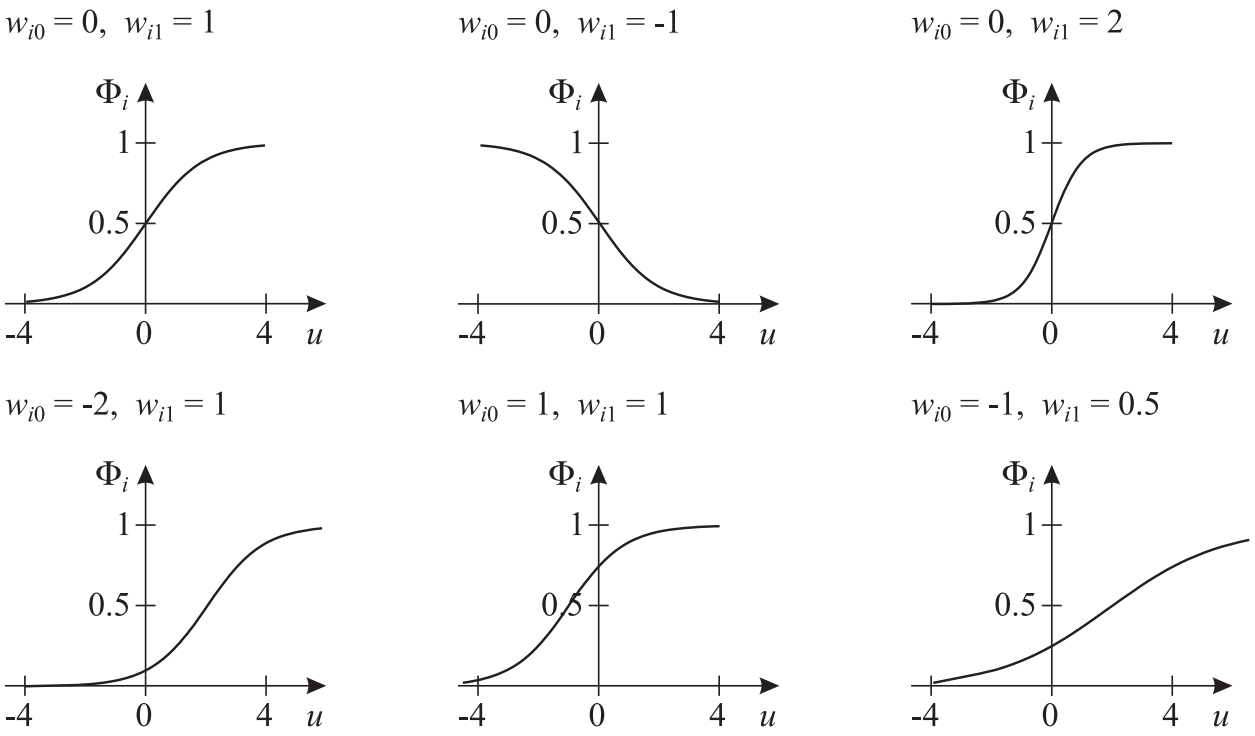


Figure 9: The effect of the parameters of the i-th neuron of the perceptron's hidden layer with a single input u. The weight w_{i0} defines the position of the activation function and the weight w_{i1} defines the slope of the activation function. In this case, the activation function is $\Phi_i(x) = 1/(1 + exp(-(w_{i1}u + w_{i0})))$

MLP neuron

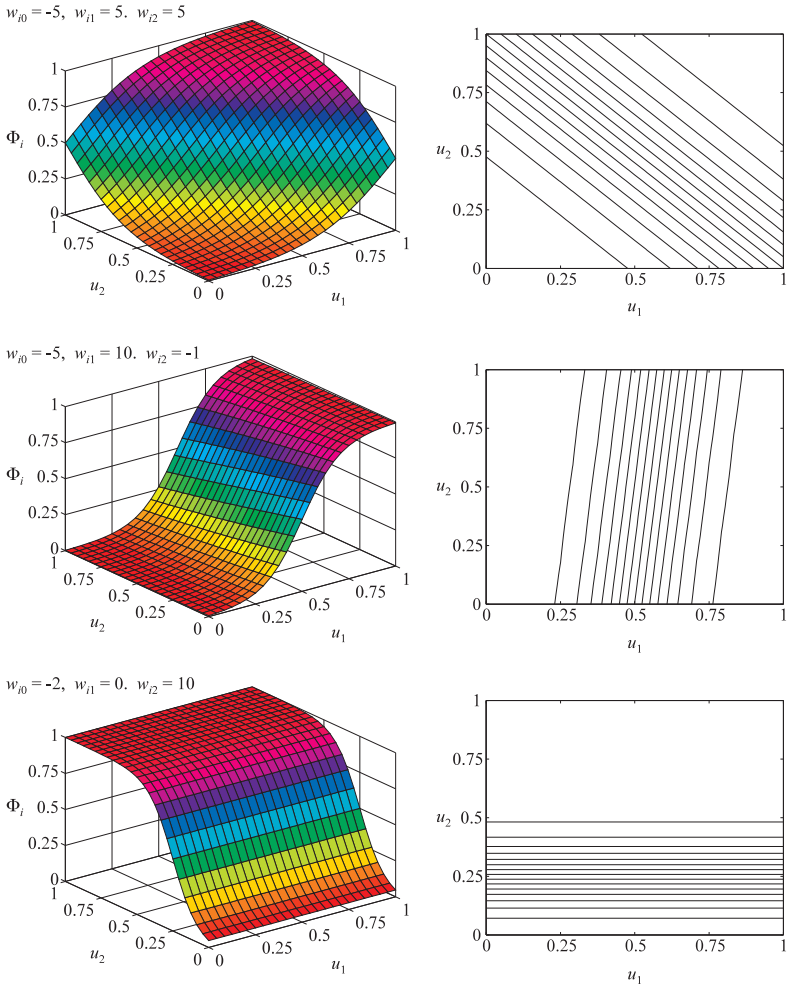


Figure 10: The effect of the parameters of the i-th neuron of the perceptron's hidden layer with two inputs u_1 and u_2. While the weight of w_{i0} defines the distance of the activation function to the coordinate starting point, the weights w_{i1} and w_{i2} define the slope in the direction of the variables u_1 and u_2. The weight vector $[w_{i1}\ w_{i2}]^{\mathrm{T}}$ points in the direction of nonlinearity of the basis function. Perpendicular to $[w_{i1}\ w_{i2}]^{\mathrm{T}}$ the activation function is constant

Network structure

If several neurons are parallel combined through the output neuron, we get **MLP network** with one hidden layer

Network output

$$\hat{y} = \sum_{i=0}^{M} w_i \Phi_i \left(\sum_{j=0}^{p} w_{ij} u_j \right) \qquad \text{za } \Phi_0(\cdot) = 1 \text{ in } u_0 = 1 \tag{10}$$

- w_i are the output layer weights and w_{ij} are the hidden layer weights.
- The activation function Φ_i depends on an argument which is a linear combination of p inputs and one additional input due to the operating point and nonlinear weights $w_{ij},\ j = 0, \ldots, p$.
- The output of the activation function equals

$$\Phi_i \left(\sum_{j=0}^{p} w_{ij} u_j \right)$$

- The total number of MLP network parameters is

$$M(p+1) + M + 1\,, \tag{11}$$

Network structure

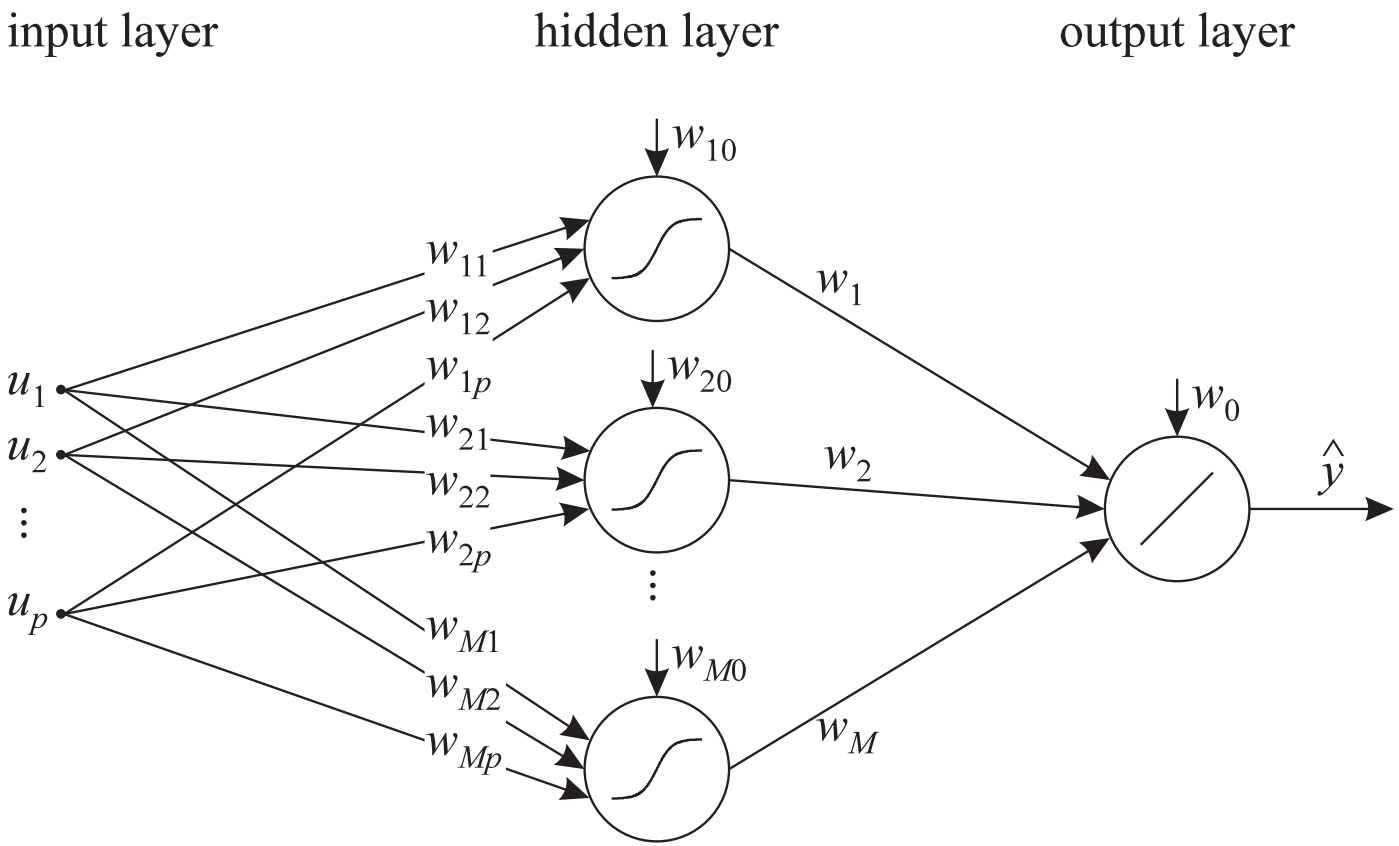


Figure 11: Multilayer perceptron network

Network parameters and training

- *Output layer weights* are linear parameters. They define the basis function amplitudes and the operating point.
- *Hidden layer weights* are nonlinear parameters and they define directions, slopes, and basis function positions.
- We train MLP network by optimizing these weights. Many learning strategies have been developed and addressed.
- The backpropagation algorithm is important.

Backpropagation training

- Backpropagation training is a way of calculating the gradients of the MLP network output according to the weights.
- The derivatives of MLP network output with respect to the i-th weight *in the output layer* are $(i = 0, \ldots, M)$

$$\frac{\partial \hat{y}}{\partial w_i} = \Phi_i \qquad \text{in } \Phi_0 = 1\,. \tag{12}$$

- The derivatives of MLP network output with respect to the *hidden layer* weights are $(i = 1, \ldots, M, j = 0, \ldots, p)$

$$\frac{\partial \hat{y}}{\partial w_{ij}} = w_i \frac{dg(x)}{dx} u_j \qquad \text{in } u_0 = 1 \tag{13}$$

- Influence of weights on the model output: the connection between j-th input and i-th hidden neuron. In equation 13 we implement equation 7 or equation 8, which define the derivatives of the activation function $g(x)$.
  For example, if $g(x) = \tanh(x)$, then equation 13 becomes

$$\frac{\partial \hat{y}}{\partial w_{ij}} = w_i(1 - \Phi_i^2)u_j \qquad \text{in } u_0 = 1\,. \tag{14}$$

$$\frac{\partial \hat{y}}{\partial w_{i0}} = w_i(1 - \Phi_i^2)\,. \tag{15}$$

- For the above notation of the output derivative with respect to the weights, we use backpropagation through the network. The algorithm is called the backpropagation algorithm. This becomes even more obvious if we have more hidden layers in the network.

MLP training

- We usually train MLP network using *nonlinear optimization*, where all weights get optimized simultaneously with local or global nonlinear optimization.
- The method requires a good initialization of weights.

- **Weight initialization**: Weights of neural networks are very difficult to interpret even with good knowledge of the physical process. For this reason, we practically can not initialize them reasonably.
- The simplest is random initialization, which leads to a large number of neurons in saturation due to the selection of large weights.

Nonlinear optimization of MLP network

- The most common way of optimizing the MLP network is to use nonlinear local optimization for determining the weights. Initially MLP was trained by using a gradient method, such as

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \eta \frac{\partial J_k}{\partial \underline{\theta}_{k-1}} \,, \tag{16}$$

where

$$\underline{\theta} = [w_0 \; w_1 \; \cdots \; w_M \; w_{10} \; w_{11} \; \cdots \; w_{1p} \; w_{20} \; w_{21} \; \cdots \; w_{Mp}]^{\mathrm{T}} \tag{17}$$

Simulation example

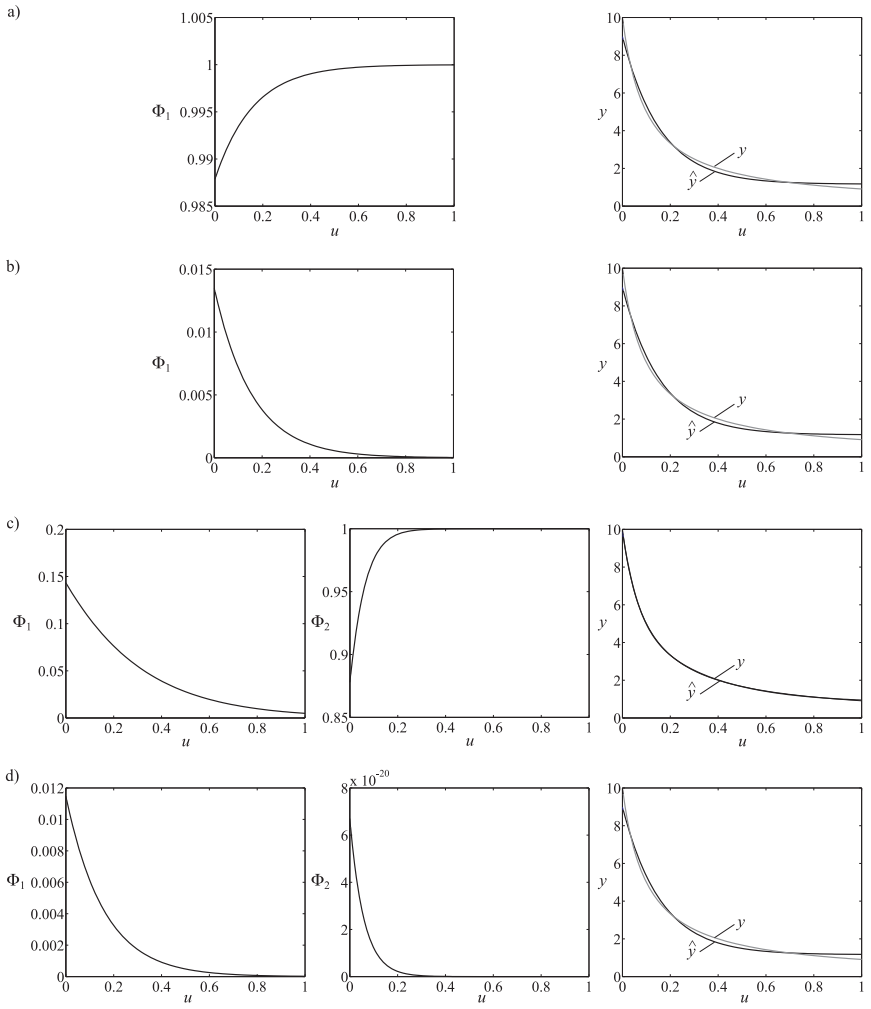


Figure 12: Function approximation $y = 1/(u + 0.1)$ with the MLP network with one hidden neuron (upper figures) and two hidden neurons (figures below). There are two possible training results with respect to the initialization of parameters, for example with one and two neurons, they are shown in figures a, b, and c, d

Simulation example

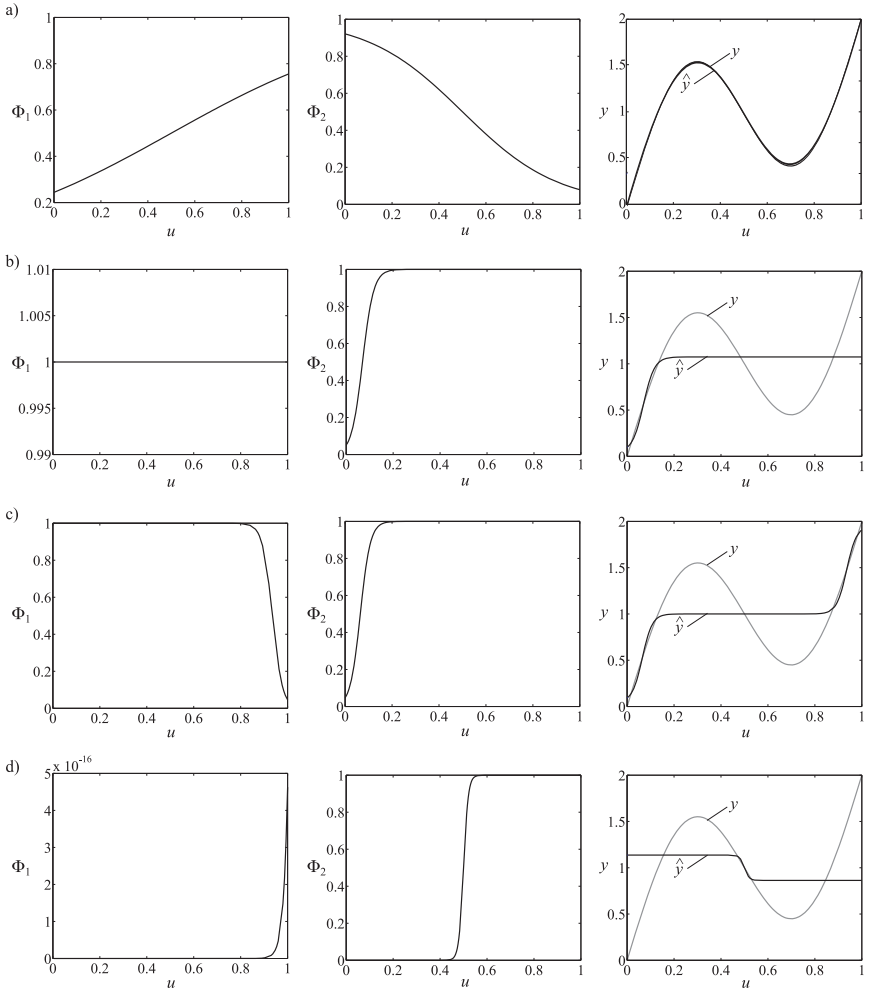


Figure 13: Function approximation $y = \sin(2\pi u) + 2u$ with the MLP network with two neurons in a hidden layer. Four different results of training, depending on the initial conditions, are shown in figures a-d

MLP network properties

The most important properties of the MLP network are the following:

- *Interpolation* is smooth with respect to the sigmoid activation function.
- *Extrapolation* is constant in a wide range with respect to the saturation property of the sigmoid function. Alternatively, extrapolation can also be linear, if the slope of the sigmoid function is small.
- *Locality* does not exist, because the change in a single weight affects the output of the model for a wide interval of input space. However, since activation functions are not strictly global, MLP has approximation flexibility only in the part where the activation function is not saturated. Therefore, the approximation mechanism has a certain locality.
- *Accuracy* is very high. Due to optimization of weights at a hidden layer, the MLP network is a very effective approximator and usually needs fewer parameters than other structures to achieve the same accuracy. This can also be interpreted as a property of effective information compression.
- *Smoothness* is high.
- *Noise sensitivity* is low, as we can use all the data for learning.
- *Parameter optimization* must be performed with nonlinear optimization and is therefore slow.
- *Structural optimization* requires computationally demanding methods for (*growing and pruning*).
- *Online adaptation* is very unreliable, since a small parameter change in an operating point has an impact on all areas.
- *Learning speed* is very slow because we need to use non-linear optimization techniques and repeat them several times due to local minima.
- *Model calculation speed* is fast due to the relatively small number of neurons.
- *Dimension problem* is very small due to the construction with the projection. MLP network is the most suitable network for high-dimensional problems.
- *Interpretation* is barely possible.
- *Incorporation of restrictions* is difficult because the interpretation is poor.
- *Incorporation of a priori knowledge* is almost impossible.
- *Usage* is very widespread. MLP is a standard neural network.

Multiple hidden layers networks

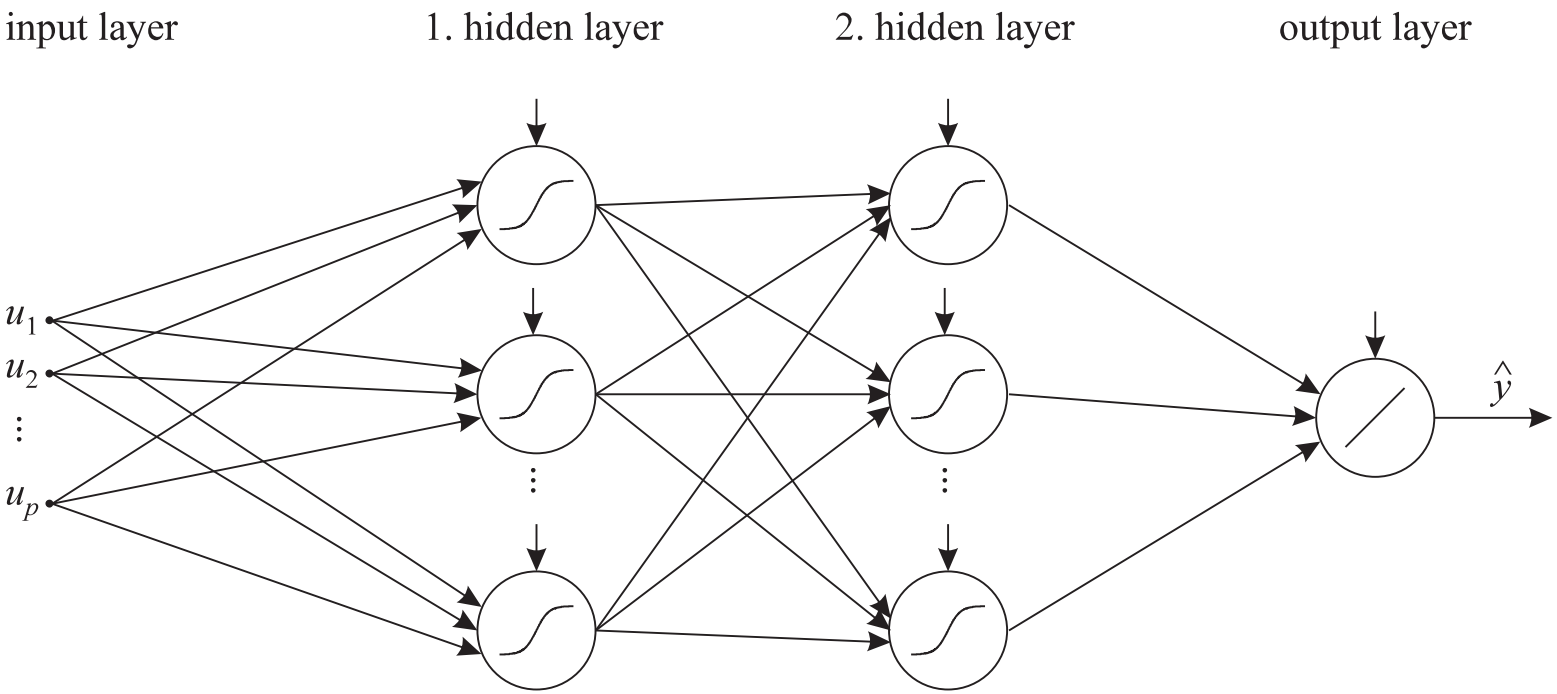


Figure 14: Multilayer perceptron with two hidden layers (weights are not written for more transparent presentation)

Multiple hidden layers networks

- The presentation of basis functions is more complex, since neurons at the second layer depend on the outputs of neurons on the first layer.
- If M_1 and M_2 are the numbers of neurons on the first and second hidden layers, w_i are weights at the output layer, $w_{jl}^{(1)}$ and $w_{ij}^{(2)}$ are the weights of the first and second hidden layers, then the output of the network is written with the basis functions as

$$\hat{y} = \sum_{i=0}^{M_2} w_i \Phi_i \left(\sum_{j=0}^{M_1} w_{ij}^{(2)} \xi_j \right) \qquad \text{and } \Phi_0(\cdot) = 1\,, \tag{18}$$

and with the output of neurons on the first layer

$$\xi_j = \Psi_j \left(\sum_{l=0}^{p} w_{jl}^{(1)} u_l \right) \qquad \text{at } \Psi_0(\cdot) = 1 \text{ and } u_0 = 1\,. \tag{19}$$

- Usually the activation functions of both hidden layers, Φ_i for the first layer, and Ψ_j for the second layer, are of the saturated type.
- The derivative of the network output with respect to the output weights is $(i = 0, \ldots, M)$ and is independent of the structure:

$$\frac{\partial \hat{y}}{\partial w_i} = \Phi_i \qquad \text{in } \Phi_0 = 1\,. \tag{20}$$

Multiple hidden layer networks

- Derivative with respect to the weights of the second hidden layer when choosing $\tanh(\cdot)$ activation function is ($i = 1, \ldots, M_2, j = 0, \ldots, M_1$):

$$\frac{\partial \hat{y}}{\partial w_{ij}^{(2)}} = w_i(1 - \Phi_i^2)\Psi_j \qquad \text{pri } \Psi_0 = 1\,. \tag{21}$$

- Derivative with respect to the weights of the first hidden layer when choosing $\tanh(\cdot)$ activation function ($j = 1, \ldots, M_1$, $l = 0, \ldots, p$) is

$$\frac{\partial \hat{y}}{\partial w_{jl}^{(1)}} = \sum_{i=0}^{M_2} w_i(1 - \Phi_i^2)w_{ij}^{(2)}(1 - \Psi_j^2)u_l \qquad \text{in } u_0 = 1 \tag{22}$$

- It is important to note that the value of the activation function Φ_i depends on the inputs and the weights at the hidden layer. We did not write this down due to the simplicity of the notation.
- The number of weights for the MLP network with two hidden layers is

$$M_1(p + 1) + M_2(M_1 + 1) + M_2 + 1 \tag{23}$$

- Due to the term with the product $M_1 M_2$, the number of weights grows quadratically with the number of neurons at the hidden layer.

Comparison between MLP, RBF and normalized RBF network

Table 2: Comparison between MLP, RBF and normalized RBF network

Properties	MLP	RBF	normalized RBF
Interpolation	+	−	+
Extrapolation	0	−	+
Locality	−	++	+
Accuracy	++	0	0
Smoothness	++	0	+
Noise sensitivity	++	+	+
Parameter optimization	−−	++*/−−**	++*/−−**
Structural optimization	−	+	−
Online adaptation	−−	+	+
Learning speed	−−	+*/−−**	+*/−−**
Model calculation speed	+	0	0
Dimension problem	++	−	−
Interpretation	−−	0	0
Restriction incorporation	−−	0	0
A priori knowledge incorporation	−−	0	0
Usage	++	0	−

* = linear optimization, ** = nonlinear optimization,
++ / −− = very positive / very negative.

Fuzzy models

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for lectures, April 2017

Table of contents

1. Fuzzy logic
   - Membership functions
   - Logical operators
   - Degree of rule fulfillment
   - Accumulation and defuzzification
2. Types of fuzzy systems
   - Linguistic fuzzy system
     - Fuzzy inference
   - Special fuzzy system
   - Takagi-Sugeno fuzzy system
3. Neuro-Fuzzy Networks
   - Fuzzy basis functions
   - Optimization of the fuzzy model
     - Optimization of consequent parameters
     - Optimization of premise parameters
     - Rule structure optimization
   - Interpretation of fuzzy models
   - Simulation example
4. Summary

Fuzzy logic

- Fuzzy logic was developed in 1965 as an extension of the classical (Boolean) logic.
- The classical logic assigns to a variable or a statement the value of 1 for "true" or the value of 0 for "false", fuzzy logic allows the value assignment at an interval between $[0,\ 1]$.
- The reason for this can be found in the observation of the way of human thinking when deciding on the very approximate estimates of various facts in the form of rules.

- A mechanism for recording knowledge based on rules in the form of *approximate reasoning* based on fuzzy logic was introduced to address such a concept.

- First, we introduce some basic concepts of fuzzy logic and approximative reasoning, which are necessary for the understanding of fuzzy models.

- Fuzzy logic records relationships, knowledge and decisions in the form of rules. In the example below, the influence of the wind on the feeling of cold is recorded, i.e. the human feeling of warmth depends not only on the temperature, but also on wind speed:

$$\text{IF } temperature = \text{low AND } wind = \text{strong}$$

Rule Elements:

IF - THEN	rules structure,
$temperature, wind, feeling$	linguistic variable,
low, strong, very cold	linguistic expressions or labels,
AND	operator, here: conjunction,
$temperature =$ low, $wind =$ strong	linguistic statement,
$temperature =$ low AND $wind =$ strong	premise, the causal part of the rule,
feeling = very cold	effect, consequence, consequent part of the rule

- The linguistic variables $temperature$ and $wind$ are often simply presented as input variables, and the linguistic variable $feeling$ as the output.
- In the *complete set of rules*, we write down all possible combinations for $temperature$ and $wind$:

$$
\begin{array}{llll}
R_1 : \text{IF } temp = \text{low} & \text{AND } wind = \text{strong} & \text{THEN } feeling = \text{very cold} \\
R_2 : \text{IF } temp = \text{medium} & \text{AND } wind = \text{strong} & \text{THEN } feeling = \text{cold} \\
R_3 : \text{IF } temp = \text{high} & \text{AND } wind = \text{strong} & \text{THEN } feeling = \text{moderate} \\
R_4 : \text{IF } temp = \text{low} & \text{AND } wind = \text{weak} & \text{THEN } feeling = \text{cold} \\
R_5 : \text{IF } temp = \text{medium} & \text{AND } wind = \text{weak} & \text{THEN } feeling = \text{moderate} \\
R_6 : \text{IF } temp = \text{high} & \text{AND } wind = \text{weak} & \text{THEN } feeling = \text{hot.}
\end{array}
$$

Membership functions

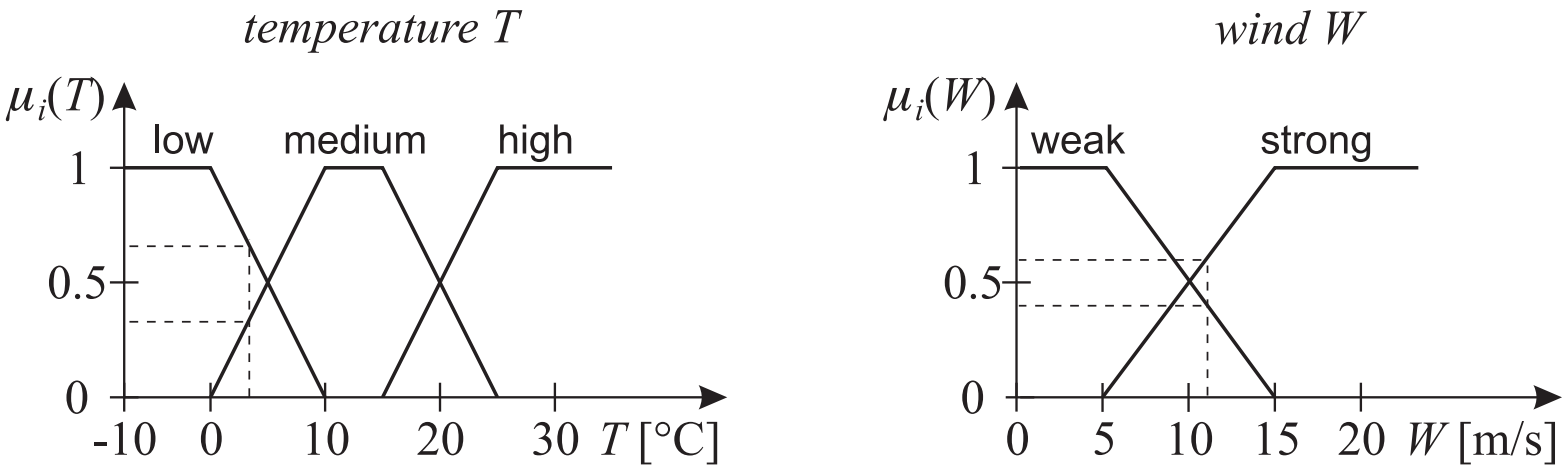


Figure 1: Membership functions for the linguistic variable *temperature T* (left) and *wind V* (right)

Membership functions

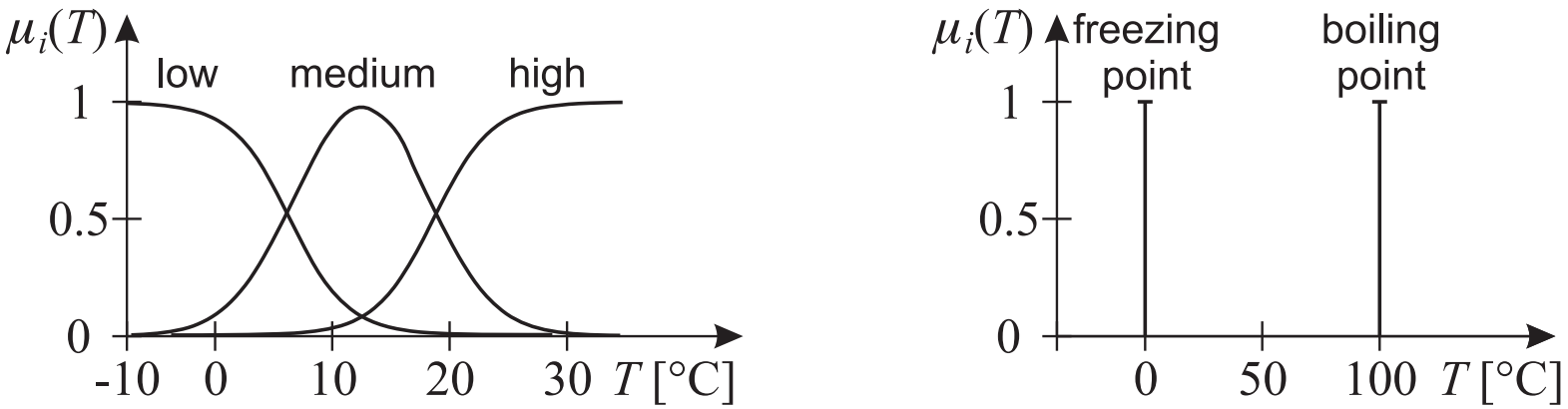


Figure 2: Membership functions can be smooth (left), which does not reduce information and is more appropriate for learning, or they can be special (singleton), which represents crisp sets (right)

Logical operators

- Fuzzy operators are extensions of classical logical operators.
- **Negation**
- The negation of the linguistic statement $T =$ low is calculated:

$$\mathrm{NOT}(\mu_i(T)) = 1 - \mu_i(T)\,. \tag{1}$$

- **Conjunction**
- For *conjunction* of two linguistic statements, such as $T =$ low in $V =$ strong, there are several different operators, also called *t-norms* . The most basic t-norms are (see figure 3):

$$\text{Min:} \quad \mu_i(T)\ \mathrm{AND}\ \mu_j(V) = \min\left(\mu_i(T), \mu_j(V)\right). \tag{2}$$

$$\text{Product:} \quad \mu_i(T)\ \mathrm{AND}\ \mu_j(V) = \mu_i(T)\mu_j(V). \tag{3}$$

$$\text{Bounded difference:} \quad \mu_i(T)\ \mathrm{AND}\ \mu_j(V) = \max\left(0, \mu_i(T) + \mu_j(V) - 1\right). \tag{4}$$

Membership functions

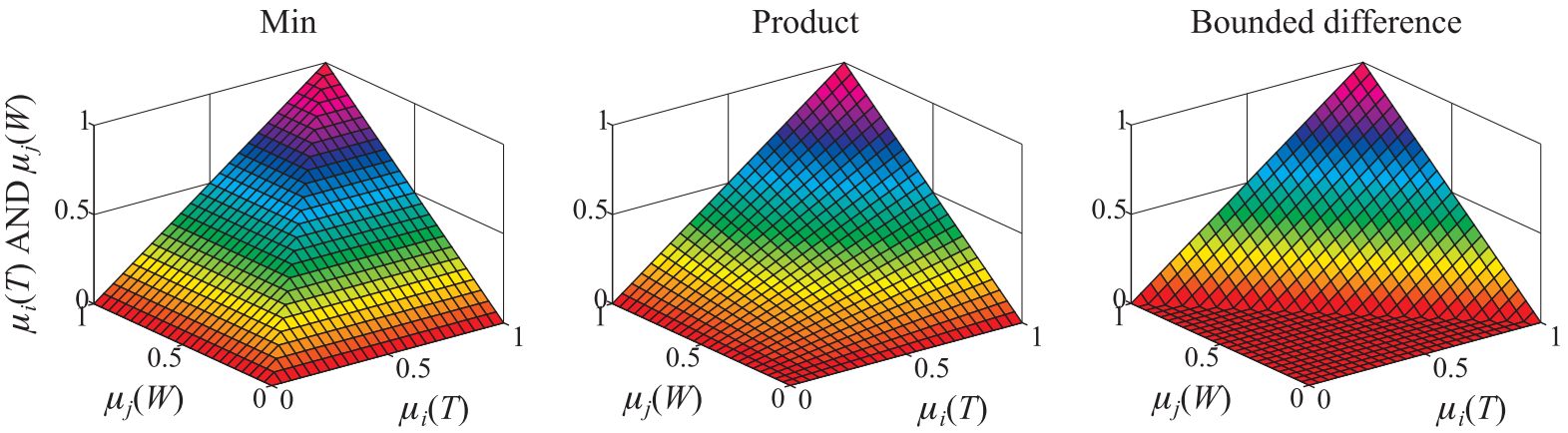


Figure 3: A display of different t-norms (conjunction operators)

Membership functions

- **Disjunction**
- For *disjunction* of two linguistic variables, there are also several different operators, also called *t-conorms or s-norms* .
- The most basic t-conorms are:

$$\text{Max:} \quad \mu_i(T) \text{ OR } \mu_j(V) = \max\left(\mu_i(T), \mu_j(V)\right). \tag{5}$$

$$\text{Algebraic sum:} \quad \mu_i(T) \text{ OR } \mu_j(V) = \mu_i(T) + \mu_j(V) - \mu_i(T)\mu_j(V). \tag{6}$$

$$\text{Bounded sum:} \quad \mu_i(T) \text{ OR } \mu_j(V) = \min\left(1, \mu_i(T) + \mu_j(V)\right). \tag{7}$$

- For the purposes of classification, a selection of min and max operators is very common.
- Product and algebraic product are more suitable for approximation problems due to their smoothness and differentiability.

Degree of rule fulfillment

- With logical operators, we can construct individual linguistic expressions into linguistic statements and draw up the premise of the rule.
- For example, $temperature$ can be $T = 3°$C and $wind$ $V = 11$ m / s. This corresponds to the following membership values : $\mu_{\text{low}}(T) = 0.7$, $\mu_{\text{medium}}(T) = 0.3$, $\mu_{\text{high}}(T) = 0$, $\mu_{\text{weak}}(V) = 0.4$ in $\mu_{\text{strong}}(V) = 0.6$; see figure 1.
- The R_1 rule calculates the conjunction between $\mu_{\text{low}}(T) = 0.7$ and $\mu_{\text{strong}}(V) = 0.6$. This results in a value of 0.6 if the operator min is used, 0.42 for the operator of the product, and 0.3 for the operator of the bounded difference.
- The output of the fuzzy system is in that way also dependent on the choice of operators.
- The combination of the affiliation of all linguistic expressions determines *degree of rule fulfillment* or *rule firing strength* because it expresses how well the premise matches the given values of input variables (in this case $T = 3°$C in $V = 11$ m/s).
- For the entire fuzzy system, only fulfillment degrees greater than zero are important.
- For strictly local MSFs, such as triangular ones in figure 1, only part of the rules for a given input is activated. For $T = 3°$ C and $V = 11$ m / s rules R_3 and R_6 have zero activation, since $\mu_{\text{high}}(T) = 0$.
- We need to ensure that the rules complete the entire possible input space to avoid situations where no rule gets activated for certain input values.
- In the case of non-explicit local affiliation functions, this problem does not exist because *all* rules are always fulfilled, although with very small values.

Accumulation and defuzzification

- After the degree of fulfillment of an individual rule is calculated, we have to determine the contributions of individual consequent parts and assemble them to obtain the output of the fuzzy system. This is called *accumulation*.
- Usually, the output of the fuzzy system is a fuzzy set that needs to be transformed into a crisp form for further work. This is called the process of *defuzzification*.
- This is of course not necessary in case we have a crisp value on the consequent part, or if the result serves qualitative estimates.

Types of fuzzy systems

Three basic types of fuzzy systems:

- linguistic or Mamdani
- special fuzzy system or singleton
- Takagi-Sugeno Fuzzy System.

Linguistic or also Mamdani fuzzy system

- notation of a fuzzy system rule in the form

$$R_i: \quad \text{IF } u_1 = \mathsf{A}_{i1} \text{ AND } u_2 = \mathsf{A}_{i2} \text{ AND } \ldots u_p = \mathsf{A}_{ip} \text{ THEN } y = \mathsf{B}_i\,,$$

- where $u_1, \ldots, u_p$ are inputs into a fuzzy system that are written into the input vector $\underline{u}$
- y is the output, index $i = 1, \ldots, M$ goes over all M rules of the fuzzy system
- A_{ij} is a fuzzy set for the variable u_j in the rule i and B_i is a fuzzy set of the output variable for the i rule. In case of processing of such a system, it is necessary to:

Fuzzy inference

fuzzification $\rightarrow$ aggregation $\rightarrow$ activation $\rightarrow$ accumulation $\rightarrow$ defuzzification

- With **fuzzification** based on MSFs we map the specified exact value of the variable into membership.
- During the *fuzzification* step the degree of membership for all linguistic statements is calculated. We write them with $\mu_{ij}(u_j)$, $i = 1, \ldots, M$, $j = 1, \ldots, p$.

Fuzzy inference

- By **aggregation** we compose individual linguistic statements into the level of activation of the rule with respect to the operators between them.
- These steps are the same for all types of fuzzy systems. The last two steps depend on the type of the fuzzy system.
- In *aggregation* step, these levels of affiliation within one rule are made according to the required operators. If the fuzzy system is written in a conjunction form and we use the operator of the product for the t-norm, the degree of activation of the i rule is equal to

$$\mu_i(\underline{u}) = \mu_{i1}(u_1) \cdot \mu_{i2}(u_2) \cdot \ldots \cdot \mu_{ip}(u_p) \,. \tag{8}$$

- In **activation** step , the levels of activation of the rules are passed to the output fuzzy set. For example, we can do this by cutting off the corresponding membership function, which is the output of this rule, at a value defined by the degree of rule fulfillment

$$\mu_i^{\mathrm{akt}}(\underline{u}, y) = \min\left(\mu_i(\underline{u}), \mu_i(y)\right) \,, \tag{9}$$

- where $\mu_i(y)$ is the output membership function of the corresponding fuzzy set B_i and $\mu_i(\underline{u})$ is the degree of fulfillment of the rule R_i.
- Figure 4 displays the MSFs and the output fulfillment of the rule R_1 from the example, where the fulfillment of the first rule is equal to 0.42.

Fuzzy inference

- In the **accumulation** we compose all the contributions of the output activations. We can do this by calculating the maximum of all output activations

$$\mu^{\mathrm{aku}}(\underline{u}, y) = \max_i [\mu_i^{\mathrm{akt}}(\underline{u}, y)] \,. \tag{10}$$

- Accumulation results in one fuzzy set, which is the fuzzy output of the system. This can be the final result if we do not need crisp values.
- If a crisp value is required for the output of a fuzzy system, then we need to execute the process of **defuzzification**. The most common way for this step is to use the *center of gravity* method, which is carried out as follows,

$$\hat{y} = \frac{\int_{y_{\min}}^{y_{\max}} y \, \mu^{\mathrm{aku}}(\underline{u}, y) \, dy}{\int_{y_{\min}}^{y_{\max}} \mu^{\mathrm{aku}}(\underline{u}, y) \, dy} \,, \tag{11}$$

where $\mu^{\mathrm{aku}}(\underline{u}, y)$ is a fuzzy set, i.e. the accumulation of output activations.

Fuzzy inference

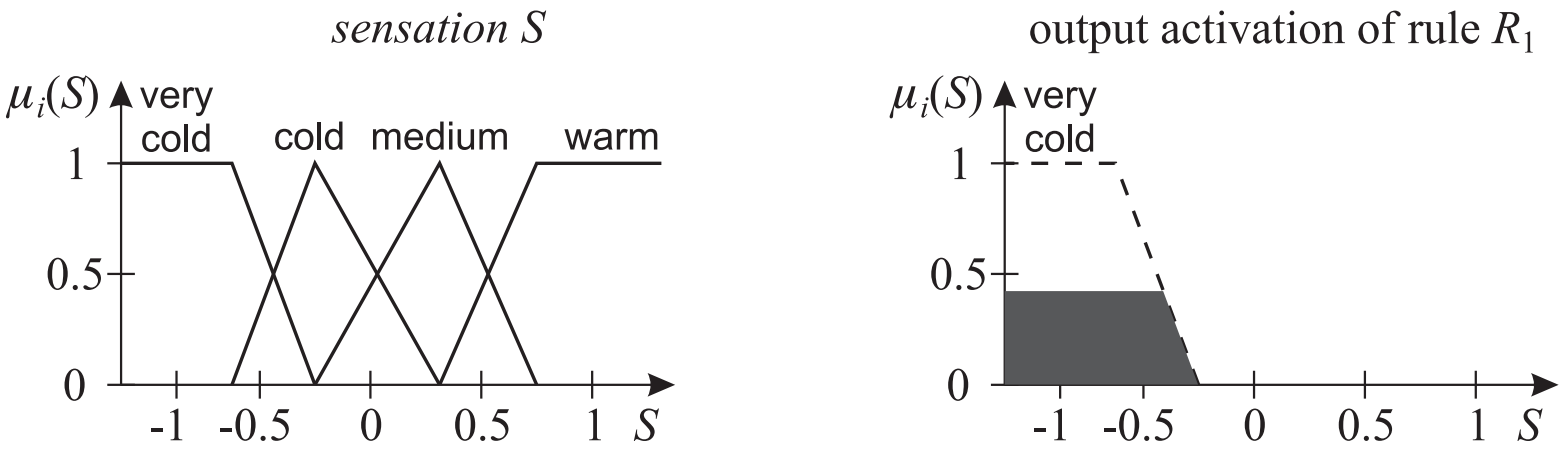


Figure 4: Membership functions for the linguistic variable *feeling* (left) and the calculation of the output activation (right). The black area represents the output activation for the rule R_1 and is obtained by the cut of the output MSF at the rule fulfillment degree of 0.42

Fuzzy inference

Fuzzy inference method with *min* and *max* operators for conjunction and disjunction, with the *max* operator for accumulation and the center of gravity method for defuzzification.

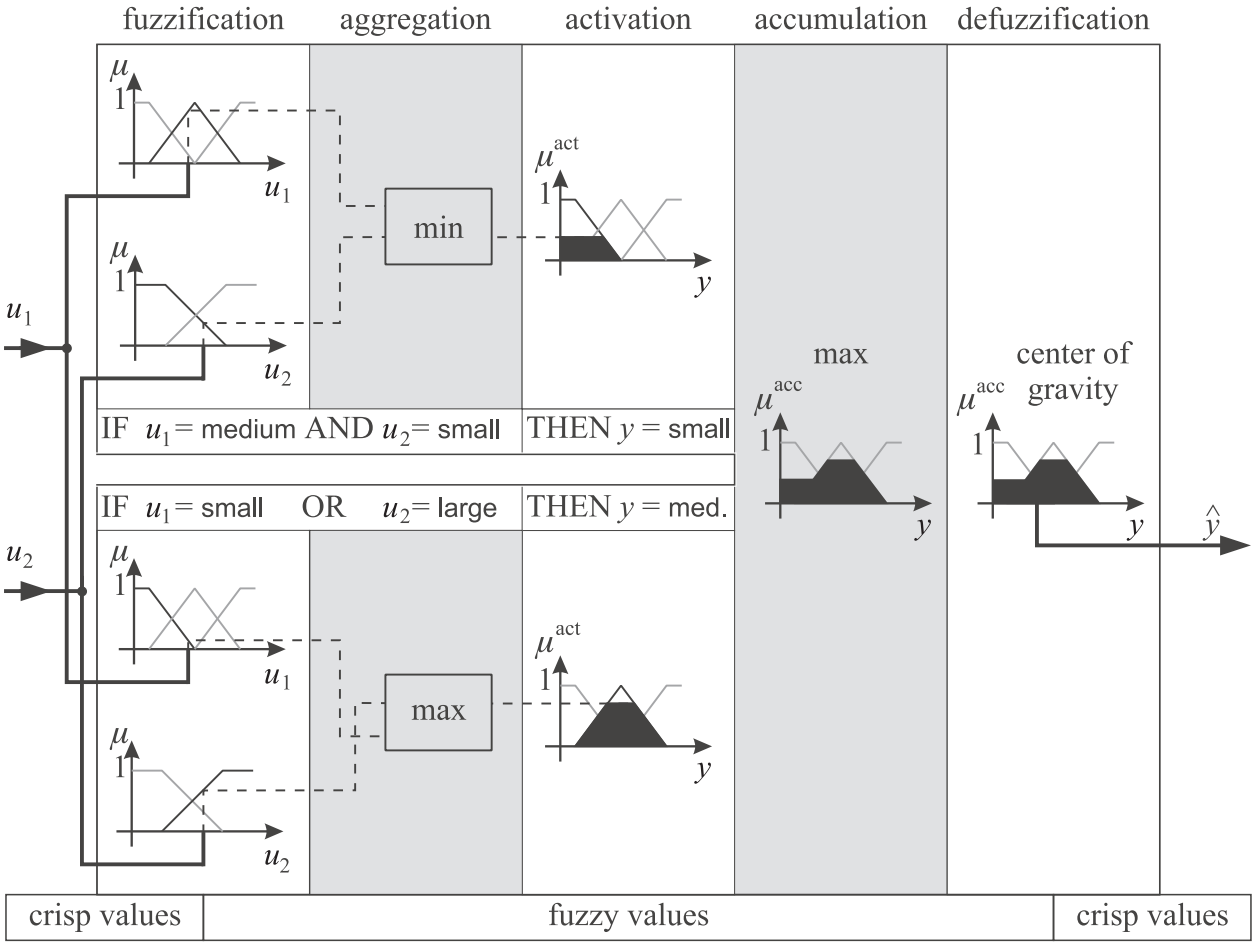


Figure 5: Inference in the case of a fuzzy system with two inputs and two rules

Special fuzzy system

Special fuzzy system

- A special fuzzy system eliminates and simplifies calculations at the defuzzification step.
- The rule of a special fuzzy system has the following form:

$$R_i: \quad \text{IF } u_1 = \mathsf{A}_{i1} \text{ AND } u_2 = \mathsf{A}_{i2} \text{ AND } \ldots u_p = \mathsf{A}_{ip} \text{ THEN } y = s_i \,,$$

- where s_i is the real value on the consequent part of the rule and is called the *singleton* of rule i.
- The output of a fuzzy system is defined as:

The output of a special fuzzy model

$$\hat{y} = \frac{\sum_{i=1}^{M} s_i \, \mu_i(\underline{u})}{\sum_{i=1}^{M} \mu_i(\underline{u})} \,, \tag{12}$$

- where M is the number of rules and $\mu_i(\underline{u})$ is the degree of fulfillment of rule R_i with regard to equation 8.
- A fuzzy output is the weighted sum of individual singletons, where the weights are equal to the degrees of the rule fulfillment.
- The normalization denominator in equation 12 is present in all forms of fuzzy systems and provides the right interpretation of the consequent parts of the rules.

Special fuzzy system

- A set of rules is complete if each combination of input linguistic labels is written with one rule.
- The rule for a special fuzzy system simplifies

$$\hat{y} = \sum_{i=1}^{M} s_i \, \mu_i(\underline{u}) \, . \tag{13}$$

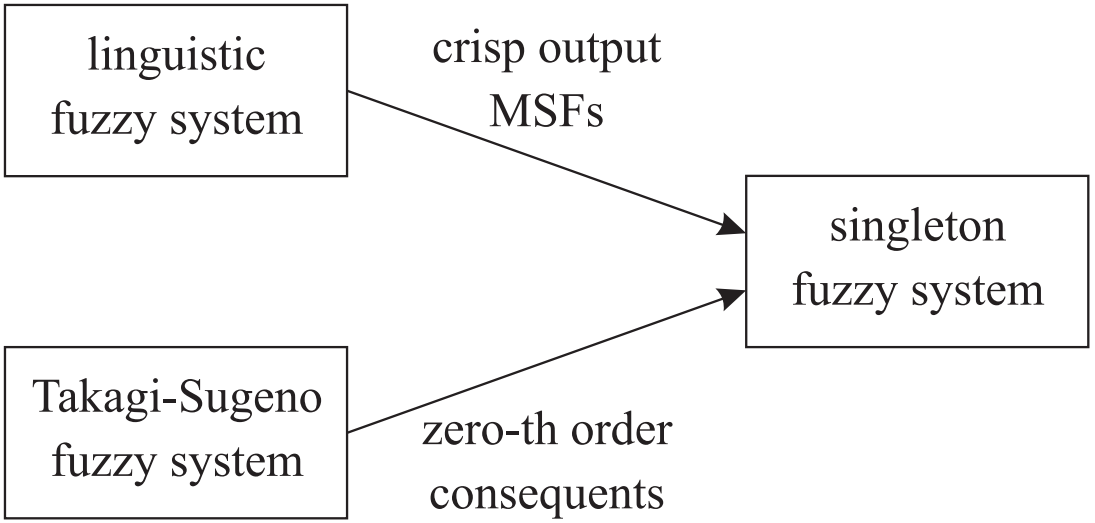


Figure 6: Singleton fuzzy system as a special example of a linguistic or Takagi-Sugeno fuzzy system

Takagi-Sugeno fuzzy system

- In 1985, Takagi and Sugeno proposed a new type of fuzzy system with the following rule notation:

$$R_i: \quad \text{IF } u_1 = \mathsf{A}_{i1} \text{ AND } \ldots \text{ AND } u_p = \mathsf{A}_{ip} \text{ THEN } y = f_i(u_1, u_2, \ldots, u_p)\,.$$

- This notation is the extension of special fuzzy systems.

Output of the fuzzy model

$$\hat{y} = \frac{\sum_{i=1}^{M} f_i(\underline{u})\, \mu_i(\underline{u})}{\sum_{i=1}^{M} \mu_i(\underline{u})}\,, \tag{14}$$

which is a simple extension for the output of a special fuzzy system in equation 12. Figure 7 shows the calculation for a one-dimensional Takagi-Sugeno fuzzy system with two rules:

$$\begin{aligned} R_1 : &\qquad \text{IF } u = \text{small} \;\; \text{THEN } y = u \\ R_2 : &\qquad \text{IF } u = \text{big} \;\; \text{THEN } y = 2u - 0.2\,. \end{aligned}$$

Takagi-Sugeno fuzzy system

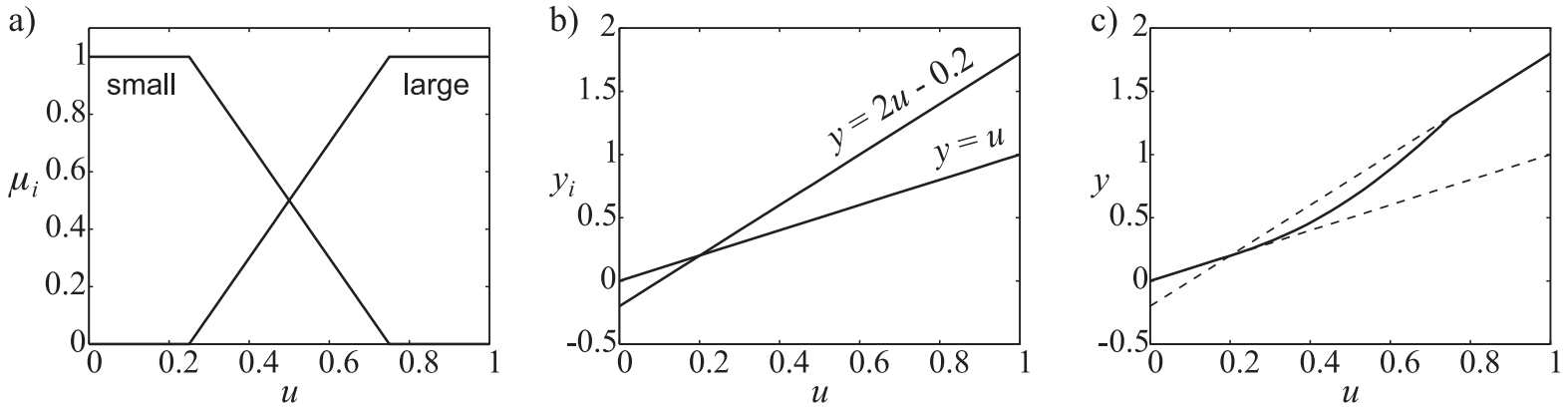


Figure 7: One-dimensional Takagi-Sugeno fuzzy system with two rules: a) membership functions, b) linear functions of both consequent parts, c) output of the fuzzy system (full line) and linear functions (dashed line). Note that the denominator in equation 14 is equal to 1 because MSFs form a unified space distribution

- All that applies to special fuzzy systems can be extended to Takagi-Sugeno fuzzy systems by replacing s_i with the function $w_{i0} + w_{i1}u_1 + w_{i2}u_2 + \ldots + w_{ip}u_p$.
- Takagi-Sugeno fuzzy models have been widely used and are highly researched mainly due to their excellent interpretability.

Fuzzy basis functions

- A special fuzzy model and a TS fuzzy model can be written in the basis formulation.
- If the fuzzy model is written in a conjunctive form and the product is used for the t-norm, then the degree of fulfillment of the rule is

$$\mu_i(\underline{u}) = \prod_{j=1}^{p} \mu_{ij}(u_j) \,. \tag{15}$$

- The function $\mu_i(\underline{u})$ represents a *multi-dimensional* membership function that is composed of several one-dimensional membership functions $\mu_{ij}(u_j)$ with a product structure.
- Figure 8 shows how one-dimensional triangular and Gaussian membership functions can be assembled in a two-dimensional MSF with the operator of the product and the minimum.
- As shown in figure 8 d, the multi-dimensional MSF can be smooth only when all one-dimensional MSFs are smooth and the smooth operator is used.
- The fuzzy models constructed in this way have rules that are a conjunction of all possible combinations of one-dimensional membership functions. In this way, such a fuzzy model is very subject to the problem of a large number of dimensions.
- The interpretation of a fuzzy model with multi-dimensional basis functions is as follows

$$\hat{y} = \sum_{i=1}^{M} s_i \Phi_i(\underline{u}) \qquad \text{for } \Phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum_{j=1}^{M} \mu_j(\underline{u})} \,. \tag{16}$$

Fuzzy basis functions

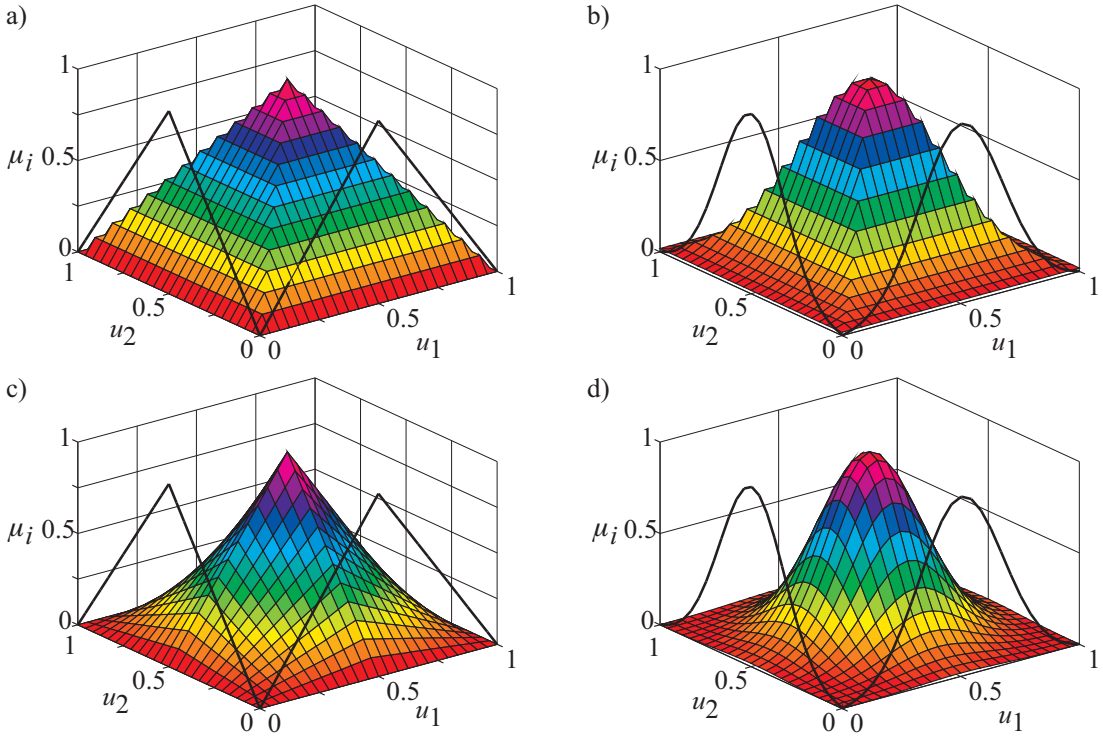


Figure 8: The construction of a multi-dimensional membership function: a) triangular MSFs with operator min, b) Gaussian MSFs with the operator min, c) triangular MSFs with the product operator, d) Gaussian with the product operator

Optimization of the fuzzy model

- In fuzzy models we have to optimize a large number of different parameters.
- We optimize nonlinear parameters in the premise and the linear parameters (usually) in consequent part of the rules
- Structure optimization

Optimization of consequent parameters

- Consequent parameters of the fuzzy model correspond to the output weights in the model formulation with the basis functions.
- Special fuzzy models and TS models have linear output functions $f_i(\cdot)$ and these parameters are linear.
- Because of this, they can be easily estimated using the least squares method.
- We often optimize the parameters so that the premise parameters (parameters at the hidden level in the neural networks) are constant and equidistant in dividing the input variables, and only the parameters in the output part of the rule are optimized.

Optimization of premise parameters

- Parameters in the premise of rules are those that define membership functions, their positions and width.
- They correspond to parameters on a hidden level of neural networks and are nonlinear.
- Because of this, we try to avoid their optimization, if possible, and optimize only linear parameters in the consequent part.
- The question arises when it makes sense to use a nonlinear local or global optimization technique to optimize nonlinear parameters.
- One of the methods often encountered in such cases is the GA method by which we can optimize the parameters in a premise. It is reasonable to use it when the knowledge about the process is really small or non existent.
- The NLS method is also very successful.

Rule structure optimization

- Optimizing the rule structure is very important because in this way we can reduce the dimension problem.
- Optimizing the rule structure is a combination problem that can be solved using a choice of linear subsets or using nonlinear global optimization, for example GA.

Interpretation of fuzzy models

- The main difference between fuzzy systems and other nonlinear approximators is in the possibility of interpretation based on rules that are very understandable to the way of human thinking.
- Therefore, this is one of the most important aspects in assessing the quality of the fuzzy model.
- Interpretability of a fuzzy system can be affected by:
  - *The number of rules:* If the number of rules is too high, the interpretability is questionable. In the case of a large number of input variables this is usually true, because we try to complete all possible combinations on the input side of the rule. This can only partially be remedied with only a small granulation of inputs, at the expense of smaller complexity.
  - *The number of linguistic statements in the rules premise:* If the rules contain several linguistic statements, they are difficult to understand. The ceiling is about three.
  - *Normalization of input membership functions (partition units):*

Interpretation of fuzzy models

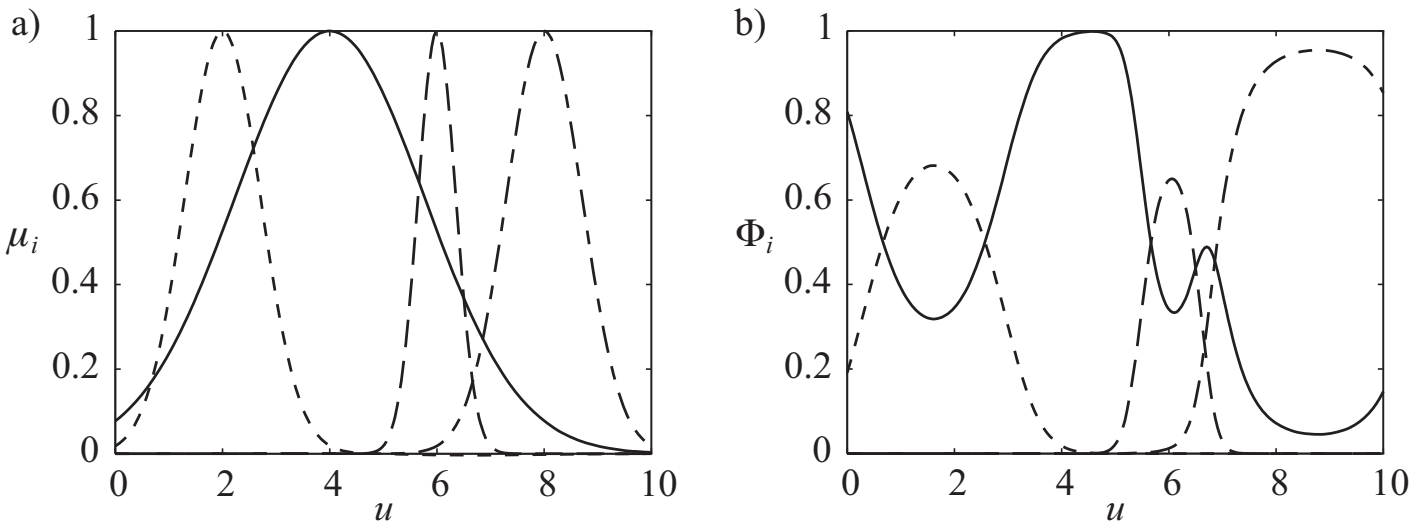
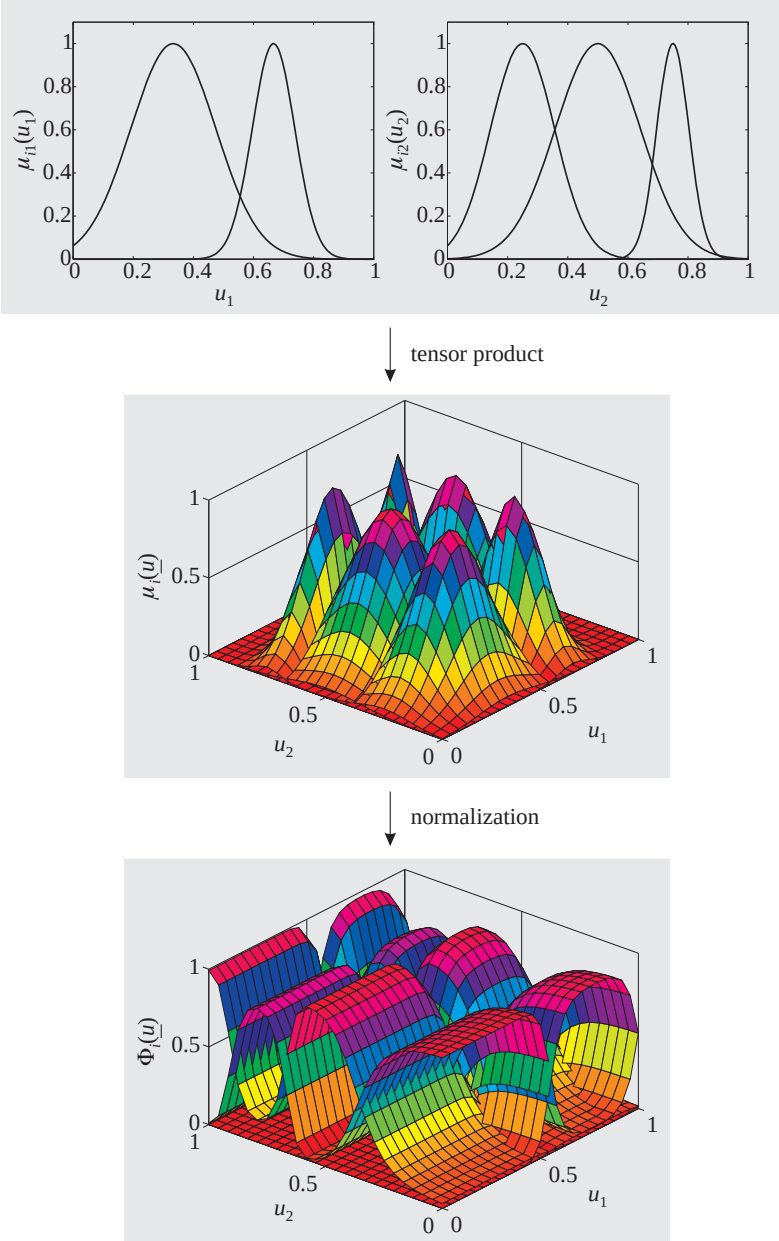


Figure 9: a) Gaussian membership functions with different standard deviations. b) A normalized Gaussian membership function that has a unit partition with the sum of 1. The second membership function (full line) has a higher standard deviation and therefore becomes dominant for $u \to -\infty$ and $u \to \infty$. This makes it nonlocal. The rule that contains it has a dominant influence not only around its center $u = 4$ but also around $u = 0$. This is an undesirable influence of the MSFs normalization. The same effect can also be detected if we use the non-normalized functions (a)

Interpretation of fuzzy models



Interpretation of fuzzy models

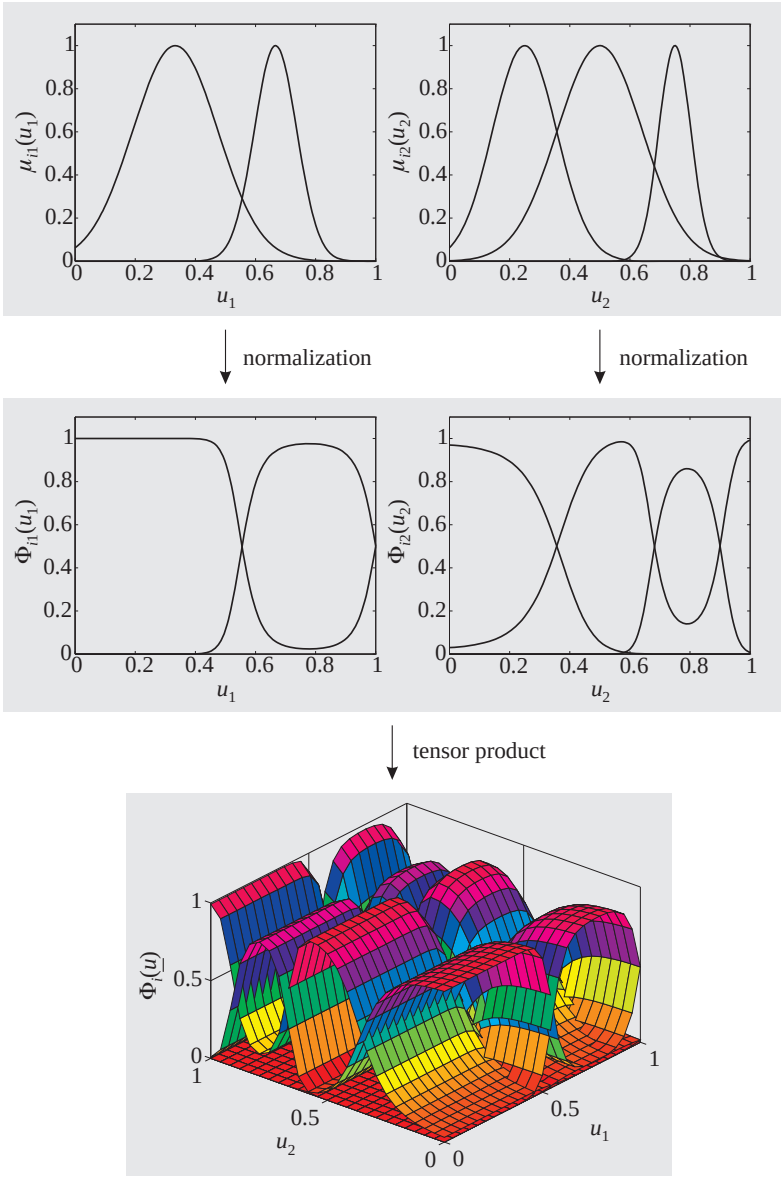


Figure 11: The normalization side effect also occurs when single-dimensional MSFs are normalized to obtain normalized one-dimensional MSFs. This effect occurs when a tensor product is formed. The result is the same

Interpretation of fuzzy models

- Figure 12 shows an unwanted effect of the reactivation of a membership function that appears when there are functions of different standard deviations.
- If they have the same width, this can not happen. It occurs outside the input learning domain and is undesirable only in case of extrapolation.

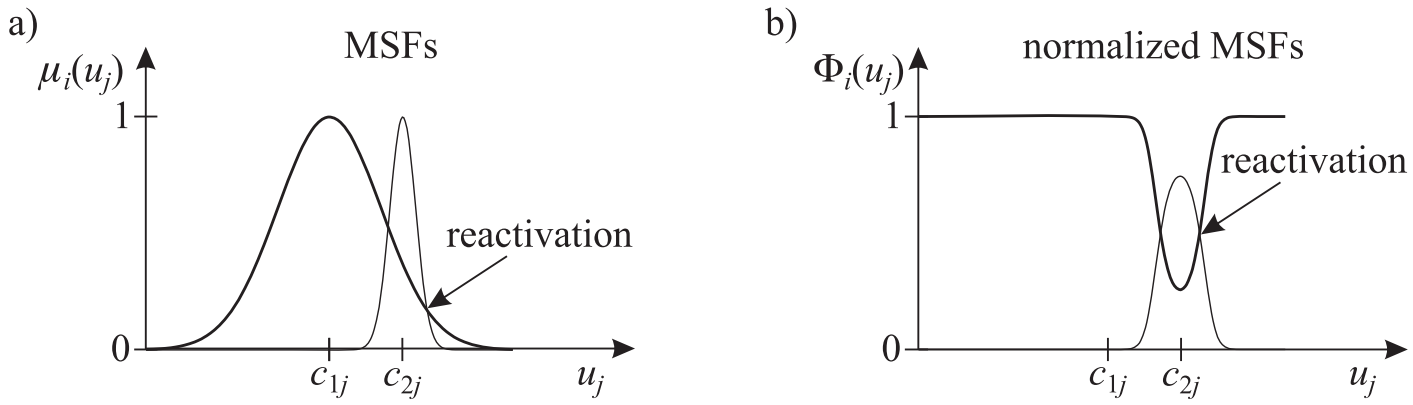


Figure 12: Reactivation of the basis function occurs if we have: a) Gaussian MSFs of different widths, b) normalized MSFs

- The above discussion presents certain limits for fuzzy models, which is the price for excellent interpretability obtained with fuzzy models.

Simulation Example

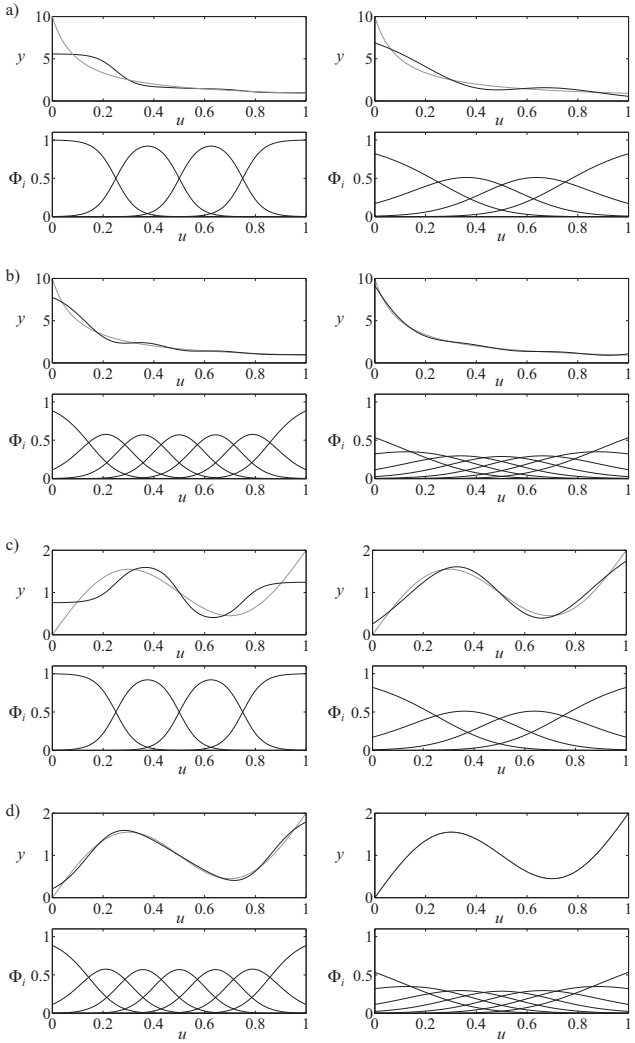


Figure 13: The approximation of the function $y = 1/(u + 0.1)$ (a, b) and $y = \sin(2\pi u) + 2u$ (c, d) with a neuro-fuzzy model with four (a, c) or seven (b, d) rules. The standard deviations of the membership functions are before the normalization equal to 0.1 (left) and 0.2 (right).

Comparison between Mamdani, a special fuzzy and Takagi-Sugeno fuzzy models

Table 1: Comparison between Mamdani, a special fuzzy and Takagi-Sugeno fuzzy models

Features	Mamdani	Special	Takagi-Sugeno
Interpolation	0	+	0
Extrapolation	+	+	++
Locality	0	+	+
Accuracy	-	0	++
Smoothness	+	+	0
Sensitivity to noise	+	+	++
Optimization of parameters	$-$	$++^{*}/--^{**}$	$++^{*}/--^{**}$
Structural optimization	$--$	0	++
Real-time adaptation	$--$	+	++
Learning speed	$-$	$+^{*}/--^{**}$	$++^{*}/--^{**}$
Speed calculation model	$--$	$-$	$-$
Dimension problem	$-$	$-$	0
Interpretation	$++$	+	0
Inclusion of limits	0	0	0
Use of a priori knowledge	+	0	0
Usage	+	++	0

* = linear optimization, ** = nonlinear optimization,
$++ / --$ = very positive / very negative.

Local linear neuro-fuzzy models

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for the 12th lecture, April 2016

Table of contents

1. Local linear neuro-fuzzy models
2. Basic idea
   - Display of local linear neuro-fuzzy models
   - Interpretation of an affine term in LLM
   - Interpretation in the form of Takagi-Sugeno model
3. Optimization of local linear model parameters
   - Global estimation
   - Local estimation
   - Comparison between global and local estimations
4. Structural optimization of the rule premise
   - Local linear model tree algorithm (LOLIMOT)

Local linear neuro-fuzzy models

- Presentation of local linear neuro-fuzzy models or Takagi-Sugeno fuzzy models.
- Algorithms for data based identification of parameters of fuzzy models are also presented.
- In their construction the main problem is the partitioning of the space of input-output data (related algorithms for partitioning of the space).
- Once we divide the space into the corresponding subspaces, we get the structure of the premise of fuzzy rules.
- The estimation of the model parameters in the consequent part is usually quite simple, since the consequent part is by definition linear in the parameters.

Basic idea

Network structure of local linear neuro-fuzzy models

- Each neuron realizes one *local linear model (LLM)* and the associated *validity function*, which defines the area of validity for LLM.
- According to the constant term the outputs of LLMs are local affine models and not linear. Nevertheless, they are called linear in literature.
- The function in the consequent is the following

$$\hat{y}_i = w_{i0} + w_{i1}u_1 + w_{i2}u_2 + \ldots + w_{ip}u_p \,, \tag{1}$$

where w_{ij} represents the LLM parameters for the $i-$th local model or neuron.

Basic idea

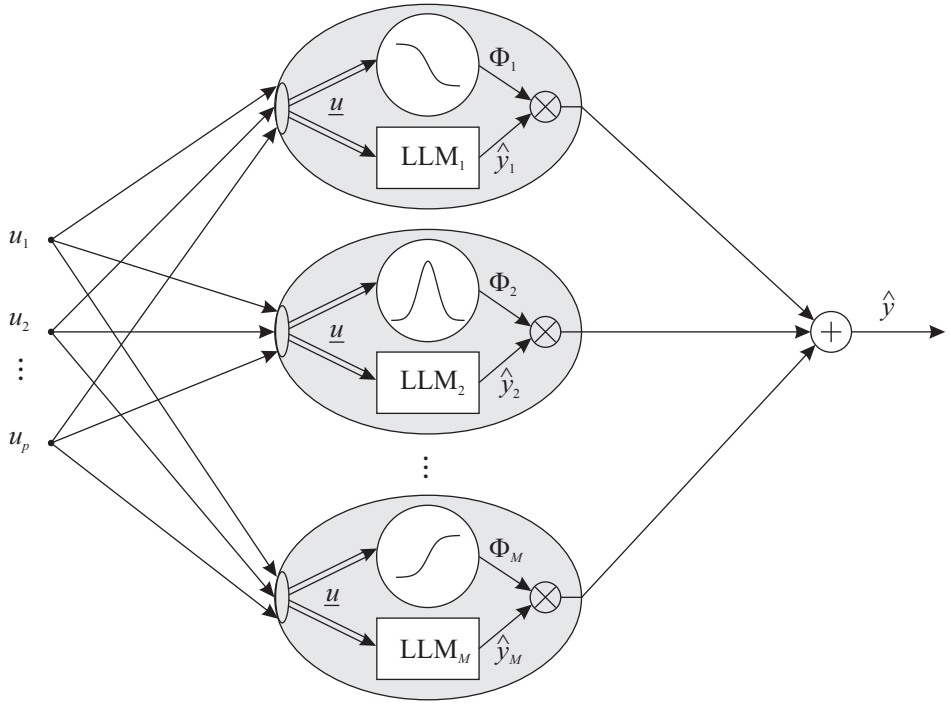


Figure 1: Structure of network for a static local linear neuro-fuzzy model with M neurons and p inputs

Basic idea

- Validity functions form a unit partition, i.e. they are normalized so that

$$\sum_{i=1}^{M} \Phi_i(\underline{u}) = 1 \tag{2}$$

  for any input vector $\underline{u} = [u_1 \; u_2 \; \cdots \; u_p]^{\mathrm{T}}$.
- This property is important for the correct interpretation of the validity functions $\Phi_i(\cdot)$, which ensures that the total contribution of local linear models equals 1.
- The output of LLM is equal to

$$\hat{y} = \sum_{i=1}^{M} \underbrace{(w_{i0} + w_{i1}u_1 + w_{i2}u_2 + \ldots + w_{ip}u_p)}_{\hat{y}_i} \Phi_i(\underline{u}) \,. \tag{3}$$

Output of the model

- The network's output is calculated as the weighted sum of the outputs of local linear models, where $\Phi_i(\cdot)$ are interpreted as weights that depend on the operating point.
- The network interpolates between different LLMs based on the weights of validity functions.
- Weights w_{ij} are linear parameters of the network.
- Validity functions are usually normalized Gaussian functions and if they are axially orthogonal, then the validity functions are equal to

$$\Phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum_{j=1}^{M} \mu_j(\underline{u})} \tag{4}$$

with

$$\begin{aligned} \mu_i(\underline{u}) &= \exp\left(-\frac{1}{2}\left(\frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2} + \ldots + \frac{(u_p - c_{ip})^2}{\sigma_{ip}^2}\right)\right) \\ &= \exp\left(-\frac{1}{2}\frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2}\right) \cdot \ldots \cdot \exp\left(-\frac{1}{2}\frac{(u_p - c_{ip})^2}{\sigma_{ip}^2}\right). \end{aligned} \tag{5}$$

Basic idea

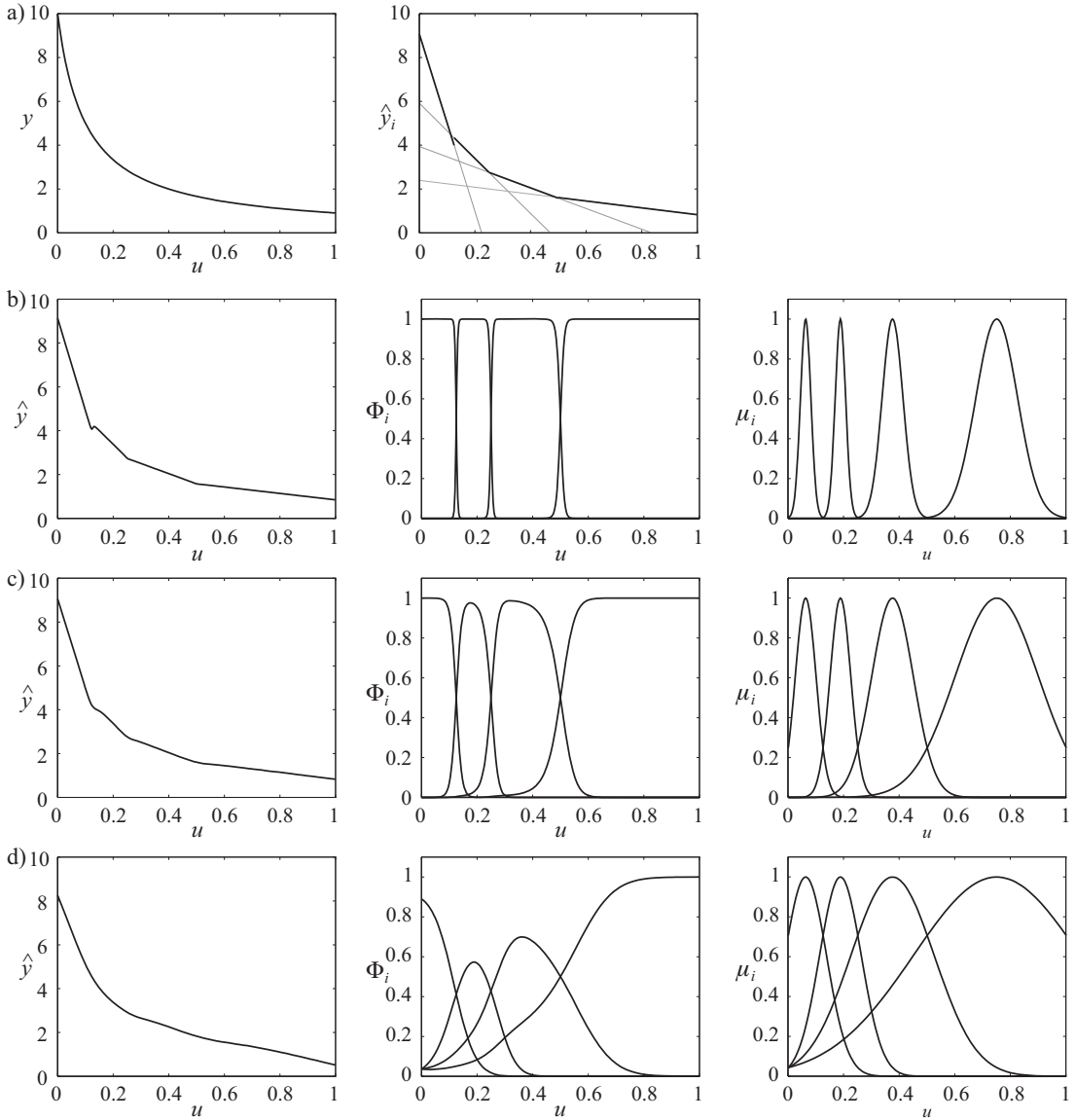


Figure 2: Approximation with local linear models: a) approximated function (left) and local linear model (right), b) small, c) medium, d) large standard deviation of the validity function Φ_i

Display of local linear neuro-fuzzy models

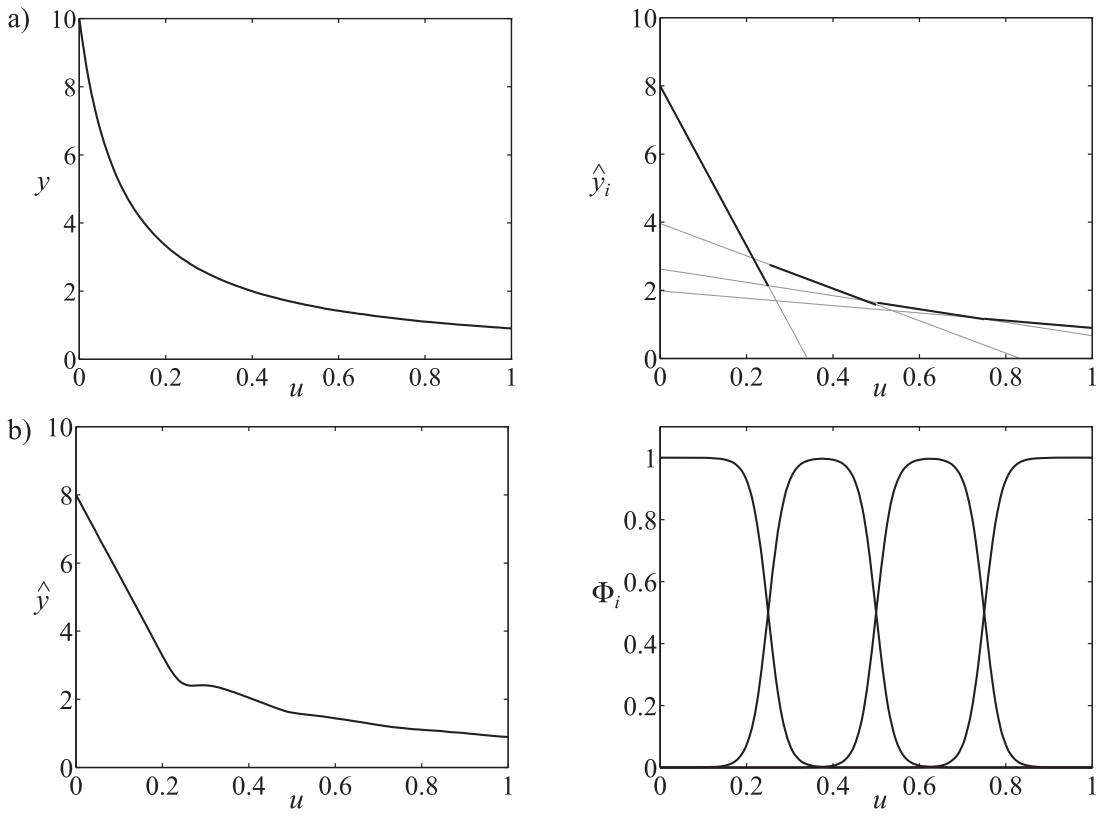


Figure 3: Local linear model such as in figure 2c, but with equidistant validity function

Display of local linear neuro-fuzzy models

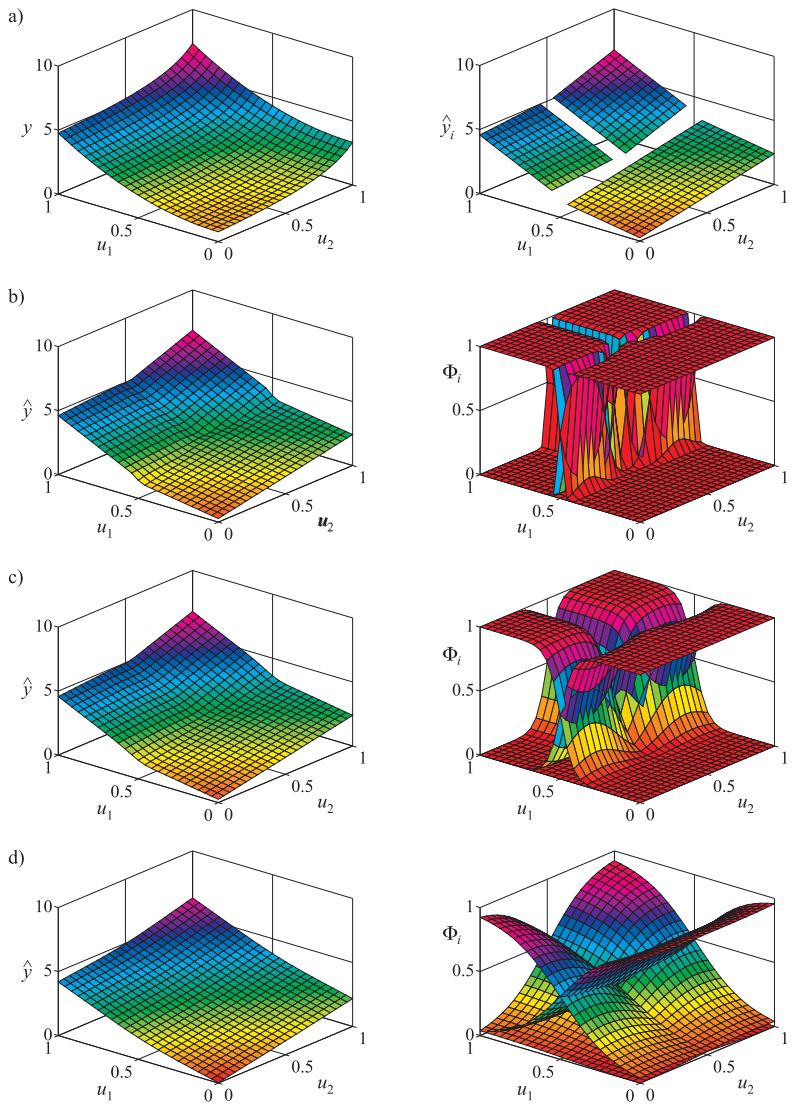


Figure 4: Effect of local linear models: a) approximated function (left) and local linear model (right), b) small, c) medium, d) large standard deviation of the validity functions Φ_i

Interpretation of an affine term in LLM

Affine term

- Article w_{i0} in equation 3 is independent of the input variables and determines the operating point or a DC component at the measurements in dynamic systems.
- For the interpretation, we write down the next form of the output of the LLM model

$$\hat{y} = \sum_{i=1}^{M} \left(\widetilde{w}_{i0} + w_{i1}(u_1 - c_{i1}) + \ldots + w_{ip}(u_p - c_{ip})\right) \Phi_i(\underline{u}) \tag{6}$$

where we have transformed the affine term into $\widetilde{w}_{i0}$. Figure 5 shows the interpretation of this term

$$\widetilde{w}_{i0} = \hat{y}_i(\underline{u} = \underline{c}_i) \tag{7}$$

where $\hat{y}_i$ is the output of i-th local linear model. The relation between $\widetilde{w}_{i0}$ in equation (6) and w_{i0} in equation (3) is equal to:

$$\widetilde{w}_{i0} = w_{i0} + w_{i1}c_{i1} + \ldots + w_{ip}c_{ip} \,. \tag{8}$$

- Both formulations in equations 3 and 6 are equivalent, but for the estimation of the parameters equation 6 is much more numerically robust due to better conditionality.

Interpretation in the form of Takagi-Sugeno model

Takagi-Sugeno fuzzy model

- Equivalence with Takagi-Sugeno fuzzy model.
- Membership functions have to be Gaussian and for the t-norm we have to choose the product.
- In addition, the validity functions must be axially orthogonal to permit a complete projection to the input axes.
- Figure 6 shows a Takagi-Sugeno fuzzy model with three rules:

$$
\begin{array}{lll}
R_1: & \text{IF } u_1 = \text{small} & \text{THEN } y = w_{10} + w_{11}u_1 + w_{12}u_2 \\
R_2: & \text{IF } u_1 = \text{large AND } u_2 = \text{small} & \text{THEN } y = w_{20} + w_{21}u_1 + w_{22}u_2 \\
R_3: & \text{IF } u_1 = \text{large AND } u_2 = \text{large} & \text{THEN } y = w_{30} + w_{31}u_1 + w_{32}u_2.
\end{array}
$$

- The collection of the rules covers the entire input space, although it is not complete, i.e. it does not contain all possible combinations of membership functions (small/small, small/large, large/small, large/large).

Interpretation in the form of Takagi-Sugeno model

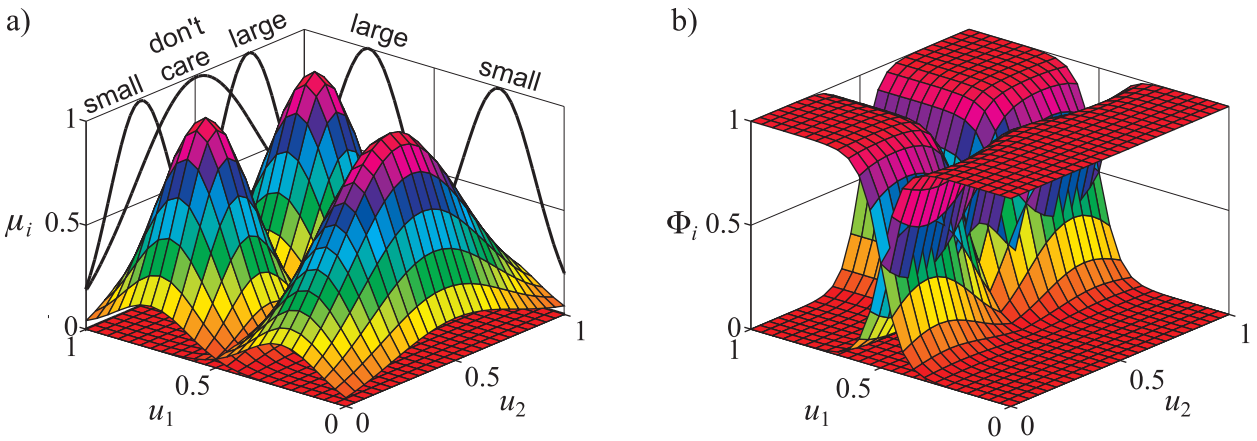


Figure 5: a) with a t-norm (operator of a product) of a one-dimensional Gaussian MSFs μ_{ij} we construct a multi-dimensional MSFs μ_i. b) with normalization, the validity function Φ_i is obtained from multidimensional MSFs

Interpretation in the form of Takagi-Sugeno model

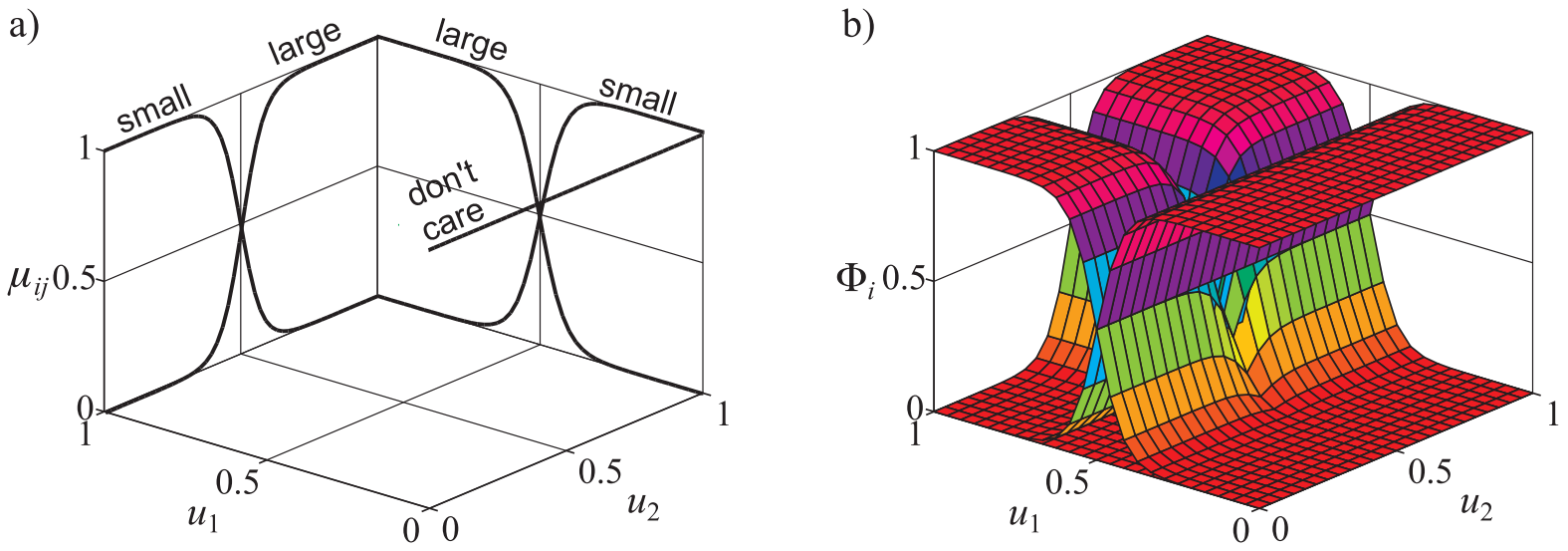


Figure 6: With *normalized* Gaussian MSFs we get a similar (not the same) result as in the figure 5 b

Optimization of local linear model parameters

- The estimation of local linear model parameters is a linear optimization problem provided that we know the validity functions. This is also assumed below.
- We distinguish two basic approaches in determining the local linear parameters of the model: **global and local estimation.**
- **The global estimation represents the simple use of the least squares method, local estimation neglects the overlap of validity functions and takes into account only the local characteristic of the model.**

Global estimation

- In this case, *all* linear parameters of the model are estimated simultaneously with a single LS optimization.
- The parameter vector contains all the $n = M(p+1)$ parameters of the local linear model from equation 3 with M neurons and p inputs:

$$\underline{w} = [w_{10}\ w_{11}\ \cdots\ w_{1p}\ w_{20}\ w_{21}\ \cdots\ w_{2p}\ \cdots\ w_{M0}\ w_{M1}\ \cdots\ w_{Mp}]^{\mathrm{T}}. \tag{9}$$

- In this case, the regression matrix is equal to $\underline{X}$ and contains the values of *regressor variables* for all learning data $i = 1, \ldots, N$. For N data, the regression matrix equals:

$$\underline{X} = \left[\ \underline{X}_1^{(\mathrm{sub})} \quad \underline{X}_2^{(\mathrm{sub})} \quad \cdots \quad \underline{X}_M^{(\mathrm{sub})}\ \right] \tag{10}$$

Global estimation

- with regression submatrices $\underline{X}_i^{(\mathrm{sub})}$

$$\underline{X}_i^{(\mathrm{sub})} = \begin{bmatrix} \Phi_i(\underline{u}(1)) & u_1(1)\Phi_i(\underline{u}(1)) & u_2(1)\Phi_i(\underline{u}(1)) & \cdots & u_p(1)\Phi_i(\underline{u}(1)) \\ \Phi_i(\underline{u}(2)) & u_1(2)\Phi_i(\underline{u}(2)) & u_2(2)\Phi_i(\underline{u}(2)) & \cdots & u_p(2)\Phi_i(\underline{u}(2)) \\ \vdots & \vdots & \vdots & & \vdots \\ \Phi_i(\underline{u}(N)) & u_1(N)\Phi_i(\underline{u}(N)) & u_2(N)\Phi_i(\underline{u}(N)) & \cdots & u_p(N)\Phi_i(\underline{u}(N)) \end{bmatrix}.$$

- The output of the model $\underline{\hat{y}}$ is equal to

$$\underline{\hat{y}} = \underline{X}\,\underline{w}\,. \tag{11}$$

where $\underline{\hat{y}} = [\hat{y}(1)\ \hat{y}(2)\ \cdots\ \hat{y}(N)]^{\mathrm{T}}$.

Global estimation

- If we estimate the parameters with equation 6, then we subtract the operating point and we must replace all u_j in matrix $\underline{X}$ with $u_j - c_{ij}$.
- With global estimation, we optimize the criterion function according to the model parameters:

$$I = \sum_{j=1}^{N} e^2(j) \longrightarrow \min_{\underline{w}} \tag{12}$$

where $e(j) = y(j) - \hat{y}(j)$ is the error between the output of the process $y(j)$ and the output of the model $\hat{y}(j)$.

Global estimation

- Global optimal parameters can be calculated in the case if $N \geq M(p+1)$

Global estimation

$$\underline{\hat{w}} = \left(\underline{X}^{\mathrm{T}}\underline{X}\right)^{-1}\underline{X}^{\mathrm{T}}\underline{y} \tag{13}$$

where $\underline{y} = [y(1)\ y(2)\ \cdots\ y(N)]^{\mathrm{T}}$ contains the measured outputs of the process.

- A global LS estimation is a very effective way to evaluate the consequent parameters, it is computationally demanding since its complexity grows with the third power according to the number of parameters, i.e. by the number of neurons M:

Computational complexity for global estimation

$$\mathcal{O}\left(M^3(p+1)^3\right) \approx \mathcal{O}\left(M^3p^3\right) . \tag{14}$$

- The local estimation, which will be presented below, is much more numerically robust and computationally less demanding.

Local estimation

- In this method of parameter estimation we deal with data within a particular area completely separately and we neglect interactions with other local models.
- As we have already noted, this increases the bias error of the model.
- In this case, when the width of the membership function goes $\sigma \to 0$, and we have no interaction between local models, the output of the model depends only on one model.
- Local estimation is made for each of M local models separately, and each time we calculate $p+1$ parameters of a particular local model.
- The parameter vector for each of $i = 1, \ldots, M$ models is equal to

$$\underline{w}_i = [w_{i0}\ w_{i1}\ \cdots\ w_{ip}]^{\mathrm{T}}\,. \tag{15}$$

- The corresponding regression matrix

$$\underline{X}_i = \begin{bmatrix} 1 & u_1(1) & u_2(1) & \cdots & u_p(1) \\ 1 & u_1(2) & u_2(2) & \cdots & u_p(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & u_1(N) & u_2(N) & \cdots & u_p(N) \end{bmatrix}. \tag{16}$$

Local estimation

- If we want to estimate the parameters in the form from equation 6, where the free term defines the operating point, then we have to observe only the deviations from the centers of membership functions, and replace all u_j with $u_j - c_{ij}$.
- Note that all regression matrices in this case are the same ($i = 1, \ldots, M$) since the elements of the $\underline{X}_i$ matrix do not depend on i.
- The output of the local model is defined by $\underline{\hat{y}}_i$

$$\underline{\hat{y}}_i = \underline{X}_i \, \underline{w}_i \tag{17}$$

where $\underline{\hat{y}}_i = [\hat{y}_i(1) \; \hat{y}_i(2) \; \cdots \; \hat{y}_i(N)]^{\mathrm{T}}$ is the output of the local model and it is valid only in the area where the validity function $\Phi_i(\cdot)$ is greater than a specified value that is close to 1.

- These are data that are close to the center $\Phi_i(\cdot)$ and are relevant for estimating parameters $\underline{w}_i$. With respect to this, of course, one of the possible good solutions is also the use of weighted least squares, where the weight is equal to the validity function

$$I_i = \sum_{j=1}^{N} \Phi_i(\underline{u}(j)) \, e^2(j) \longrightarrow \min_{\underline{w}_i} \tag{18}$$

and $e(j) = y(j) - \hat{y}(j)$ is the error between the output of the model and the measurement.

Local estimation

- We weight the data with the matrix of dimension $N \times N$, which is called weight matrix

$$\underline{Q}_i = \begin{bmatrix} \Phi_i(\underline{u}(1)) & 0 & \cdots & 0 \\ 0 & \Phi_i(\underline{u}(2)) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Phi_i(\underline{u}(N)) \end{bmatrix} \tag{19}$$

- We obtain the solution of the weighted least squares for the parameters of the consequent part of i-th local model in the case of a predefined equation system ($N \geq p + 1$)

Local estimation

$$\underline{\hat{w}}_i = \left(\underline{X}_i^{\mathrm{T}} \underline{Q}_i \underline{X}_i\right)^{-1} \underline{X}_i^{\mathrm{T}} \underline{Q}_i \underline{y} \,. \tag{20}$$

Local estimation

- In this way we calculate estimates for all local models ($i = 1, \ldots, M$).
- The main advantage of this calculation is in the reduced complexity and good conditionality of the regression matrix. Computational complexity grows linearly with the number of models.

Computational complexity for local estimation

$$\mathcal{O}\left(M(p+1)^3\right) \approx \mathcal{O}\left(Mp^3\right) . \tag{21}$$

- In this way we can see that the computational complexity is much lower than in the case of the global estimation. Of course, the price for this is poorer quality of the model, as we have ignored the interactions between local models.

Comparison between global and local estimations

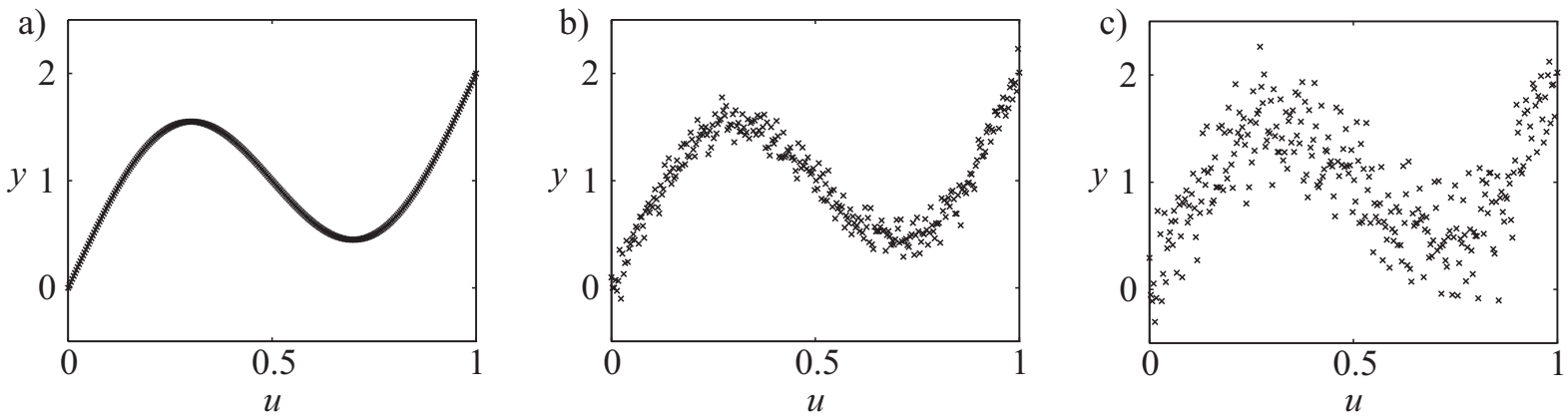


Figure 7: Learning data for the function $y = \sin(2\pi u) + 2u$ with different noise values: a) $\sigma_n = 0$, b) $\sigma_n = 0.1$, c) $\sigma_n = 0.3$

Comparison between global and local estimations

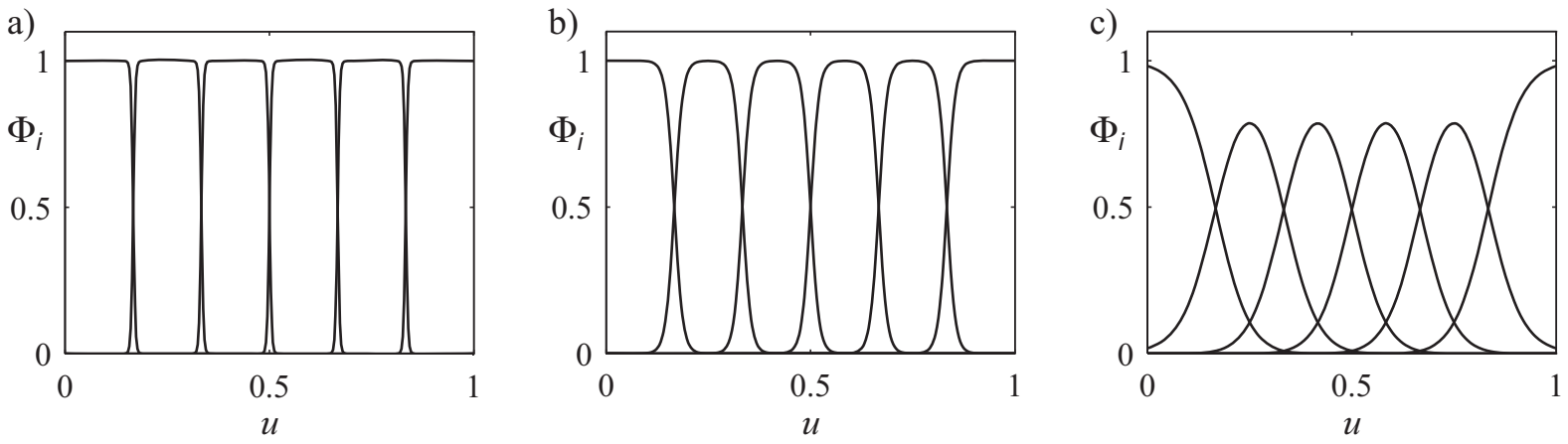


Figure 8: Validity functions for a neuro-fuzzy model with different overlaps: a) $\sigma = 0.125/M$, b) $\sigma = 0.25/M$, c) $\sigma = 0.5/M$ at $M = 6$

Comparison between global and local estimations

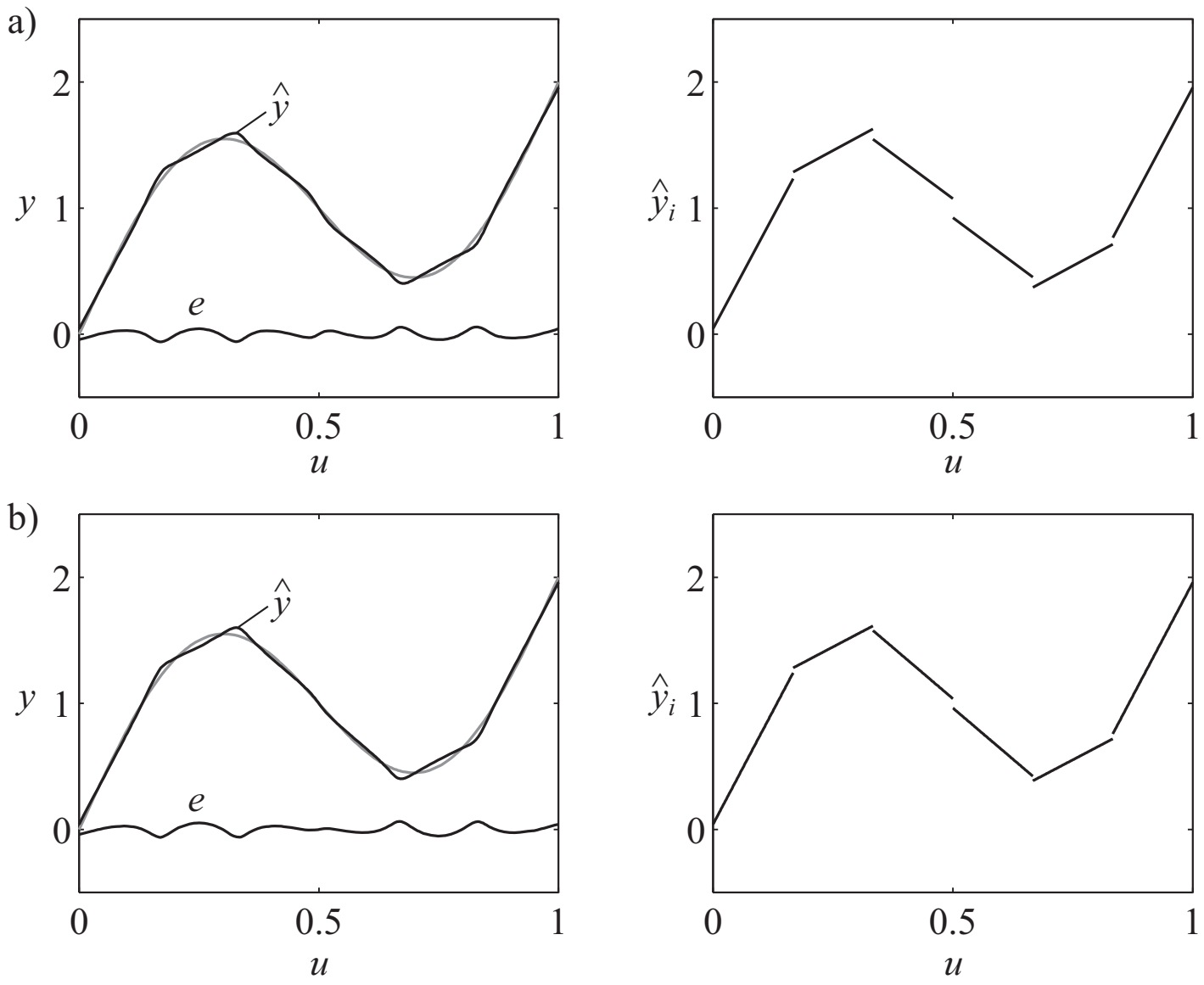


Figure 9: Process and model output (left) and local linear models for a) global and b) local estimation (right) at $\sigma_n = 0$ and $\sigma = 0.125/M$

Comparison between global and local estimations

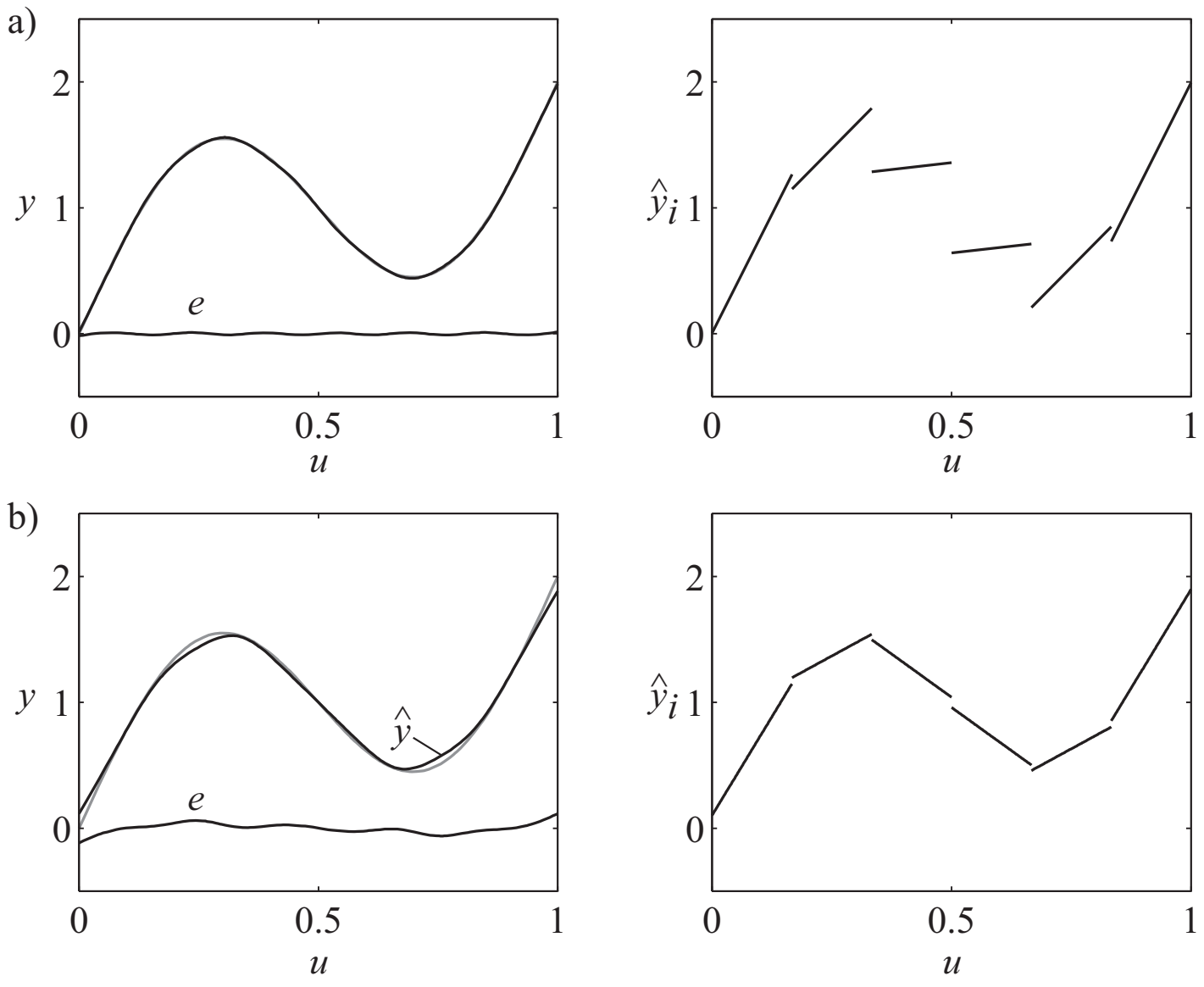


Figure 10: Process and model output (left) and local linear models for a) global and b) local estimation (right) at $\sigma_n = 0$ and $\sigma = 0.5/M$

Comparison between global and local estimations

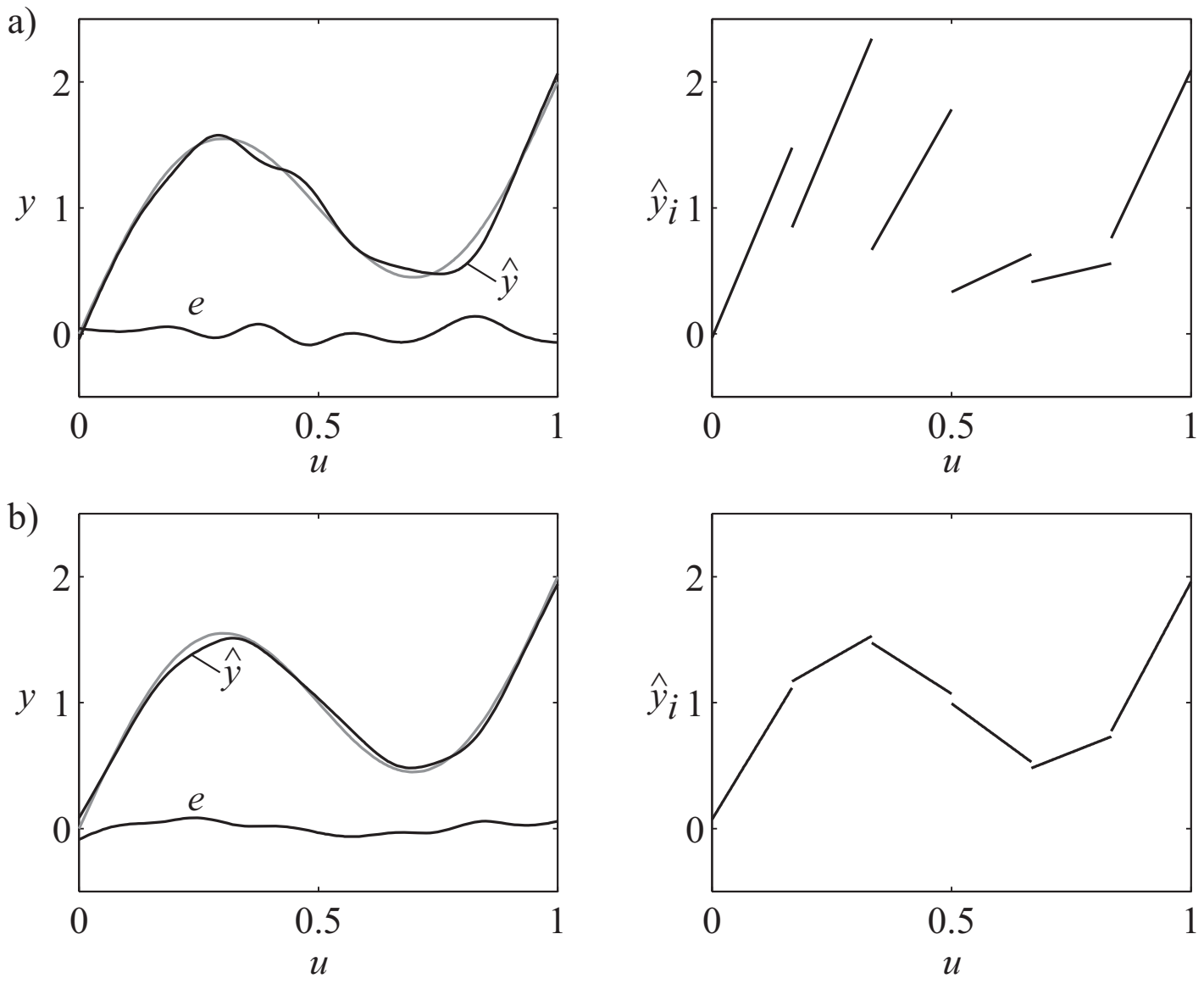


Figure 11: Process and model output (left) and local linear models for a) global and b) local estimation (right) at $\sigma_n = 0.3$ in $\sigma = 0.5/M$

Comparison between global and local estimations

- Depending on the individual applications, we can come across many different criteria which, in addition to approximation, are also important in deciding for an evaluation technique.
- Often the interpretability of local linear models is a very important criterion. In this sense, the local estimation has a great advantage.
- It turns out that the properties of the local estimation are often an advantage. Below are the advantages:
  - *Quick learning*
  - *Interpretation*
  - *Possibility of on-line learning with recursive identification*
  - *High flexibility:* due to a small number of parameters, some of the local models can be parameterized linearly and the others nonlinearly. We also estimate each model separately. We can also choose completely different structures.

Structural optimization of the rule premise

Identification of parameters of membership functions is the problem of nonlinear optimization. There are the following basic strategies for determining these parameters:

- *Network partitioning:* The number of MSFs at the input is determined by preknowledge. This approach creates a dimension problem. The network can be used where we have enough data, but not elsewhere. The approach is limited to small dimension problems.
- *Clustering of the input space:* Membership functions are obtained by the clustering of the input space and they are set according to the data distribution. The approach does not take into account the nature of the process.
- *Nonlinear local optimization:* Originally, the MSFs and the parameters of the consequent part of the rules are optimized simultaneously. Usually, things are solved in two steps: by nonlinear local optimization we set the parameters in the rule premise and estimate the consequent parameters with the global least squares. The same approach is used in ANFIS algorithm (adaptive neuro-fuzzy inference system). The approach is computationally demanding, but it gives very good results. The algorithm optimizes a very large number of parameters and we often encounter the problem of overparameterization.
- *Genetic algorithms:* are one of the possible ways to optimize the structure in a premise, although they are computationally very demanding.

Structural optimization of the rule premise

- *Cluster projection:* In the optimization of the structure of fuzzy models, Gustafson-Kessel fuzzy clustering is used very often. By its use we can find clusters which have hyperellipsoidal shapes, which are usual if we tackle the approximation problems. Covariance matrices are joined to the clusters. If covariance matrices are divided to their principal components, we can also obtain data model in the form of different hyperplanes under certain conditions
- *Heuristic construction:* The most widely used algorithms for constructing are algorithms that increase the complexity of the model with learning iterations. They start with one rule, then they decrease the resolution of the input space and add rules to areas where models do not fit the learning set. Methods can result in very flexible structures that can achieve almost any space partitioning, or slightly less flexible partitions, which are axially orthogonal and non-orthogonal.

Local linear model tree algorithm (LOLIMOT)

LOLIMOT is an incremental algorithm with a tree structure that divides the input space with an axial orthogonal partitioning. It is also used by the SUHICLUST algorithm.

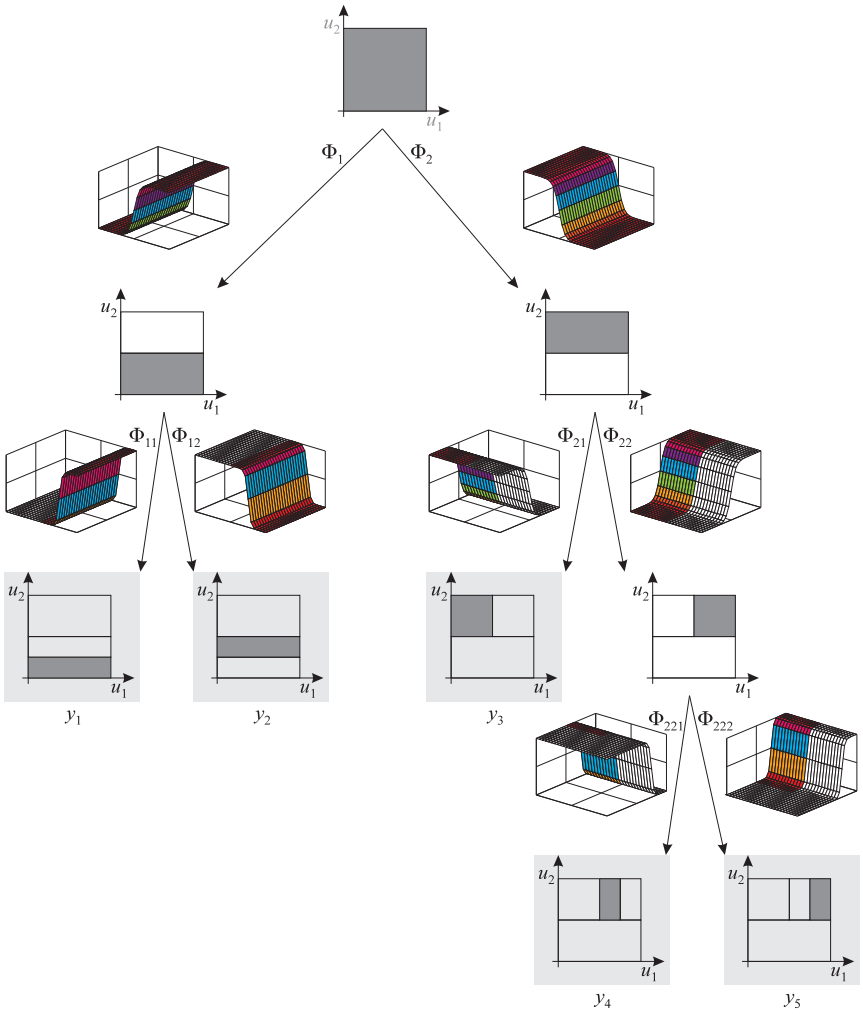


Figure 12: The operation of the LOLIMOT algorithm in the first few iterations for a two-dimensional problem ($p = 2$)

Identification of linear dynamic systems

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for 13th Lecture, April 2016

Table of contents

1. Identification of linear dynamic systems
2. Overview of linear system identification methods
3. Excitation signals
4. General model structures
   - Terminology and classification
   - Optimal predictor
   - Simulation
   - Prediction
   - Prediction Error Models - PEM
5. Output feedback models
   - Autoregressive model with external input - ARX
     - Bias and consistency
   - autoregression model of a moving average with an external input - ARMAX
   - Output error models - OE
6. Some important aspects of identification
   - Consistency
   - Operating point

Overview of linear system identification methods

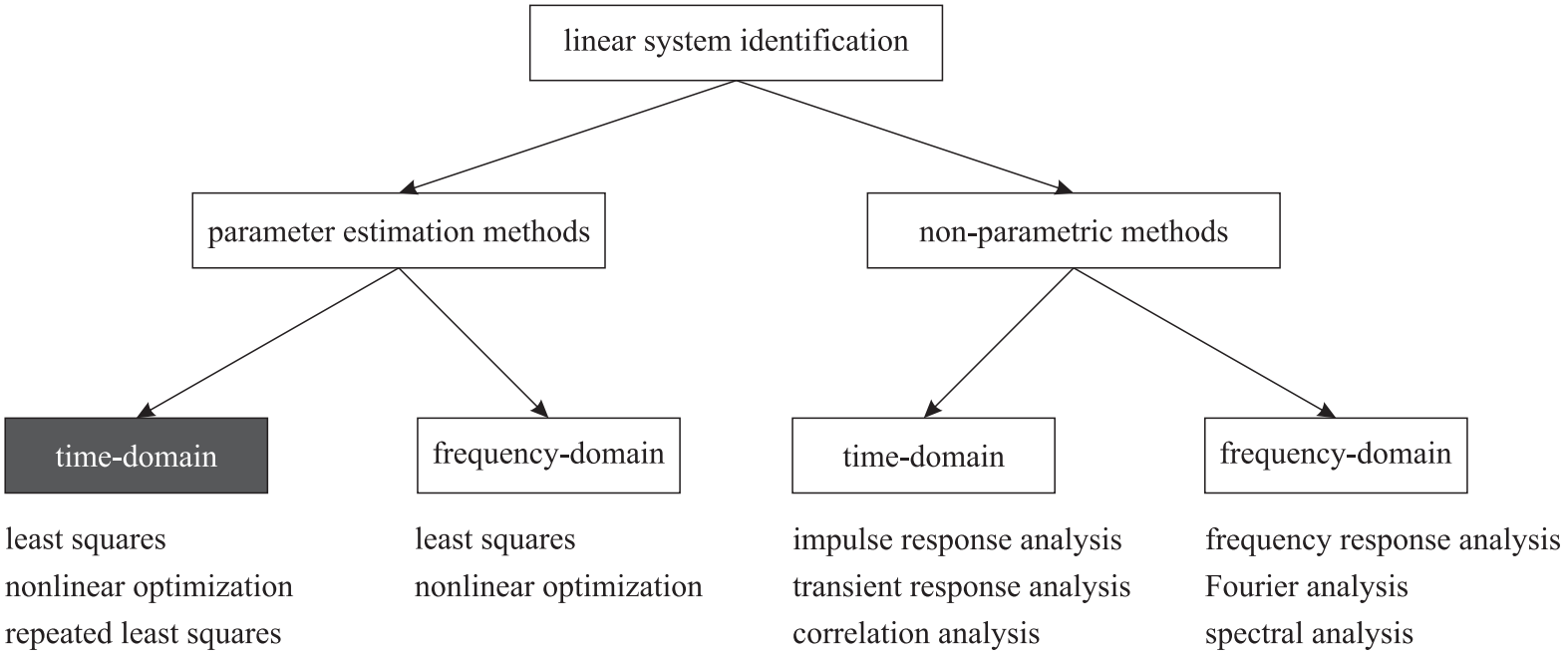


Figure 1: Overview of methods for identifying linear systems. In this chapter, we treat only methods in black shaded blocks. Let's emphasize that we are talking about identification methods. A non-parametric impulse response model can also be obtained using the parametric method if we assume the finite number of model parameters.

Excitation signals

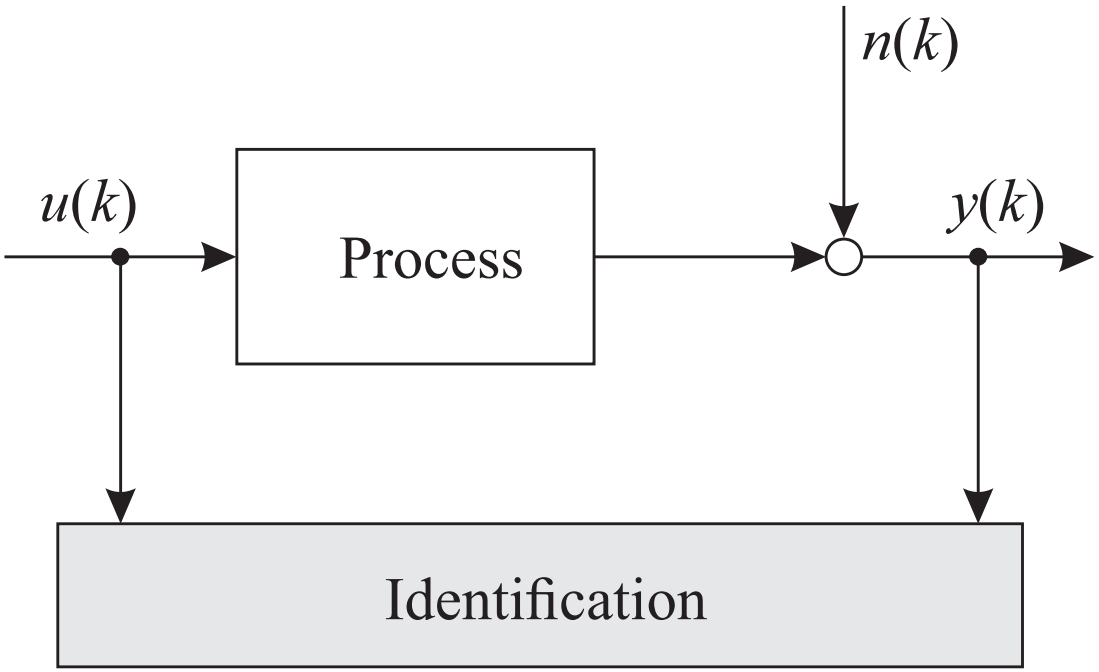


Figure 2: Input and output with a disturbance at the process output that is measured for use in identification

Example: Excitation input signals

Below we consider various excitation signals and corresponding output signals for the first-order system with a gain of $K = 1$ and a time constant of $T = 8s$, which is sampled with the sampling time $T_0 = 1$ s and we obtain a difference equation ($a_1 = \exp(-T_0/T),\ b_1 = K(1 - a_1)$)

$$y(k) \ = \ 0.1175u(k-1) + 0.8825y(k-1) \tag{1}$$

- excitation of the process with the input signals in figure 3–7, with 100 measurements,
- added colored noise with variance 0.01 to the process output,
- the noise filter has a denominator that is identical to the process denominator $1/A$ to meet the assumptions for the ARX model,
- identification of ARX first-order model; see subsection 1.

Since the structure of the model and the structure of the process are the same, the model bias is zero and the whole error is present only due to noise.

Example

The results of identification for various excitations are summarized in table 1. The comparison is as follows:

- *Constant:* Suitable for identification of only one parameter at zero frequency. In this case we have identified a DC gain K, which is defined as $K = b_1/(1 - a_1)$ in a discrete time. The signal is not suitable for the identification of dynamic systems since it does not excite the dynamics. Parameters b_1 and a_1 can not be estimated individually, we can only estimate the ratio $b_1/(1 - a_1)$.
- *Impulse:* It is not suitable for identification, since the estimation of the gain is very inaccurate.
- *Step:* Very suitable for identification. We estimate the dynamics and the low frequency behavior of the process very well, since the low frequencies are dominant.
- *Rectangular pulse:* Very suitable for identification. By changing the frequency of the pulses, we can excite a certain frequency band to a greater extent. For the signal in figure 6, the estimate of time constant is very accurate.
- *PRBS (pseudo-random binary signal):* Very suitable for identification since it imitates white noise in a discrete time with a deterministic signal and, therefore, it excites an uniformly wide frequency band.

Example

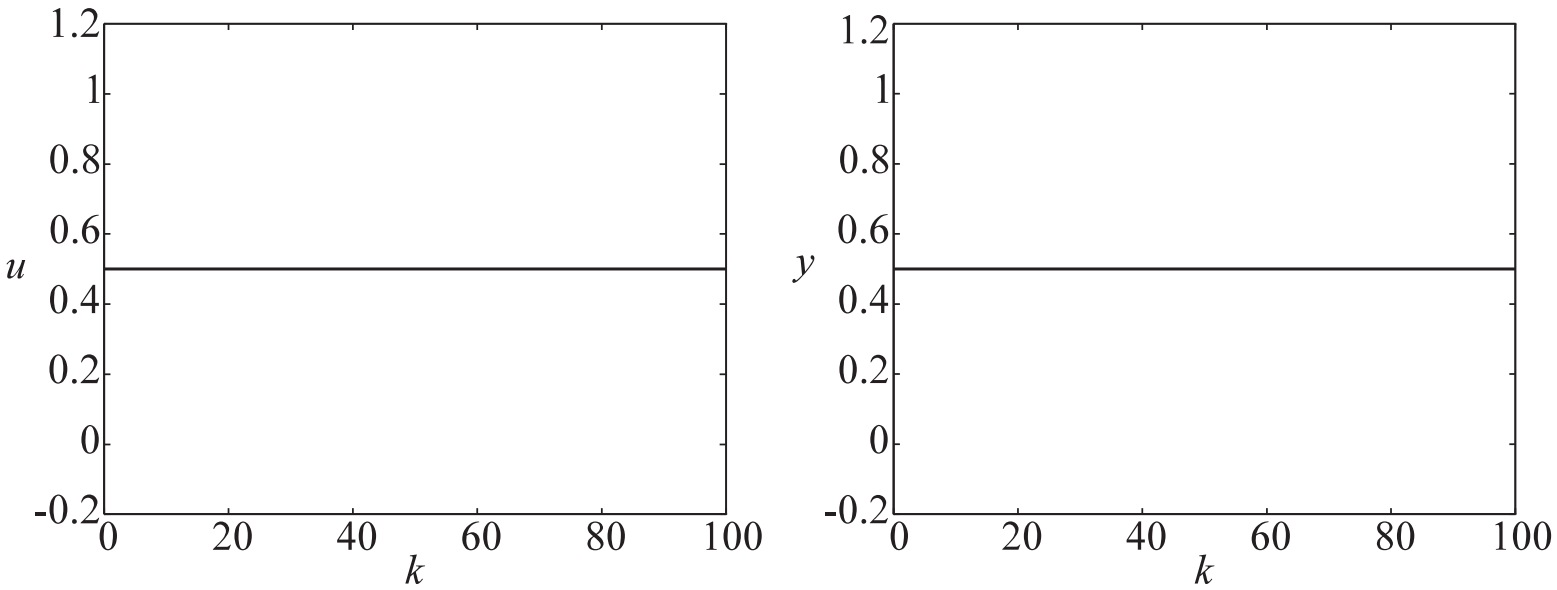


Figure 3: Excitation with constant signal (left) and output of the process without disturbance (right)

Example

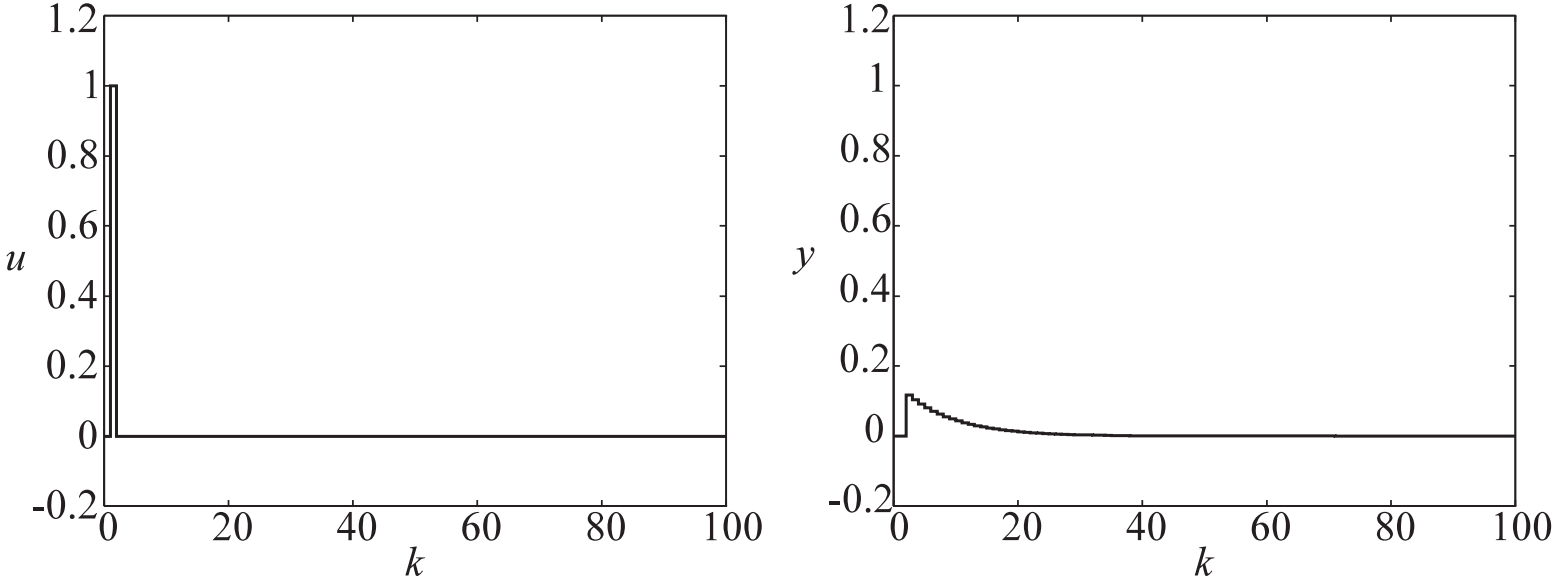


Figure 4: Excitation with impulse signal (left) and output of the process without disturbance (right)

Example

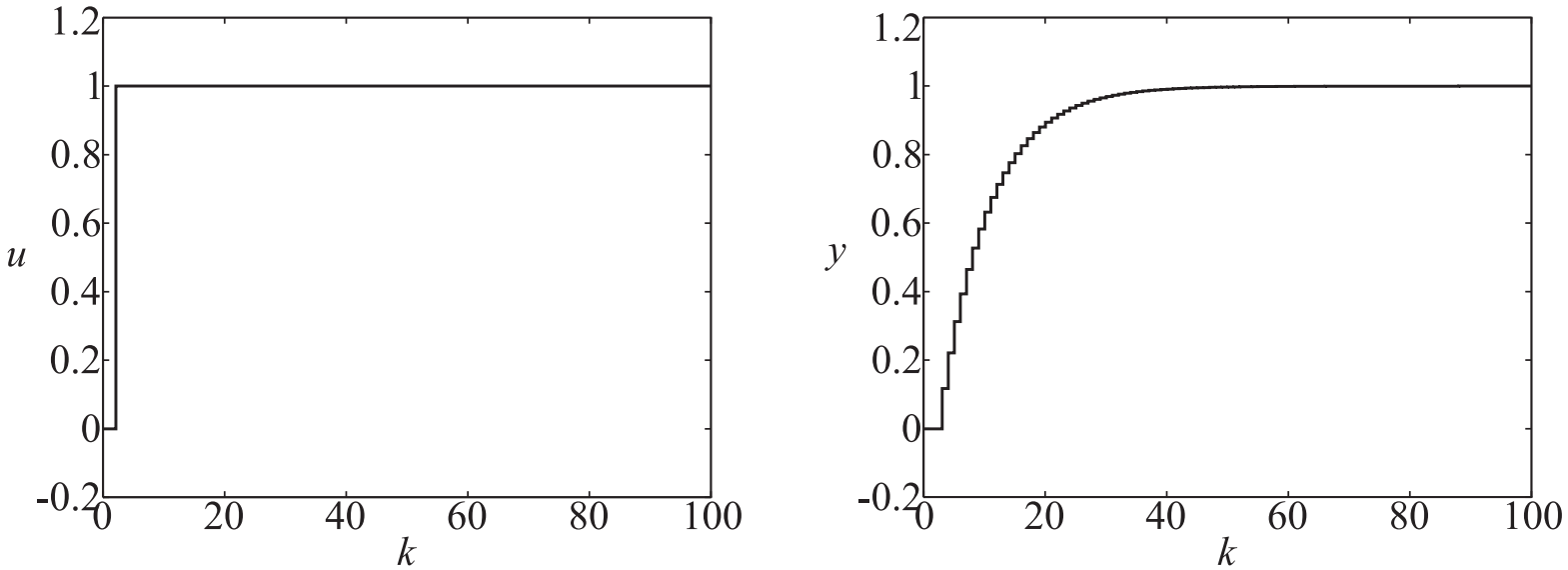


Figure 5: Excitation with step signal (left) and output of the process without disturbance (right)

Example

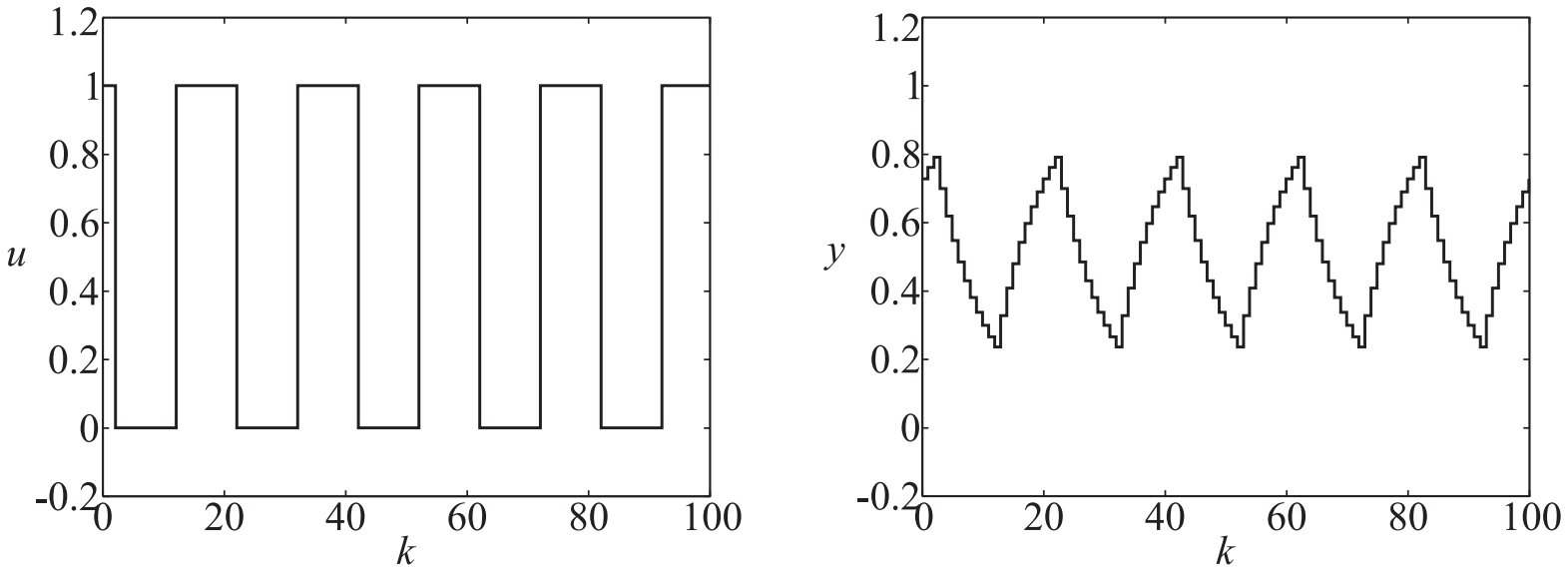


Figure 6: Excitation with rectangular pulses (left) and output of the process without disturbance (right)

Example

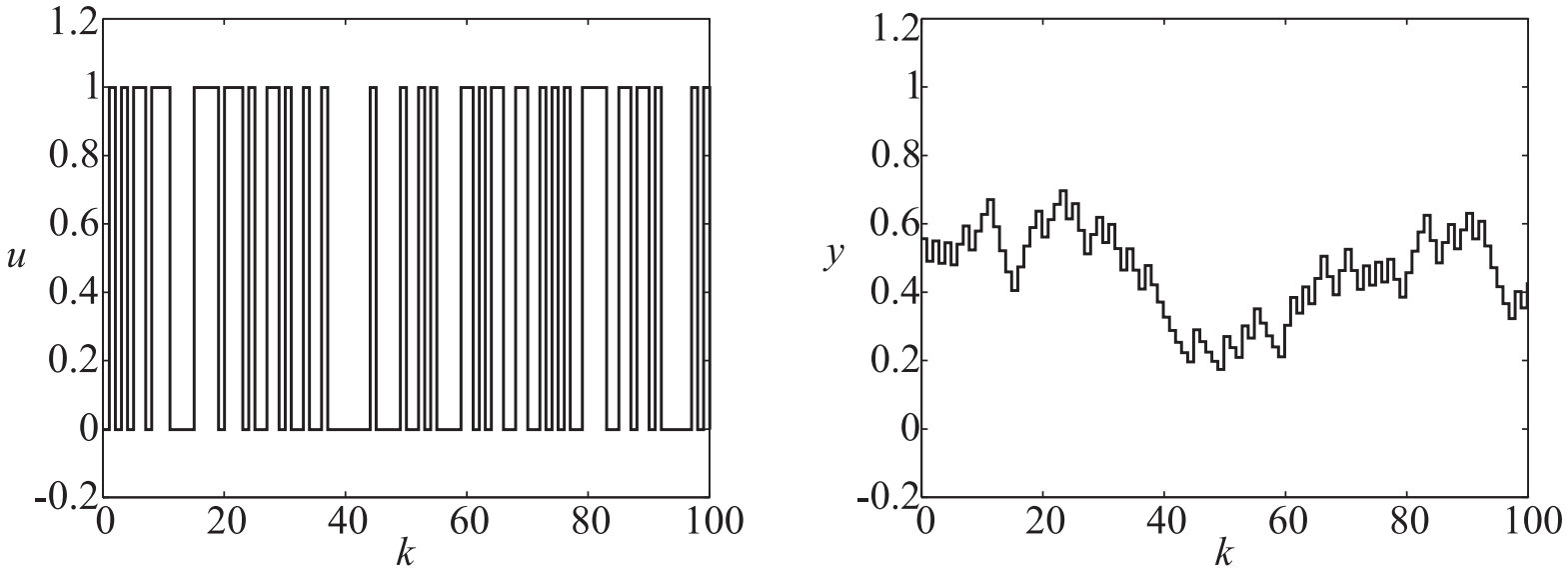


Figure 7: Excitation with PRBS (pseudo-random binary signal) (left), which is the deterministic approximation of white noise in discrete time and the output of the process without disturbance (right)

Example

Table 1: Process identification for various excitation signals

Input signal	b_1	a_1	K	T [s]
Constant	0.2620	-0.7392	1.0048	3.3098
Impulse	0.0976	-0.8570	0.6826	6.4800
Step	0.1220	-0.8780	0.9998	7.6879
Rectangular pulses	0.1170	-0.8834	1.0033	8.0671
PRBS	0.1201	-0.8796	0.9980	7.7964
Actual process parameters	0.1175	-0.8825	1	8

General model structures

The general linear structure of the model from which we can obtain all other structures by means of simplifications. Usually we do not use the general model.

The deterministic part of model The output of the linear system $y(k)$ at time k can be obtained by filtering the input signal $u(k)$ through the transfer function of the process $G(q)$ (q represents the shift operator, i.e. $q^{-1}x(k) = x(k-1)$, and it is time equivalent to the operator $z = e^{j\omega}$ in the frequency domain). In general, the transfer function of the process $G(q)$ contains the numerator $\widetilde{B}(q)$ and the denominator $\widetilde{A}(q)$.

$$y(k) \;=\; G(q)u(k) \;=\; \frac{\widetilde{B}(q)}{\widetilde{A}(q)}u(k) \tag{2}$$

The non- deterministic part of model By filtering the white noise $v(k)$ through a linear filter $H(q)$, we can model any noise characteristic (colored noise). In this way, we can get any colored noise $n(k)$ that we write down as

$$n(k) \;=\; H(q)v(k) \;=\; \frac{\widetilde{C}(q)}{\widetilde{D}(q)}v(k) \tag{3}$$

A general linear model describing the deterministic and stochastic part of the process can be written by combining both parts (see figure 8a)

$$y(k) \;=\; G(q)u(k) \;+\; H(q)v(k) \tag{4}$$

General model structures

- The filter $G(q)$ is called the *input transfer function or transfer function of the process*, since it connects the input $u(k)$ to the output $y(k)$,
- and the filter $H(q)$ is called *noise transfer function* because it connects the input white noise $vk)$ to the process output $y(k)$.
- The transfer functions $G(q)$ and $H(q)$ can be decoupled into their numerator and denominator; see figure 8b.
- For the further analysis, it is recommended to separate the denominator $A(q)$ from $G(q)$ and $H(q)$; see figure 8c and 8d.

The general form of the linear model can be written as

$$y(k) = \frac{B(q)}{F(q)A(q)}u(k) + \frac{C(q)}{D(q)A(q)}v(k) \tag{5}$$

or equivalently as

$$A(q)y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}v(k) \tag{6}$$

General model structures

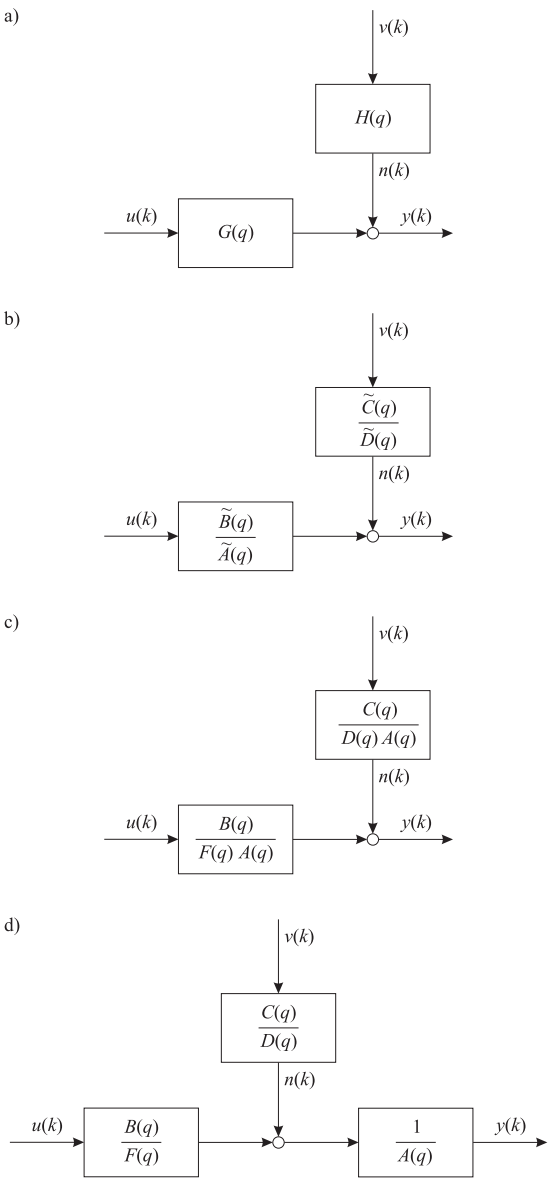


Figure 8: General linear model

Terminology and classification

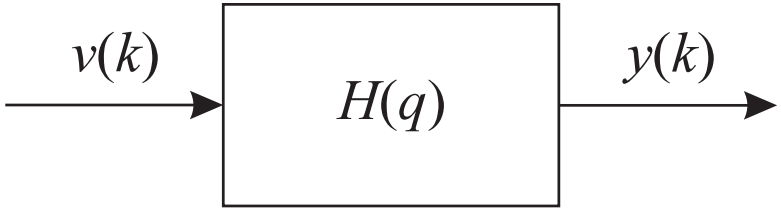


Figure 9: General linear model of the time series. The input of the model $v(k)$ is white noise, but we do not have the deterministic input $u(k)$

Terminology and classification

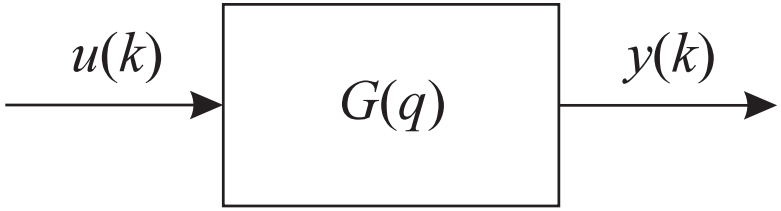


Figure 10: General linear deterministic model. The input of the $u(k)$ model is a deterministic signal, but we do not have the stochastic signal $v(k)$

Terminology and classification

A time series model which includes only denominator (figure 11) is called the *autoregressive model (AR)*. It is written as:

$$y(k) = \frac{1}{D(q)} v(k) \tag{7}$$

The time series model which includes only numerator (figure 11) is called the *moving average model (MA)*. It is written as:

$$y(k) = C(q)v(k) \tag{8}$$

The time series model which has both the numerator and the denominator in the transfer function (figure 11) is called the *autoregressive moving average model (ARMA)*. It is written as:

$$y(k) = \frac{C(q)}{D(q)} v(k) \tag{9}$$

Terminology and classification

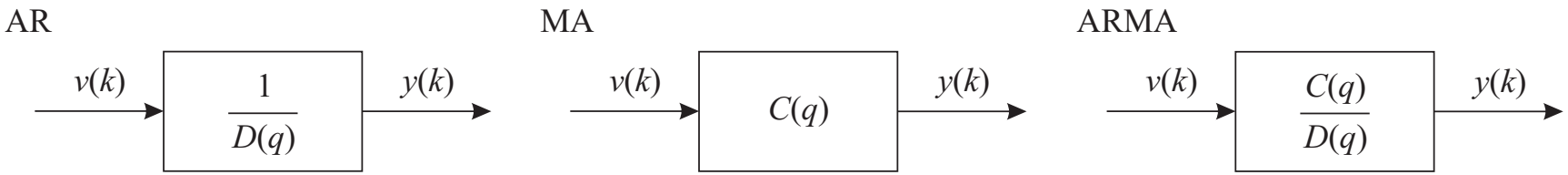


Figure 11: Overview of time series models: autoregressive model (AR), moving average model (MA) and autoregressive moving average model (ARMA)

Terminology and classification

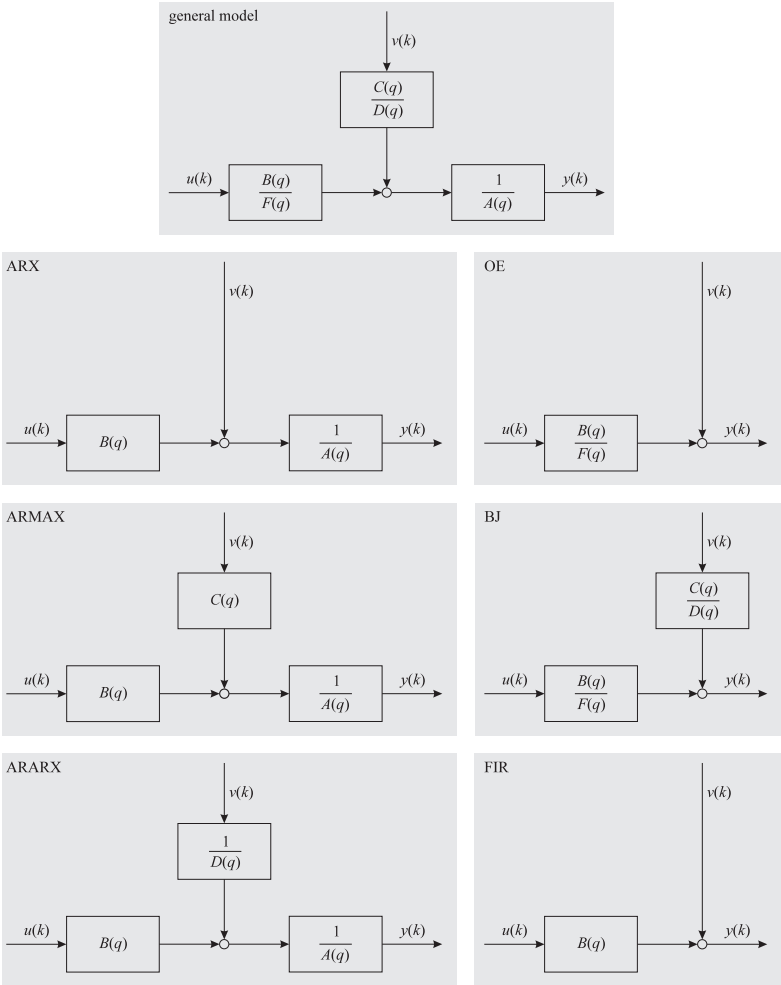


Figure 12: Overview of the most important linear dynamic models

Optimal predictor

The most common use of models is in predicting the behaviour of the process in the future. In this prediction, two different cases can be distinguished:

- *simulation* if we calculate the output of the model according to the process input signal without measuring intermediate values of the process output
- *prediction* if the process outputs are known or they are measurable when calculating predictions.

Figures 13 and 14 show schemata for simulation and prediction.

Simulation

The use of Equation 10 as simulation of the completely deterministic process:

$$\hat{y}(k) = G(q)u(k) \tag{10}$$

- the model $H(q)$ is completely irrelevant for the simulation.
- the model of the noise $H(q)$ influences the estimation of the parameters of the transfer function $G(q)$ and in this way indirectly affects the quality of the simulation,
- $H(q)$ is not directly included in equation (10).

- Since we do not know the process output, we do not have any information about disturbances.
- To get at least some insight into the disturbances, usually the best way is to generate the white noise $w(k)$ with the appropriate variance, which is filtered through the filter $H(q)$ and then added to the deterministic output of the model

$$\hat{y}(k) = G(q)u(k) + H(q)w(k) \tag{11}$$

Simulation

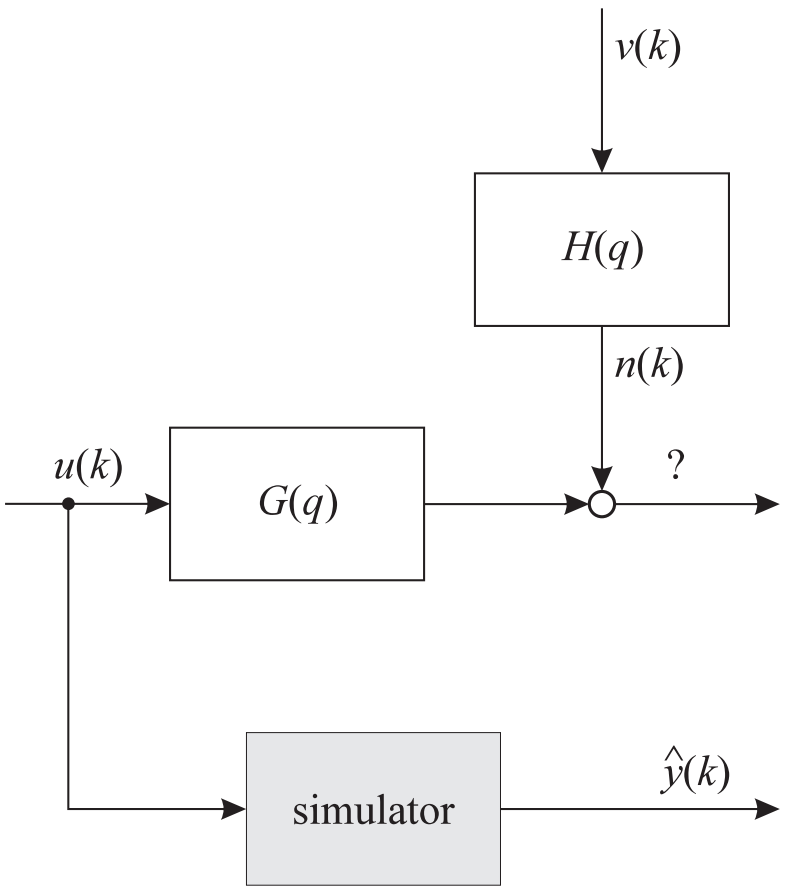


Figure 13: Only inputs are required for the simulation; information about output is not available

Prediction

- in the case of a prediction, we use information about delayed measured values of the process output signal.
- the *optimal predictor* combines the input and delayed measured outputs of the process.
- The optimal *linear* predictor can be defined as a linear combination of filtered inputs and outputs of the process

$$\begin{aligned}\hat{y}(k|k-1) \;=\; & s_0 u(k) + s_1 u(k-1) + \ldots + s_m u(k-m) \\ & + t_1 y(k-1) \;+ \ldots + \; t_n y(k-n)\end{aligned} \tag{12}$$

or

$$\hat{y}(k|k-1) \;=\; S(q)u(k) \;+\; T(q)y(k) \tag{13}$$

Prediction

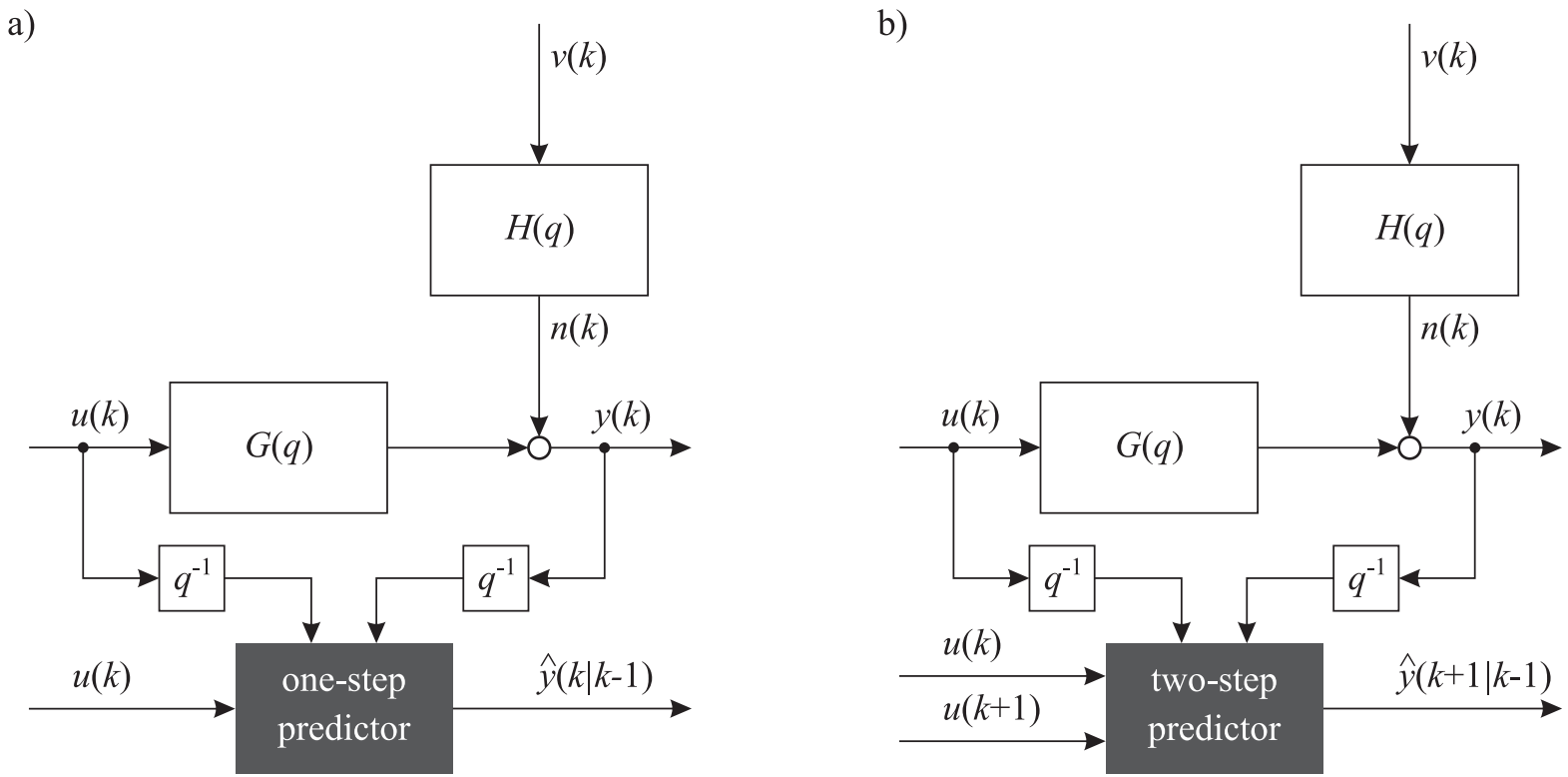


Figure 14: a) One step ahead prediction and b) Two step ahead prediction. The term " $|k-1$ " means that this " information is available at time $k-1$ ". In addition to the inputs, we need delayed process outputs for the prediction. If the prediction horizon l is large, the information about the previous outputs of the process becomes less important, and if $l \to \infty$ then the prediction goes towards simulation, the figure 13

Prediction

In the literature it has been shown that the optimal predictor, which gives the smallest quadratic prediction error (variance of prediction error), can be determined as

$$\hat{y}(k|k-1) = \frac{G(q)}{H(q)}u(k) + \left(1 - \frac{1}{H(q)}\right)y(k) \tag{14}$$

or

$$H(q)\hat{y}(k|k-1) = G(q)u(k) + (H(q)-1)\,y(k) \tag{15}$$

This means that $S(q) = G(q)/H(q)$ and $T(q) = 1 - 1/H(q)$.

- The optimal predictor must be able to extract all the information from the data.
- The prediction error, i.e. the difference between the output of the process $y(k)$ and the prediction $\hat{y}(k|k-1)$, should be equal to the white noise $v(k)$, since this is the only thing that is unpredictable:

$$v(k) = y(k) - \hat{y}(k|k-1) \tag{16}$$

This equation can be put into the equation that is represented by a block in figure 13) and we get:

$$y(k) = G(q)u(k) + H(q)\,(y(k) - \hat{y}(k|k-1)) \tag{17}$$

If we express $\hat{y}(k|k-1)$ from this equation, we obtain the optimal predictor from equation (14).
The optimal predictor results in residual of white noise, which can be used to test the model's structure.

Predictive Error Models - PEM

Usually, we use the optimal predictor for assessing the quality of the model. The prediction error is the difference between the process output (what we want to achieve) and the one-step ahead prediction of the model at the same input

$$e(k) = y(k) - \hat{y}(k|k-1) \tag{18}$$

Hereinafter, the term *prediction error* is used as a synonym for *one step ahead prediction error*. By the use of the optimal predictor from equation (14) we get a prediction error that is equal to

$$e(k) = \frac{1}{H(q)} y(k) - \frac{G(q)}{H(q)} u(k) \tag{19}$$

Most of the identification algorithms are based on the minimization of the criterion function, which is a function of prediction error (one-step ahead prediction). So the criterion function is the following

$$J = \sum_{i=1}^{N} e^2(i) \tag{20}$$

Autoregressive model with external input - ARX

- The ARX model is by far the most common linear dynamic model.
- Usually, first we try to model the system with the ARX model and, if the result is not satisfactory, then we try with more complex structures. However, it rarely happens since its structure is very close to real processes.
- The popularity of ARX model should also be attributed to a very simple way of estimating the parameters where the least squares method can be used since the prediction error is linear in the parameters.
- At the same time there is also a reliable recursive calculation option.
- The ARX model is shown in figure 15 and it can be written as

$$A(q)y(k) = B(q)u(k) + v(k) \tag{21}$$

Autoregressive model with external input - ARX

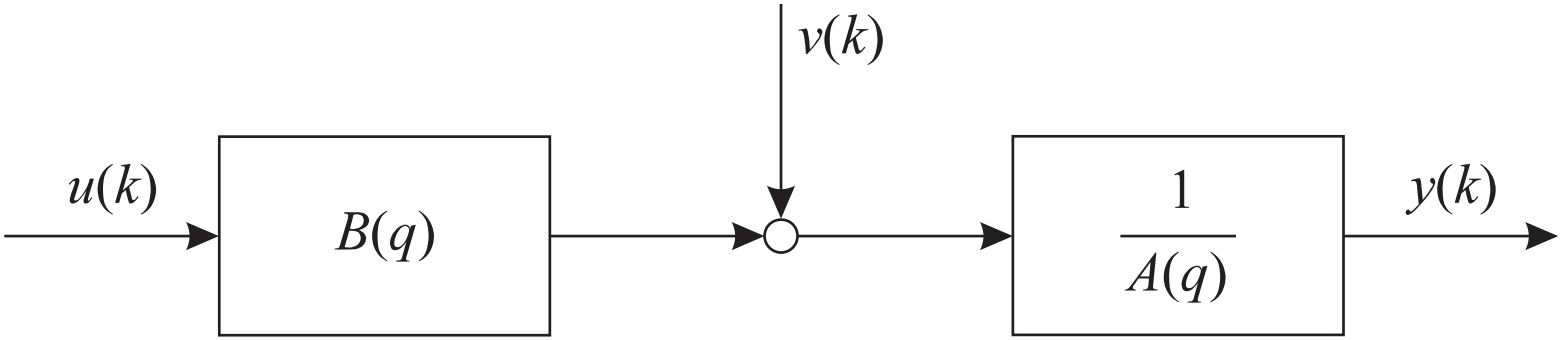


Figure 15: ARX model

The optimal ARX predictor can be written with the polynomials of the transfer function as

$$\hat{y}(k|k-1) \;=\; B(q)u(k) \;+\; (1 - A(q))y(k) \tag{22}$$

or it can be written in the form of a differential equation as

$$\begin{aligned} \hat{y}(k|k-1) \quad &= \quad b_1 u(k-1) + \ldots + b_m u(k-m) \\ &- \quad a_1 y(k-1) - \ldots - a_m y(k-m) \end{aligned} \tag{23}$$

assuming that order(A) = order(B) = m.

Autoregressive model with external input - ARX

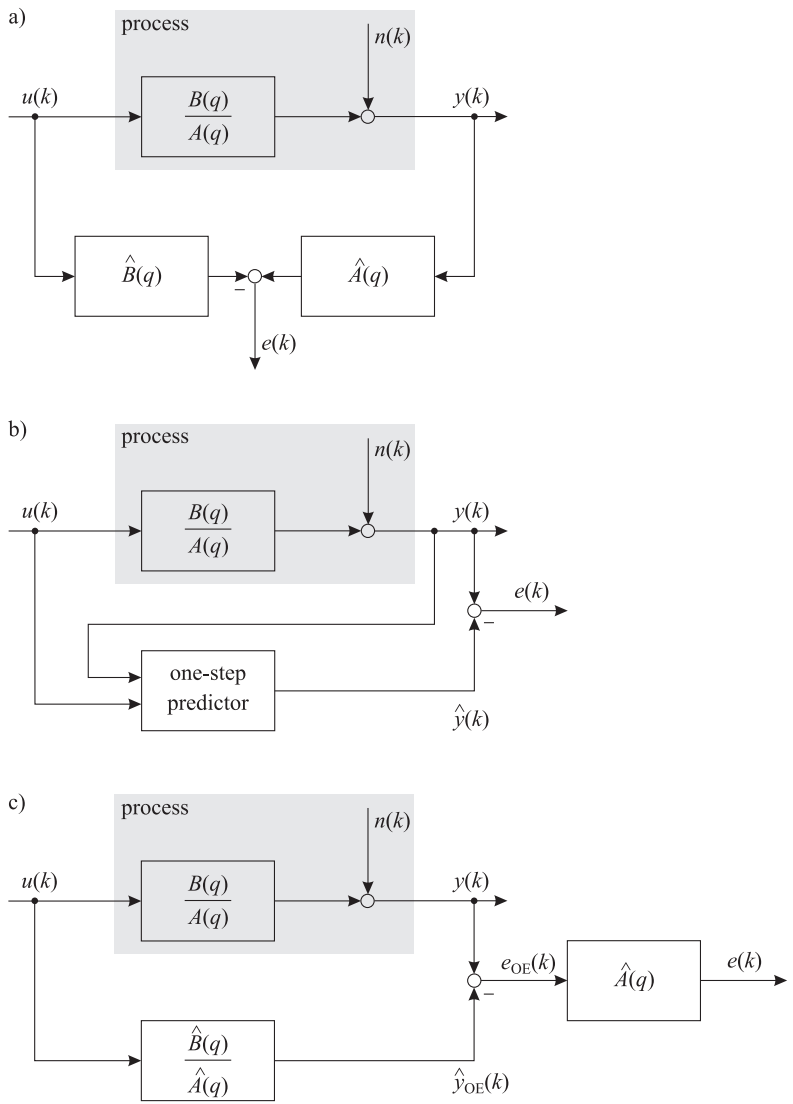


Figure 16: Different schemes of ARX models: a) configuration in the form of a generalized model or model of equation error, b) configuration in the form of a predictor, c) pseudo-parallel configuration with error filtering. All configurations realize the same ARX model.

Bias and consistency

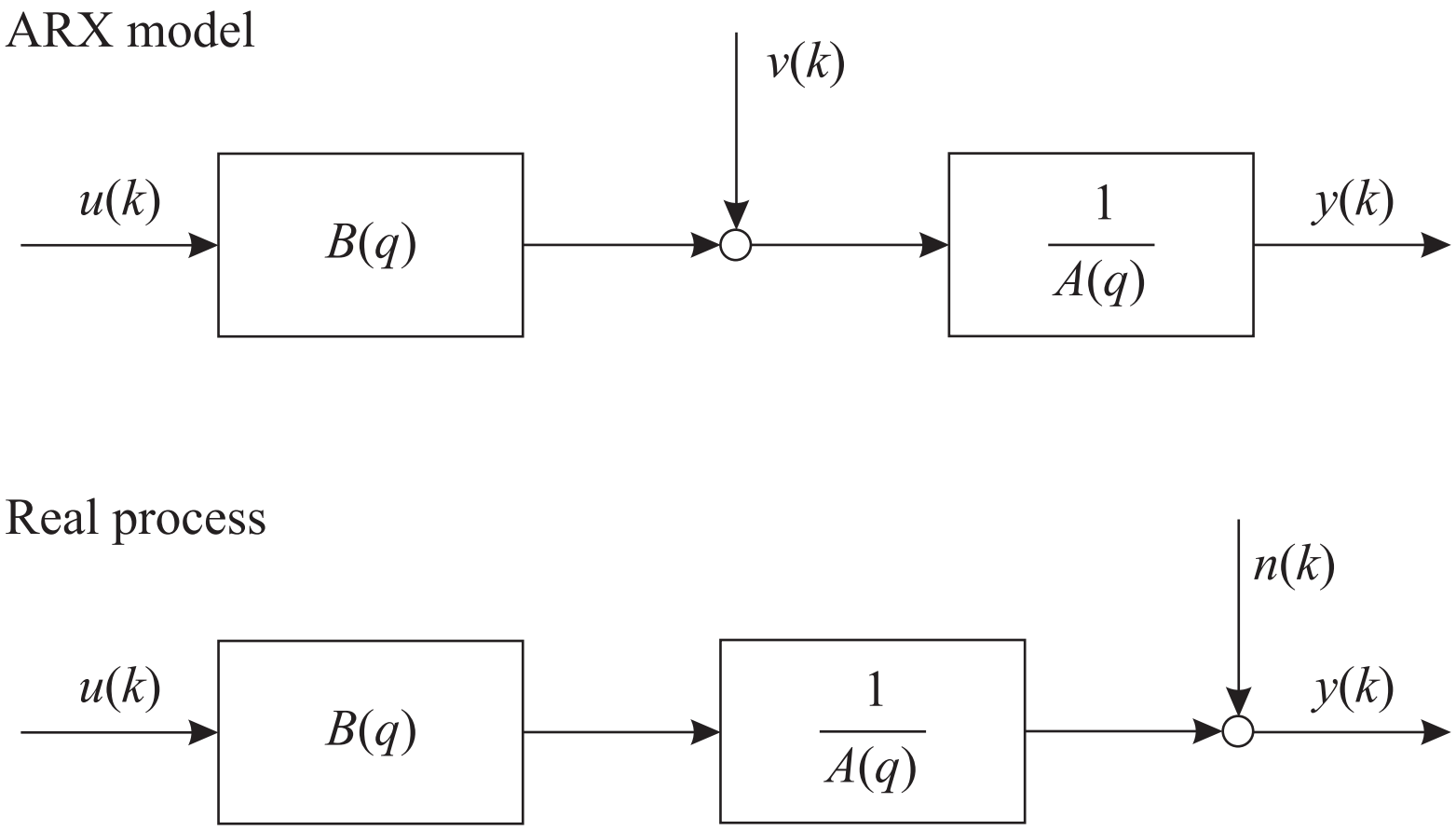


Figure 17: The ARX model assumes a noise model with a transfer function $1/A(q)$, although the additive noise $n(k)$ at the output of the model is much more realistic. The noise $n(k)$ can be white noise or color noise $n(k) = C(q)/D(q)v(k)$

Output error models - OE

- In addition to the ARX and ARMAX models, the OE is the third most common model.
- Is the simplest representative of the class of output error models.
- The model assumes an additive noise disturbance at the output of the model, and not in the middle of the model, as is the case with general error models (equation error).
- OE models are often closer to real problems and frequently they turn out to be better than general error models. However, since the noise model does not contain the denominator $1/A(q)$, all forms of the output error model are nonlinear in parameters and consequently more difficult to estimate.
- The OE model is shown in figure 18 and it is represented by the following equation

$$y(k) = \frac{B(q)}{F(q)}u(k) + v(k) \tag{24}$$

- In the field of linear system identification standard is that in the case of output error model, the denominator is written with the polynomial $F(q)$, and in the case of general error models (such as ARX, ARMAX, and ARARX) with polynomial $A(q)$. This is just a way to highlight the differences between models.

Output error models - OE

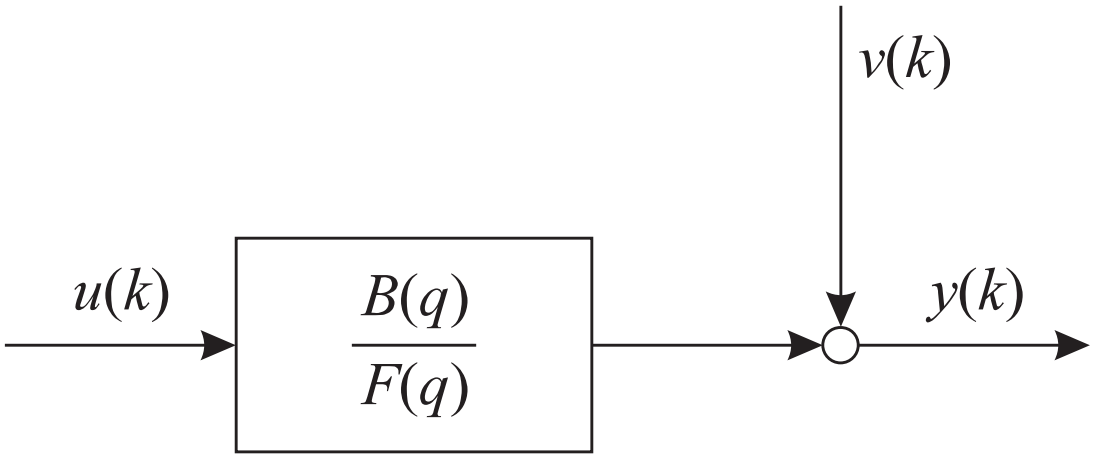


Figure 18: OE model

Output error models - OE

The optimal OE predictor is just a simulator since it does not use any measured output signal $y(k)$:

$$\hat{y}(k|k-1) \;=\; \hat{y}(k) \;=\; \frac{B(q)}{F(q)}u(k) \tag{25}$$

Let us note that in this case we can omit "$|k-1''$ in the notation since the OE model is not based on the measured signals for calculating the prediction.

We can also find that the OE predictor is unstable if the polynomial $F(q)$ is unstable. Therefore, it can not be used for modeling unstable processes. The same applies to all output error models. Considering equation 25, the prediction error for the OE model is equal to

$$e(k) \;=\; y(k) - \frac{B(q)}{F(q)}u(k) \tag{26}$$

Output error models - OE

Figure 19 shows the OE model in parallel with the process. The prediction error of the OE model is the difference between the process output and the simulated output of the model. We assume that a disturbance $n(k)$ is white noise.

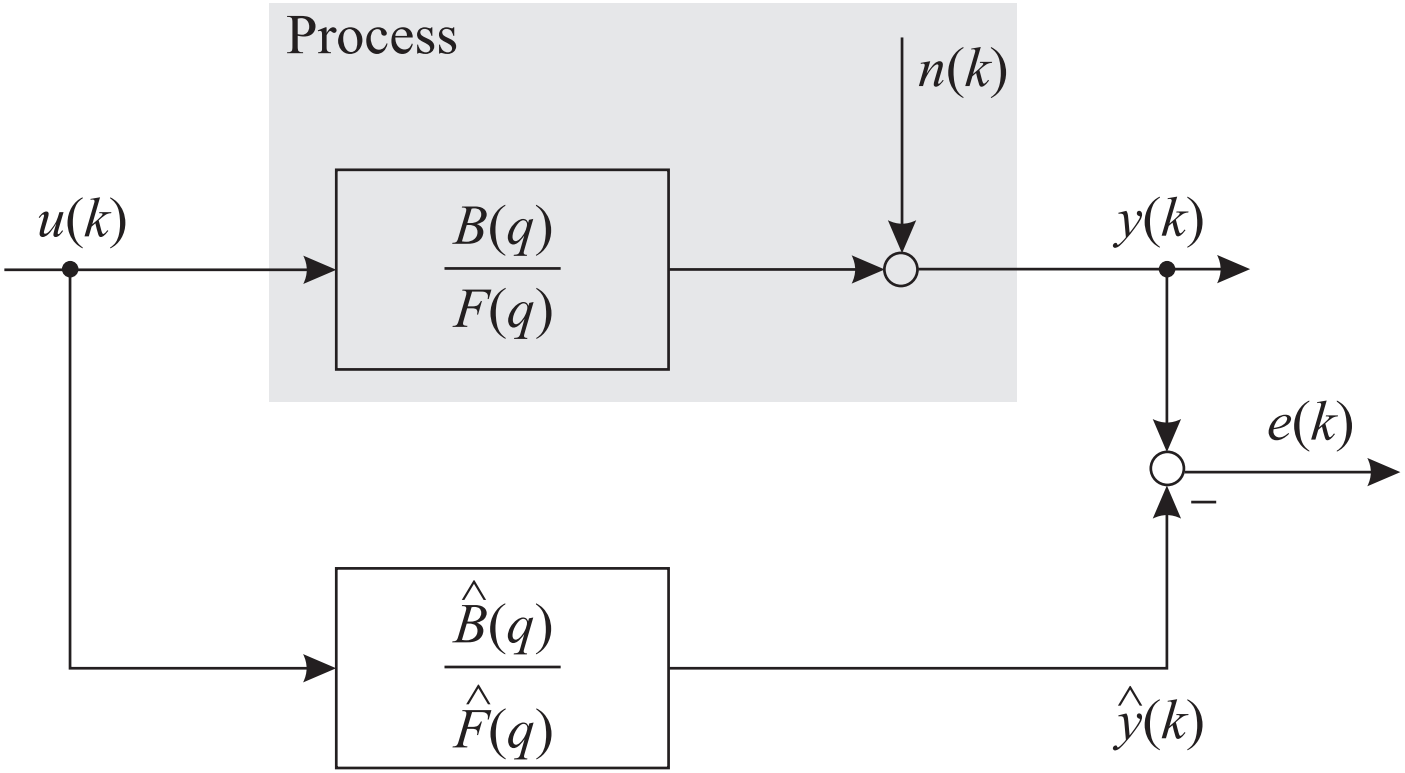


Figure 19: OE model in parallel with the process

Output error models - OE

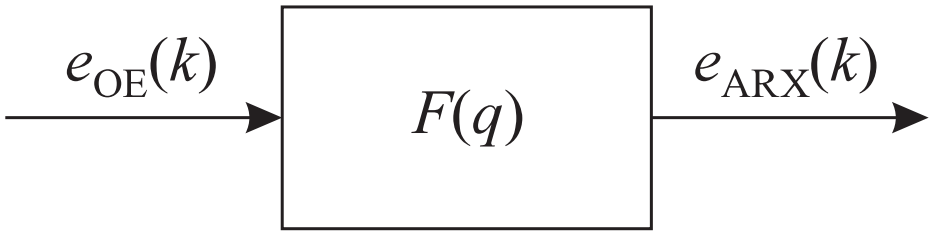


Figure 20: The relationship between ARX and OE residuals. ARX residuals are filtered residuals of the OE model through the filter $F(q)$

Figure 20 shows the residuals of the OE and ARX model. According to the equations for the ARX and OE model, the residuals of the ARX model can be obtained by filtering the OE residuals:

$$e_{\mathrm{ARX}}(k) \;=\; F(q)\, e_{\mathrm{OE}}(k) \tag{27}$$

Some important aspects of identification

Some important aspects of identification:

- consistency
- operating point

Consistency

- For the consistency of the estimation of the model parameters, we have to choose transfer function $\hat{G}(q)$ of the model, which has the same structure and complexity as the process. Otherwise, the process $G(q)$ can not be completely described.
- Parameters of the ARX model are estimated consistently only when the noise can be properly modeled with the transfer function $1/A(q)$. This is very rare in practice. If the noise models are properly structured and in this way they are capable of describing the actual noise, then the parameters are estimated consistently. However, since credible knowledge of noise is rarely accessible, in practice we try to use methods that, despite the inadequate noise modeling, enable a consistent estimation of the parameters.
- *All model structures*, that have an independent input transfer function $G(q)$ and a transfer function of the noise $H(q)$, enable consistent estimation of the parameters of the transfer function even if the noise model is inadequate. An “independently parameterized” transfer function signifies that $G(q)$ and $H(q)$ do not contain common parameters.
- In the case of output error models, the parameters are independent, i.e. in the case of OE or BJ model. In the case of general error models, i.e. ARX and ARMAX, the parameters are dependent, since $G(q)$ and $H(q)$ have common polynomial $A(q)$. This is the main advantage of OE and BJ models over ARX and ARMAX. Of course it is clear that these parameters of the OE and BJ model are more difficult to estimate.

Operating point

- The data for linear identification are typically measured near the operating point $\bar{U}$ and $\bar{Y}$.
- Figure 21 shows the operating point that lies on the static characteristic of the process.
- If we want to approximate the process with a linear model, then the process must be excited in such a way that deviations from the operating point u and y are sufficiently small with respect to the degree of process nonlinearity at a given operating point $\bar{U}$ and $\bar{Y}$).
- A problem arises in the identification of the linear model, since we measure $U(k)$ and $Y(k)$, but for identification we need deviations from the operating point $u(k)$ and $y(k)$.
- If we write the deviations from the operating point as $u(k) = U(k) - \bar{U}$ and $y(k) = Y(k) - \bar{Y}$, we get the following difference equations:

$$\begin{aligned}(Y(k) - \bar{Y}) + a_1(Y(k-1) - \bar{Y}) + \ldots + a_m(Y(k-m) - \bar{Y}) \quad &= \\ b_1(U(k-1) - \bar{U}) + \ldots + b_m(U(k-m) - \bar{U})\end{aligned} \tag{28}$$

which can also be written as

$$\begin{aligned}Y(k) = \quad & - \quad a_1 Y(k-1) - \ldots - a_m Y(k-m) \\ & + \quad b_1 U(k-1) + \ldots + b_m U(k-m) \\ & + \quad \underbrace{(1 + a_1 + \ldots + a_m)\bar{Y} - (b_1 + \ldots + b_m)\bar{U}}_{C}\end{aligned} \tag{29}$$

Operating point

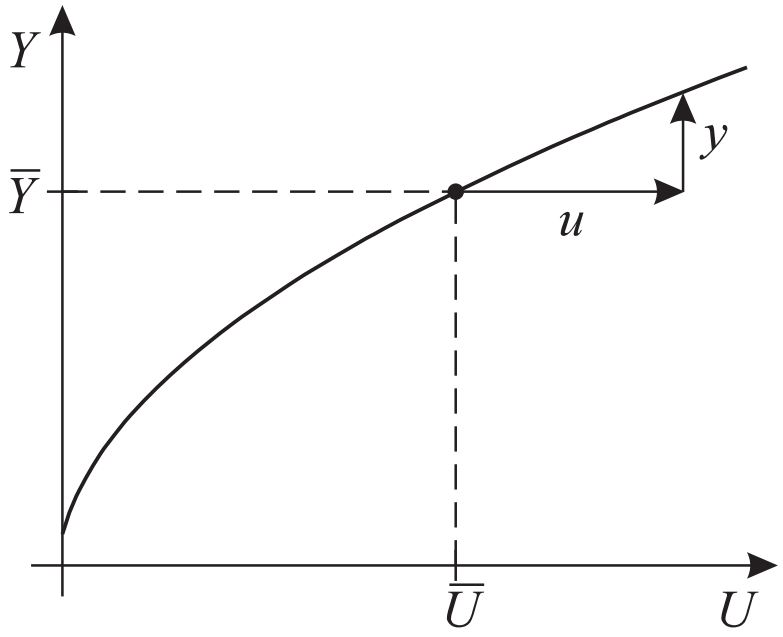


Figure 21: Data for linear identification must lie near the operating point $\bar{U}$, $\bar{Y}$

Operating point

The *offset* C contains information about operating point $\bar{U}$, $\bar{Y}$. We have three options to deal with this offset:

- remove the value of the operating point from the data and processing of the offsets,
- offset estimation,
- extension of the model.

Operating point

1. If we know the equilibrium point, then the deviation from the equilibrium point can be calculated explicitly as

$$u(k) = U(k) - \bar{U}\,, \qquad y(k) = Y(k) - \bar{Y} \tag{30}$$

   and then $u(k)$ and $y(k)$ are used in the identification.

2. If the equilibrium point is not known, it can be approximated by $U(k)$ and $Y(k)$

$$\bar{U} = \frac{1}{N}\sum_{i=1}^{N} U(i)\,, \qquad \bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y(i) \tag{31}$$

   and then we use this approximated equilibrium point in the first approach.

3. An offset can be calculated explicitly if we perform an estimate of the parameters with the use of equation (29). For the ARX model, the regression vector is extended to

$$\underline{x} = [U(k-1) \;\cdots\; U(k-m) \quad -Y(k-1) \;\cdots\; -Y(k-m) \quad 1]^{\mathrm{T}} \tag{32}$$

$$\underline{\theta} = [b_1 \;\cdots\; b_m \quad a_1 \;\cdots\; a_m \quad C]^{\mathrm{T}} \tag{33}$$

   For other model structures the extension is also performed. We need to estimate an additional parameter, which is reflected as a constant 1 in the regression vector.

4. We can eliminate the offset by differentiation of the data. This can be done by pre-filtering the data $U(k)$ and $Y(k)$ with the next filter

$$L(q) = 1 - q^{-1}\,, \tag{34}$$

   which results in differences between adjacent samples $U(k) - U(k-1)$ and $Y(k) - Y(k-1)$.

Identification of nonlinear dynamical systems

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for lectures, May 2018

Table of contents

1. Identification of nonlinear dynamical systems
2. Extension of linear to nonlinear identification
3. Model with external dynamics
   - Model with external dynamics
     - Sampling time effect
     - One-step ahead prediction hyperplane
   - Series-parallel and parallel models
   - Nonlinear classes of dynamic input-output models
   - Models with feedback loop
   - Models without feedback loop
   - Limitations of nonlinear input-output models
4. Excitation of systems
5. Order determination
   - The Lipschitz coefficient method for determination of system order
   - Structural identification with the use of the PCA method

Extension of linear to nonlinear identification

- The simplest linear input-output model in discrete time is an *autoregressive model with exogenous input (ARX)*
- or *output error (OE)* model.
- The optimal prediction of the output of the m-order model is obtained with the ARX model as follows

$$\hat{y}(k) = b_1 u(k-1) + \ldots + b_m u(k-m) - a_1 y(k-1) - \ldots - a_m y(k-m). \tag{1}$$

- This principle can be extended from a linear notation to a nonlinear NARX model in a simple way by replacing the linear relation in equation 1 with an unknown non-linear function $f(\cdot)$ and get

$$\hat{y}(k) = f\left(u(k-1), \ldots, u(k-m), y(k-1), \ldots, y(k-m)\right) . \tag{2}$$

Model with external dynamics

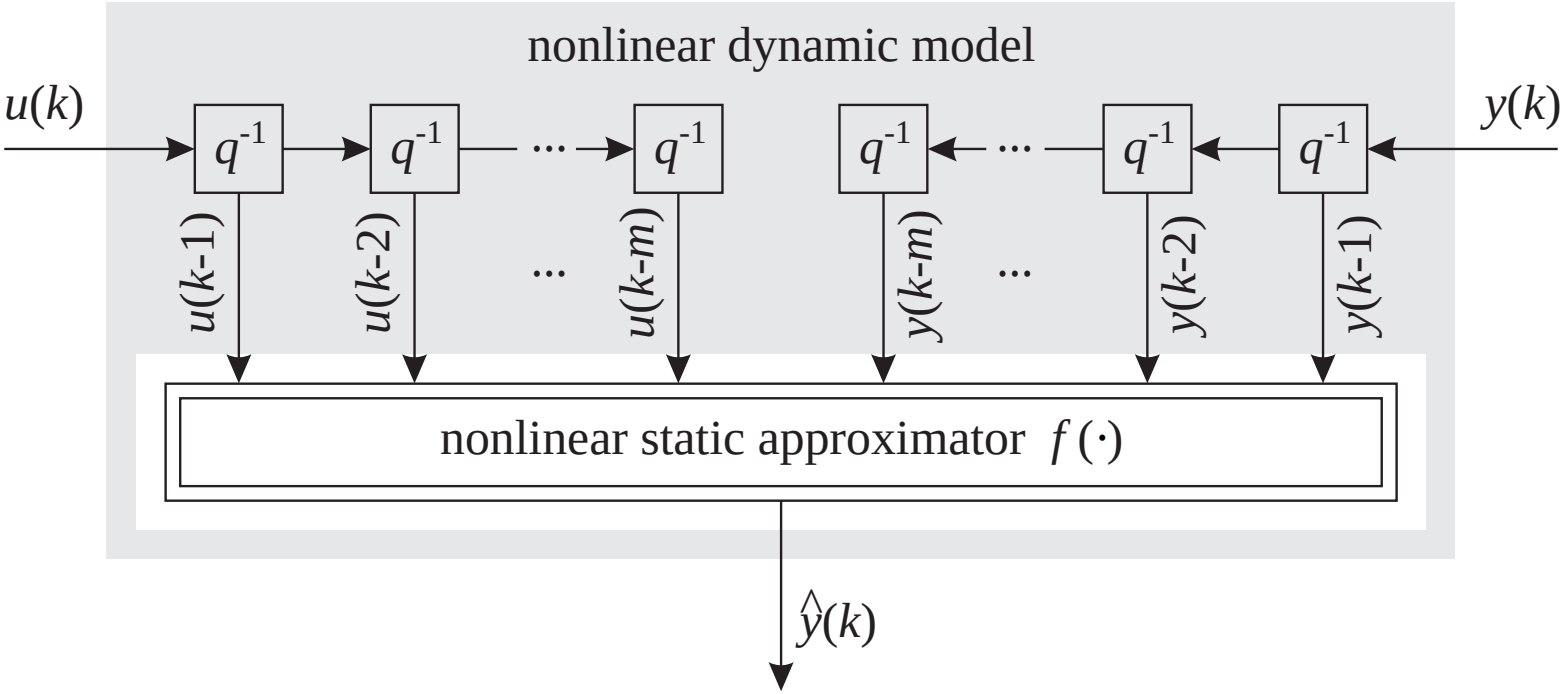


Figure 1: External dynamics model: the model can be divided into a static approximator and an external filter series (TDL)

Model with external dynamics

For example, we will look at the Hammerstein model, determined by the static nonlinearity in a series with the first-order process

$$y(k) = 0.1 \arctan(u(k-1)) + 0.9\, y(k-1) \tag{3}$$

Figure 2 (left) shows the process.

- For stepped shape input signals as shown in Fig. 3a, the input-output space is filled with data that are in vertical lines at constant $u(k)$.
- For sinusoidal signals with low and high frequency, the distribution of data is shown in figures 3b and 3c.

We can observe the following:

- We can not randomly choose all of the inputs of the approximator. Only the input $u(k)$ is selected directly, and all other inputs of the approximator are the result of this.
- The lower the frequency of the input signal, the closer the data to the static nonlinearity (equilibrium point) of the process.
- Of course, the number (distribution) of points in the vicinity of static nonlinearity is considerably greater than outside the equilibrium points.
- A very dynamic input signal is required to cover with data the wide area of the input space.

Model with external dynamics

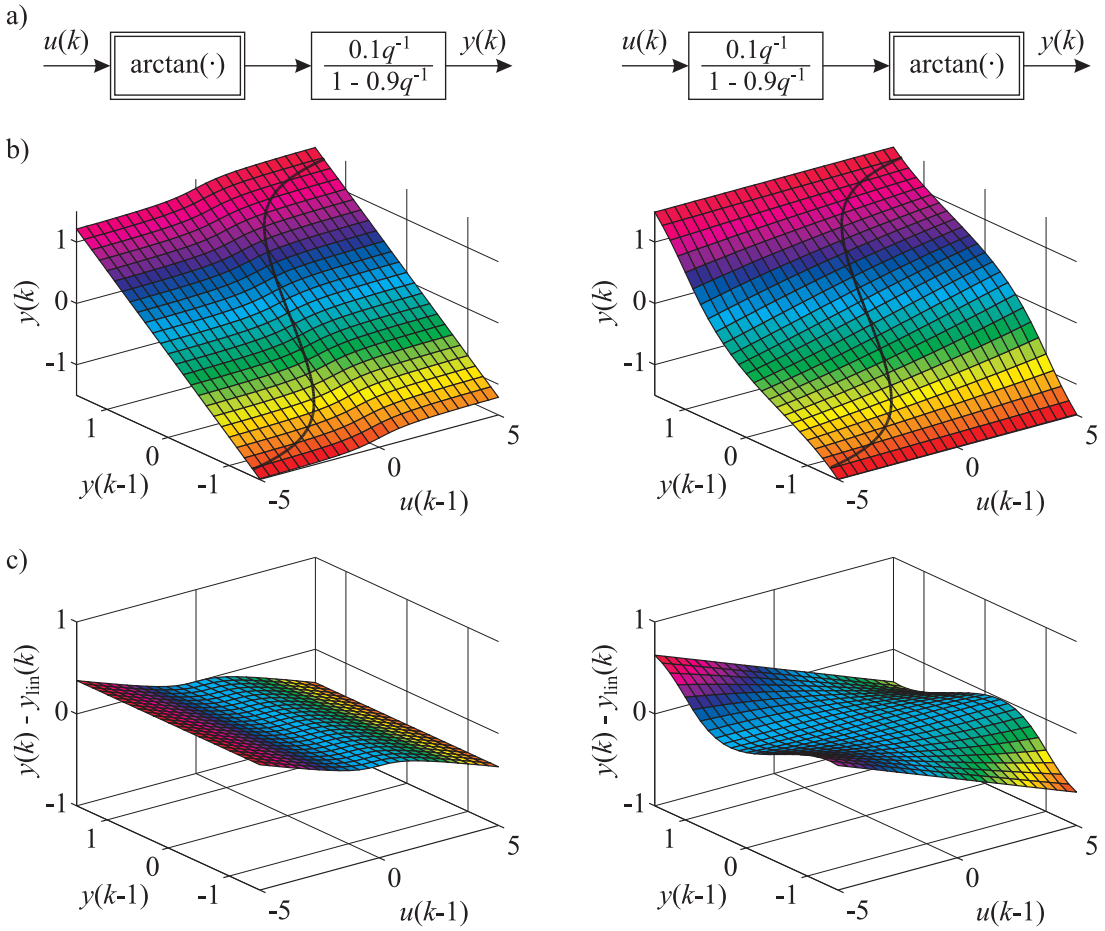


Figure 2: Hammerstein (left) and Wiener (right) models: a) a block diagram of the model, b) a one-step ahead prediction with a surface and a static curve with a thick line, c) a surface showing the differences between the output of the nonlinear and linear models

The relation between the input-output space and the input space of the approximator

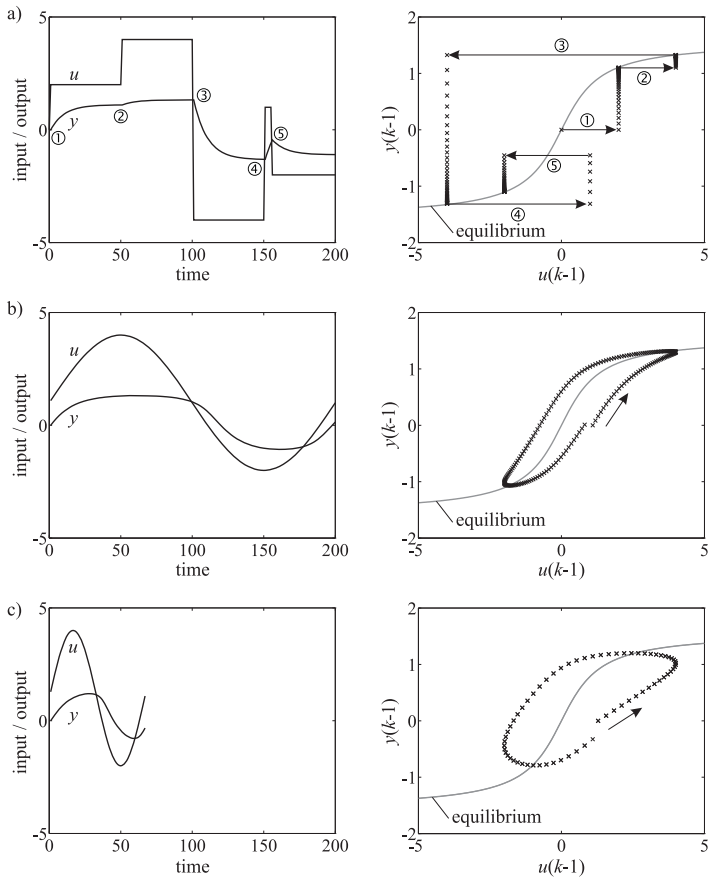


Figure 3: The dependence between the input-output system data (left) and the input space of the approximator in the approach with external dynamics (right): a) step-shaped excitation signals, b) low frequency sinusoidal excitation signal, c) high frequency sinusoidal excitation signal. It can be noted that points far from the equilibrium state can be achieved only with large input steps. The area above static nonlinearity represents the dynamics with decreasing the input signal, and the area below represents the dynamics when the input signal is increased.

Sampling time effect

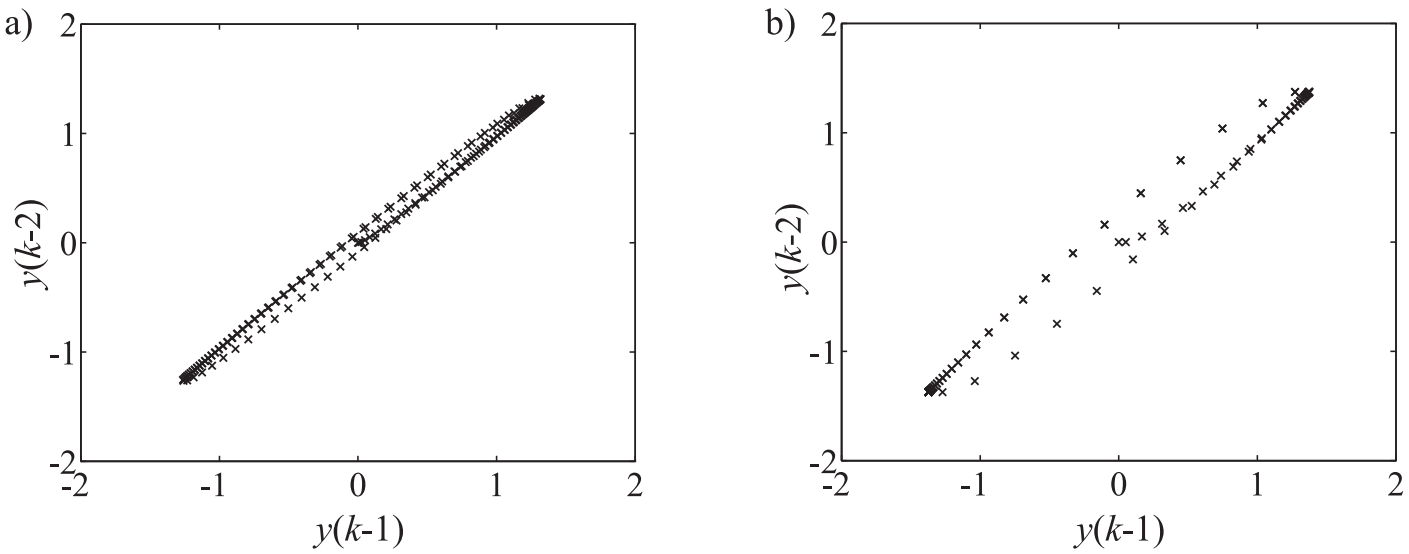


Figure 4: Correlation between delayed output variables for the second order system for a) small and b) large sampling time

Sampling time effect

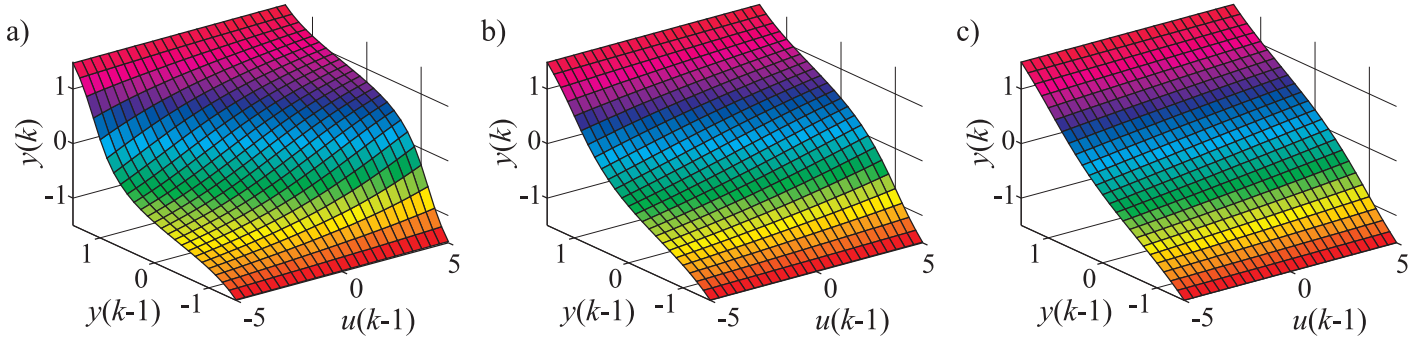


Figure 5: One-step ahead prediction surface for the Wiener system in Fig. 2 (right) for different sampling times (longer times on the left): $T_{95}/T_0 =$ a) 15, b) 30, c) 60

Sampling time effect

One-step ahead predictor

Hereafter, we will highlight the fact of relatively weak nonlinearities in the case of a presentation with the one-step ahead prediction even when systems with strong nonlinearities are concerned.

- The function that is modelled and represents the nonlinear system, $f(\cdot)$ is a one-step ahead predictor.
- If, of course, we want to make a prediction for several steps in the future, we need to make a multiple prediction.
- For the first-order system, a one-step ahead prediction is given by

$$y(k+1|k) = f\left(u(k), y(k)\right) \tag{4}$$

- The two-step ahead forecast is determined by using a one-step ahead forecast and we get

$$y(k+2|k) = f\left(u(k+1), y(k+1)\right) = f\left[u(k+1), f\left(u(k), y(k)\right)\right] \tag{5}$$

- The prediction is a multiple nested function. By increasing the prediction step, the number of embeddings increases.
- For simplicity, we will assume that the input is constant, as in the case of a step excitation function, i.e. $u(k) = u(k+1) = \ldots = u(k+h)$. Then equation (5) becomes $y(k+2|k) = f(f(u(k), y(k)))$ and the output of the model for h steps in the future can be determined by h-step function $f(\cdot)$ nesting

$$y(k+h|k) = \underbrace{f(\cdots f(f}_{h \text{ times}}(u(k), y(k)))\cdots) \tag{6}$$

One-step ahead prediction hyperplane

We are interested in the shape of the hyperplane of function $f(\cdot)$ in equation 2, which is actually modelled with a nonlinear static approximator.

- For the first-order system we can visualize it as $y(k) = f(u(k-1), y(k-1))$.
- The function $f(\cdot)$ is a one-step ahead predictor because it maps the previous values of the inputs and outputs to the current output of the model.
- Figure 2 compares one-step ahead predictions for the first-order Hammerstein and Wiener models .
- Hammerstein model in Fig. 2a (left) is written in equation 3.
- Wiener model is obtained by moving the static nonlinearity to the end, behind a dynamic block, as we can see in Fig. 2a (right). In this way the equation is

$$y(k) = \arctan\left(0.1\, u(k-1) + 0.9 \tan(y(k-1))\right) \tag{7}$$

One-step ahead prediction hyperplane

Linearized Hammerstein model in the operating point (u_0, y_0)

$$y(k) = 0.1 \arctan(u(k-1)) + 0.9y(k-1) \tag{8}$$

$$\Delta y(k) = b_1 \Delta u(k-1) + a_1 \Delta y(k-1) \tag{9}$$

where $\Delta u(k) = u(k) - u_0$ and $\Delta y(k) = y(k) - y_0$ are deviations from the operating point.

- The derivative is obtained in the following way:

$$\begin{aligned} b_1 &= \left.\frac{\partial f(u(k-1), y(k-1))}{\partial u(k-1)}\right|_{(u_0, y_0)} = \frac{0.1}{1 + u_0^2(k-1)} \\ a_1 &= \left.\frac{\partial f(u(k-1), y(k-1))}{\partial y(k-1)}\right|_{(u_0, y_0)} = 0.9 \end{aligned} \tag{10}$$

$$K = \frac{\frac{0.1}{1+u_0^2(k-1)}}{1 - 0.9} \,. \tag{11}$$

- The system expresses a strong nonlinearity, the gain changes between 1 at $u_0 = 0$ and 1/26 at $u_0 = \pm 5$. Nevertheless, the prediction surfaces are only slightly nonlinear.

Series-parallel and parallel models

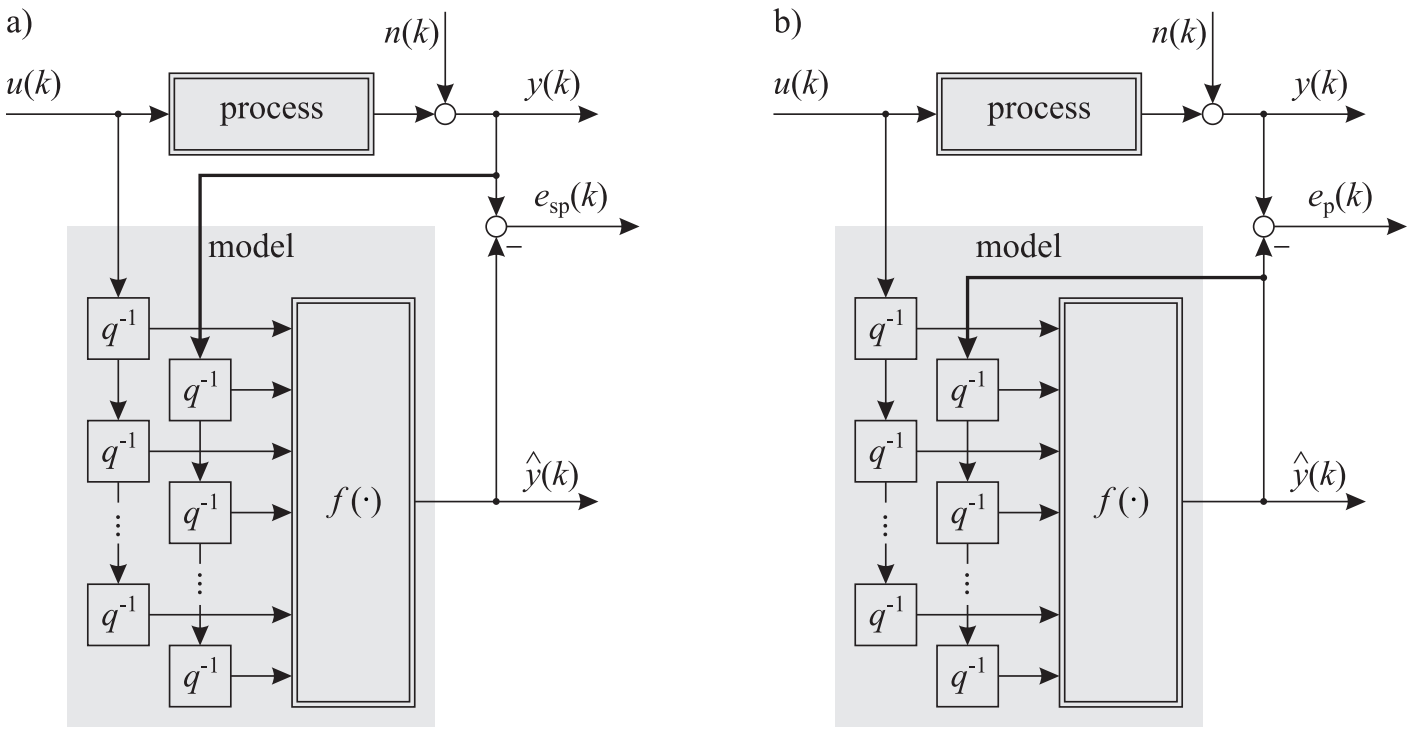


Figure 6: a) One-step ahead prediction and series-parallel model. b) Simulation and parallel model

Series-parallel and parallel models

- The configurations shown in Fig. 6 do not differ only in the operating phase, but also in the learning phase.
- Models can be trained by minimizing the criterion function, which depends on the error between the output of the process and the model.
- For series-parallel models, the error is called *input-output error or equation error* $e_{\mathrm{sp}}(k)$
- for parallel models *output error* $e_{\mathrm{p}}(k)$.
- This is also in line with the classical terminology in the field of linear identification.

- For a second-order model, a one-step ahead prediction is calculated based on the delayed process outputs

$$\hat{y}(k) = f(u(k-1), u(k-2), y(k-1), y(k-2)) \tag{12}$$

- while in the case of a simulation model it is calculated according to the delayed outputs of the model as

$$\hat{y}(k) = f(u(k-1), u(k-2), \hat{y}(k-1), \hat{y}(k-2)) \tag{13}$$

Series-parallel and parallel models

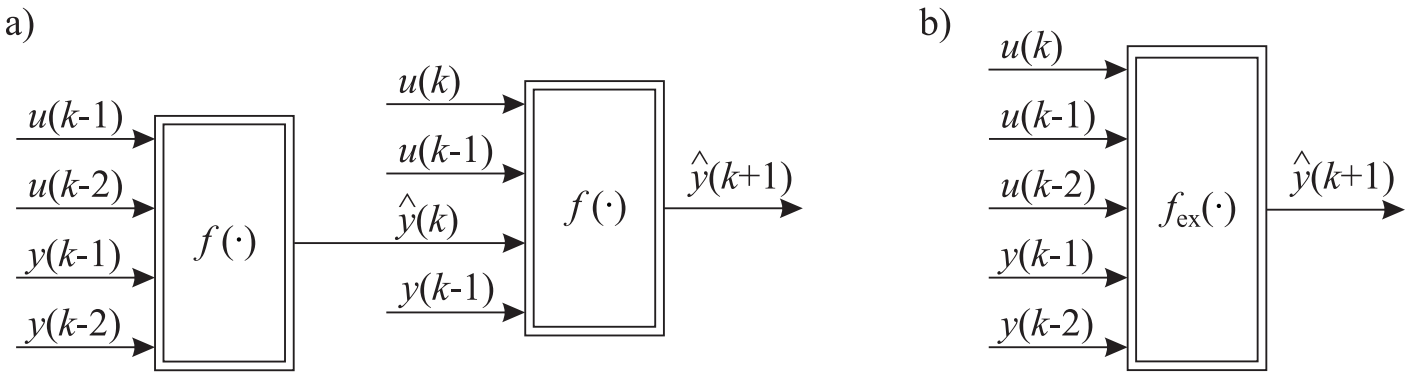


Figure 7: Two-step ahead prediction with a) cascading application of the one-step ahead prediction and b) an expanded predictor that performs a two-step ahead prediction in one step

Nonlinear classes of dynamic input-output models

- By analogy with linear models (Ljung), nonlinear models can be labelled with “ N ” for “ nonlinearity ” in front of the linear model class.
- All nonlinear models can be written in the form of an input-output relation

$$\hat{y}(k) = f\left(\underline{\varphi}(k)\right) \tag{14}$$

where in the regression vector $\underline{\varphi}(k)$ the delayed and current inputs, the delayed outputs of the model and the delayed prediction errors are written. We can distinguish between the models with the feedback loop and models without the feedback loop.

Models with feedback loop

- As for linear systems, models with the feedback loop are most common for nonlinear models.
- Regression vector $\underline{\varphi}(k)$ contains delayed inputs and outputs of a process or model, and possibly also errors in the prediction.
- The three most common structures of linear models are ARX, ARMAX and OE models.
- Analogously, there are models with the following regressors in the nonlinear field:

$$\begin{aligned}
\text{NARX}: \quad & \underline{\varphi}(k) = [u(k-1) \ \cdots \ u(k-m) \ \ y(k-1) \ \cdots \ y(k-m)]^{\mathrm{T}} \\
\text{NARMAX}: \quad & \underline{\varphi}(k) = [u(k-1) \ \cdots \ u(k-m) \ \ y(k-1) \ \cdots \ y(k-m) \\
& \qquad e(k-1) \ \cdots \ e(k-m)]^{\mathrm{T}} \\
\text{NOE}: \quad & \underline{\varphi}(k) = [u(k-1) \ \cdots \ u(k-m) \ \ \hat{y}(k-1) \ \cdots \ \hat{y}(k-m)]^{\mathrm{T}}
\end{aligned}$$

Models without feedback loop

- If there is no feedback loop, then the regression vector $\underline{\varphi}(k)$ contains only delayed or filtered input signals.
- The number of required regressors in the case of models without the feedback loop is much higher.

- *Nonlinear Finite Impulse Response (NFIR)*
- Models with the feedback loop have many disadvantages (learning, problem of stability, error accumulation); therefore, we try to solve certain problems using nonlinear finite impulse response models (NFIR) because they do not introduce any feedback loops.
- The regression vector for the NFIR model consists only of delayed process inputs:

$$\text{NFIR}: \ \underline{\varphi}(k) = [u(k-1)\ u(k-2)\ \cdots\ u(k-m)]^{\mathrm{T}}\,. \tag{15}$$

- The price for not using the feedback loop is a large number of m delays, which are needed to describe the process dynamic.

Limitations of nonlinear input-output models

- Input-output models can describe a very large class of different systems.
- They are not as general as nonlinear state models, where nonlinearities can be realized on states.
- They are limited to nonlinearities that are invertible. We can not use them in cases of hysteresis and backlash, where unmeasurable states play an important role.
- The last limitation can be illustrated on the example shown in Fig. 8. A Wiener type process with a noninvertible nonlinearity, which can not be modelled with an input-output model, since the state $x(k)$ is not measurable.
- Suppose there is a dynamic system in Fig. 8, where the Wiener process is defined by equation

$$x(k) = b_1 u(k-1) - a_1 x(k-1)\,, \qquad y(k) = g\left(x(k)\right) \tag{16}$$

- The output of the Wiener process is written as

$$y(k) \quad = \quad g\left(b_1 u(k-1) - a_1 x(k-1)\right) \tag{17}$$

- If $g^{-1}(\cdot)$ exists, then equation 17 can be written as a nonlinear input-output model of the first order with respect to equation 2

$$y(k) \quad = \quad g\left(b_1 u(k-1) - a_1 g^{-1}(y(k-1))\right) = g\left(u(k-1), y(k-1)\right)) \tag{18}$$

- Otherwise we can not formulate it as an input-output model.
- We can, of course, approximate it, but not with any precision. The requirement of invertibility of function $g(\cdot)$ is a rather strict requirement limiting the use of this type of model to monotone nonlinearities $g(\cdot)$.

Limitations of nonlinear input-output models

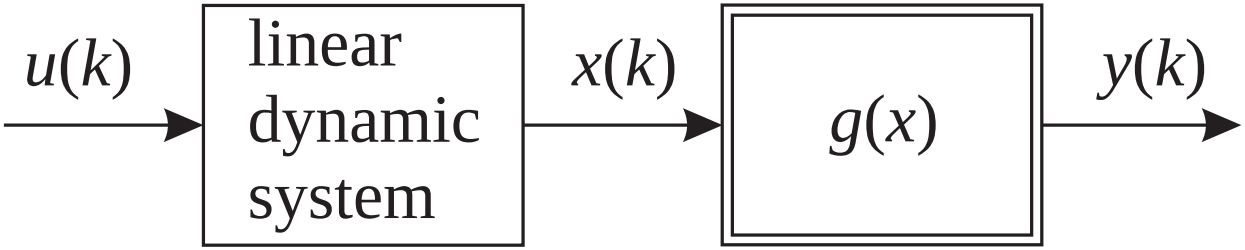


Figure 8: We can not model the Wiener system with an input-output model if $g(\cdot)$ is not invertible

Excitation of systems for generating data

- The most important steps in identifying systems include the planning of appropriate measurements.
- Excitation of the system with appropriate excitation signals.
- This step is even more important in the case of nonlinear systems, because nonlinear systems are significantly more complex and therefore the measurements must contain significantly more information.
- In this case, of course, the a priori information on the basis of which we can adequately plan the measurements is very important.

Excitation of systems

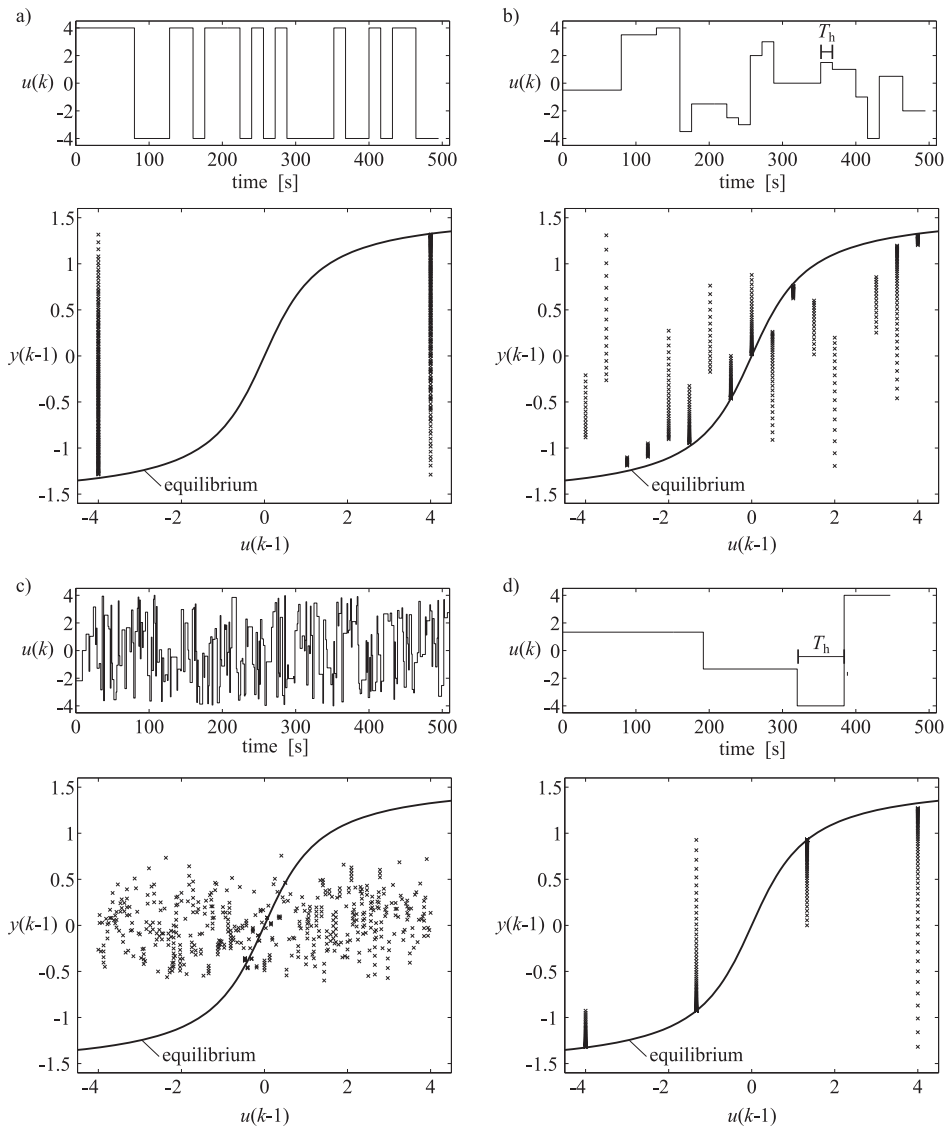


Figure 9: Excitation of nonlinear dynamic systems: a) binary PRBS, b) APRBS with appropriate hold time, c) APRBS with short hold time, d) APRBS with long hold time ($T_h \approx T_{dominant}$)

Excitation of systems

In addition to the more general instructions given above, we can also add some instructions for designing excitation signals:

1. *Purpose of the model:* First, we need to find out for what purpose we are building the model. Will we use it for control, fault detection, prediction, or optimization? From this we can determine the necessary precision in different operating modes and frequency ranges. For example, for control purposes, the accuracy around the cutoff frequency is the most important, and the errors at low frequencies are simply eliminated by an integrated controller action. Errors over the cutoff frequency are outside the range of control elements and the closed-loop frequency range.
2. *Maximum length of the learning set:* With a larger learning set, the model will be more precise in the appropriate distribution of data. In most real cases, the measurement time is limited. However, the size of the set may also depend on hardware limitations.
3. *Properties of different input signals:* For each input signal of the system it is necessary to determine whether it is necessary to perform static or dynamic measurements. In other words, we need to determine the functionality of the input variable. If the input variable is a control signal, dynamic measurements will surely be required. In other cases, however, static measurements can suffice, as in the case of slowly changing measured disturbances in the control systems.
4. *Input signal space:* The process must be excitated throughout the operating space, which can occur during system operation. Learning data must contain data between the limit values of input signals, since interpolation is essentially more important than extrapolation.
5. *Even data distribution:* Especially in modelling for control purposes, it is necessary to ensure that the output data is evenly distributed. In this way they contain the same amount of information about all operating points.
6. *Dynamic properties:* The dynamic of signals must be appropriately selected so that different dynamics can be excitated in different operating points.

Order determination

Order determination:

- **Lipschitz coefficients**
- **PCA method**

- In the case of nonlinear models with external dynamic, determining the order of the system is equivalent to determining the corresponding inputs for the function $f(\cdot)$ in equation 14.
- Order determination is, in this way, equal to input determination, and therefore the procedure explained below is applicable in the case of dynamic as well as in the case of static systems.
- It is important to understand that the delayed input $u(k-i)$ and the delayed output $y(k-i)$ can be two inputs in function $f(\cdot)$ and that they contain properties that cause the order determination to be essentially more demanding.
- For example, $y(k-1)$ and $y(k-2)$ are highly correlated (which brings redundancy in the information), but both are relevant.

Lipschitz coefficient method for determination of system order

- The method is based on data and does not require any assumption about the architecture or structure of the model.
- The only method constraint is the smooth behaviour of the process output, which is also common to models with external dynamic and a general identification model.
- The basic idea of the method is presented in Fig. 10a.

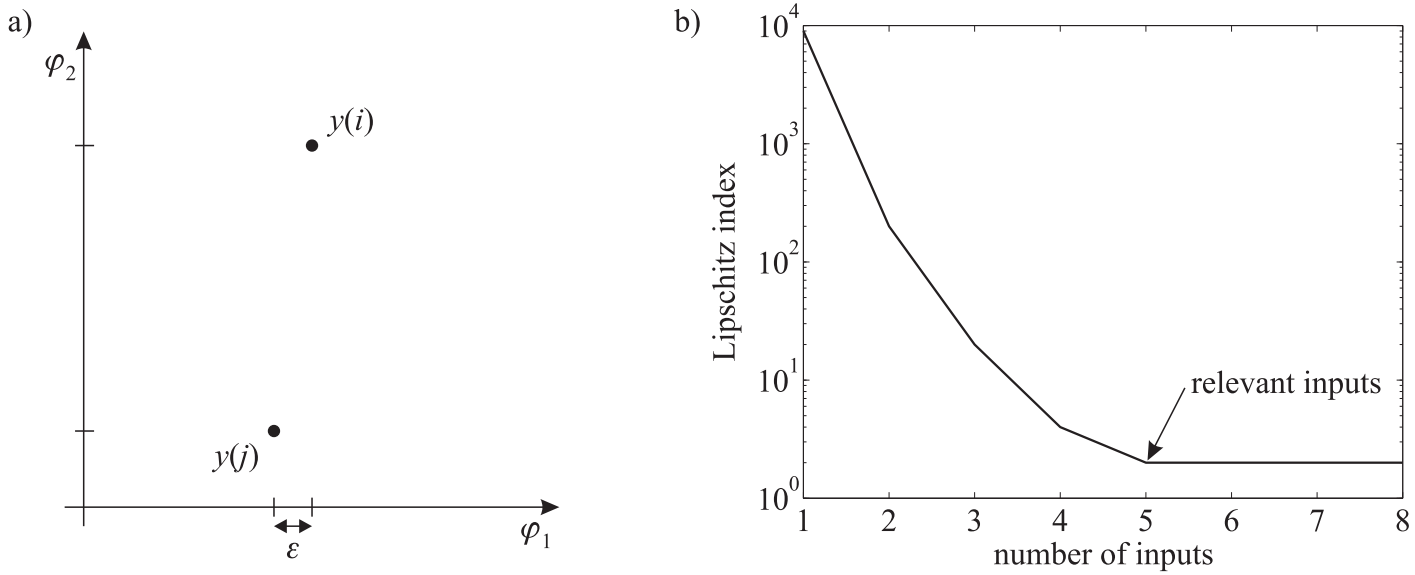


Figure 10: a) Let's take two samples that are close to each other in the space of variable φ_1. If we look at the value of the function for two different values of the variable φ_2, then the output may have very different values of $y(i)$ and $y(j)$ if the function also depends on another variable, or if $y(i) \approx y(j)$ if the function depends only on φ_1. b) The Lipschitz index detects an instance when all relevant inputs are included.

Lipschitz coefficient method for determination of system order

- First of all, we need to define the appropriate inputs of the function

$$y = f(\varphi_1, \varphi_2, \ldots, \varphi_n) \tag{19}$$

  from the set of potential measured inputs $\varphi_1, \varphi_2, \ldots, \varphi_o$ $(o > n)$ that are available.
- If φ_i are determined physical input signals, then equation 19 describes the problem of static approximation; but if there are delayed input and output signals, then this is the model with external dynamic.

  Idea
- If we assume that function f depends on $n-1$ inputs, but in fact, the function depends on n inputs, then a set of learning data contains at least two points that are very close to each other (maybe even identical) in the space dimension $n-1$, but they are very different with regard to the input n.
- This example is in Fig. 10a at $n = 1$. Two points i and j are close to each other in the input space determined by variable φ_1, but they are far apart if we look at the points in the space of the variables φ_1 and φ_2.
- Since these two points are close to each other in space defined by $n-1$ inputs (in our case φ_1), it would be expected that the $y(i)$ and $y(j)$ are also close together. If one or more inputs are missing, then it is natural that $y(i)$ and $y(j)$ have completely different values.
- In such a case, we can conclude that $n-1$ inputs are not sufficient and we need to take the $n-$th input into consideration.
- The Lipschitz quotient is an index that is large if one or more inputs are missing (larger if multiple inputs are missing) and small in the opposite case.
- Using the Lipschitz index in Fig. 10b, we obtained the result $n = 5$ and $o = 8$. In this way, the real number $(n = 5)$ is determined where the Lipschitz index stops falling.

Lipschitz quotient in a one-dimensional case (input φ)

- We define the item as

$$l_{ij} = \frac{|y(i) - y(j)|}{|\varphi(i) - \varphi(j)|} \quad \text{for } i = 1, \ldots, N,\ j = 1, \ldots, N \ \text{ and } i \neq j\,, \tag{20}$$

  where N is the number of measurements in the learning set. Let us mention that in this way we define $N(N-1)$ Lipschitz quotients and we must calculate them $N(N-1)/2$ since $l_{ij} = l_{ji}$.
- Since the final differential is in equation 20 as the approximation of the derivative $df(\varphi)/d\varphi$, we must limit it with the maximum slope of the function $f(\cdot)$ if $f(\cdot)$ is smooth.
- For the multi-dimensional example, equation 20 must be expanded:

$$l_{ij}^{(n)} = \frac{|y(i) - y(j)|}{\sqrt{(\varphi_1(i) - \varphi_1(j))^2 + \ldots + (\varphi_n(i) - \varphi_n(j))^2}} \tag{21}$$

  for $i = 1, \ldots, N, j = 1, \ldots, N$ and $i \neq j$ and the upper index (n) in $l_{ij}^{(n)}$ defines the number of inputs.

Lipschitz coefficient method for determination of system order

- **Lipschitz index** is defined as the maximum Lipschitz quotient

$$l^{(n)} = \max_{i,j,(i \neq j)} \left(l_{ij}^{(n)} \right) \tag{22}$$

- If n is too small, and not all relevant inputs are included, then the Lipschitz index is high due to the situation in Fig. 10a.
- If we include all relevant inputs, then the maximum Lipschitz coefficient in equation 22 remains low and constant.

Lipschitz coefficient method for determination of system order

- The strategy requires that inputs are sorted.
- For example, for a nonlinear dynamic system we can choose the inputs $\varphi_1 = u(k-1)$, $\varphi_2 = y(k-1)$, $\varphi_3 = u(k-2)$, $\varphi_4 = y(k-2)$, etc. This can lead to finding the order of the system, but it is not possible to identify the dead time.

- Two important weaknesses can be identified with this method. The first is noise sensitive and the second is the distribution of data.
- Regardless of all, the method is very useful because it is not very computationally demanding, especially in the case of minor complexities.

PCA method

- We add an appropriate number of time-delayed input data to the input matrix

$$\underline{X}_l(k) = [\underline{X}(k),\ \underline{X}(k-1), \cdots, \underline{X}(k-l)] \tag{23}$$

- We need to find the number of time-delayed input data l which does not add any new collinear relations between data in the covariance matrix.
- Optimal number with which we can describe the dynamic of the measured data is $l-1$.
- Data matrix consists of delayed input matrices $\underline{X}_i(k),\ i = 1, ..., l$. With the right number l (order of the system) we assure that the static as well as dynamic components appear in the data space.
- Most often the number l is determined iteratively.
- **Step without delay** $l = 0$. Firstly, we deal with a static case ($l = 0$) and determine the number of static linear relations which is equal to the difference between the number of all variables (columns $\underline{X}(l)$) and the number of specified main components.

PCA method for determination of system order

Step with one delay $l = 1$

- In the next step, we take $l = 1$ and determine the number of newly discovered dynamic relations. We get this by subtracting from all the variables the number of main components and the number of the discovered collinear relations r_l in the previous steps, where with index l we define the number of time delays.
- Assume that we have an example of a data matrix $\underline{X}(k)$ (static data) in which we have r_0 collinear relations. If we add a matrix of delayed data to this matrix, we get a new data matrix, which is defined as

$$\underline{X}_1(k) = [\underline{X}(k),\ \underline{X}(k-1)]$$

- The number of collinear columns in this matrix is equal to $r_1 = m - s - 2r_0$, where m is the number of columns in matrix $\underline{X}_1(k)$, s is the number of main components of matrix $\underline{X}_1(k)$, where we take into account all components that have non-zero eigenvalues and r_0 number of collinear columns in the $\underline{X}(k)$ matrix.
- Since the statistical properties of the matrix $\underline{X}(k)$ are equal to the properties of the shifted matrix $\underline{X}(k-1)$, the number of collinear columns r_0 is double in the composite matrix $\underline{X}_1(k)$.

PCA method for determination of system order

Step with two delays $l = 2$

- By adding time-delayed data, we get a matrix

$$\underline{X}_2(k) = [\underline{X}(k),\ \underline{X}(k-1),\ \underline{X}(k-2)]$$

The number of collinear columns in this matrix is equal to $r_2 = m - s - 2r_1 - 3r_0$, where m is the number of columns in matrix $\underline{X}_2(k)$, s is the number of principal components of matrix $\underline{X}_2(k)$ and r_1 is the number of collinear columns in matrix $\underline{X}_1(k)$. Because statistical properties of matrix $\underline{X}_1(k)$ are equal to the properties of the shifted matrix $[\underline{X}(k-1),\ \underline{X}(k-2)]$, the number of collinear columns r_1 in composite matrix $\underline{X}_2(k)$ is double. The number of collinear columns r_0 is tripled because there are three subsets of columns in matrix $\underline{X}_2(k)$ with the same statistical properties as in matrices $\underline{X}(k),\ \underline{X}(k-1)$ and $\underline{X}(k-2)$.

- Generally we write the number of newly discovered collinear columns with the following formula that applies to $l \geq 1$:

$$r_l = m - s - \sum_{i=1}^{l} (i+1)\, r_{l-i} \tag{24}$$

In the case of $l = 0$, the number of discovered collinear relations is $r_0 = m - s$. Iterative process of calculating newly discovered collinear columns ends when the number of detected collinear relations r_l equals zero. The number of delays that represents the optimal structure of the notation of the dynamic system is in this case equal to $l - 1$.

Example of the structural identification of the second-order system with the PCA method

Example

- The second order system is given with a difference equation

$$y(k+2) = 1.92y(k+1) - 0.93y(k) + 0.02u(k)$$

- The system is excitated with the pseudorandom binary signal $u(k)$

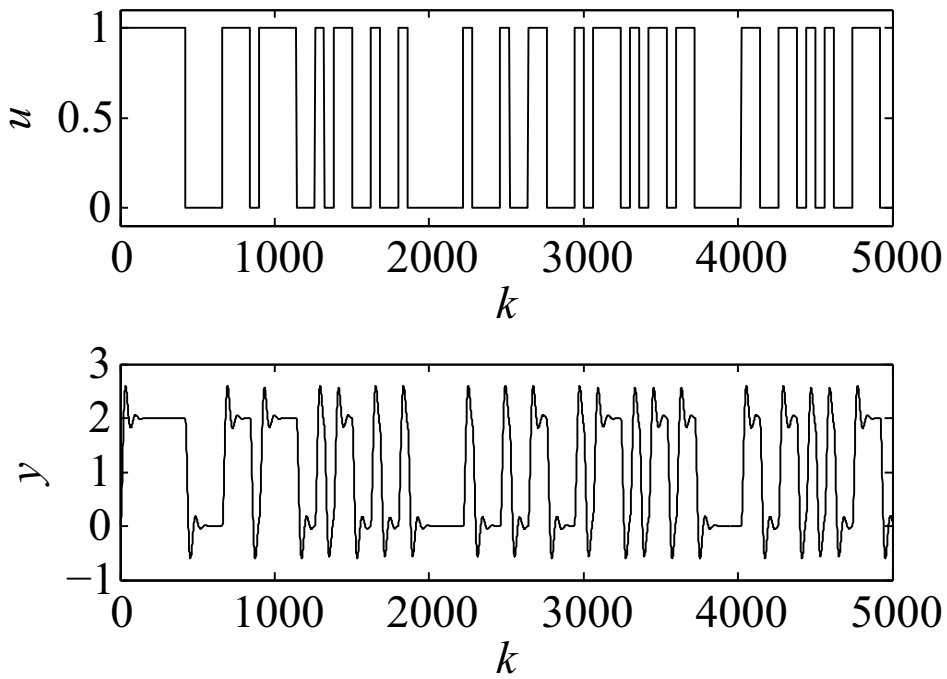


Figure 11: Input and output signals in the case of structural identification with the PCA method.

Example of the structural identification of the second-order system with the PCA method

- In the data analysis we write the results in Tab. 1.
- Eigenvalues in the case of data matrix without shifting $\underline{X}_0(k)$ are different from zero, so there are no collinear columns in the matrix.
- In the case of matrix $\underline{X}_1(k)$, which has four columns $(m = 4)$, the number of principal components $s = 1$.
- For $l = 2$ we get $s = 1$.
- To move $l = 3$, the number of principal components $s = 1$.
- With this shift value we stop adding delayed data, because $r_3 = 0$.

l	m	s	r_l
0	2	2	$r_0 = m - s = 0$
1	4	1	$r_1 = m - s - 2r_0 = 3$
2	6	1	$r_2 = m - s - 2r_1 - 3r_0 = -1$
3	8	1	$r_3 = m - s - 2r_2 - 3r_1 - 4r_0 = 0$

Table 1: Procedure for structural identification using the PCA method.

Example of the structural identification of the second-order system with the PCA method

- From Tab. 1 we find that by adding the matrix that is delayed for three samples $(l = 3)$ we do not detect any new collinear data and therefore we can say that the dynamic system we are dealing with is the second-order system.
- In the case of two delays $(l = 2)$ we have one collinear relation between columns $(r_2 = 1)$ in matrix $\underline{X}_2(k)$.
- Residuum $\underline{E}_r^T \underline{E}_r$ can be written in the form of a singular value decomposition

$$\underline{E}_r^T \underline{E}_r = \underline{\lambda}_{26} \underline{p}_{26} \underline{p}_{26}^T$$

- Eigenvector that belongs to the zero eigenvalue describes the process model, which can be written with the following equation

$$\underline{x}^T \underline{p}_{26} = 0 \tag{25}$$

- In our case, we get:

$$-0.4244y(k) + 0.8149y(k-1) + 0.0085u(k-2) - 0.3947y(k-2) = 0$$

If in the above equation we explicitly express the variable $y(k)$, we get a notation of the data model that is in the considered (ideal) case:

$$y(k+2) = 1.9201y(k+1) - 0.9300y(k) + 0.0200u(k)$$

Example of the structural identification of the second-order system with the PCA method

Example with noise

- Assume the noise of output data measurements $\mathcal{N}(0, 0.01)$.
- In this case the measurements are shown in Fig. 12.

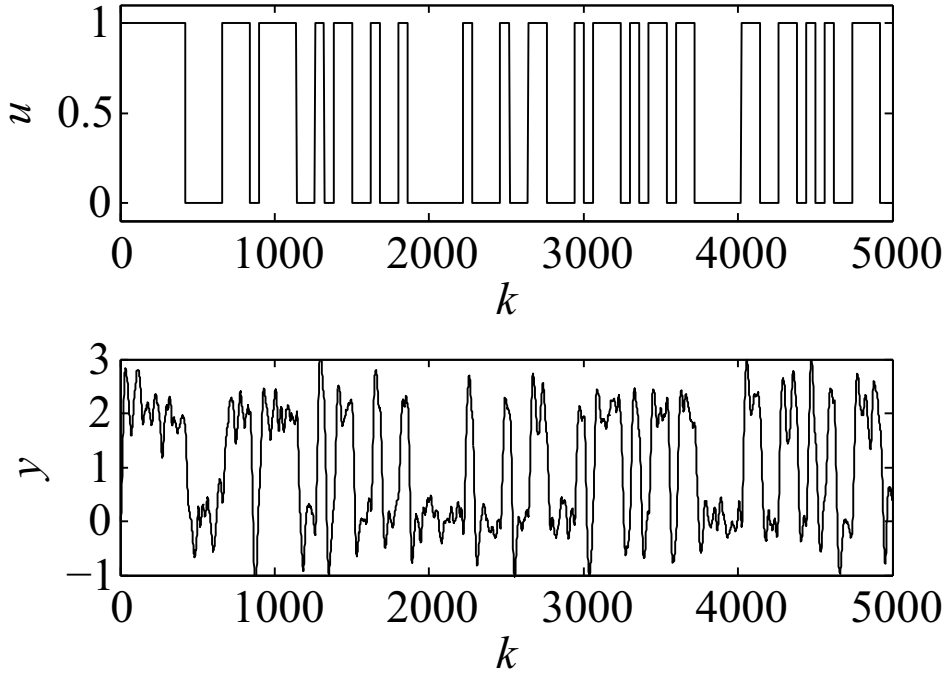


Figure 12: Input and output signals with added noise in case of structural identification with the PCA method

- We get to the same table as in the case of data without noise (Tab. 1).
- The identified difference equation of the model is:

$$y(k+2) = 1.9447y(k+1) - 0.9535y(k) + 0.0177u(k)$$

.

Adaptive control

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for the 15th Lecture, May 2016

Table of contents

1. Adaptive control
2. Adaptive control principles
3. Classification of adaptive systems according to the control scheme
   - Open-loop adaptive systems
   - Closed-loop adaptive systems
     - Model-reference adaptive control
     - Example: Lyapunov-based direct model-reference adaptive controller
     - Model-identification adaptive control
4. Classification of adaptive systems according to their functional subunits
   - Direct adaptive control
   - Indirect adaptive control
5. Direct fuzzy adaptive control
   - Direct fuzzy adaptive control
     - Example: Lyapunov-based direct fuzzy model-reference adaptive controller
   - Indirect fuzzy adaptive control

Adaptive control

The term *adaptation* appears for the first time in biology and represents *useful adjustment of the organism to changes in its environment*. Inspired by this idea, in 1957 *Drenick* and *Shahbender* introduced the term *adaptive system* into the field of control theory. By their definition, the adaptive systems are systems that control their own behaviour and correct their own states and parameters in terms of more appropriate behaviour.

Definition 1

Adaptive system is a system that constantly supervises its behaviour and changes its parameters and states in the sense of approaching to the optimal solution of the selected criterion.

Definition 2

Adaptive control is an intelligent feedback control that changes its parameters and states in a changing environment in a way that it operates optimally according to the selected criterion.

Definition 3

In 1959, Bellman and Kalaba introduced the term adaptive control in terms of control of a system for which we do not have complete information. According to this definition, adaptive control is the last level in development of the process control.

Adaptive control

- If we have complete information about a process that is the subject of control, then we can talk about a deterministically controllable process.
- When unknown elements which can be mathematically written as random variables with known distribution are present in the process, then we are talking about a stochastically controllable process.
- The third level is represented by the systems where even less information is available. In this case, the controller needs to improve its behaviour based on observation of system outputs (adaptive mechanism). With sufficient information, the controller's decisions become correct in terms of improving the behaviour of the system. Bellman and Kalaba define these systems as adaptively controllable systems.

Adaptive control principles

In the field of automatic process control the implication of the above thinking was first given by Feldbaum in 1965. Assume that we have **controller** A and **process** B for which we design the control. The problem can be defined as **deterministic, stochastic** or **adaptive** according to the available information about A and B.

- If characteristics of the process B are known, then we can design a selected control strategy with a known classical method.
- On the other hand, in case of unknown characteristics of system B, these need to be identified first. In this case, the control mechanism has to simultaneously address two problems that are closely related, but of a different nature.
- First, the mechanism has to determine the characteristics of the system based on collected information. Then by the use of this information it has to determine the way of action required for successful control.
- The first problem is the system identification problem and the second one is the control problem. The whole mechanism is called *dual control*.

Classification of adaptive systems according to the control scheme

In this section, we will discuss the most important structures of **adaptive systems**.

- The main property of adaptive systems is acquisition of information about an unknown process during its operation and changing the control laws according to this information.
- We can say that generally accepted definition of *adaptive systems* actually does not exist.
- Most adaptive systems are divided according to their structure into **open-loop adaptive systems** and **closed-loop adaptive systems**, depending on the structure of the adaptive mechanism, which can be in a closed-loop or in an open-loop, without feedback.

Open-loop adaptive systems

Open-loop adaptive system is based on the assumption that the changing of the process behaviour can be established by using measurements of signals that are present in the system.

- We need to know the impact of the measured signals on the behaviour of the system to realize an open-loop adaptive system.
- The first examples of the above mechanism have been known under the name *gain scheduling* since 1950.
- The principle has been expanded in a way that enables setting other parameters.
- The principle is based on the fact that we can calculate the parameters of the controller for each operating range or the value of a particular measured signal.
- Parameter values are stored in a rule table. When a certain value of the measured signal appears, we can determine the values of the associated controller's parameters from the rule table.

- **The advantage** of open-loop adaptive systems is in a quick reaction to process changes that are already known and foreseen in advance and therefore they do not need to be identified at the time of the experiment.
- **The disadvantages** are that all unforeseen changes in the system are ignored, the number of stored parameters that are intended for certain operating conditions can be very large and the principle is limited to processes where the parameters change slowly.

Open-loop adaptive systems

- A typical diagram of an open-loop adaptive system is shown in figure 1, where $z(t)$ is a measurable signal that gives us information on operating conditions and on the basis of which we calculate or select the predefined parameters of the controller.

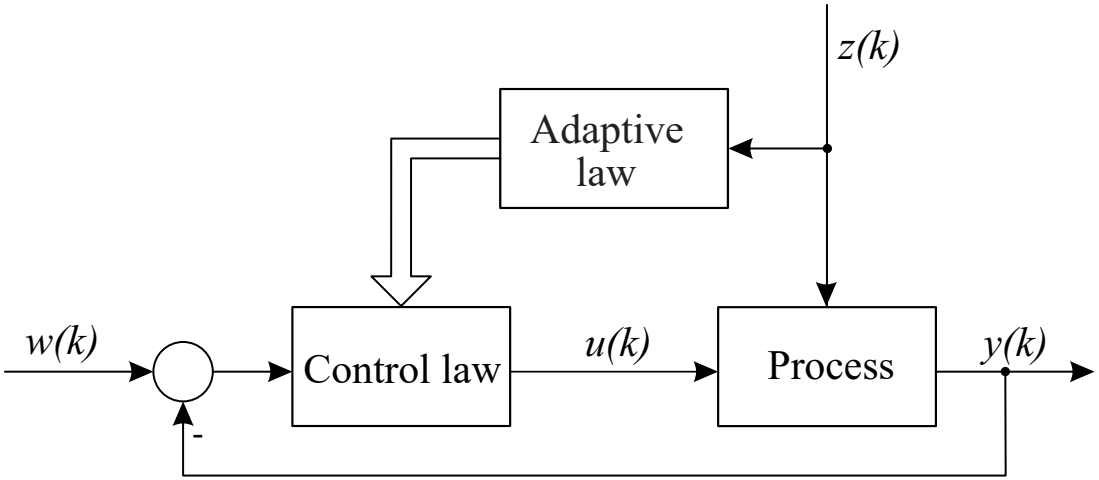


Figure 1: Open-loop adaptive system

Closed-loop adaptive systems

- In the case when the changes in the process operation can not be uniquely detected by measuring external signals which are present in the system, the problem needs to be solved with a closed-loop adaptive principle.
- The structure of the closed-loop adaptive system is shown in figure 2.

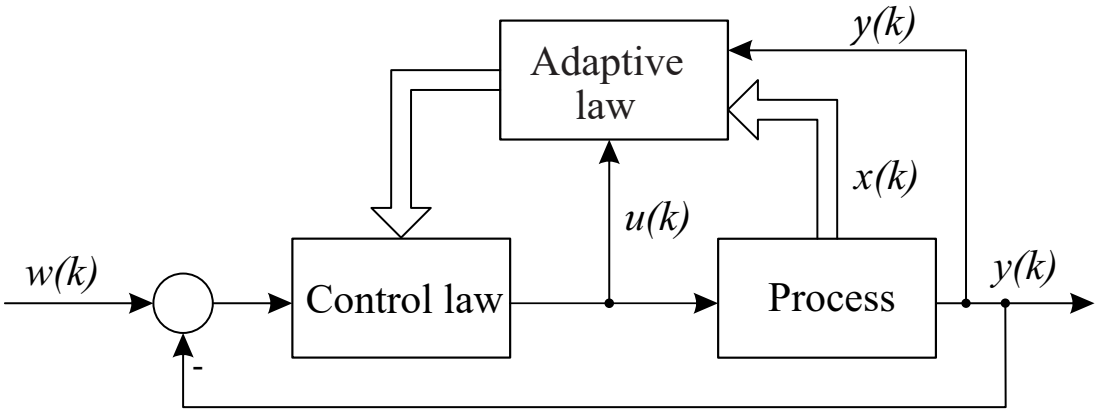


Figure 2: Closed-loop adaptive system

Model-reference adaptive control

- Model-reference adaptive control is formed in a way that closed-loop response of the system tracks the desired reference which is given by the reference model.
- The principle requires a measurable reference $w(t)$, which at the same time is an input to the reference model.
- Controller parameters are tuned according to the adaptive mechanism in a way that error between reference model output and process output is minimized.
- Adaptive mechanism is obtained on the basis of optimum theory. Minimization, along with the assumption of certain constraints, leads to an adaptive algorithm that is nonlinear and time-varying.
- In the case of model-reference adaptive systems the adaptation mechanism is divided into three parts: **comparison of the closed-loop operation with the reference model**, **calculation of controller parameters according to the adaptive law** (which is obtained on the basis of optimum theory) and **change of controller parameters**.
- A typical diagram of model-reference adaptive control is represented in figure 3.

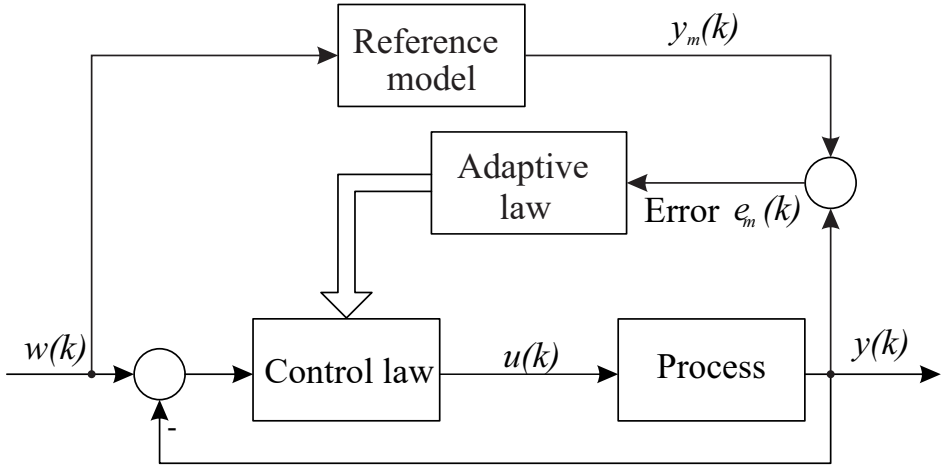


Figure 3: Model-reference adaptive control diagram

Example: Lyapunov-based direct model-reference adaptive controller

- When control law is being derived, we assume that we have a first-order process that can be written in the form of a transfer function such as:

$$G_p(s) = \frac{Y_p(s)}{U(s)} = \frac{b}{s+a} \tag{1}$$

  where $Y_p(s)$ is the output of the process and $U(s)$ is the input to the process.
- We want that the output of the process tracks the first-order reference model with transfer function

$$G_r(s) = \frac{Y_r(s)}{W(s)} = \frac{b_r}{s+a_r} \tag{2}$$

  where $Y_r(s)$ is the output of the reference model and $W(s)$ is the reference.
- To ensure that the process tracks the reference model, we need to determine the values of the open-loop gain f and the feedback loop gain q, which are present in the following control law:

$$u(k) = fw(k) - qy_p(k) \tag{3}$$

Example: Lyapunov-based direct model-reference adaptive controller

- Block diagram of the control law.

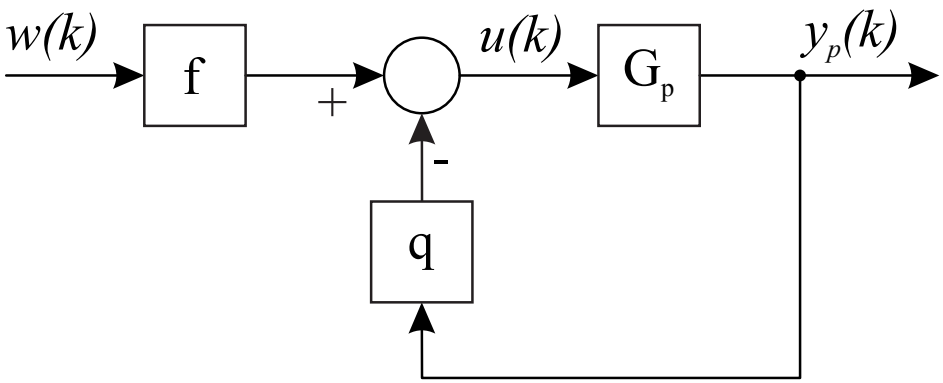


Figure 4: Basic block diagram of model-reference adaptive controller.

Example: Lyapunov-based direct model-reference adaptive controller

- The closed-loop transfer function of the process G_p, which is controlled by the control law 3, is:

$$G_w(s) = \frac{Y_p(s)}{W(s)} = \frac{fb}{s + a + bq} \tag{4}$$

- As we mentioned at the beginning, we want that the process output tracks the output of the reference model. That means that the error between the reference model and our closed-loop transfer function must be kept to a minimum. The error is obtained by subtracting the differential equations of both transfer functions. The differential equation for the reference model is:

$$\dot{y}_r(t) + a_r y_r(t) = b_r w(t) \tag{5}$$

and for a closed-loop transfer function:

$$\dot{y}_p(t) + (a + bq) y_p(t) = fbw(t) \tag{6}$$

- If we subtract the equations, we get:

$$\dot{y}_p - \dot{y}_r + (a + bq) y_p - a_r y_r = fbw - b_r w \tag{7}$$

Then on both sides of the equation we add $a_r y_p$ and take into account the error $e = y_p - y_r$

$$\dot{e} + a_r e = (fb - b_r) w - ((a + bq) - a_r) y_p \tag{8}$$

- For the parameters we introduce auxiliary variables:

$$\tilde{b} = fb - b_r \tag{9}$$

$$\tilde{a} = a + qb - a_r \tag{10}$$

Example: Lyapunov-based direct model-reference adaptive controller

- An adaptive law for the gains f and q is obtained by means of positive definite Lyapunov function:

$$V(e, \tilde{a}, \tilde{b}) = e^2 + \frac{1}{\gamma_f}\tilde{b}^2 + \frac{1}{\gamma_q}\tilde{a}^2 \tag{11}$$

  whose derivative is

$$\dot{V}(e, \tilde{a}, \tilde{b}) = -2a_r e^2 + 2\tilde{b}we - 2\tilde{a}y_p e + \frac{2}{\gamma_f}\tilde{b}\dot{\tilde{b}} + \frac{2}{\gamma_q}\tilde{a}\dot{\tilde{a}} \tag{12}$$

  where equation 8 was taken into account for calculating the derivative of the error.
- Stability criterion is provided by $\dot{V}(e, \tilde{a}, \tilde{b}) \leq 0$. If we assume that the process is time-invariant, then we can obtain the following adaptive law:

$$\dot{\tilde{b}} = -\gamma_f ew \tag{13}$$

$$\dot{\tilde{a}} = \gamma_q ey_p \tag{14}$$

  since $\dot{\tilde{b}} = \dot{f}b + f\dot{b}$ and $\dot{\tilde{a}} = \dot{q}b + q\dot{b}$ and since we assume slow changes in the process parameters ($\dot{b} \approx 0$, $\dot{\tilde{b}} = \dot{f}b$ and $\dot{\tilde{a}} = \dot{q}b$). Then the adaptive law for gains f and q equals to

$$\dot{f} = -\frac{\gamma_f}{b}ew \tag{15}$$

$$\dot{q} = \frac{\gamma_q}{b}ey_p \tag{16}$$

Example: Lyapunov-based direct model-reference adaptive controller

- If we know the sign of the gain or parameter b, the adaptive law can be simplified in the following form:

$$\dot{f} = -\gamma_f^* sign(b) ew \tag{17}$$

$$\dot{q} = \gamma_q^* sign(b) ey_p \tag{18}$$

where γ_f^* and γ_q^* are arbitrary positive constants.

- The present controller is globally stable in the vicinity of the operating point, where the process is sufficiently described by the linear model.
- If we are dealing with a nonlinear process that we want to control throughout the entire operating area, the controller needs to be expanded. The easiest way to do so is by introducing the concept of fuzzy logic.

Model-identification adaptive control

- Model-identification adaptive control (MIAC) is sometimes called self-organizing or auto-tuning adaptive control.
- For this principle three phases are characteristic: gathering information on current process behaviour, optimization of certain criterion function and adjustment of the controller.
- Depending on the way information is obtained, several different types of model-identification adaptive systems can be distinguished.
- Parameters of the controller are calculated by fulfilling certain performance criteria or closed-loop behaviour.
- Adjustment of the controller means replacing the old parameters of the controller with the new calculated values.

Model-identification adaptive control

- The essence of the approach is to achieve the optimal process modelling and optimal behaviour of the closed-loop system irrespective of input signal and operating conditions.
- Characteristic structure is shown in figure 5.

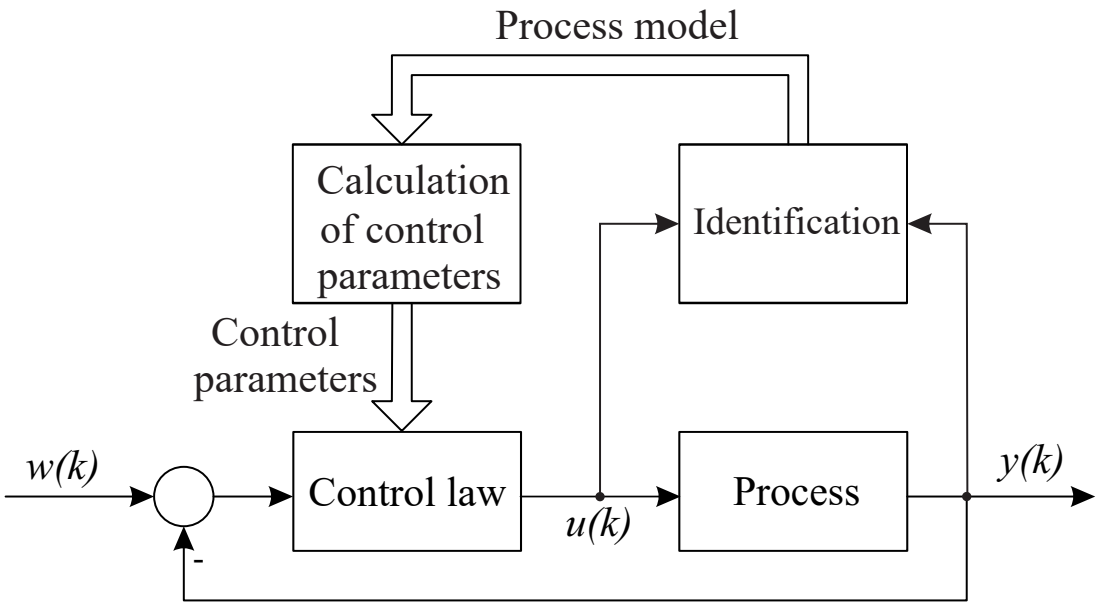


Figure 5: Diagram of model-identification adaptive control

Classification of adaptive systems according to their functional subunits

Another classification of adaptive systems that was introduced by Narendra and Annawamy represents a very rough partition and is based on functional subunits of the adaptive system. Accordingly we divide adaptive control into:

- **direct**
- and **indirect**.

Direct adaptive control

- The first approach is called **direct adaptive control**.
- This approach is possible in some cases when we can modify the system directly by modifying the controller parameters with respect to certain performance criteria, which the whole system must meet.
- The principle of direct adaptive control is shown in figure 6.

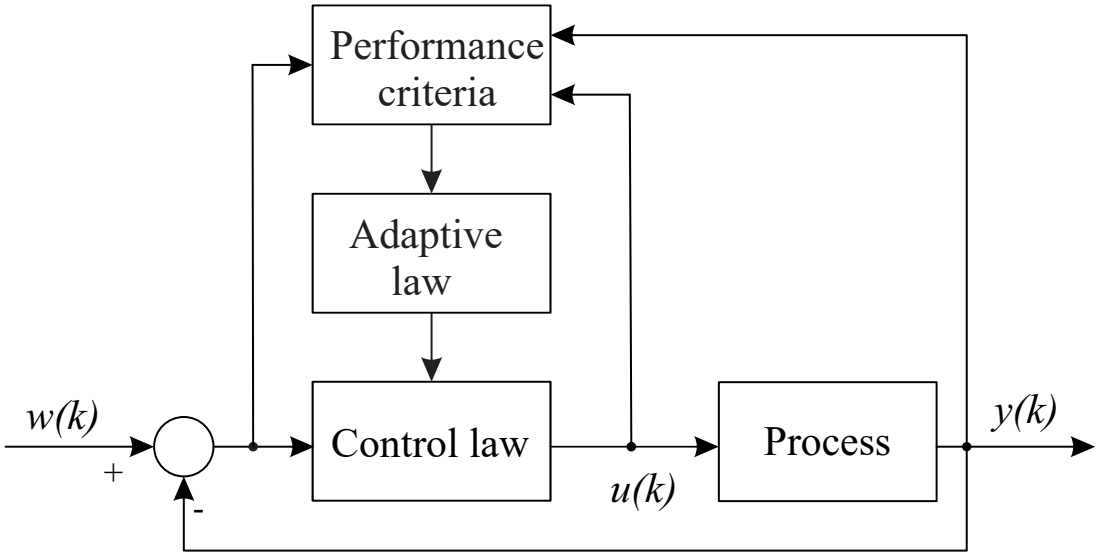


Figure 6: Block diagram of direct adaptive system

Indirect adaptive control

- The second approach is called **indirect control**.
- In this case the process parameters are identified by the use of an online identification method and then based on these parameters we modify the control parameters.
- A generalized diagram of indirect parametric adaptive control is shown in figure 7.
- Vector $\underline{\theta}$ represents the vector of process parameters and vector $\underline{\Gamma}$ represents the vector of controller parameters.

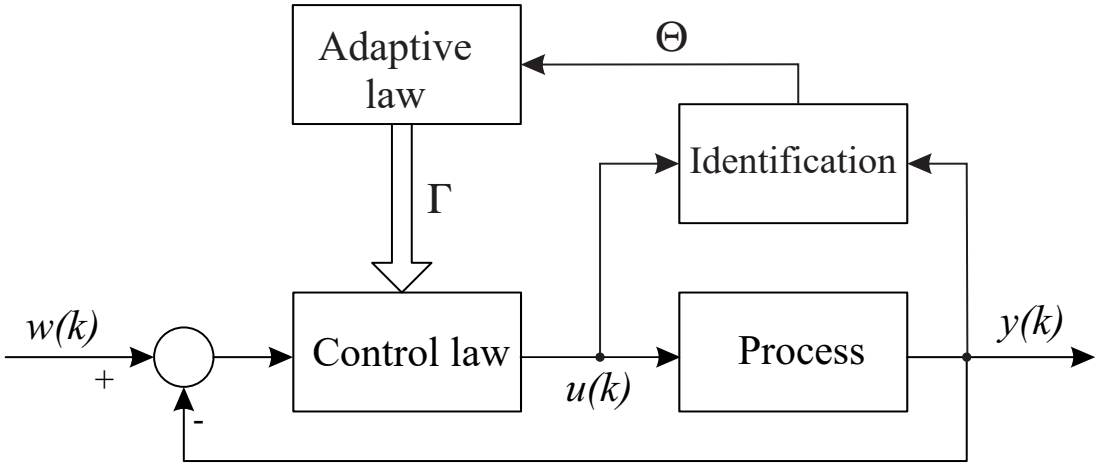


Figure 7: Block diagram of the indirect adaptive system

Direct fuzzy adaptive control

- **Direct fuzzy** adaptive system can be designed in the form of self-adjusting (self-regulating) control.
- A typical diagram of a direct fuzzy adaptive system consists of an auxiliary part, which may be a reference model, auxiliary controller, supervisor or mechanism for setting parameters of a fuzzy controller in forward path.
- Diagram of direct fuzzy adaptive control is shown in figure 8.

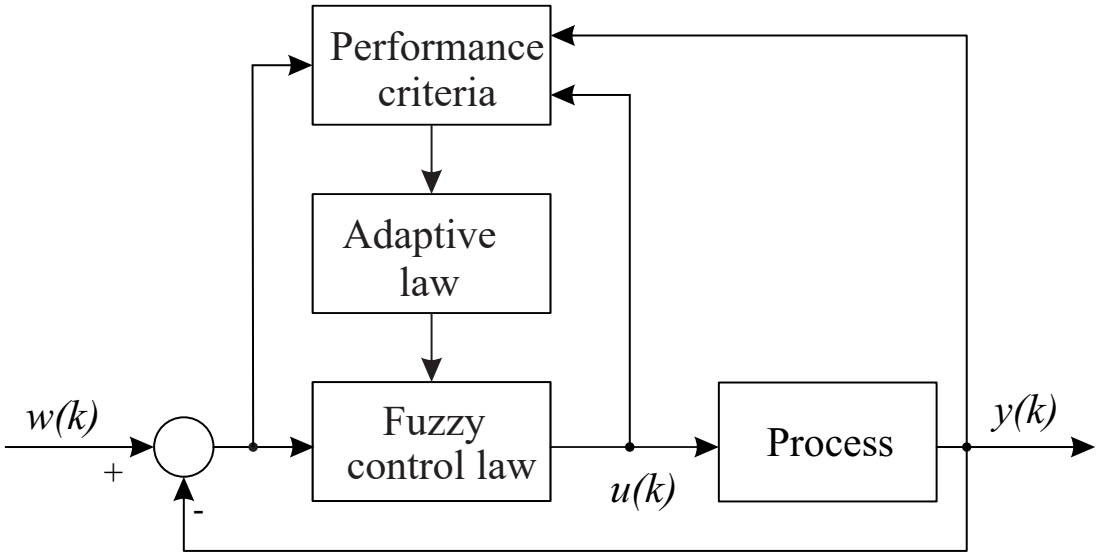


Figure 8: Block diagram of direct fuzzy adaptive control

Example: Lyapunov-based direct fuzzy model-reference adaptive controller

- As in the case of MRAC, we can also derive Direct Fuzzy Model Reference Adaptive Controller (DFMRAC).
- To make the derivation simple, we assume a first-order process.
- In case of a direct fuzzy model-reference controller, we divide the operating area into several smaller fuzzy sub-regions or clusters in which the process can be adequately described with a linear model.
- Areas can be determined manually or by the use of clustering methods.
- For each sub-area, we define gains f and q. We get gains vectors:

$$\underline{f} = [f_1, f_2, ..., f_c]^T \tag{19}$$

$$\underline{q} = [q_1, q_2, ..., q_c]^T \tag{20}$$

Example: Lyapunov-based direct fuzzy model-reference adaptive controller

A nonlinear process can be described by a fuzzy model as a weighted sum of linear models which are valid on their fuzzy sub-regions. The linear model is described by the differential equation of the first order:

$$\dot{y}_{p,i} = -a_i y_p + b_i u \tag{21}$$

where y_p is the process output and $\dot{y}_{p,i}$ is derivative according to the i-th local model. The total derivative $\dot{y}_p$ is equal to the sum:

$$\dot{y}_p = \sum_{i=1}^{c} \beta_i \dot{y}_{p,i} \tag{22}$$

where c is the number of clusters or local linear models, and β_i is normalized membership to a particular cluster. The output of a fuzzy model can be written in vector form as:

$$\dot{y}_p = -\underline{\beta}^T \underline{a} y_p + \underline{\beta}^T \underline{b} u \tag{23}$$

where $\underline{\beta} = [\beta_1, \beta_2, ..., \beta_c]^T$, $\underline{a} = [a_1, a_2, ..., a_c]^T$ and $\underline{b} = [b_1, b_2, ..., b_c]^T$. Here we deal with global linear model whose current parameters are:

$$\tilde{a} = \underline{\beta}^T \underline{a} \tag{24}$$

$$\tilde{b} = \underline{\beta}^T \underline{b} \tag{25}$$

Example: Lyapunov-based direct fuzzy model-reference adaptive controller

- To derive adaptive algorithm DFMRAC, we assume the following control law:

$$u = \frac{1}{\underline{\beta}^T \underline{b}} (\underline{\beta}^T \underline{f}^* w - \underline{\beta}^T \underline{q}^* y_p) \tag{26}$$

- Control law given by equation 26 can not be realized, since we do not usually know the vector of parameters $\underline{b}$.
- As in the linear case, we write the equation of the closed-loop system

$$\dot{y}_p = -\underline{\beta}^T \underline{a} y_p + \underline{\beta}^T \underline{f}^* w - \underline{\beta}^T \underline{q}^* y_p \tag{27}$$

- The reference model is given in the form of a fuzzy model as:

$$\dot{y}_r = -\underline{\beta}^T \underline{a}_r y_r + \underline{\beta}^T \underline{b}_r w \tag{28}$$

$$\underline{a}_r = [a_{r,1}, a_{r,2}, \ldots, a_{r,c}]^T \tag{29}$$

$$\underline{b}_r = [b_{r,1}, b_{r,2}, \ldots, b_{r,c}]^T \tag{30}$$

- If we substract differential equations 27 and 28, we get

$$\dot{e} + \underline{\beta}^T \underline{a}_r e = \underline{\beta}^T (\underline{f}^* - \underline{b}_r) w - \underline{\beta}^T (\underline{q}^* + \underline{a} - \underline{a}_r) y_p \tag{31}$$

- We use the following auxiliary variables for a compact notation of equations:

$$\begin{aligned} \tilde{\underline{b}} &= \underline{f}^* - \underline{b}_r \\ \tilde{\underline{a}} &= \underline{q}^* + \underline{a} - \underline{a}_r \end{aligned} \tag{32}$$

Example: Lyapunov-based direct fuzzy model-reference adaptive controller

- Equation 31 is simplified to

$$\dot{e} + \underline{\beta}^T \underline{a}_r e = \underline{\beta}^T \underline{\tilde{b}} w - \underline{\beta}^T \underline{\tilde{a}} y_p \tag{33}$$

- We define a positive definite Lyapunov function

$$V(e, \underline{\tilde{a}}, \underline{\tilde{b}}) = e^2 + \sum_{i=1}^{c} \frac{1}{\gamma_{bi}} \tilde{b}_i^{\,2} + \sum_{i=1}^{c} \frac{1}{\gamma_{ai}} \tilde{a}_i^{\,2} \tag{34}$$

whose time derivative is:

$$\dot{V}(e, \underline{\tilde{a}}, \underline{\tilde{b}}) = 2e\dot{e} + 2\sum_{i=1}^{c} \frac{1}{\gamma_{bi}} \dot{\tilde{b}}_i \tilde{b}_i + 2\sum_{i=1}^{c} \frac{1}{\gamma_{ai}} \dot{\tilde{a}}_i \tilde{a}_i \tag{35}$$

Example: Lyapunov-based direct fuzzy model-reference adaptive controller

- If we take into account Lyapunov stability conditions ($V(e,\underline{\tilde{a}},\underline{\tilde{b}}) > 0$ and $\dot{V}(e,\underline{\tilde{a}},\underline{\tilde{b}}) \leq 0$), we get the following equations:

$$\begin{aligned} ew\sum_{i=1}^{c}\beta_i\tilde{b}_i + \sum_{i=1}^{c}\frac{1}{\gamma_{bi}}\dot{\tilde{b}}_i\tilde{b}_i &= 0 \\ ey_p\sum_{i=1}^{c}\beta_i\tilde{a}_i + \sum_{i=1}^{c}\frac{1}{\gamma_{bi}}\dot{\tilde{a}}_i\tilde{a}_i &= 0 \end{aligned} \tag{36}$$

from which we express derivatives $\dot{\tilde{b}}_i$ and $\dot{\tilde{a}}_i$

$$\begin{aligned} \dot{\tilde{b}}_i &= -\gamma_{bi}ew\beta_i \\ \dot{\tilde{a}}_i &= \gamma_{ai}ey_p\beta_i \end{aligned} \tag{37}$$

- If we assume that the process is time invariant then we can write:

$$\begin{aligned} \dot{\underline{\tilde{b}}} &= \dot{\underline{f}}^* \\ \dot{\underline{\tilde{a}}} &= \dot{\underline{q}}^* \end{aligned} \tag{38}$$

- This way we get the adaptive law for gains $\dot{f}_i^*$ in $\dot{q}_i^*$:

$$\begin{aligned} \dot{f}_i^* &= -\gamma_{bi}ew\beta_i \\ \dot{q}_i^* &= \gamma_{ai}ey_p\beta_i, \qquad i = 1,\ldots,c \end{aligned} \tag{39}$$

Example: Lyapunov-based direct fuzzy model-reference adaptive controller

- Gains $\dot{f}_i^*$ and $\dot{q}_i^*$ form the following gains vectors

$$\begin{aligned} \underline{f}^* &= [f_1^*, \ldots, f_k^*]^T \\ \underline{q}^* &= [q_1^*, \ldots, q_k^*]^T \end{aligned} \tag{40}$$

- The control law in equation 26 can be written in the following form:

$$u = \underline{\beta}^T \underline{f} w - \underline{\beta}^T \underline{q} y_p \tag{41}$$

where vectors $\underline{f}$ and $\underline{q}$ are equal to:

$$\begin{aligned} \underline{f} &= \frac{\underline{f}^*}{\underline{\beta}^T \underline{b}} \\ \underline{q} &= \frac{\underline{q}^*}{\underline{\beta}^T \underline{b}} \end{aligned} \tag{42}$$

- Depending on equations 39 and 42, the algorithm for adaptation of gains f_i and q_i is equal to:

$$\begin{aligned} \dot{f}_i &= -\frac{\gamma_{bi}}{\underline{\beta}^T \underline{b}} e w \beta_i, \qquad \gamma_{bi} > 0 \\ \dot{q}_i &= \frac{\gamma_{ai}}{\underline{\beta}^T \underline{b}} e y_p \beta_i, \qquad \gamma_{ai} > 0 \end{aligned} \tag{43}$$

Example: Lyapunov-based direct fuzzy model-reference adaptive controller

- In case we know the signs of the parameters b_i, the above adaptation algorithm can be implemented. In most cases, we do not know these values. Therefore the adaptation law needs to be adjusted accordingly.

- If the vector of parameters $\underline{b}$ is constant, we know the sign of the parameters b_i, which is the same for all b_i ($sign(b_i) = b_*,\ i = 1, ..., c$). Then the adaptive law can be simplified in the following form:

$$\begin{aligned} \dot{\underline{f}} &= -\gamma_f sign(b_*) e w \underline{\beta} \\ \dot{\underline{q}} &= \gamma_q sign(b_*) e y_p \underline{\beta} \end{aligned} \tag{44}$$

Indirect fuzzy adaptive control

- Indirect fuzzy adaptive control is based on fuzzy process identification.
- In this case the entire control algorithm consists of fuzzy process identification and fuzzy model based calculation of the control signal.
- The principle of indirect fuzzy adaptive control is shown in figure 9.

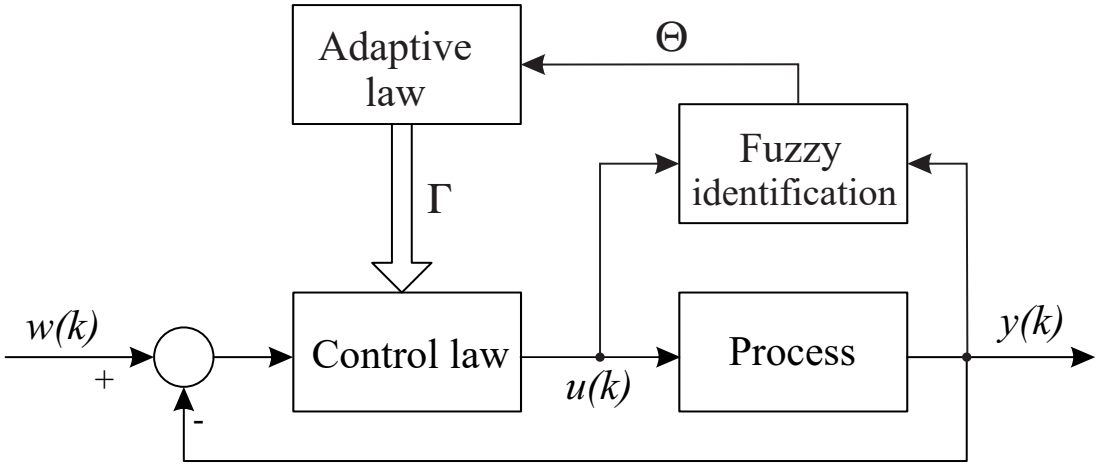


Figure 9: Block diagram of indirect fuzzy adaptive control

- Fuzzy model-based adaptive control can be realized in the form of a predictive algorithm.

Model based predictive control

Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for the 16th lecture, May 2018

Table of contents

1. Predictive control concept
2. Basic principles of predictive control
   - Impulse-response model
   - Step-response model
   - Transfer function model
   - Example: Calculation of the output signal prediction using Diophantine equation
   - State space model
3. Development of the basic predictive control through example
4. Predictive functional control as control design methodology
   - Predictive functional control with first-order model
   - PFC controller based on state space model
   - Model-based PFC controller for ramp reference signal

Predictive control concept

From the point of view of usefulness in industrial applications, predictive control methods have **important advantages** over other control strategies:

- they are suitable for the control of systems with complex dynamics,
- they are suitable for systems with dead time and non-minimum phase,
- the general concept enables the control of univariable and multivariable systems,
- they enable the pre-compensation (feedforward) of measurable disturbances,
- during control we can take into account the constraints of control and the controlled signal and the speed limits of control signal changes,
- substantial designing freedom, where the design parameters are considered as specifications,
- the predictive controller can generate the appropriate control signal in advance if we know the reference trajectory beforehand (looking-ahead capability),
- there is no explicit derivation (therefore the measurement noise does not cause problems),
- there is no explicit integration (therefore we have no problem with integral windup),
- the basic principle is understandable also for non-experts.

Crucial disadvantages of predictive control methods are:

- we need a good dynamic model of the process (the process of model construction can represent up to 80 % of the total time needed to develop a real predictive control application),
- some of the methods are computationally demanding, which can lead to problems in control of processes with faster dynamics.

Basic principles of predictive control

Predictive control methodology is as follows (figure 1):

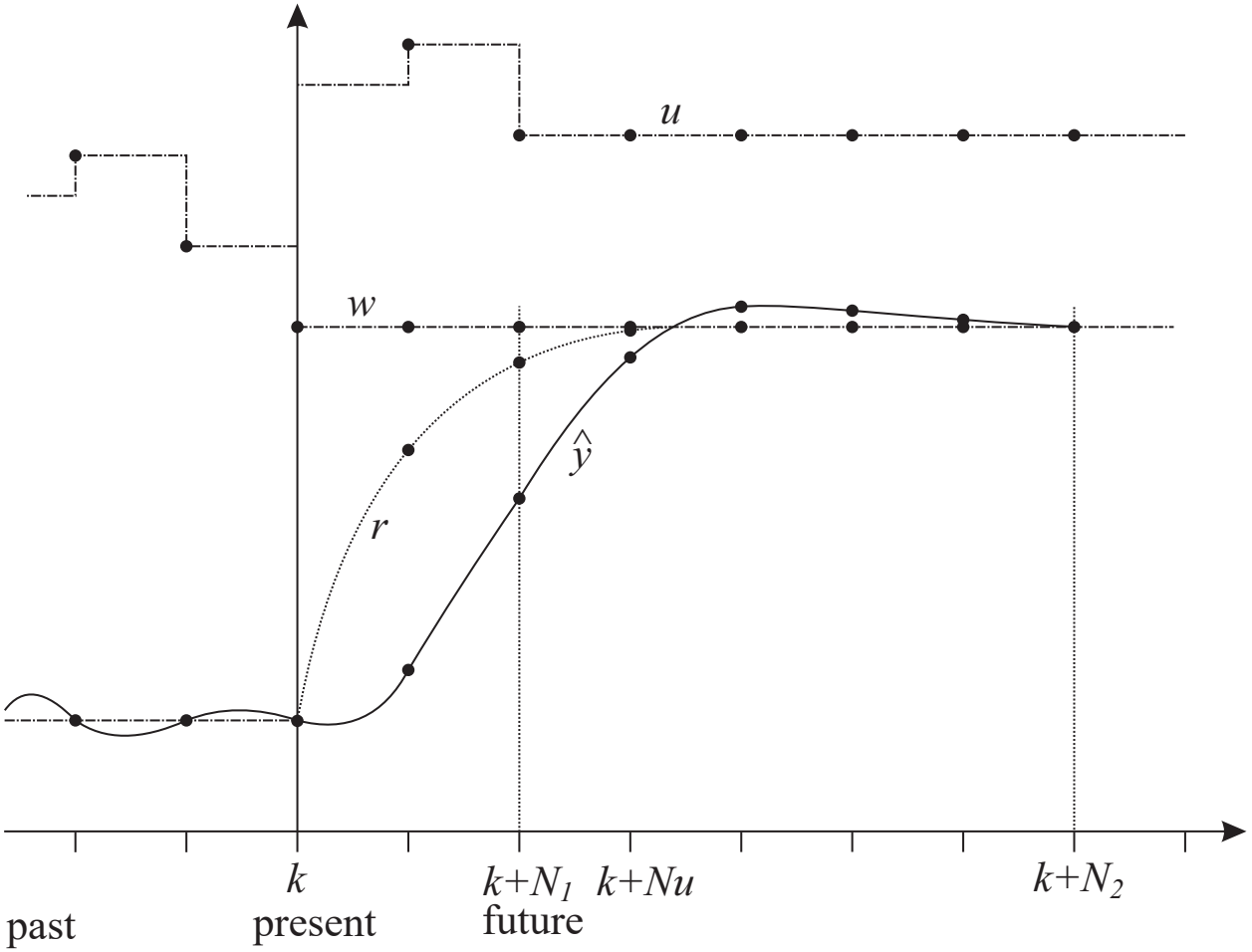


Figure 1: Predictive control principle

Basic principles of predictive control

- At each discrete time k we calculate **prediction of the process output** $y(k+j)$ for a certain observation horizon in the future $(j = N_i, \ldots, N_2)$. The predicted values are labelled as $\hat{y}(k+j|k)$ and they represent j-step prediction of the model.
- Values N_1 and N_2 denote **lower** and **upper values of the prediction horizon**. The lower and upper values of output signal prediction horizon determine the **coincidence interval**, within which we want to obtain the matching between the output signal and the required behaviour. The prediction of the process output is calculated by the use of the process model. The predicted values also depend on the control scenario in the future $u(k+j|k)$, $j = 0, \ldots, N_u - 1$, which we intend to use from time k onward.
- If we want to achieve fitting at one point, we have to deal with **coincidence horizon**.
- We define reference trajectory $r(k+j|k), j = N_1, \ldots, N_2$, which determines how the process response should behave or how it should attain the reference values $w(k)$ (*setpoint trajectory*).
- The vector of future control signals $(u(k+j|k), j = 0, \ldots, N_u - 1)$ is calculated by **minimization** of the corresponding **objective function**, i.e. by minimization of predicted error between $r(k+j|k)$ and $\hat{y}(k+j|k), j = N_1, \ldots, N_2$. In some cases, we use **structuring** of future control signals, which will be described below.
- Only the first element $u(k|k)$ of the **optimal vector** of control signals $u(k+j|k)$, $j = 0, \ldots, N_u - 1$ is used **for control** purposes.
- In the next discrete time, we have a new measured process output, and the whole process is repeated. This principle is called **receding horizon strategy**.

Basic principles of predictive control

The basic diagram of the closed-loop control system can be seen in figure 2. Ideally, when there are no disturbances and when the model ideally fits the process, the diagram results in an open-loop optimal control (in practice, the feedback loop always works).
From the described strategy, the following important elements of predictive control can be obtained:

- prediction based on a process model,
- generating of model-reference trajectory,
- algorithm based calculation of control scenario.

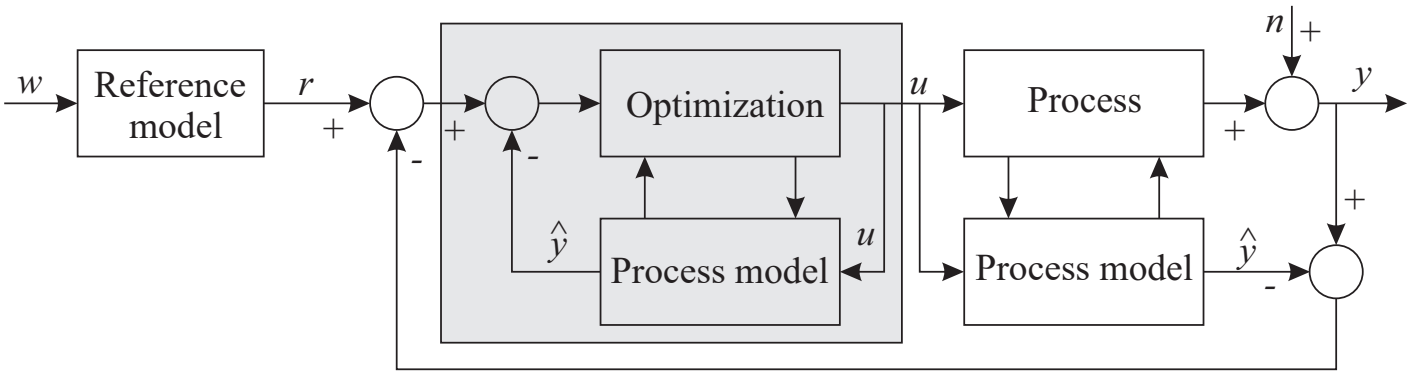


Figure 2: Diagram of predictive control

Impulse-response model

- Impulse-response model of the process (also called weight function, convolution model or FIR (*Finite Impulse-Response) model*)

$$y(k) = \sum_{i=1}^{\infty} h_i u(k-i) \approx \sum_{i=1}^{n} h_i u(k-i) = H(q^{-1})u(k) \tag{1}$$

  where h is the value of the impulse-response process at individual times and $H(q^{-1}) = h_1 q^{-1} + h_2 q^{-2} + \ldots + h_n q^{-n}$, q^{-1} is time delay operator $q^{-1}u(k) = u(k-1)$.
- Advantages of impulse-response model:
  - we do not need almost any prior knowledge about the process (order of the process), it is also suitable for processes with more complex dynamics, we get it with nonparametric identification,
  - simple calculation of the j-step prediction of the output signal:

$$\hat{y}(k+j) = \sum_{i=1}^{n} h_i u(k+j-i) = H(q^{-1})u(k+j), \tag{2}$$

    where we take into account that $u(k+j)$ are known. Since the calculation of prediction does not require recursion, it is simple and less sensitive to numerical errors.
- Disadvantages of impulse-response model:
  - it is useless for unstable processes,
  - it requires a large number of parameters that we need to know or estimate $(n = 30 \div 50)$.

Step-response model

- Process step-response model (*used with DMC method*), also called the FSR model (*Finite Step Response*)

$$y(k) = y_0 + \sum_{i=1}^{\infty} g_i \Delta u(k-i) \approx y_0 + \sum_{i=1}^{n} g_i \Delta u(k-i) = y_0 + G(q^{-1})(1-q^{-1})u(k), \qquad (3)$$

- g_i is the value of process unit step-response, $\Delta u(k) = u(k) - u(k-1)$, n must be large enough so that all g_i are equal to each other from n onward, y_0 is a cumulative contribution of the impact of all changes Δu before n samples.

- Prediction for the next j-steps is calculated as:

$$\hat{y}(k+j) = \sum_{i=1}^{n} g_i \Delta u(k+j-i) = G(q^{-1})(1-q^{-1})u(k+j), \qquad (4)$$

Step-response model has the same advantages and disadvantages as impulse-response model.

Transfer function model

- The model is represented by a difference equation:

$$y(k) + a_1 y(k-1) + \ldots + a_{n_a} y(k-n_a) = b_1 u(k-d-1) + \ldots + b_{n_b} u(k-d-n_b) \tag{5}$$

- or in the form of:

$$y(k) = \frac{q^{-d} B(q^{-1})}{A(q^{-1})} u(k) \tag{6}$$

where d is dead time of the process ($d \geq 0$), $A(q^{-1})$ and $B(q^{-1})$ are polynomials:

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \\ B(q^{-1}) &= b_1 + q^{-1} + \ldots + b_{n_b} q^{-n_b} \end{aligned} \tag{7}$$

- Let's assume ($\hat{B} = B$, $\hat{A} = A$, $\hat{d} = d$, therefore we use notations B, A, d)

Transfer function model

- Calculation of the output signal prediction

$$\hat{y}(k+j) \quad = \quad q^j \frac{q^{-d}B(q^{-1})}{A(q^{-1})} u(k) \tag{8}$$

- If shift operator is taken into account at the process input, we get

$$\begin{aligned} \hat{y}(k+j) &= \frac{q^{-d}B(q^{-1})}{A(q^{-1})} u(k+j) \\ &= q^{-d}B(q^{-1})u(k+j) - q\left(A(q^{-1})-1\right)\hat{y}(k+j-1) \end{aligned} \tag{9}$$

- In this case $q\left(A(q^{-1})-1\right) = a_1 + a_2 q^{-1} + \ldots + a_{n_a} q^{-(n_a-1)}$, since $A(q^{-1})$ is a monic polynomial.
- All $\hat{y}(k+j)$ for $j \geq 1$ can be calculated recursively according to equation 10 starting at $j = 1$:

$$\hat{y}(k+1) = q^{-d}B(q^{-1})u(k+1) - q\left(A(q^{-1})-1\right)\hat{y}(k) \tag{10}$$

- The above predictor (in simulation form) is not useful due to recursive calculation of the prediction since in practice we can not avoid the differences between the model and the process, which results in accumulation of error.
- A possible improvement is a replacement of the output prediction with the measured process output $y(k)$. This requires the elimination of the recursion.

Transfer function model

- Output signal prediction

$$\hat{y}(k+j) \quad = \quad \frac{q^{-d}B(q^{-1})}{A(q^{-1})}u(k+j)$$

- is written in a way that term $\frac{1}{A(q^{-1})}$ is transformed into the sum of polynomial and the remainder

$$\frac{1}{A(q^{-1})} = E_j(q^{-1}) + q^{-j}\frac{F_j(q^{-1})}{A(q^{-1})} \Longrightarrow E_j(q^{-1})A(q^{-1}) + q^{-j}F_j(q^{-1}) = 1 \tag{11}$$

- The polynomials E_j and F_j are obtained by solving the Diophantine equation

$$E_j(q^{-1})A(q^{-1}) + q^{-j}F_j(q^{-1}) = 1 \tag{12}$$

Transfer function model

- What must be the order of these two polynomials to obtain an equation that gives us an unambiguous solution? Assuming that the order of the polynomial E_j is equal to n_e and the order of the polynomial F_j is n_f, then on the left we have a polynomial of order $\max\big((n_e + n_a), (j + n_f)\big)$. The order n_e and n_f are chosen so that $n_e + n_a = j + n_f$.
- This means that we have $n_e + n_a + 1$ equations on the basis of which we have to calculate unknown parameters for the polynomials E_j and F_j. The number of parameters is $n_e + 1 + n_f + 1$.
- To obtain the solution, the number of equations has to be equal to the number of parameters, which means that $n_f = n_a - 1$ and $n_e = j - 1$.
- This means that the degree of polynomial $E_j(q^{-1})$ equals $(j-1)$ and the degree of polynomial $F_j(q^{-1})$ is $(n_a - 1)$. The polynomials are uniquely determined by j and $A(q^{-1})$.
- If the term from equation 11 is used in equation 10, we get

$$\hat{y}(k+j) = q^{-d}E_j(q^{-1})B(q^{-1})u(k+j) + q^{-j}F_j(q^{-1})\frac{q^{-d}B(q^{-1})}{A(q^{-1})}u(k+j) \tag{13}$$

- If we assume that process outputs are measurable, then we can replace $\hat{y}(k)$ with $y(k)$. This way we get:

$$\hat{y}(k+j) = q^{-d}E_j(q^{-1})B(q^{-1})u(k+j) + F_j(q^{-1})y(k) \tag{14}$$

- From this follows that for each j-th prediction horizon we can calculate polynomials E_j and F_j, which determine the prediction of the output signal in this horizon.
- The polynomials can also be determined by long division of the polynomials or the recursive algorithm, which is described in the Appendix. At this point, let's take a look at a simple example of calculation with Diophantine equation.

Example: Calculation of the output signal prediction using Diophantine equation

- Let's assume that we have a model of a process that can be written with a transfer function in Z-space

$$G_p(z) = \frac{Y(z)}{U(z)} = \frac{B(q^{-1})}{A(q^{-1})} = \frac{0.2z^{-1} - 0.4z^{-2}}{(1-0.5z^{-1})(1-0.8z^{-1})} \tag{15}$$

  where the nominator $B(z^{-1})$ is of the order of $n_b = 2$ and the denominator $A(z^{-1})$ is of the order of $n_a = 2$ as well. Due to clarity, we set the pure delay d to zero.
- Let's write the Diophantine equation for horizon j, where the order of E_j is equal to $n_e = j - 1$ and the order of F_j is equal to $n_f = n_a - 1$.
- The polynomials E_j and F_j are obtained by solving the Diophantine equation. For our example, we get a general solution:

$$\left(e_0 + e_1 q^{-1} + \ldots + e_{j-1} q^{-(j-1)}\right)\left(1 + a_1 q^{-1} + a_2 q^{-2}\right) + q^{-j}\left(f_0 + f_1 q^{-1}\right) = 1 \tag{16}$$

  where $a_1 = -1.3$ and $a_2 = 0.4$.
- If a multiplication is performed and we sum the terms with the same exponents for $j = 1$, then

$$e_0 + (e_0 a_1 + f_0)\, q^{-1} + (e_0 a_2 + f_1)\, q^{-2} = 1 \tag{17}$$

- By comparing the coefficients on both sides of the polynomial equation, we get $e_0 = 1$, $f_0 = -a_1 = 1.3$ and $f_1 = -a_2 = -0.4$. This means that the polynomial is

$$E_1(q^{-1}) = 1$$

  and

$$F_1(q^{-1}) = 1.3 - 0.4\, q^{-1}$$

Example: Calculation of the output signal prediction using Diophantine equation

- In the case when horizon equals to $j = 2$, we get

$$e_0 + (e_0 a_1 + e_1)\, q^{-1} + (e_0 a_2 + e_1 a_1 + f_0)\, q^{-2} + (e_1 a_2 + f_1)\, q^{-3} = 1 \tag{18}$$

- By equating the coefficients of the polynomial we obtain $e_0 = 1$, $e_1 = -e_0 a_1 = 1.3$, $f_0 = -e_0 a_2 - e_1 a_1 = 1.29$ in $f_1 = -e_1 a_2 = -0.52$. The polynomials are equal to

$$E_2(q^{-1}) = 1 + 1.3 q^{-1}$$

and

$$F_2(q^{-1}) = 1.29 - 0.52 q^{-1}$$

- In the case when horizon equals to $j = 3$, we get

$$E_3(q^{-1}) \;=\; 1 \;+\; 1.3 q^{-1} \;+\; 1.29\, q^{-2}$$

and

$$F_3(q^{-1}) \;=\; 1.157 \;-\; 0.516\, q^{-1}$$

- By considering equation 14, we get the following process output predictions

$$\begin{aligned} \hat{y}(k+1) &= E_1(q^{-1}) B(q^{-1}) u(k+1) + F_1(q^{-1}) y(k) \\ \hat{y}(k+2) &= E_2(q^{-1}) B(q^{-1}) u(k+2) + F_2(q^{-1}) y(k) \\ \hat{y}(k+3) &= E_3(q^{-1}) B(q^{-1}) u(k+3) + F_3(q^{-1}) y(k) \end{aligned} \tag{19}$$

Example: Calculation of the output signal prediction using Diophantine equation

- If we write the values for our example, we get

$$
\begin{aligned}
\hat{y}(k+1) &= \left(0.2q^{-1} - 0.4q^{-2}\right) u(k+1) + \left(1.3 - 0.4q^{-1}\right) y(k) \\
\hat{y}(k+2) &= \left(0.2q^{-1} - 0.14q^{-2} - 0.52q^{-3}\right) u(k+2) + \left(1.29 - 0.52q^{-1}\right) y(k) \\
\hat{y}(k+3) &= \left(0.2q^{-1} - 0.14q^{-2} - 0.26q^{-3} - 0.516q^{-4}\right) u(k+3) + \left(1.157 - 0.516q^{-1}\right) y(k)
\end{aligned}
$$

or

$$
\begin{aligned}
\hat{y}(k+1) &= 0.2u(k) - 0.4u(k-1) + 1.3y(k) - 0.4y(k-1) \\
\hat{y}(k+2) &= 0.2u(k+1) - 0.14u(k) - 0.52u(k-1) + 1.29y(k) - 0.52y(k-1) \\
\hat{y}(k+3) &= 0.2u(k+2) - 0.14u(k+1) - 0.26u(k) - 0.516u(k-1) + 1.157y(k) - 0.516y(k-1)
\end{aligned}
$$

- or in the matrix form, where the unknown and known values of the input and output signals are separated

$$
\begin{bmatrix} \hat{y}(k+1) \\ \hat{y}(k+2) \\ \hat{y}(k+3) \end{bmatrix} = \begin{bmatrix} 0.2 & 0 & 0 \\ -0.14 & 0.2 & 0 \\ -0.26 & -0.14 & 0.2 \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ u(k+2) \end{bmatrix} + \begin{bmatrix} -0.4 \\ -0.52 \\ -0.516 \end{bmatrix} u(k-1) + \begin{bmatrix} 1.3 & -0.4 \\ 1.29 & -0.52 \\ 1.157 & -0.516 \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \end{bmatrix}
$$

- The notation of the output signal prediction is:

$$
\underline{y}^{+} = \underline{G}\underline{u}^{+} + \underline{\Gamma}\underline{u}^{-} + \underline{F}\underline{y}^{-}
$$

Example: Calculation of the output signal prediction by polynomial division

- Calculation of the output signal prediction $\hat{y}(k+j) = q^j \frac{0.2q^{-1} - 0.4q^{-2}}{(1-0.5q^{-1})(1-0.8q^{-1})} u(k)$
- Polynomial division

$$\frac{q^j}{A(q^{-1})} \tag{20}$$

- In the case when horizon equals to $j = 1$, we get

$$\frac{q^1}{A(q^{-1})} = q^1 + \frac{1.3 - 0.4q^{-1}}{A(q^{-1})} \tag{21}$$

- In the case when horizon equals to $j = 2$, we get

$$\frac{q^2}{A(q^{-1})} = q^2 + 1.3q^1 + \frac{1.29 - 0.52q^{-1}}{A(q^{-1})} \tag{22}$$

- In the case when horizon equals to $j = 3$, we get

$$\frac{q^3}{A(q^{-1})} = q^3 + 1.3q^2 + 1.29q^1 + \frac{1.157 - 0.516q^{-1}}{A(q^{-1})} \tag{23}$$

- Calculation of the output signal prediction

$$\hat{y}(k+3) = \left(q^3 + 1.3q^2 + 1.29q^1 + \frac{1.157 - 0.516q^{-1}}{A(q^{-1})}\right) B(q^{-1})u(k) \tag{24}$$

$$\hat{y}(k+3) = \left(q^3 + 1.3q^2 + 1.29q^1\right) B(q^{-1})u(k) + \left(1.157 - 0.516q^{-1}\right) y(k) \tag{25}$$

State space model

- State space model representation:

$$\underline{x}(k+1) = \underline{A}\underline{x}(k) + \underline{b}u(k) \tag{26}$$

$$y(k) = \underline{c}^T\underline{x}(k), \tag{27}$$

  where $\underline{x}$ are process states, $\underline{A}$ is system matrix, $\underline{b}$ is input vector, and $\underline{c}^T$ is output vector. Since in real processes usually we do not detect direct connections between the input and output of the process, we leave them out in the above notation.

- The output signal prediction is calculated using the following equation:

$$\hat{y}(k+j) = \underline{c}^T\underline{x}(k+j) = \underline{c}^T\left(\underline{A}^j\underline{x}(k) + \sum_{i=1}^{j}\underline{A}^{i-1}\underline{b}u(k+j-i)\right) \tag{28}$$

- State space model is useful in control design of multivariable systems. In this case the computational complexity is higher and if the states are not directly measurable, we need to use state observers.
- Using state space representation, we can easily solve a control problem, where the reference (setpoint) signals are given in different polynomial forms.

Development of the basic predictive control through example

- The criterion function takes into account the deviation of the prediction of the controlled signal from the desired reference on the selected time interval, which is called the coincidence interval $[N_1, N_2]$.
- In addition to the deviation of the controlled signal from the desired reference, the criterion function also takes into account the costs necessary for tracking the reference value. These costs are expressed by deviations of the control signal from its settled value in the interval to the horizon of the control signal N_u.

$$J = \sum_{j=N_1}^{N_2} \Big(y(k+j) - r(k+j) \Big)^2 + \lambda \sum_{j=0}^{N_u} \Big(\Delta u(k+j) \Big)^2 \tag{29}$$

- We control a process with the following transfer function

$$G_p(z) = \frac{Y(z)}{U(z)} = \frac{0.4z^{-1}}{1 - 0.6z^{-1}} \tag{30}$$

- We require the control law in the incremental form, and therefore we multiply the transfer function (in numerator and denominator) with the operator $\Delta = 1 - z^{-1}$, and in this way we get

$$(1 - 0.6z^{-1})(1 - z^{-1})Y(z) = 0.4z^{-1}\Delta U(z) \tag{31}$$

- or in time domain

$$y(k) = 1.6y(k-1) - 0.6y(k-2) + 0.4\Delta u(k-1) \tag{32}$$

Example: Development of the basic predictive control through example

- We select the parameters $N_1 = 1, N_2 = 3$ and $N_u = 1$. For the other part of the control signal at $N_u > 1$, we assume that the changes are equal to zero.
- The criterion function is as follows

$$J = \sum_{j=1}^{3} \Big(y(k+j) - r(k+j) \Big)^2 + \lambda \sum_{j=0}^{1} \Big(\Delta u(k+j) \Big)^2 \tag{33}$$

  where λ is the parameter that regulates the ratio between the quality of control and the necessary costs.
- The prediction is obtained by dividing, recursion or using Diophantine equation

$$\begin{aligned}
\hat{y}(k+1) &= 1.6y(k) - 0.6y(k-1) + 0.4\Delta u(k) \\
\hat{y}(k+2) &= 1.6\hat{y}(k+1) - 0.6y(k) + 0.4\Delta u(k+1) = \\
&= 1.96y(k) - 0.96y(k-1) + 0.4\Delta u(k+1) + 0.64\Delta u(k) \\
\hat{y}(k+3) &= 1.6\hat{y}(k+2) - 0.6\hat{y}(k+1) + 0.4\Delta u(k+2) = \\
&= 2.1760y(k) - 1.1760y(k-1) + 0.4\Delta u(k+2) + 0.64\Delta u(k+1) + 0.784\Delta u(k)
\end{aligned}$$

- or in matrix form, where we take into account the fact that the closed-loop control variable does not change after the interval $(\Delta u(k+2) = 0)$

$$\begin{bmatrix} \hat{y}(k+1) \\ \hat{y}(k+2) \\ \hat{y}(k+3) \end{bmatrix} = \begin{bmatrix} 0.4 & 0 \\ 0.64 & 0.4 \\ 0.7840 & 0.64 \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \end{bmatrix} + \begin{bmatrix} 1.6 & -0.6 \\ 1.96 & -0.96 \\ 2.1760 & -1.1760 \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \end{bmatrix} \tag{34}$$

Example: Development of the basic predictive control through example

- The shortened notation of matrix equation for the process output prediction is

$$\underline{y}^{+} = \underline{G}\Delta\underline{u}^{+} + \underline{F}\underline{y}^{-}$$

where $+$ signs label future signals (which are unknown) and $-$ signs label past signals (which are known).

- If we had a system that had a polynomial of higher order in the numerator of the process transfer function, the term $\Gamma\Delta\underline{u}^{-}$ would appear on the right side of the equation above. This term will be taken into account due to complete derivation of control law. In our case, we have $\underline{\Gamma} = \underline{0}$
- The criterion function in equation 33 can be written in the form

$$\begin{aligned} J &= \left(\underline{y}^{+} - \underline{r}^{+}\right)^{T}\left(\underline{y}^{+} - \underline{r}^{+}\right) + \lambda\Delta\underline{u}^{+T}\Delta\underline{u}^{+} = \\ &= \left(\underline{G}\Delta\underline{u}^{+} + \underline{\Gamma}\Delta\underline{u}^{-} + \underline{F}\underline{y}^{-} - \underline{r}^{+}\right)^{T}\cdot\left(\underline{G}\Delta\underline{u}^{+} + \underline{\Gamma}\Delta\underline{u}^{-} + \underline{F}\underline{y}^{-} - \underline{r}^{+}\right) + \lambda\Delta\underline{u}^{+T}\Delta\underline{u}^{+} \end{aligned}$$

where the vector $\underline{r}^{+}$ contains three future values of the reference signal.

- We need to find a vector of the future control signal values which minimizes the criterion function:

$$\frac{\partial J}{\partial\Delta\underline{u}^{+}} = 2\underline{G}^{T}\left(\underline{G}\Delta\underline{u}^{+} + \underline{\Gamma}\Delta\underline{u}^{-} + \underline{F}\underline{y}^{-} - \underline{r}^{+}\right) + 2\lambda\Delta\underline{u}^{+} = 0$$

from which follows

$$\Delta\underline{u}^{+} = \left(\underline{G}^{T}\underline{G} + \lambda\underline{I}\right)^{-1}\underline{G}^{T}\left(\underline{r}^{+} - \underline{\Gamma}\Delta\underline{u}^{-} + \underline{F}\underline{y}^{-}\right) \tag{35}$$

Example: Development of the basic predictive control through example

- By inserting matrices $\underline{G}$ and $\underline{F}$ in our example and $\underline{\Gamma}$ (in our case 0) and by selecting $\lambda = 0.1$ we get a solution for the prediction $\Delta u(k)$ and $\Delta u(k+1)$.
- For the control, we use only the first line for the change at the moment of observation $\Delta u(k)$, which is equal to:

$$\Delta u(k) = -2.6648y(k) + 1.1485y(k-1) + 0.9374r(k+1) + 0.4390r(k+2) + 0.1400r(k+3)$$

- This equation represents the control law.
- We can observe that this law requires knowledge of the future reference signal values. If we do not know them, in practice we replace them with a current reference signal value.
- The general characteristics of the predictive controller and its advantages over PID controller are a smoother control signal and a response with less overshoot.

Predictive functional control as control design methodology

- PFC control strategy is based on the use of a dynamic process model for the process output prediction in a given coincidence horizon.
- The goal of this approach is to determine such an output of the controller that the changes between the reference value prediction $y_r(k+H)$ and the current output value $y_p(k)$ and between the model output prediction $y_m(k+H)$ and $y_m(k)$ match in the horizon H:

$$y_r(k+H) - y_p(k) = y_m(k+H) - y_m(k) \tag{36}$$

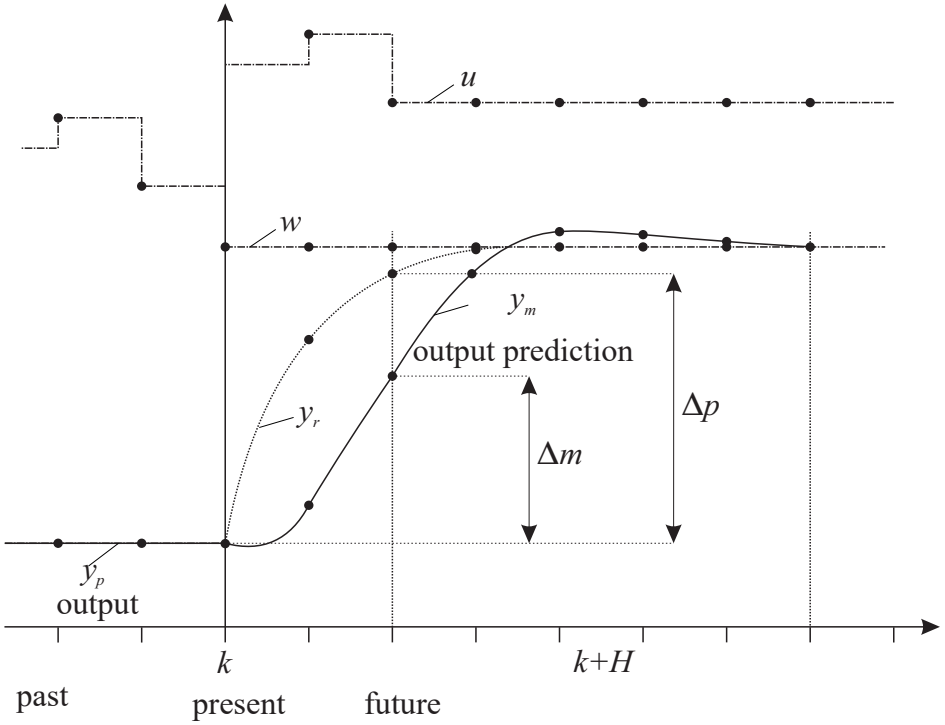


Figure 3: PFC principle

Predictive functional control with first-order model

- PFC algorithm for a process that is described by the first-order dynamic linear model in a continuous space:

$$Y_m(s) = \frac{K_m}{T_m s + 1} U(s) \tag{37}$$

- In a discrete space equation 37 is written as:

$$y_m(k+1) = a_m y_m(k) + b_m u(k) \tag{38}$$

where $y_m(k)$ is the output of the model, and $u(k)$ is the input into the model or process. The relationship between the time constant and gain and between the parameters a_m and b_m is as follows:

$$a_m = e^{-\frac{T_s}{T_m}} \tag{39}$$

$$b_m = K_m(1 - a_m) \tag{40}$$

where T_s is the sampling time.

Predictive functional control with first-order model

- To derive the control law, the reference trajectory of the closed-loop process should be given. It is given in the form of a reference model that provides the tracking of the reference w:

$$y_r(k+1) = a_r y_r(k) + b_r w(k) \tag{41}$$

- To ensure the tracking of the reference, the gain of this model should be equal to one. Based on this condition, we get the value of the parameter b_r:

$$b_r = (1 - a_r) \tag{42}$$

- The reference model defines the desired behaviour of the entire closed-loop system in terms of the relationship between the reference and process output.
- The control law is obtained by solving equation 36. PFC expects the coincidence horizon H ($N_1 = N_2 = H$) and the weighting factor of the control signal $\lambda = 0$.
- At the coincidence horizon, the output of the process model should match the reference response. This means that the changes in the output of the process model and the reference model must be equal from the step k to the step $k + H$ (see figure 3):

$$\Delta_p = \Delta_m \tag{43}$$

Predictive functional control with first-order model

- As we can see, the model-based control is practically the same as the open-loop process control. We optimize the output of the model according to the reference model.
- The change of the reference model can be written as:

$$\Delta_r = y_r(k+H) - y_r(k) \tag{44}$$

- Closed loop is included in the control algorithm indirectly, in a way that the initial values of the reference model are equal to the current process values. So, the output of the reference model equals to the output of the process at the time moment k ($y_r(k) = y_p(k)$).
- The change in the process output, which is necessary to obtain the reference, is written as:

$$\Delta_p = y_r(k+H) - y_p(k) \tag{45}$$

- and the change in the process model output as:

$$\Delta_m = y_m(k+H) - y_m(k) \tag{46}$$

- In the k-th step (for the step $k+H$) the output of the process model or reference model is obtained by solving the following system of equations:

$$\begin{aligned}
y_m(k+1) &= a_m y_m(k) + b_m u(k) \\
y_m(k+2) &= a_m y_m(k+1) + b_m u(k+1) = \\
&= a_m^2 y_m(k) + a_m b_m u(k) + b_m u(k) \\
&\vdots \\
y_m(k+H) &= a_m y_m(k+H-1) + b_m u(k+H-1) = \\
&= a_m^H y_m(k) + a_m^{H-1} b_m u(k) + a_m^{H-2} b_m u(k+1) + \ldots + b_m u(k+H-1)
\end{aligned} \tag{47}$$

Predictive functional control with first-order model

- We assume that the future control signal is constant (*mean level control*) when solving the optimization problem.
- This means that during optimization we do not calculate all input values from the time moment k to the time moment $k+H$.
- In some algorithms, the values of the control signal u are optimized for each time moment from k to $k+H$. However, only the value $u(k)$ is used as the output of the controller. This principle is called the principle of receding horizon.
- Considering the assumption of the constant future control signal ($u(k) = u(k+1) = ... = u(k+H-1)$) we can write a prediction for $k+H$-th step from the system of equations (47) in the following form:

$$y_m(k+H) = a_m^H y_m(k) + a_m^{H-1} b_m u(k) + a_m^{H-2} b_m u(k) + ... + b_m u(k) \tag{48}$$

- Considering that $(1 + a_m + ... + a_m^{H-2} + a_m^{H-1})(1 - a_m) = (1 - a_m^H)$, then equation (48) can be simplified to:

$$y_m(k+H) = a_m^H y_m(k) + \frac{b_m}{1-a_m}(1 - a_m^H)u(k) \tag{49}$$

Predictive functional control with first-order model

- A similar equation is obtained for the prediction of the reference model output where we consider the constant future reference ($w(k) = w(k+1) = ... = w(k+H-1)$):

$$y_r(k+H) = a_r^H y_r(k) + (1 - a_r^H) w(k) \tag{50}$$

- Then the changes Δ_p and Δ_m are written as:

$$\Delta_p = a_r^H y_r(k) + (1 - a_r^H) w(k) - y_p(k) \tag{51}$$

$$\Delta_m = a_m^H y_m(k) + \frac{b_m}{1 - a_m}(1 - a_m^H) u(k) - y_m(k) \tag{52}$$

Predictive functional control with first-order model

- If we solve equation (43), considering that $y_r(k) = y_p(k)$, we get the following control law:

$$u(k) = \frac{(1 - a_r^H)(w(k) - y_p(k))}{\frac{b_m}{1-a_m}(1 - a_m^H)} + \frac{y_m(k)}{\frac{b_m}{1-a_m}} \tag{53}$$

- Taking into account

$$K_m = \frac{b_m}{(1 - a_m)} \tag{54}$$

and

$$g = \frac{1 - a_r^H}{K_m(1 - a_m^H)} \tag{55}$$

equation (53) can be written in a more transparent form:

$$u(k) = g(w(k) - y_p(k)) + \frac{1}{K_m} y_m(k) \tag{56}$$

Predictive functional control with first-order model

- The control law (equation (56)) is illustrated with a block diagram in figure 4.

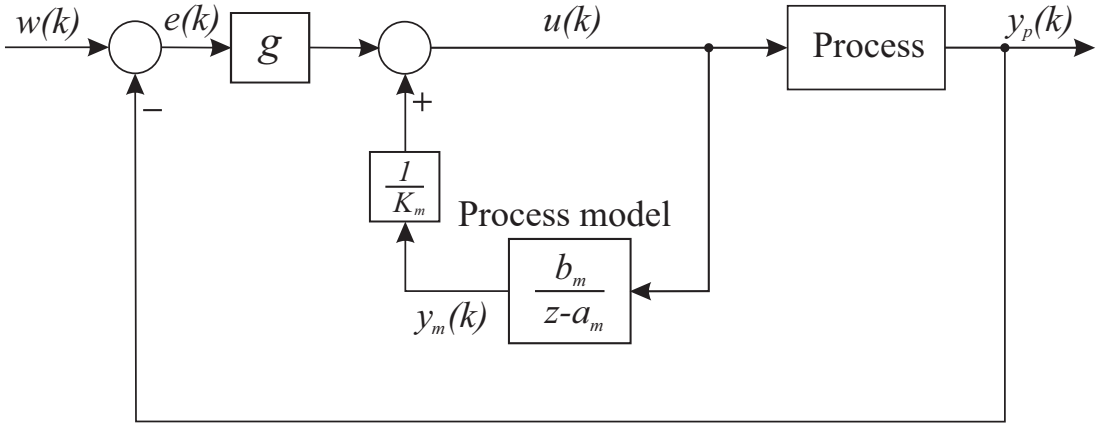


Figure 4: Block diagram of PFC controller

PFC controller based on state space model

- In order to control **high order processes and multivariable processes** we develop a predictive controller based on state space model.
- The derivation of the control law is based on the same idea as in the case of PFC controller with first-order model.
- The process model is written in state space as:

$$\underline{x}_m(k) = \underline{A}_m \underline{x}_m(k) + \underline{B}_m \underline{u}(k) \tag{57}$$

$$\underline{y}_m(k) = \underline{C}_m \underline{x}_m(k) \tag{58}$$

- The reference model is also given in state space:

$$\underline{x}_r(k) = \underline{A}_r \underline{x}_r(k) + \underline{B}_r \underline{w}(k) \tag{59}$$

$$\underline{y}_r(k) = \underline{C}_r \underline{x}_r(k) \tag{60}$$

- Since the gain of the reference model must be equal to one, the matrices $\underline{A}_r$, $\underline{B}_r$ and $\underline{C}_r$ must fulfil the following condition:

$$\underline{C}_r(\underline{I} - \underline{A}_r)^{-1}\underline{B}_r = \underline{I} \tag{61}$$

PFC controller based on state space model

- At the time moment k the prediction (for H steps in the future) is obtained in the same way as in the case of PFC controller with the first-order model (constant control signal in the future):

$$\underline{x}_m(k+H) = \underline{A}_m^H \underline{x}_m(k) + (\underline{A}_m^H - \underline{I})(\underline{A}_m - \underline{I})^{-1} \underline{B}_m \underline{u}(k) \tag{62}$$

$$\underline{y}_m(k+H) = \underline{C}_m \underline{x}_m(k+H) \tag{63}$$

- Similarly we write the equation for the reference model:

$$\underline{x}_r(k+H) = \underline{A}_r^H \underline{x}_r(k) + (\underline{A}_r^H - \underline{I})(\underline{A}_r - \underline{I})^{-1} \underline{B}_r \underline{w}(k) \tag{64}$$

$$\underline{y}_r(k+H) = \underline{C}_r \underline{x}_r(k+H) \tag{65}$$

- Once again we need to solve equation (43), where the increment of the model is:

$$\Delta_m = \underline{C}_m(\underline{A}_m^H \underline{x}_m(k) + (\underline{A}_m^H - \underline{I})(\underline{A}_m - \underline{I})^{-1} \underline{B}_m \underline{u}(k)) - \underline{y}_m(k) \tag{66}$$

- and the increment of the process is:

$$\Delta_p = \underline{C}_r(\underline{A}_r^H \underline{x}_r(k) + (\underline{A}_r^H - \underline{I})(\underline{A}_r - \underline{I})^{-1} \underline{B}_r \underline{w}(k)) - \underline{y}_p(k) \tag{67}$$

PFC controller based on state space model

- The obtained control law has the following form:

$$\underline{u}(k) = G_0^{-1}(\underline{y}_r(k+H) + \underline{y}_m(k) - \underline{y}_p(k) - \underline{C}_m\underline{A}_m^H\underline{x}_m(k)) \tag{68}$$

- where:

$$\underline{y}_r(k+H) = \underline{C}_r(\underline{A}_r^H\underline{x}_r(k) + (\underline{A}_r^H - \underline{I})(\underline{A}_r - \underline{I})^{-1}\underline{B}_r\underline{w}(k)) \tag{69}$$

- and G_0:

$$G_0 = \underline{C}_m(\underline{A}_m^H - \underline{I})(\underline{A}_m - \underline{I})^{-1}\underline{B}_m \tag{70}$$

- If the reference model is determined in a way that $\underline{C}_r = \underline{I}$ and $\underline{B}_r = (\underline{I} - \underline{A}_r)$ then the control law (equation (68)) is simplified. The prediction of the reference signal can be written as:

$$\underline{y}_r(k+H) = \underline{A}_r^H\underline{y}_r(k) + (\underline{I} - \underline{A}_r^H)\underline{w}(k) \tag{71}$$

PFC controller based on state space model

- We solve equation (43) by taking into account $\underline{y}_p(k) = \underline{y}_r(k)$, where:

$$\Delta_m = \underline{C}_m(\underline{A}_m^H \underline{x}_m(k) + (\underline{A}_m^H - \underline{I})(\underline{A}_m - \underline{I})^{-1}\underline{B}_m \underline{u}(k)) - \underline{y}_m(k) \tag{72}$$

- and

$$\Delta_p = \underline{A}_r^H \underline{y}_r(k) + (\underline{I} - \underline{A}_r^H)\underline{w}(k) - \underline{y}_p(k) \tag{73}$$

- We obtain a simplified control law:

$$\underline{u}(k) = \underline{G}(\underline{w}(k) - \underline{y}_p(k)) + G_0^{-1}\underline{y}_m(k) - G_0^{-1}\underline{C}_m \underline{A}_m^H \underline{x}_m(k) \tag{74}$$

- where:

$$\underline{G} = G_0^{-1}(\underline{I} - \underline{A}_r^H) \tag{75}$$

  and G_0:

$$G_0 = \underline{C}_m(\underline{A}_m^H - \underline{I})(\underline{A}_m - \underline{I})^{-1}\underline{B}_m \tag{76}$$

- Equations 68 and 74 show that the controller exists only if the value of the expression G_0 is non-zero.
- The term G_0 is non-zero for stable, controllable and observable processes.

Model-based PFC controller for ramp reference signal

- Regular controllers are inadequate for tracking a ramp signal without error in case of proportional process since second order open-loop transfer function (2 poles at the origin) is required. Consequently this causes problems with the stability of the system.
- PFC based on state space model successfully solves the ramp tracking problem.
- In the case of ramp tracking the control signal (for proportional processes) has the form of a ramp.

$$u(k+H) = u(k) + H\Delta u(k)$$

- To determine the control signal, we must define $u(k)$ and $\Delta u(k)$.
- We solve a system of two equations obtained for two coincidence horizons H_1 and H_2.
- At the time moment k the prediction for H steps in the future is obtained as

$$y_m(k+H) = \underline{C}_m \underline{A}_m^H \underline{x}_m(k) + \Sigma_0(H)u(k) + \Sigma_1(H)\Delta u(k) \tag{77}$$

$$\tag{78}$$

- where we consider the following simplification:

$$\begin{aligned}
\Sigma_0(H) &= \underline{C}_m \left(\underline{A}_m^{H-1} + \underline{A}_m^{H-2} + \cdots \underline{I}\right) \underline{B}_m = \underline{C}_m \left(\underline{I} - \underline{A}^H\right) \left(\underline{I} - \underline{A}^H\right)^{-1} \underline{B}_m \\
\Sigma_1(H) &= \underline{C}_m \left(\underline{A}_m^{H-2} + 2\underline{A}_m^{H-3} + \cdots + (H-2)\underline{A}_m + (H-1)\underline{I}\right) \underline{B}_m \\
\Sigma_1(H) &= \underline{C}_m \left(H\underline{I} - \Sigma_0(H)\right) \left(\underline{I} - \underline{A}^H\right)^{-1} \underline{B}_m
\end{aligned}$$

Model-based PFC controller for ramp reference signal

- The error between the reference trajectory w and the process output y_p is denoted by e.
- The PFC controller introduces a dynamics that requires an exponential reduction of error $e(k)$. This means that

$$e(k+H) = a_r^H e(k) \tag{79}$$

  where a_r is a constant that determines the speed of error reduction, $0 < a_r < 1$.
- The process output value (desired, $y_p(k+H) = y_r(k+H)$) for horizon H is defined according to the reference value and error

$$y_p(k+H) = w(k+H) - a_r^H e(k) \tag{80}$$

- At the time moment k the process output value is also defined according to the reference value and error

$$y_p(k) = w(k) - e(k) \tag{81}$$

- The process increment at the coincidence horizon H is defined as the difference between $y_p(k+H)$ and the output signal value $y_p(k)$

$$\Delta_p(H) = y_p(k+H) - y_p(k) = \left(w(k+H) - a_r^H e(k)\right) - \left(w(k) - e(k)\right) \tag{82}$$

- The process increment depends on the change of reference and exponential reduction of error between reference and output

$$\Delta_p(H) = w(k+H) - w(k) + \left(1 - a_r^H\right)\left(w(k) - y_p(k)\right) \tag{83}$$

Model-based PFC controller for ramp reference signal

- The increment of the process model $\Delta_m(H)$ at the coincidence horizon H is defined as the difference between $y_m(k+H)$ and the model output value $y_m(k)$

$$\Delta_m(H) = \underline{C}_m \underline{A}_m^H \underline{x}_m(k) + \Sigma_0(H)u(k) + \Sigma_1(H)\Delta u(k) - y_m(k) \tag{84}$$

- We write the basic equation in which we equate the process increment and the model at the coincidence horizon H, $\Delta_p(H) = \Delta_m(H)$

$$w(k+H) - w(k) + (1 - a_r^H)\,(w(k) - y_p(k)) = \underline{C}_m \underline{A}_m^H \underline{x}_m(k) + \Sigma_0(H)u(k) + \Sigma_1(H)\Delta u(k) - y_m(k) \tag{85}$$

- Let's assume that we have a ramp reference, which can be written as

$$w(k+H) = w(k) + H\Delta w \tag{86}$$

- We get a simplified equality at the coincidence horizon H

$$H\Delta w + (1 - a_r^H)\,(w(k) - y_p(k)) - \underline{C}_m \underline{A}_m^H \underline{x}_m(k) + y_m(k) = \Sigma_0(H)u(k) + \Sigma_1(H)\Delta u(k) \tag{87}$$

Model-based PFC controller for ramp reference signal

- We have one equation and two unknown values which determine the control signal, $u(k)$ and $\Delta u(k)$
- We define two equations at two (sufficiently different) coincidence horizons H_1 and H_2

$$H_1 \Delta w + (1 - a_r^{H_1})\,(w(k) - y_p(k)) - \underline{C}_m \underline{A}_m^{H_1} \underline{x}_m(k) + y_m(k) = \Sigma_0(H_1) u(k) + \Sigma_1(H_1) \Delta u(k) \quad (88)$$

$$H_2 \Delta w + (1 - a_r^{H_2})\,(w(k) - y_p(k)) - \underline{C}_m \underline{A}_m^{H_2} \underline{x}_m(k) + y_m(k) = \Sigma_0(H_2) u(k) + \Sigma_1(H_2) \Delta u(k) \quad (89)$$

- in the matrix form

Introduction to statistical methods

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for lectures, March 2016

Table of contents

1. Introduction to statistical methods
2. Deterministic and random variables
3. Probability density function distribution
4. Stochastic processes and ergodicity
5. Expected value and variance
6. Variance
7. Correlation and covariance
8. Estimator properties
9. Concepts of bias and convergence

Introduction to statistical methods

This chapter includes:

- a brief introduction to statistics,
- deterministic and random variables,
- elementary methods and
- statistical properties.

If we perform a certain test on a process and always get the same output, then we can say that the output **is deterministic** and it can be written in this way. Therefore, the output can be predicted for the selected input. If we cannot predict the exact result, then we deal with a **non-deterministic system or variable**. The reason for unrepeatability of the experiment is often attributed to the process noise, which cannot be described and therefore we cannot predict its value.

Random variables are very important for describing the nature of processes, although the classical physics notation is deterministic. We cannot model many deterministic phenomena since we do not understand the physical background. Therefore they seem to be random.
Certain phenomena are the result of the effect of variables that are not measured (or cannot be measured). Therefore they are represented as a disturbance which we call noise.

Introduction to statistical methods

Depending on the nature of the noise, for its description it makes sense to use random variables and stochastic processes which are described with expected values, variances, etc. To explain the concept of deterministic and random variables we selected an example in figure 1. The immeasurable output of the process y_u is disturbed by the additive noise n, which results in the measurable output signal y.

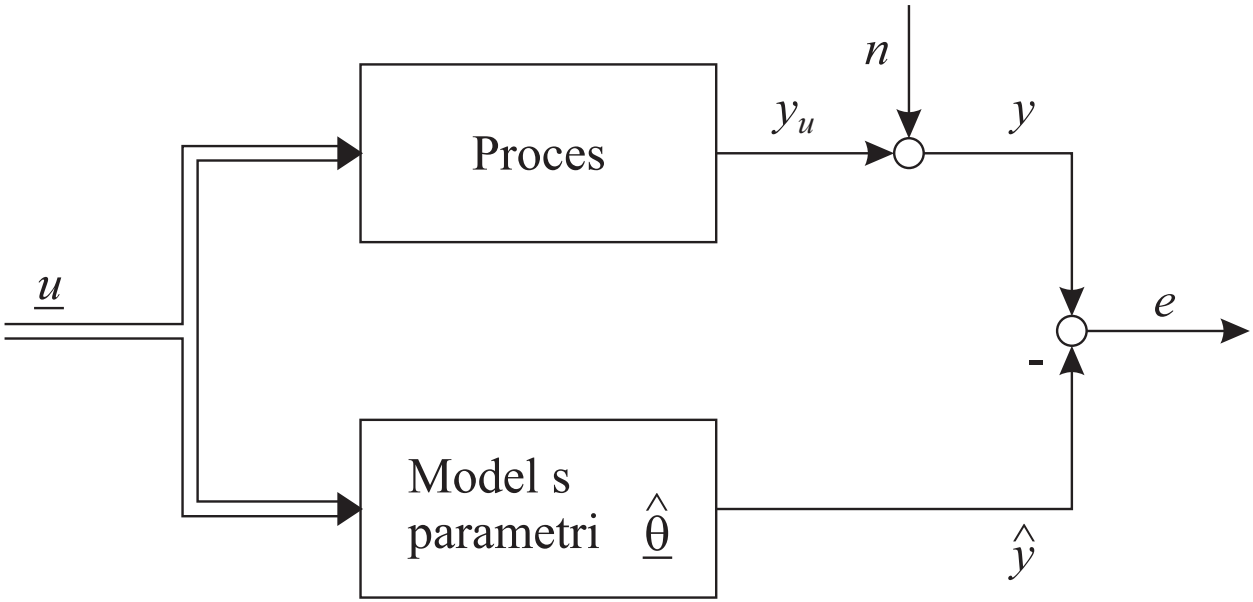


Figure 1: Deterministic and random variables in process modeling

It should be emphasized that the model output $\hat{y}$ is also of a random nature, since it is the result of the input and the estimated parameters of the model which take random values. Therefore, statistical analysis methods should be used when dealing with the estimated parameters. **The expected values of the estimated parameters and the covariance matrices give information about the quality of the estimate.**

Probability density function (pdf)

Random variables:

- are given with probability.
- Probability density function is the most comprehensive way of presenting information about random variables.
- Figure 2: *Gaussian* probability density distribution $p(x)$; *normal* distribution .
- The probability that x lies in the interval between $[a, b]$ is equal to the gray coloured surface under probability density function; x is a continuous variable; the probability for an exact value of x is equal to 0; in the interval between $[-\infty, \infty]$ the probability is equal to 1.

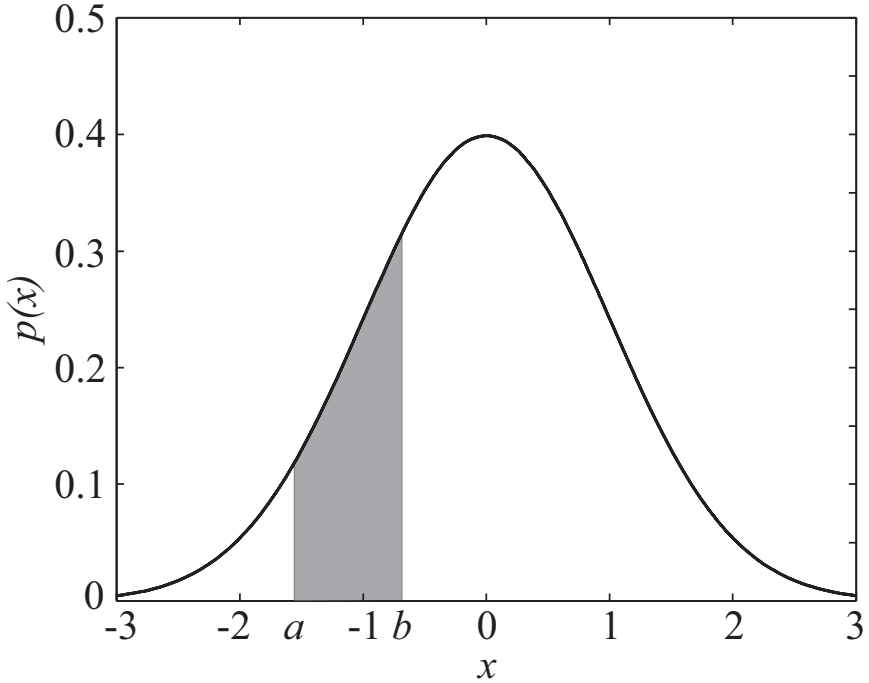


Figure 2: One-dimensional Gaussian pdf. The probability that x lies in the interval between $[a, b]$ is equal to the gray coloured surface under pdf.

Probability density function (pdf)

Pdf determines the probability of **realization of a certain random variable** if we perform an experiment. Figure 3: realization of Gaussian distribution; 100 experiments; each experiment results in a certain realization; most likely values are close to zero; experiments are not time-dependent and can be realized simultaneously.

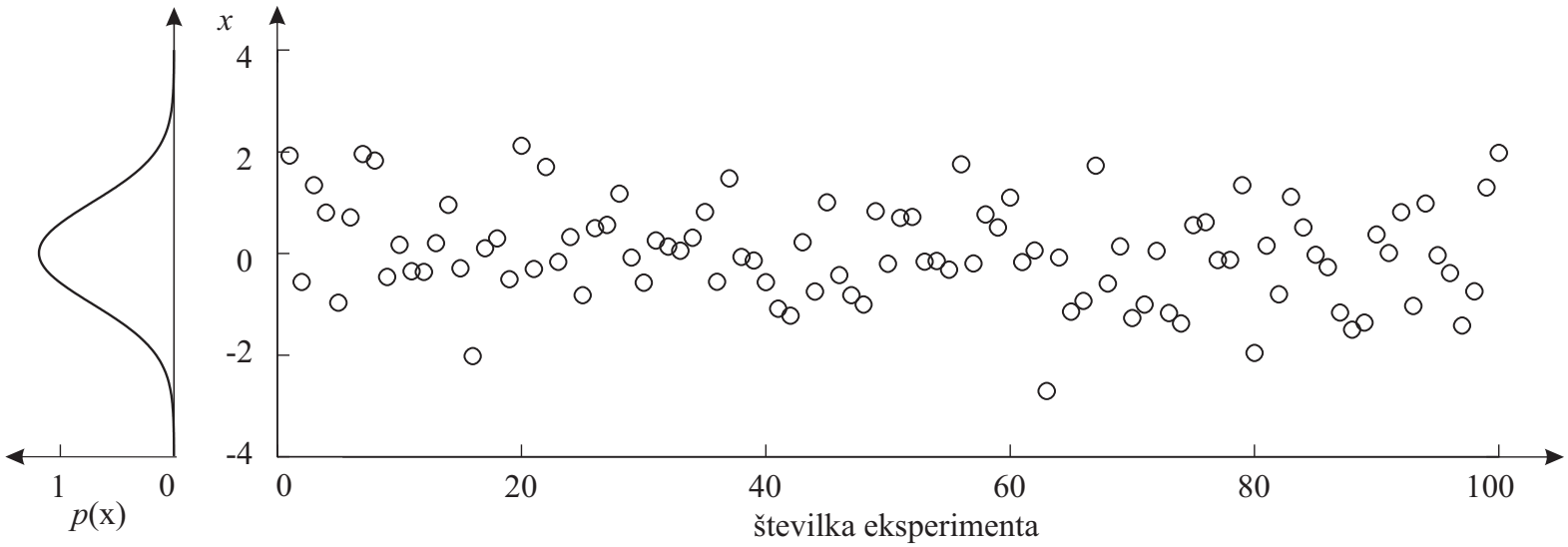


Figure 3: A random variable with Gaussian distribution (pdf) has different realizations for each experiment. The probability for each experiment is determined by pdf. The experiment number is not time-related.

- **More random variables**: vector $\underline{x} = [x_1 \; x_2 \; \cdots \; x_n]^{\mathrm{T}}$; the distribution function $p(\underline{x})$ is multidimensional; **combined pdf** in figure 4 and its contour diagrams.
- The pdf on the left is **axially orthogonal**, which means that it can be constructed as a product of two one-dimensional pdfs $p_{x_1}(x_1)$ and $p_{x_2}(x_2)$. This does not apply to the pdf on the right.

Probability density function (pdf)

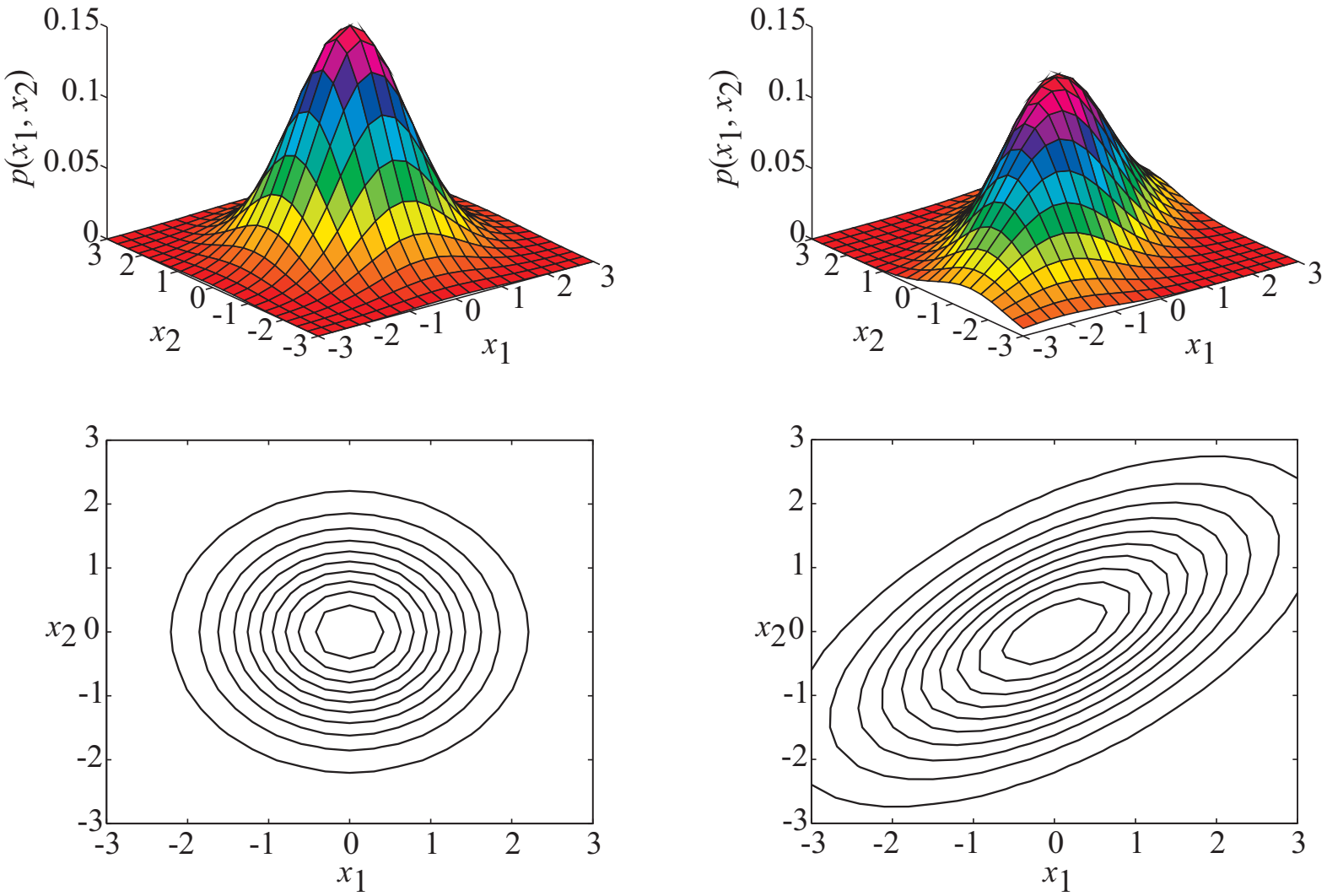


Figure 4: Two-dimensional Gaussian pdf. The left pdf describes two independent random variables. The information about the realization of one of them does not provide any information about the distribution of another random variable. The right figure shows two dependent variables. The information about one variable also provides information about another variable.

Probability density function (pdf)

Two random variables x_1 and x_2 are *independent* if the product of their probability density distribution functions leads to a common probability density distribution function:

$$p(x_1, x_2) = p_{x_1}(x_1) \cdot p_{x_2}(x_2) \,. \tag{1}$$

This means that independent random variables can be described just with one-dimensional pdf $p_{x_i}(x_i)$. In figure 4 (right) we can see two dependent variables. If we know the realization for x_1, we get also the information about x_2. For small values of x_1, small values of x_2 are more likely than large ones.

Stochastic processes and ergodicity

Stochastic processes:

- A random variable is described by the probability density function that leads to the realization of the variable when the experiment is performed.
- A random variable x, which depends not only on the experiment, but also on the time k, $x(k)$ is called the *stochastic process* .
- A stochastic process $x(k)$ is described with pdf $p(x, k)$, which also depends on the time k.
- In this way, the stochastic process can also be presented as an infinite number of random variables, each for one time moment k.

Stochastic processes and ergodicity

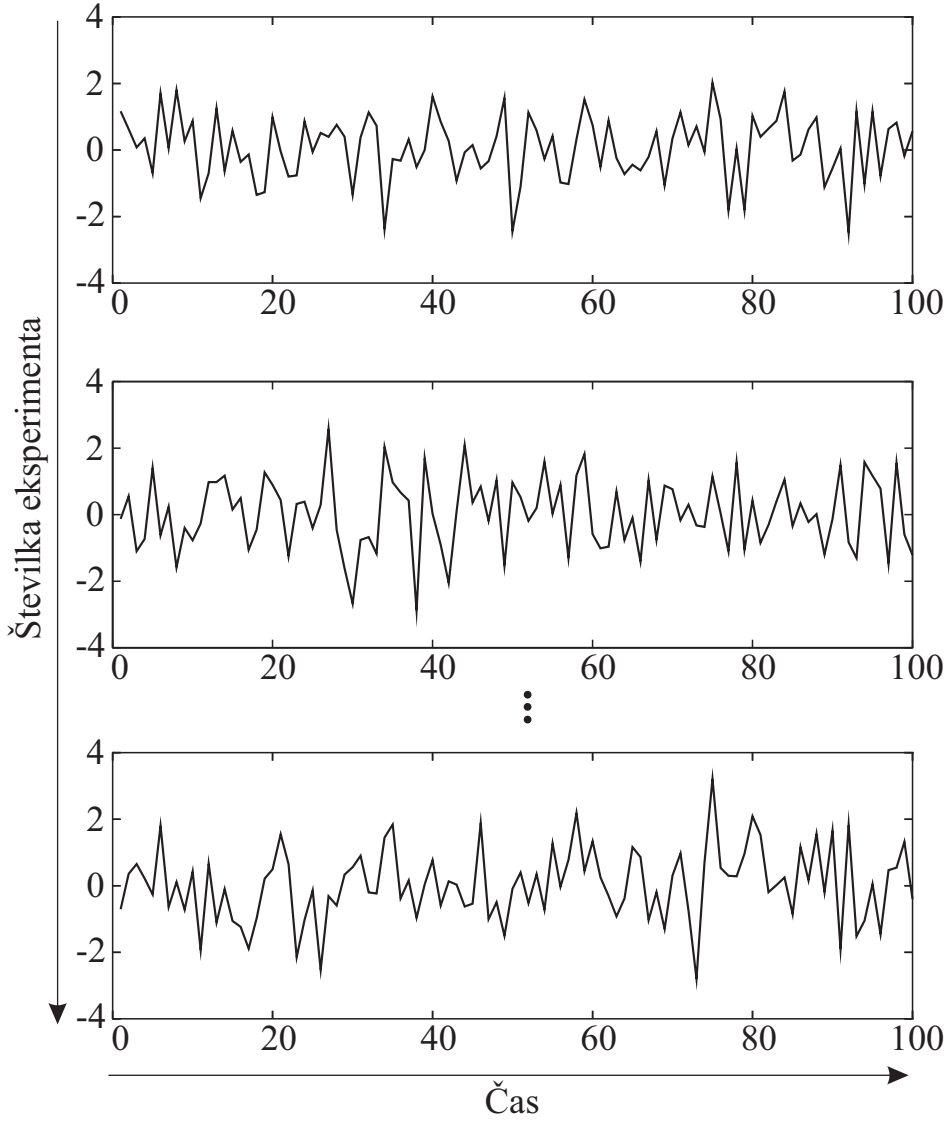


Figure 5: The stochastic process depends on experimental realization and time. Even if the statistical properties of the process do not change, we get a different second realization of the stochastic process in the second measurement. This is a stochastic property of a random variable.

Stochastic processes and ergodicity

Random variables with Gaussian distribution (i.e. white noise) are common. This means that we have an amplitude distribution, which is described with $p(x, k_0)$ of the stochastic process at every time moment k_0. This signifies the probability of occurrence of small and larger values of $x(k_0)$. On the contrary, white noise means the relation between two consecutive time samples $p(x, k_1)$ and $p(x, k_2)$ of the stochastic process. If the statistical properties of the stochastic process do not change with time, then such stochastic processes are called stationary processes.

The stochastic process is *stationary* if the statistical properties of the process do not change with time. This means that

$$p(x, k_1) = p(x, k_2) \qquad \textit{for all } k_1 \textit{ and } k_2 \,. \tag{2}$$

- Noise is usually modelled as a stationary process. Of course, it can happen that noise changes over time, e.g. it increases monotonously, which we call the signal drift.
- Usually, in the case of noise, we assume that the expectations for *all* different realizations, from top to bottom in figure 5, are equal to the expectation of the *single* time realization in figure 5 .

Stochastic processes and ergodicity

If we can calculate the expected value for a stochastic process from one time realization, then this is an *ergodic* process.

The stochastic process is *ergodic* if the expected value can be calculated through various realizations as the time average of one realization.

Ergodicity implies that when $k \to \infty$ one realization contains all the information about the statistical properties of the stochastic process. For this process, basic statistical properties, mean value and variance can be calculated.

- Ergodic process is stationary.
- The time average eliminates time dependency. For this reason, the statistical properties of an ergodic process are independent of time. Let's take a look at some examples of non-stationary processes that are also non-ergodic.

Stochastic processes and ergodicity

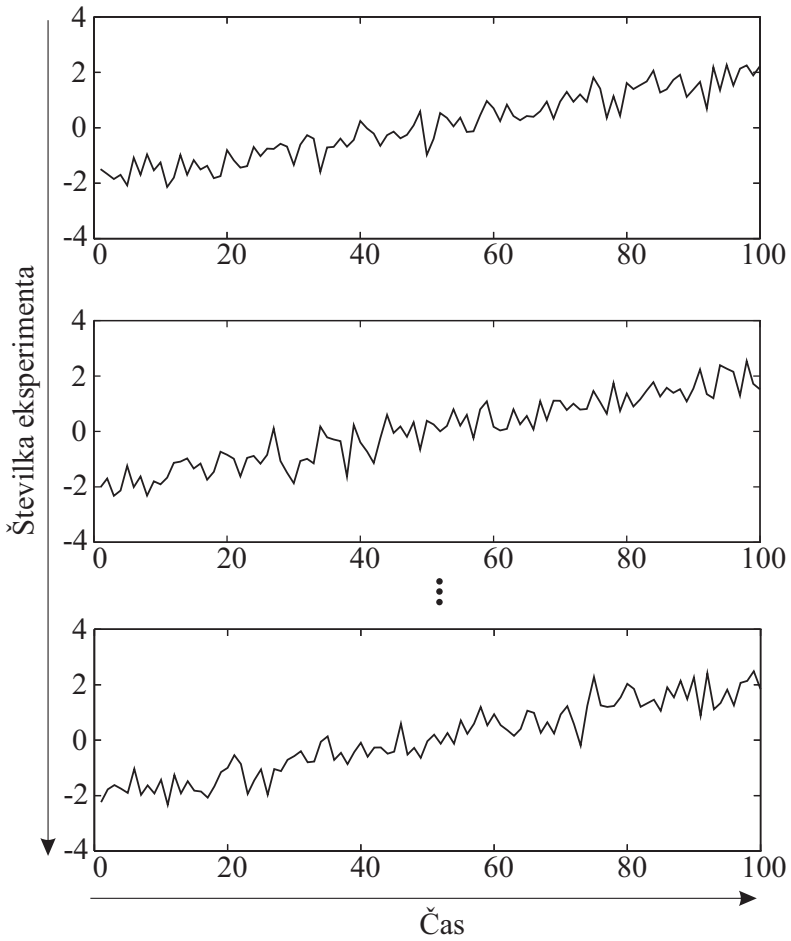


Figure 6: Non-stationary and non-ergodic stochastic process. The mean value of the stochastic process varies with time. Therefore, the statistical properties change over time and the process is non-stationary. For this reason, the time average (from left to right) is completely different from the average with respect to the realizations (top-down)

Stochastic processes and ergodicity

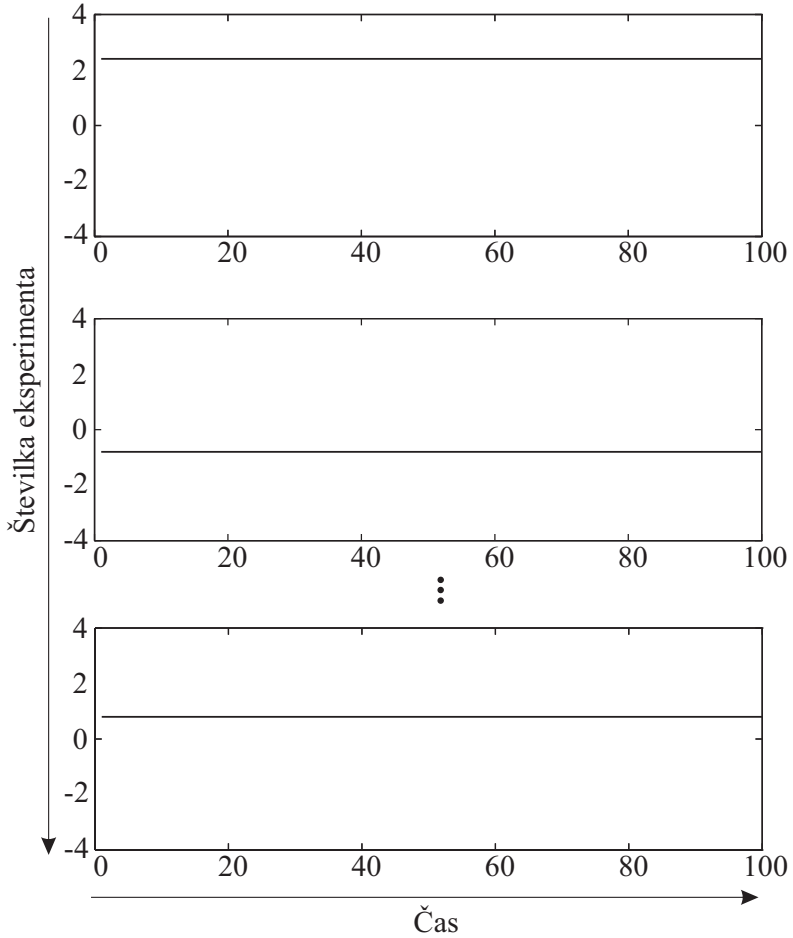


Figure 7: Stationary but non-ergodic processes. Since the statistical properties of the process are not time dependent, the process is stationary. However, this process is not ergodic, since one realization does not contain information about the statistical properties of the process

Expected value and variance

The two most important quantities that determine the properties of random variables and stochastic processes are their **expected value** or **mean** and **variance** .

Expected value operator. The argument is usually a random variable or a stochastic process, and the result is the expected value of a variable or a stochastic process. If the argument of the operator **is deterministic**, then we get

$$\mathrm{E}\{x\} = x$$

If we want to calculate the expected value of the random variable, all realizations must be weighted with their probability and integrated over all values. The definition of the expected value

of the random variable x is

$$\text{mean}\{x\} \;=\; \mu_x \;=\; \mathrm{E}\{x\} \;=\; \int_{-\infty}^{\infty} x\, p(x)\, dx\,. \tag{3}$$

Consequently, the expected value of the stochastic process is

$$\text{mean}\{x(k)\} \;=\; \mu_x(k) \;=\; \mathrm{E}\{x(k)\} \;=\; \int_{-\infty}^{\infty} x(k)\, p(x,k)\, dx\,. \tag{4}$$

Expected value

The calculation of the expected value requires a probability density distribution function for a random variable or process. However, this is usually not known in practice. For this reason, we assume that the stochastic process is ergodic and instead of averaging over all realizations, we average the values over one realization according to time. If $x(k)$ is the realization of a stochastic process for the time moment $k = 1, \ldots, N$, then the expected value is estimated as

$$\mathrm{E}\{x\} \approx \frac{1}{N} \sum_{k=1}^{N} x(k) = \bar{x} \tag{5}$$

- $\bar{x}$ is experimentally determined mean value.
- We note that in equation 5 $\mathrm{E}\{x\}$ does not depend on the time k, since we calculate the average with respect to this variable.
- For this reason it is essential that the stochastic process be stationary if we want to use equation 5.
- If the stochastic process is ergodic, then the estimate in equation 5 converges to the true expected value if $N \to \infty$.
- Hereinafter we assume the ergodicity of processes, although this is not particularly emphasized.

Expected value

The definition of the expected value operator can be extended to vectors and matrices as follows:

$$\mathrm{E}\{\underline{x}\} = \mathrm{E}\left\{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\right\} = \begin{bmatrix} \mathrm{E}\{x_1\} \\ \mathrm{E}\{x_2\} \\ \vdots \\ \mathrm{E}\{x_n\} \end{bmatrix} \tag{6}$$

$$\begin{aligned} \mathrm{E}\{\underline{X}\} &= \mathrm{E}\left\{\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}\right\} \\ &= \begin{bmatrix} \mathrm{E}\{x_{11}\} & \mathrm{E}\{x_{12}\} & \cdots & \mathrm{E}\{x_{1n}\} \\ \mathrm{E}\{x_{21}\} & \mathrm{E}\{x_{22}\} & \cdots & \mathrm{E}\{x_{2n}\} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}\{x_{m1}\} & \mathrm{E}\{x_{m2}\} & \cdots & \mathrm{E}\{x_{mn}\} \end{bmatrix} \end{aligned} \tag{7}$$

Variance

Simplified: **variance** defines the width of pdf. A low variance means that realizations close to the expected value are very likely, and a high variance means that there is a high likelihood that the realizations could be very far from the expected value.

Variance of a random variable is calculated as the **second moment**, i.e. the sum of squared deviations from the mean, weighted by the corresponding probability:

$$\mathrm{var}\{x\} \;=\; \sigma_x^2 \;=\; \mathrm{E}\left\{(x-\mu_x)^2\right\} \;=\; \int_{-\infty}^{\infty} (x-\mu_x)^2\, p(x)\, dx\,. \tag{8}$$

Consequently, the variance of the stochastic process is

$$\begin{aligned} \mathrm{var}\{x(k)\} \quad &= \quad \sigma_x^2(k) = \mathrm{E}\left\{(x(k)-\mu_x(k))^2\right\} \\ &= \quad \int_{-\infty}^{\infty} (x(k)-\mu_x(k))^2\, p(x,k)\, dx\,. \end{aligned} \tag{9}$$

Variance

Two different cases can be distinguished in calculating the variance:

- The mean value of the stochastic process is known,
- The mean value of the stochastic process is estimated using equation 5.

For a known mean value μ_x, we get (in the first case) **an unbiased estimate of variance**:

$$\mathrm{E}\{(x-\mu_x)^2\} \approx \frac{1}{N}\sum_{i=1}^{N}(x(i)-\mu_x)^2 \ . \tag{10}$$

For the second case, with an experimentally estimated mean value $\bar{x}$, the unbiased estimate of variance equals to

$$\mathrm{E}\{(x-\bar{x})^2\} \approx \frac{1}{N-1}\sum_{i=1}^{N}(x(i)-\bar{x})^2 = s^2 \ . \tag{11}$$

Variance

Since the variance σ^2 is a quadratic distance measure, for illustration it is often easier to use **standard deviation** σ, which is the square root of variance.

If we have a Gaussian distribution with a mean value μ and a standard deviation σ, then:

- the intervals around the mean value $[\mu \pm \sigma]$, $[\mu \pm 2\sigma]$ and $[\mu \pm 3\sigma]$, which we call one, two and three-sigma intervals, represent 68.3 %, 95.4 % and 99.7 % of the total area under the pdf curve.
- In other words, only 31.7 %, 4.6 % and 0.3 % of all realizations of the random variable are outside these intervals.
- For this reason, the standard deviation is a good intuitive estimate of the pdf width.
- E.g., confidence intervals represent the interval $[\mu - \sigma, \mu + \sigma]$, where σ is the standard deviation of noise. Also, the noise-signal ratio is given by the ratio between the noise's standard deviation and the signal.

Correlation and covariance

Correlation and **covariance** are defined for **two random variables or stochastic processes**, e.g. x and y or $x(k)$ and $y(k)$.

Interpretation of correlation and covariance. They measure the linear dependence or similarity between two random variables or processes. For two random variables x and y, **correlation is defined** as

$$\mathrm{corr}\{x, y\} \;=\; \mathrm{corr}_{xy} \;=\; \mathrm{E}\{x \cdot y\}\,, \tag{12}$$

and **covariance** as

$$\mathrm{cov}\{x, y\} \;=\; \mathrm{cov}_{xy} \;=\; \mathrm{E}\{(x - \mu_x) \cdot (y - \mu_y)\}\,. \tag{13}$$

Correlation and covariance

- If μ_x and μ_y are equal to zero, then the correlation and the covariance are the same.
- The value of the correlation is large if x and y are very correlated and small.
- If the variables are equal, i.e. $x = y$, then the covariance becomes equal to the variance of the variable x: $\mathrm{cov}\{x, x\} = \mathrm{var}\{x\}$.
- Correlation $\mathrm{corr}\{x, x\}$ is also called *auto-correlation* .
- The correlation $\mathrm{corr}\{x, y\}$ (for $x \not\equiv y$) is called *cross-correlation* .

Two random variables are *uncorrelated* if

$$\mathrm{E}\{x \cdot y\} = \mathrm{E}\{x\} \cdot \mathrm{E}\{y\} = \mu_x \cdot \mu_y \,. \tag{14}$$

Correlation and covariance

- Correlation and covariance of a stochastic process are defined as

$$\mathrm{corr}\{x, y, k_1, k_2\} = \mathrm{corr}_{xy}(k_1, k_2) = \mathrm{E}\{x(k_1) \cdot y(k_2)\}\,, \tag{15}$$

$$\begin{aligned} \mathrm{cov}\{x, y, k_1, k_2\} &= \mathrm{cov}_{xy}(k_1, k_2) \\ &= \mathrm{E}\{(x(k_1) - \mu_x(k_1)) \cdot (y(k_2) - \mu_y(k_2))\}\,. \end{aligned} \tag{16}$$

- Correlation and covariance depend on the time k_1 and k_2.
- If the stochastic process is stationary, then the statistical properties do not depend on time. Consequently, both the correlation and the covariance depend only on the time shift $\kappa = k_2 - k_1$ between $x(k_1)$ and $y(k_2)$.
- We obtain the following equation, which does not depend on time

$$\mathrm{corr}\{x, y, \kappa\} \;=\; \mathrm{corr}_{xy}(\kappa) \;=\; \mathrm{E}\{x(k) \cdot y(k+\kappa)\}\,, \tag{17}$$

$$\mathrm{cov}\{x, y, \kappa\} \;=\; \mathrm{cov}_{xy}(\kappa) \;=\; \mathrm{E}\{(x(k) - \mu_x) \cdot (y(k+\kappa) - \mu_y)\}\,. \tag{18}$$

Correlation and covariance

- If **we do not know the probability density distribution** when calculating the expected values, **we assume the ergodicity and calculate the correlation and the covariance from the time average**.
- If $x(k)$ and $y(k)$ are measurements of the stochastic process realization for the time moments $k = 1, \ldots, N$, then the correlation and covariance can be estimated as

$$\mathrm{corr}_{xy}(\kappa) \approx \frac{1}{N - |\kappa|} \sum_{k=1}^{N-|\kappa|} x(k)\, y(k+\kappa)\,, \tag{19}$$

$$\mathrm{cov}_{xy}(\kappa) \approx \frac{1}{N - |\kappa| - 1} \sum_{k=1}^{N-|\kappa|} (x(k) - \bar{x})\ (y(k+\kappa) - \bar{y})\ . \tag{20}$$

- The two estimates above are unbiased, since the sum is divided by a factor which equals to the number of terms in the sum. Since we have only N samples available, $|\kappa| = N - 1$ represents the maximum possible time shift.

Correlation and covariance

Although the estimates are unbiased, the error variance estimate is very large ($\kappa \to N$) for large shifts, which is a consequence of a small number of terms. Therefore, in this case, the estimate is completely useless. So the problem is usually solved with the following estimate:

$$\mathrm{corr}_{xy}(\kappa) \approx \frac{1}{N} \sum_{k=1}^{N-|\kappa|} x(k)\, y(k+\kappa)\,, \tag{21}$$

$$\mathrm{cov}_{xy}(\kappa) \approx \frac{1}{N-1} \sum_{k=1}^{N-|\kappa|} \left(x(k) - \bar{x}\right) \left(y(k+\kappa) - \bar{y}\right)\,. \tag{22}$$

Correlation and covariance

The auto-correlation ($x = y$) $\mathrm{corr}_{xx}(\kappa)$ calculates the correlation between $x(k)$ and the time-shifted signal $x(k-\kappa)$. Auto-correlation is always symmetrical.

If $\underline{x} = [x_1 \; x_2 \; \cdots \; x_n]$ is the vector of random variables, the covariance matrix is defined by

$$\mathrm{cov}\{\underline{x}\} = \mathrm{cov}\{\underline{x}, \underline{x}\} = \begin{bmatrix} \mathrm{cov}\{x_1, x_1\} & \mathrm{cov}\{x_1, x_2\} & \cdots & \mathrm{cov}\{x_1, x_n\} \\ \mathrm{cov}\{x_2, x_1\} & \mathrm{cov}\{x_2, x_2\} & \cdots & \mathrm{cov}\{x_2, x_n\} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{cov}\{x_n, x_1\} & \mathrm{cov}\{x_n, x_2\} & \cdots & \mathrm{cov}\{x_n, x_n\} \end{bmatrix} \tag{23}$$

The diagonal of the matrix contains variances. Therefore, this matrix is sometimes called a variance-covariance matrix.

Estimator properties

Evaluation of the estimation quality

- In the estimation we usually take a large number of measurements N, which in some ways are mapped to a smaller number of model parameters $\underline{\theta} = [\theta_1 \ \theta_2 \ \cdots \ \theta_n]^{\mathrm{T}}$.
- We can represent them statistically with pdf $p(\hat{\underline{\theta}}, N)$, which is also a function of the number of measurements N.
- The most important properties of pdf are the expected value (mean) $\mathrm{E}\{\hat{\underline{\theta}}\}$ and the covariance matrix (variance in the one-dimensional case) $\mathrm{cov}\{\hat{\underline{\theta}}\}$. If $N \to \infty$, then pdf is often Gaussian, which is unambiguously defined by the mean and covariance matrix.
- A good estimation results in a sharp form of pdf that gives significantly higher probability to the values around the expected value than the rest. Optimal distribution of pdf would be the Dirac impulse at θ_{opt}. An important characteristic is also an improvement of the estimation while increasing the number of measurements.
- In figure 8 the three pdfs correspond to a different number of measurements N_1, N_2 and N_3, where $N_1 < N_2 < N_3$ is valid. Figure 8 shows $p(\hat{\underline{\theta}}, N)$, which becomes more and more sharp, which means that its variance decreases with the increase of the number of learning samples N.
- At $N \to \infty$ the variance goes to zero. For most estimators, the variance decreases with a factor $1/N$. This relation applies to almost all estimators.

Estimator properties

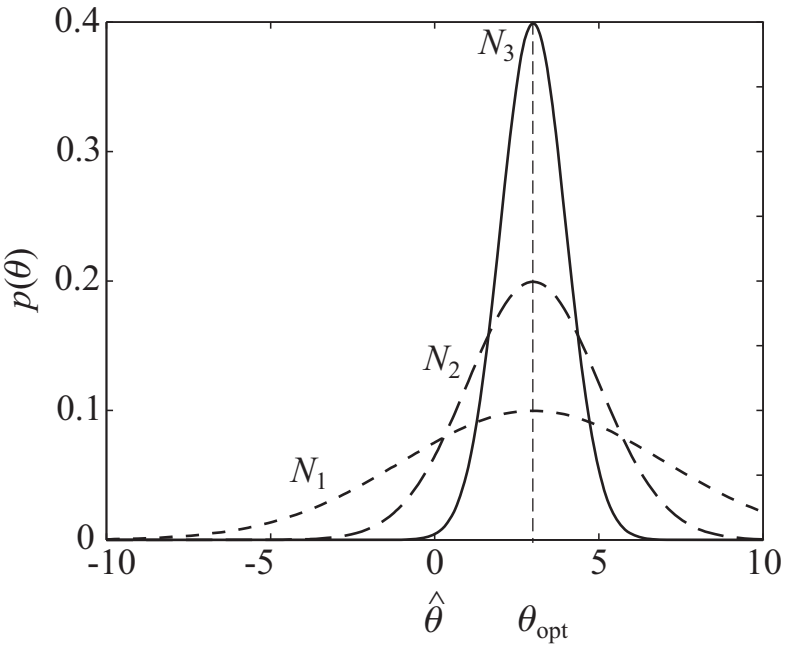


Figure 8: Pdf *o*f unbiased estimator for a different number of data $N_1 < N_2 < N_3$. The expected value of the estimated parameter $\hat{\theta}$, which is equal to the true value θ_{opt}. This property does not depend on the number of data used in the estimation

Estimator properties

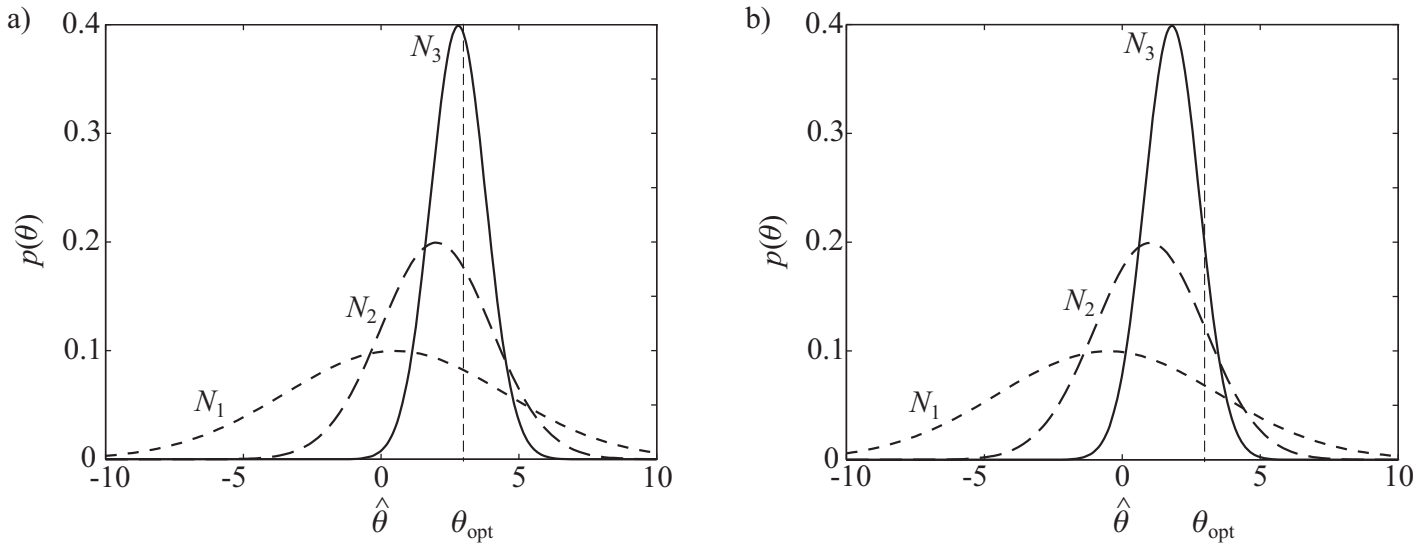


Figure 9: a) Pdf of the *biased* but *consistent* estimator for a different number of data $N_1 < N_2 < N_3$. The expected value of the parameter $\hat{\theta}$ *is not* equal to the true value θ_{opt}. With the increase of the number of measurements, the expected value of the estimated parameter approaches the true value if the number goes to infinity. b) Pdf of the *inconsistent* (and therefore also *biased*) estimator for a different number of data $N_1 < N_2 < N_3$. The expected value of the estimated parameter $\hat{\theta}$ *is not* equal to the true value θ_{opt} and does not move closer to it when the number of measurements goes to infinity. There is a systematic error between the expected value of the estimation and the true value. This is, of course, very undesirable.

Estimator properties

In figures 8 and 9, the comparison of pdfs shows a decrease of variance when the number of data is higher. The behaviour of the average value is different.

In figure 8, the average of the estimated parameters is equal to the true value of the parameter θ_{opt} irrespective of the number of data. Such an estimator is called the unbiased estimator.

The estimator is *unbiased* if for each number of data N:

$$\mathrm{E}\{\underline{\hat{\theta}}\} = \underline{\theta}_{\text{opt}} \,. \tag{24}$$

Otherwise, the estimator is *biased*. A systematic error between the expected value of the estimated parameters $\mathrm{E}\{\underline{\hat{\theta}}\}$ and the true value $\underline{\theta}_{\text{opt}}$ is called bias :

$$\underline{b} = \mathrm{E}\{\underline{\hat{\theta}}\} - \underline{\theta}_{\text{opt}} \,. \tag{25}$$

Estimator properties

- In contrast to the estimator in figure 8, in figure 9a a biased estimator is shown.
- The average estimate shows a systematic error according to the true value of the parameter. The deviation decreases with the increase of the number of data: $\underline{b}(N_1) > \underline{b}(N_2) > \underline{b}(N_3)$.
- If N goes to infinity, this error goes to zero. Such an estimator (or estimate) is called the **consistent or asymptotically unbiased** estimator. All unbiased estimators are certainly also consistent.

The estimate is *consistent* if with the increase of the number of data $N \to \infty$

$$\underline{\hat{\theta}} \to \underline{\theta}_{\text{opt}} \,. \tag{26}$$

- Although an unbiased estimator is highly desirable, a biased but consistent (we can often provide a very large amount of data) estimator is often adequate.
- It can be proved that all maximum likelihood estimators are consistent.
- In figure 9b, an inconsistent estimator is shown. This means that the parameter estimation does not converge to the true value if $N \to \infty$. **An inconsistent estimate is not acceptable**.

Estimator properties

A good estimator is unbiased or at least consistent. This property is rather unusable if the variance of the parameters is very large. For this reason, the variance must also be within certain limits.

- Figure 10 shows why a biased low-variance estimator can be better than an unbiased estimator with a large variance.
- Compromise between bias and variance is a known problem in statistics, which we often encounter when estimating parameters.
- The methods that take into account this compromise are called *regularization techniques* .

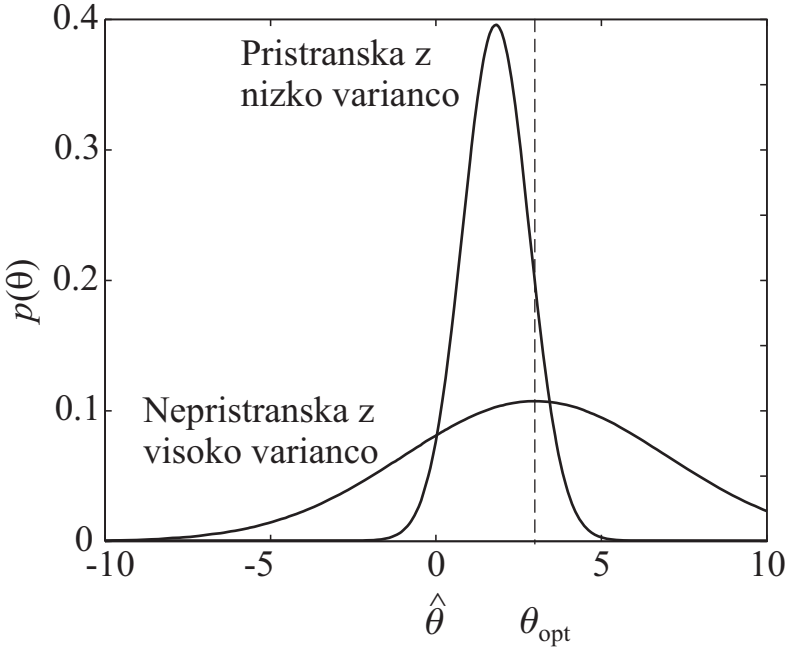


Figure 10: A biased low-variance estimator against unbiased estimator with a high variance

Concepts of bias and convergence

- In theoretical formulas the integral boundaries are infinite. In practical calculations these boundaries are replaced by the final values.
- The quantity obtained in this way is the estimate of the actual quantity which we label as ($\hat{}$).
- In the presence of stochastic disturbances two questions about the correctness of the estimate arise. We need to find out if the expected value of the estimate equals the true value. If this is the case, we say that the estimate **is unbiased**. In addition, we say that the estimate **is consistent** if with the increase of the observation interval it improves and limits to the true value when the observation interval goes to infinity.
- An important question is whether the variance of the estimation error goes to zero when the observation interval goes to infinity. If this is true, then the estimate is consistent in the mean square value.

The mean value of the stochastic variable x is equal to

$$\bar{x} = E\{x(k)\} = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} x(k) \tag{27}$$

and its estimation is

$$\hat{\bar{x}} = \frac{1}{N} \sum_{k=1}^{N} x(k) \tag{28}$$

Example: Mean value estimate of stochastic variable

Example: Mean value estimate of stochastic variable

$$\hat{\bar{x}} = \frac{1}{N}\sum_{k=1}^{N} x(k) \tag{29}$$

We need to find out if the estimate **is unbiased, consistent, and consistent in the mean square value**. Let's first look at the **expected value of the mean value estimate**

$$E\{\hat{\bar{x}}\} = E\left\{\frac{1}{N}\sum_{k=1}^{N} x(k)\right\} = \frac{1}{N}\sum_{k=1}^{N} E\{x(k)\} = \frac{1}{N}N\bar{x} = \bar{x} \tag{30}$$

The expected value of the estimate is equal to the true value already for final observation intervals N. Therefore, the estimate is unbiased and also consistent.

Example: Estimate of the variance of the mean value estimate error

Example: Estimate of the variance of the mean value estimate error
Let's take a look at how we obtain **the variance of the mean value estimate error**

$$E\{(\hat{\bar{x}}-\bar{x})^2\} = E\left\{\left[\left(\frac{1}{N}\sum_{k=1}^{N}x(k)\right)-\bar{x}\right]^2\right\} = E\left\{\left[\frac{1}{N}\sum_{k=1}^{N}(x(k)-\bar{x})\right]^2\right\} \tag{31}$$

If the individual values of the random variable x are mutually statistically independent (white noise), then $E\{[\sum_{k=1}^{N}(x(k)-\bar{x})]^2\} = \sum_{k=1}^{N} E\left\{(x(k)-\bar{x})^2\right\} = N\sigma_x^2$ (example in Matlab). In this way we get:

$$\begin{aligned} E\{(\hat{\bar{x}}-\bar{x})^2\} &= E\left\{\frac{1}{N^2}\sum_{k=1}^{N}(x(k)-\bar{x})^2\right\} \\ &= \frac{1}{N^2}\sum_{k=1}^{N}E\left\{(x(k)-\bar{x})^2\right\} \\ &= \frac{1}{N}\sigma_x^2\,, \end{aligned} \tag{32}$$

- The variance of estimation error decreases with the increase of the observation interval and it goes to zero in the limit. Therefore, the estimate is **consistent in the mean square value**.

Example: Estimate of the variance of a stochastic variable

□

Example: **Estimate of the variance of a stochastic variable**
The variance of the stochastic variable x is estimated as

$$\hat{\sigma}_x^2 = \frac{1}{N}\sum_{k=1}^{N}[x(k) - \hat{\bar{x}}]^2 \tag{33}$$

Bias and consistency of the estimate.

The expected value of the estimate of variance.

$$E\{\hat{\sigma}_x^2\} = E\left\{\frac{1}{N}\sum_{k=1}^{N}(x(k) - \hat{\bar{x}})^2\right\} \tag{34}$$

Estimate of the variance of a stochastic variable

With the transformation we get

$$\begin{aligned}\sum_{k=1}^{N}(x(k)-\hat{\bar{x}})^2 &= \sum_{k=1}^{N}\left[(x(k)-\bar{x})-(\hat{\bar{x}}-\bar{x})\right]^2 \\ &= \sum_{k=1}^{N}[x(k)-\bar{x}]^2+\sum_{k=1}^{N}(\hat{\bar{x}}-\bar{x})^2-2(\hat{\bar{x}}-\bar{x})\sum_{k=1}^{N}[x(k)-\bar{x}]\end{aligned} \tag{35}$$

since $\sum_{k=1}^{N}[x(k)-\bar{x}]=N(\hat{\bar{x}}-\bar{x})$

$$\begin{aligned}&= \sum_{k=1}^{N}[x(k)-\bar{x}]^2+N(\hat{\bar{x}}-\bar{x})^2-2(\hat{\bar{x}}-\bar{x})N(\hat{\bar{x}}-\bar{x}) \\ &= \sum_{k=1}^{N}[x(k)-\bar{x}]^2-N(\hat{\bar{x}}-\bar{x})^2\end{aligned} \tag{36}$$

Considering equation (32) we get

$$E\{\hat{\sigma}_x^2\}=\frac{1}{N}\left[\sum_{k=1}^{N}E\{[x(k)-\bar{x}]^2\}-NE\{(\hat{\bar{x}}-\bar{x})^2\}\right]=\sigma_x^2-\frac{1}{N}\sigma_x^2 \tag{37}$$

The estimate of variance is **biased** for the final values N. The smaller the bias, the larger the N. For $N\to\infty$ the bias disappears. Therefore, the estimate **is consistent**.

Addition to matrix algebra

prof. dr. Igor Škrjanc

Faculty of Electrical Engineering

PPT slides for lectures, February 2019

Table of contents

1. Vectors and matrices
2. Vector and matrix derivatives
3. Gradient, Hessian and Jacobian matrix

Vectors and matrices

Differentiation rules for vectors and matrices.

Derivative of m-dimensional column vector $\underline{x} = [x_1 \; x_2 \; \cdots \; x_m]^{\mathrm{T}}$ with respect to the scalar θ equals to

$$\frac{\partial \underline{x}}{\partial \theta} = \begin{bmatrix} \partial x_1/\partial\theta \\ \partial x_2/\partial\theta \\ \vdots \\ \partial x_m/\partial\theta \end{bmatrix} \tag{1}$$

Derivative of matrix $\underline{X}$ **with dimensions** $l \times m$ with respect to the scalar θ is defined as

$$\frac{\partial \underline{X}}{\partial \theta} = \begin{bmatrix} \partial X_{11}/\partial\theta & \partial X_{12}/\partial\theta & \cdots & \partial X_{1m}/\partial\theta \\ \partial X_{21}/\partial\theta & \partial X_{22}/\partial\theta & \cdots & \partial X_{2m}/\partial\theta \\ \vdots & \vdots & \ddots & \vdots \\ \partial X_{l1}/\partial\theta & \partial X_{l2}/\partial\theta & \cdots & \partial X_{lm}/\partial\theta \end{bmatrix} \tag{2}$$

(3)

Vectors and matrices

To obtain the derivative with respect to the n-dimensional vector $\underline{\theta} = [\theta_1 \ \theta_2 \ \cdots \ \theta_n]^{\mathrm{T}}$, we define a differentiation operator:

$$\frac{\partial}{\partial \underline{\theta}} = \begin{bmatrix} \partial/\partial\theta_1 \\ \partial/\partial\theta_2 \\ \vdots \\ \partial/\partial\theta_n \end{bmatrix} \tag{4}$$

Using equation 4 the derivative of the scalar x with respect to n-dimensional vector $\underline{\theta}$ is defined as

$$\frac{\partial x}{\partial \underline{\theta}} = \begin{bmatrix} \partial x/\partial\theta_1 \\ \partial x/\partial\theta_2 \\ \vdots \\ \partial x/\partial\theta_n \end{bmatrix} \tag{5}$$

Derivative of m-dimensional vector $\underline{x}^{\mathrm{T}}$ with respect to n-dimensional column vector $\underline{\theta}$ is defined by the outer product:

$$\frac{\partial \underline{x}^{\mathrm{T}}}{\partial \underline{\theta}} = \frac{\partial}{\partial \underline{\theta}} \cdot \underline{x}^{\mathrm{T}} = \begin{bmatrix} \partial x_1/\partial\theta_1 & \partial x_2/\partial\theta_1 & \cdots & \partial x_m/\partial\theta_1 \\ \partial x_1/\partial\theta_2 & \partial x_2/\partial\theta_2 & \cdots & \partial x_m/\partial\theta_2 \\ \cdots & \vdots & & \vdots \\ \partial x_1/\partial\theta_n & \partial x_2/\partial\theta_n & \cdots & \partial x_m/\partial\theta_n \end{bmatrix} \tag{6}$$

Vectors and matrices

The vector derivative of the scalar (dot) product is equal to:

$$\frac{\partial}{\partial \underline{\theta}} \left(\underline{x}^{\mathrm{T}} \underline{y} \right) = \frac{\partial \underline{x}^{\mathrm{T}}}{\partial \underline{\theta}} \cdot \underline{y} + \frac{\partial \underline{y}^{\mathrm{T}}}{\partial \underline{\theta}} \cdot \underline{x} \,, \tag{7}$$

where $\underline{x}^{\mathrm{T}}$ is m-dimensional row vector, $\underline{y}$ is m-dimensional column vector and $\underline{\theta}$ is n-dimensional column vector.
If one of the vectors in equation 7 does not depend on the parameter $\underline{\theta}$, and another vector depends on $\underline{\theta}$, then we get:

$$\frac{\partial}{\partial \underline{\theta}} \left(\underline{z}^{\mathrm{T}} \underline{\theta} \right) = \underline{z} \tag{8}$$

and

$$\frac{\partial}{\partial \underline{\theta}} \left(\underline{\theta}^{\mathrm{T}} \underline{z} \right) = \underline{z} \tag{9}$$

where the rows and columns in vectors $\underline{z}^{\mathrm{T}}$ and $\underline{z}$ do not depend on the vector $\underline{\theta}$. We obtain the corresponding notation if vector $\underline{z}$ is replaced with matrix $\underline{Z}$:

$$\frac{\partial}{\partial \underline{\theta}} \left(\underline{Z}^{\mathrm{T}} \underline{\theta} \right) = \underline{Z} \tag{10}$$

$$\frac{\partial}{\partial \underline{\theta}} \left(\underline{\theta}^{\mathrm{T}} \underline{Z} \right) = \underline{Z} \tag{11}$$

Finally, let's take a look at **derivative of quadratic form** $\underline{\theta}^{\mathrm{T}}\underline{Z}\,\underline{\theta}$, which is a very common and important expression:

$$\frac{\partial}{\partial\underline{\theta}}\left(\underline{x}^{\mathrm{T}}\underline{y}\right) = \frac{\partial\underline{x}^{\mathrm{T}}}{\partial\underline{\theta}}\cdot\underline{y} + \frac{\partial\underline{y}^{\mathrm{T}}}{\partial\underline{\theta}}\cdot\underline{x}\,, \tag{12}$$

$$\begin{aligned}\frac{\partial}{\partial\underline{\theta}}\left(\underbrace{\underline{\theta}^{\mathrm{T}}\underline{Z}}_{\underline{x}^{\mathrm{T}}}\underbrace{\underline{\theta}}_{\underline{y}}\right) &= \frac{\partial}{\partial\underline{\theta}}\left(\underline{\theta}^{\mathrm{T}}\underline{Z}\right)\cdot\underline{\theta} + \frac{\partial}{\partial\underline{\theta}}\left(\underline{\theta}^{\mathrm{T}}\right)\cdot\left(\underline{\theta}^{\mathrm{T}}\underline{Z}\right)^{\mathrm{T}} \\ &= \underline{Z}\,\underline{\theta} + \underline{I}\,\underline{Z}^{\mathrm{T}}\underline{\theta} = \left(\underline{Z} + \underline{Z}^{\mathrm{T}}\right)\underline{\theta}\,,\end{aligned} \tag{13}$$

where matrix $\underline{Z}$ has dimension of $n \times n$

- In the first step, equation 7 is used.
- The second step is $\partial\underline{\theta}^{\mathrm{T}}/\partial\underline{\theta}$, which is equal to identity matrix $\underline{I}$ according to equation 6.
- The notation from equation 13 can be simplified in $2\underline{Z}\,\underline{\theta}$ if $\underline{Z}$ is a symmetric matrix.

Gradient, Hessian and Jacobian matrix

Gradient $\underline{g}$ is the first derivative of a scalar function with respect to the vector of parameters $\underline{\theta}$.

- The gradient of the function $f(\underline{\theta})$ depends on n-dimensional vector of parameters $\underline{\theta} = [\theta_1 \ \theta_2 \ \cdots \ \theta_n]^{\mathrm{T}}$. According to these parameters, it can be written as n-dimensional vector:

$$\underline{g} = \frac{\partial f(\underline{\theta})}{\partial \underline{\theta}} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} = \begin{bmatrix} \partial f(\underline{\theta}) \,/\, \partial \theta_1 \\ \partial f(\underline{\theta}) \,/\, \partial \theta_2 \\ \vdots \\ \partial f(\underline{\theta}) \,/\, \partial \theta_n \end{bmatrix} \tag{14}$$

- Geometrically, the gradient is the direction of the steepest slope of the function $f(\underline{\theta})$.
- The stationary points (local or global) $f(\underline{\theta})$ are defined in a way that the gradient of the function is zero at these points.

Gradient, Hessian and Jacobian matrix

Hessian matrix $\underline{H}$ is the second derivative of the scalar function with respect to the vector of parameters.

- The Hessian matrix of the scalar function $f(\underline{\theta})$ depends on the n-dimensional vector of parameters $\underline{\theta} = [\theta_1 \ \theta_2 \ \cdots \ \theta_n]^{\mathrm{T}}$ and it is defined as $n \times n$-dimensional matrix:

$$\begin{aligned}
\underline{H} &= \frac{\partial^2 f(\underline{\theta})}{\partial \underline{\theta}^2} = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1n} \\ H_{21} & H_{22} & \cdots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{n1} & H_{n2} & \cdots & H_{nn} \end{bmatrix} \\
&= \begin{bmatrix} \partial^2 f(\underline{\theta}) \,/\, \partial\theta_1^2 & \partial^2 f(\underline{\theta}) \,/\, \partial\theta_1\partial\theta_2 & \cdots & \partial^2 f(\underline{\theta}) \,/\, \partial\theta_1\partial\theta_n \\ \partial^2 f(\underline{\theta}) \,/\, \partial\theta_2\partial\theta_1 & \partial^2 f(\underline{\theta}) \,/\, \partial\theta_2^2 & \cdots & \partial^2 f(\underline{\theta}) \,/\, \partial\theta_2\partial\theta_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 f(\underline{\theta}) \,/\, \partial\theta_n\partial\theta_1 & \partial^2 f(\underline{\theta}) \,/\, \partial\theta_n\partial\theta_2 & \cdots & \partial^2 f(\underline{\theta}) \,/\, \partial\theta_n^2 \end{bmatrix}
\end{aligned} \tag{15}$$

- Geometrically, the Hessian matrix represents the curvature (convexity or concavity) of the function $f(\underline{\theta})$, i.e. for a linear function $f(\underline{\theta})$ the Hessian matrix is zero and for a quadratic function $f(\underline{\theta})$ the Hessian matrix is constant.
- The Hessian matrix is symmetric.
- If the Hessian matrix is positive definite ($\underline{\theta}^T \underline{H\theta} > 0$ for each $\underline{\theta}$, convex function), then the stationary point of the function is a minimum, but if it is negative definite (concave), the stationary point is a maximum.

Jacobian matrix is the first derivative of a vector function with respect to a vector.

- Jacobian matrix of N-dimensional function $\underline{f}(\underline{\theta}) = [f_1(\underline{\theta})\, f_2(\underline{\theta})\ \cdots\ f_N(\underline{\theta})]^{\mathrm{T}}$, which depends on n-dimensional vector of parameters $\underline{\theta} = [\theta_1\ \theta_2\ \cdots\ \theta_n]^{\mathrm{T}}$ is defined in a way that we get $N \times n$-dimensional matrix:

$$\underline{J} = \frac{\partial \underline{f}(\underline{\theta})}{\partial \underline{\theta}} = \begin{bmatrix} (\partial f_1(\underline{\theta}) \,/\, \partial \underline{\theta})^{\mathrm{T}} \\ (\partial f_2(\underline{\theta}) \,/\, \partial \underline{\theta})^{\mathrm{T}} \\ \vdots \\ (\partial f_N(\underline{\theta}) \,/\, \partial \underline{\theta})^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} & \cdots & J_{1n} \\ J_{21} & J_{22} & \cdots & J_{2n} \\ \vdots & \vdots & & \vdots \\ J_{N1} & J_{N2} & \cdots & J_{Nn} \end{bmatrix}$$

$$= \begin{bmatrix} \partial f_1(\underline{\theta}) \,/\, \partial \theta_1 & \partial f_1(\underline{\theta}) \,/\, \partial \theta_2 & \cdots & \partial f_1(\underline{\theta}) \,/\, \partial \theta_n \\ \partial f_2(\underline{\theta}) \,/\, \partial \theta_1 & \partial f_2(\underline{\theta}) \,/\, \partial \theta_2 & \cdots & \partial f_2(\underline{\theta}) \,/\, \partial \theta_n \\ \vdots & \vdots & & \vdots \\ \partial f_N(\underline{\theta}) \,/\, \partial \theta_1 & \partial f_N(\underline{\theta}) \,/\, \partial \theta_2 & \cdots & \partial f_N(\underline{\theta}) \,/\, \partial \theta_n \end{bmatrix}. \tag{16}$$