INTERNATIONAL JOURNAL AUTOMATION AUSTRIA

HEFT 1,2 Jg. 6, (1998)

INHALT

Optimal control of ARX model

J. Štecha, R. Rázek

Department of Automatic Control, FEE, Czech technical university in Prague

Submitted on September 1, 1997

Summary: *Stochastic optimal control strategies for a discrete linear stochastic dynamic system described with an ARX model are derived in this paper. Dynamic programming approach is used to develop optimal control of discrete stochastic system considering quadratic criterion of optimality.*

Cautious and certainty equivalent optimal control strategies are obtained. Certainty equivalent optimal control strategies are obtained when only parameters mean is considered, while cautious strategies also consider uncertainty of parameters. To avoid state estimation, nonminimal state space representation of the ARX model is considered, where the states of the system are delayed inputs and outputs.

Simulations demonstrating the properties of stochastic optimal control are shown and advantages and disadvantages of proposed approach are discussed.

1 Introduction

The discrete time ARX (AutoRegressive stochastic model with eXternal input) single input–single output stochastic model is described by the difference equation

$$y(t) = \sum_{i=1}^{n} a_i(t)y(t-i) + \sum_{i=0}^{n} b_i(t)u(t-i) + e(t) \tag{1}$$

where $y(t)$, $u(t)$ is the output and input of the system, a_i, b_i are the parameters of the system and $e(t) = \mathcal{N}(0, \sigma_e^2)$ is the noise of output measurement. A Kalman filter can be used for estimation of the unknown parameters a_i, b_i based on measured data. Kalman filtering results in a conditional mean and a covariance matrix of the parameters.

The results of the estimation (conditional mean and a covariance matrix of the parameters) are used to develop an optimal control strategy. A quadratic criterion is chosen as a measure of the optimality of the control.

To obtain a linear optimal control strategy, which utilizes the discrete Riccati equation, it is necessary to have a state equation of the system. Nonminimal realization of the ARX model is used where nonminimal state of the system is formed from the old values of input and output.

The text is organized in the following way. First quadratic optimal cautious and certainty equivalent control strategies for linear stochastic state space model in general are developed. Afterwards Kalman filter for parameter estimation of ARX model is presented. Finally, nonminimal state space representation of ARX model is shown and its optimal control strategies, leading to linear state feedback control law, are presented.

In the second part simulations carried under MATLAB 4.0 are given to demonstrate the nature of stochastic optimal control in case of ARX model. The difference between cautious and certainty equivalent strategies is stressed.

2 Cautious and certainty equivalent control strategies

Let us consider linear stochastic system in the form

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}\boldsymbol{v}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{e}(t) \end{aligned} \tag{2}$$

where state noise $\boldsymbol{v}(t)$ and measurement noise $\boldsymbol{e}(t)$ are mutual independent white sequences with zero mean and covariances $\boldsymbol{P}_v$ and $\boldsymbol{P}_e$, so $\boldsymbol{v}(t) \sim N(\boldsymbol{O}, \boldsymbol{P}_v)$, $\boldsymbol{e}(t) \sim N(\boldsymbol{O}, \boldsymbol{P}_e)$.

We are looking for optimal control $\boldsymbol{u}^*(t)$ which minimizes the quadratic criterion

$$J = \mathcal{E}\left\{\boldsymbol{y}^T(N)\boldsymbol{S}_y\boldsymbol{y}(N) + \sum_{t=0}^{N-1} \boldsymbol{y}^T(t)\boldsymbol{Q}_y(t)\boldsymbol{y}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}_u(t)\boldsymbol{u}(t)\right\}$$

where $\mathcal{E}\{.\}$ is the mean. The positive semidefinite symmetric weighting matrices $\boldsymbol{S}_y$, $\boldsymbol{Q}_y(t)$ a $\boldsymbol{R}_u(t)$ are used as the tuning parameters for the optimal control design.

The optimal value of the criterion, denoted as J^*, depends on initial time $t = 0$, initial state $\boldsymbol{x}(0)$ and final time $t = N$, so $J^* = J_N^*(\boldsymbol{x}(0), 0)$. We are looking for causal optimal feedback control in the form $\boldsymbol{u}^*(t) = \boldsymbol{f}(\boldsymbol{x}(t), t)$. Due to the additive form of the criterion it is possible to express the criterion in a recursive way. Consider general initial time t and initial state $\boldsymbol{x}(t)$. Let us introduce optimal function $V(\boldsymbol{x}(t), t)$

$$J_N^*(\boldsymbol{x}(t), t) = V(\boldsymbol{x}(t), t)$$

Our original problem is embedded to the whole set of problems of optimal control with initial time t and initial state $\boldsymbol{x}(t)$ (final time $t = N$ is fixed). Consider that optimal function has the form

$$V(\boldsymbol{x}(t), t) = \boldsymbol{x}^T(t)\boldsymbol{G}(t)\boldsymbol{x}(t) + g(t)$$

Optimal function $V(\boldsymbol{x}(t), t)$ can be computed in recursive way according to the formula

$$V(\boldsymbol{x}(t), t) = \min_{\boldsymbol{u}(t)} \mathcal{E}\left\{\boldsymbol{y}^T(t)\boldsymbol{Q}_y\boldsymbol{y}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}_u\boldsymbol{u}(t) + V(\boldsymbol{x}(t+1), t+1)|\boldsymbol{x}(t)\right\}$$

which is the well known Bellman equation. Conditional mean is used in the previous formula to express the fact that by minimization with respect to $\boldsymbol{u}(t)$ the state $\boldsymbol{x}(t)$ is known. After the substitution for $\boldsymbol{y}(t)$ from the equation (2) Bellman equation has the form

$$\begin{aligned} V(\boldsymbol{x}(t), t) &= \min_{\boldsymbol{u}(t)} \mathcal{E}\left\{\boldsymbol{x}^T(t)\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}\boldsymbol{u}(t) + \right. \\ &\quad \boldsymbol{x}^T(t)\boldsymbol{Q}_u\boldsymbol{u}(t) + \boldsymbol{u}^T(t)\boldsymbol{Q}_u^T\boldsymbol{x}(t) + \boldsymbol{e}^T(t)(.) + \\ &\quad \left. (.)\boldsymbol{e}(t) + \boldsymbol{e}^T(t)\boldsymbol{Q}_y\boldsymbol{e}(t) + V(\boldsymbol{x}(t+1), t+1)|\boldsymbol{x}(t)\right\} \end{aligned}$$

where $\boldsymbol{Q} = \boldsymbol{C}^T\boldsymbol{Q}_y\boldsymbol{C}$, $\boldsymbol{R} = \boldsymbol{R}_u + \boldsymbol{D}^T\boldsymbol{Q}_y\boldsymbol{D}$ and $\boldsymbol{Q}_u = \boldsymbol{C}^T\boldsymbol{Q}_y\boldsymbol{D}$. Terms linear in $\boldsymbol{e}$ are not written in details, because their mean equals to zero. From the form of weighting matrices $\boldsymbol{Q}$, $\boldsymbol{Q}_u$ and $\boldsymbol{R}$ it is obvious, that their elements depend on system parameters.

After the substitution for state $\boldsymbol{x}(t+1)$ from (2) the optimal function equals

$$\begin{aligned} V(\boldsymbol{x}(t),t) &= \min_{\boldsymbol{u}(t)} \mathcal{E}\left\{\boldsymbol{x}^T(t)\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}\boldsymbol{u}(t) + \right. \\ & \boldsymbol{x}^T(t)\boldsymbol{Q}_u\boldsymbol{u}(t) + \boldsymbol{u}^T(t)\boldsymbol{Q}_u^T\boldsymbol{x}(t) + \boldsymbol{e}^T(t)\boldsymbol{Q}_y\boldsymbol{e}(t) + \\ & g(t+1) + (\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}\boldsymbol{v}(t))^T \boldsymbol{G}(t+1) \times \\ & \left. \times (\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}\boldsymbol{v}(t))\right\} \end{aligned} \tag{3}$$

Minimization of the previous formula is done by completing the squares

$$\begin{aligned} V(\boldsymbol{x}(t),t) &= \min_{\boldsymbol{u}(t)} \left\{(\boldsymbol{u}(t) - \boldsymbol{u}^*(t))^T \left[\mathcal{E}\{\boldsymbol{R}\} + \mathcal{E}\left\{\boldsymbol{B}^T\boldsymbol{G}(t+1)\boldsymbol{B}\right\}\right](\boldsymbol{u}(t) - \boldsymbol{u}^*(t)) + \right. \\ & \boldsymbol{x}^T(t)\left[\mathcal{E}\{\boldsymbol{Q}\} + \mathcal{E}\left\{\boldsymbol{A}^T\boldsymbol{G}(t+1)\boldsymbol{A}\right\}\right]\boldsymbol{x}(t) + \\ & \operatorname{tr}\left(\boldsymbol{Q}_y\boldsymbol{P}_e + \mathcal{E}\left\{\boldsymbol{E}^T\boldsymbol{G}(t+1)\boldsymbol{E}\right\}\boldsymbol{P}_v\right) + g(t+1) - \\ & \left. -\boldsymbol{u}^*(t)\left[\mathcal{E}\{\boldsymbol{R}\} + \mathcal{E}\left\{\boldsymbol{B}^T\boldsymbol{G}(t+1)\boldsymbol{B}\right\}\right]\boldsymbol{u}^*(t)\right\} \end{aligned}$$

From the previous form of optimal function it is obvious, that $\boldsymbol{u}^*(t)$ minimizes the relation and so it is optimal control. Comparing two previous formulae for optimal function the relation for optimal control is obtained

$$\begin{aligned} \boldsymbol{u}^*(t) &= -\left[\mathcal{E}\{\boldsymbol{R}\} + \mathcal{E}\left\{\boldsymbol{B}^T\boldsymbol{G}(t+1)\boldsymbol{B}\right\}\right]^{-1} \times \\ & \times \left[\mathcal{E}\left\{\boldsymbol{Q}_u^T\right\} + \mathcal{E}\left\{\boldsymbol{B}^T\boldsymbol{G}(t+1)\boldsymbol{A}_u\right\}\right]\boldsymbol{x}(t) \end{aligned} \tag{4}$$

The formula (4) for optimal control $\boldsymbol{u}^*(t)$ can be substituted to the formula for optimal function (3) and the recursive relations for matrix sequence $\boldsymbol{G}(t)$ and sequence $g(t)$ are obtained

$$\begin{aligned} \boldsymbol{G}(t) &= \mathcal{E}\{\boldsymbol{Q}\} + \mathcal{E}\left\{\boldsymbol{A}^T\boldsymbol{G}(t+1)\boldsymbol{A}\right\} - \\ & -\left[\mathcal{E}\{\boldsymbol{Q}_u\} + \mathcal{E}\left\{\boldsymbol{A}^T\boldsymbol{G}(t+1)\boldsymbol{B}\right\}\right] \times \\ & \times \left[\mathcal{E}\{\boldsymbol{R}\} + \mathcal{E}\left\{\boldsymbol{B}^T\boldsymbol{G}(t+1)\boldsymbol{B}\right\}\right]^{-1} \times \\ & \times \left[\mathcal{E}\left\{\boldsymbol{Q}_u^T\right\} + \mathcal{E}\left\{\boldsymbol{B}^T\boldsymbol{G}(t+1)\boldsymbol{A}\right\}\right] \end{aligned} \tag{5}$$

$$g(t) = \operatorname{tr}\left[\boldsymbol{Q}_y\boldsymbol{P}_e + \mathcal{E}\left\{\boldsymbol{E}^T\boldsymbol{G}(t+1)\boldsymbol{E}\right\}\boldsymbol{P}_v\right] + g(t+1) \tag{6}$$

Optimal function for the final time $t = N$ equals

$$V(\boldsymbol{x}(N),N) = \operatorname{tr}(\boldsymbol{S}_y\boldsymbol{P}_e),$$

and optimal control in time $t = N$ is

$$\boldsymbol{u}^*(N) = -\mathcal{E}\{\boldsymbol{S}_u\}^{-1}\mathcal{E}\left\{\boldsymbol{D}^T\boldsymbol{S}_y\boldsymbol{C}\right\}\boldsymbol{x}(N) \tag{7}$$

where $\boldsymbol{S}_u = \boldsymbol{D}^T \boldsymbol{S}_y \boldsymbol{D}$.

General formulae for cautious optimal control strategies are obtained. Uncertainty is considered to be in the system parameters only. State $\boldsymbol{x}(t)$ is supposed to be measurable. This form of results will be used later on for developing optimal control strategies of an ARX model. Certainty equivalent control strategy is obtained when only parameter mean in the previous relations is considered.

2.1 Parameter estimation of ARX model

The discrete time ARX model is described by the difference equation (1). Let us introduce parameter vector $\boldsymbol{\theta}(t)$ and vector of delayed inputs and outputs $\boldsymbol{z}(t)$

$$\begin{aligned} \boldsymbol{\theta}(t) &= \begin{bmatrix} b_0(t) & a_1(t) & b_1(t) & \dots & b_n(t) \end{bmatrix}^T \\ \boldsymbol{z}(t) &= \begin{bmatrix} u(t) & y(t-1) & \dots & u(t-n) \end{bmatrix}^T \end{aligned}$$

The ARX model (1) has then the form

$$y(t) = \boldsymbol{z}^T(t)\boldsymbol{\theta}(t) + e(t) \tag{8}$$

This equation can be considered to be the output equation of the state space model where parameter vector $\boldsymbol{\theta}(t)$ is considered to be the state of the system and the vector $\boldsymbol{z}^T(t)$ is an output matrix. Unknown parameters are supposed to be constant and so can be formally described by state equation

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) \tag{9}$$

States (in reality the parameters) of such system can be estimated by Kalman filter. Let $\widehat{\boldsymbol{\theta}}(t|\tau)$ equals to conditional mean of the parameter vector in time t conditioned by the data till time τ. In similar way is denoted conditional parameter covariance $\boldsymbol{P}_\theta(t|\tau)$.

Recursive Kalman filter consists from data and time update steps. Time update step is given by state equation (9), so $\widehat{\boldsymbol{\theta}}(t+1|t) = \widehat{\boldsymbol{\theta}}(t|t)$ and in similar way for the covariance matrix. It is possible to use only one time argument and so $\widehat{\boldsymbol{\theta}}(t)$ is mean of parameter vector (in time t or $t+1$) conditioned by data till time t.

Kalman filter for parameter estimation of the ARX model has only data update step which is usual formula for conditioning

$$\begin{aligned} \widehat{\boldsymbol{\theta}}(t) &= \widehat{\boldsymbol{\theta}}(t-1) + \boldsymbol{P}_{\theta y}\boldsymbol{P}_y^{-1}\left[y(t) - \hat{y}(t)\right] \\ \hat{y}(t) &= \boldsymbol{z}^T(t)\widehat{\boldsymbol{\theta}}(t-1) \\ \boldsymbol{P}_\theta(t) &= \boldsymbol{P}_\theta - \boldsymbol{P}_{\theta y}\boldsymbol{P}_y^{-1}\boldsymbol{P}_{y\theta} \end{aligned}$$

where covariances on the right side are in time $(t-1)$. Mutual covariance $\boldsymbol{P}_{\theta y}(t-1) = \boldsymbol{P}_\theta(t-1)\boldsymbol{z}(t)$ and output variance $\boldsymbol{P}_y(t-1) = \boldsymbol{z}^T(t)\boldsymbol{P}_\theta(t-1)\boldsymbol{z}(t) + \sigma_e^2$ follow from output equation (8).

Previous relations are recursive formulae for estimation of unknown parameters, where uncertainty is expressed by parameter mean and covariance. Recursive formulae start from prior or initial estimate $\widehat{\boldsymbol{\theta}}(0) = \widehat{\boldsymbol{\theta}}_0$ and $\boldsymbol{R}_\theta(0)$.

2.2 State space representation of ARX model

Let us consider the nonminimal realization of the ARX model in the form

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u(t) + \boldsymbol{E}e(t) \\ y(t) &= \boldsymbol{C}\boldsymbol{x}(t) + Du(t) + e(t) \end{aligned}$$

Here, the nonminimal state of the system is formed from the old values of input and output

$$\boldsymbol{x}(t) = \begin{bmatrix} y(t-1) & u(t-1) & \dots & u(t-n) \end{bmatrix}^T.$$

It is obvious that matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{E}$ equal

$$\boldsymbol{A} = \begin{bmatrix} a_1 & b_1 & \dots & b_{n-1} & a_n & b_n \\ 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \dots & & \dots & & & \\ 0 & 0 & \dots & 1 & 0 & 0 \end{bmatrix},$$

$$\begin{aligned} \boldsymbol{B} &= \begin{bmatrix} b_0 & 1 & 0 & 0 & \dots & 0 \end{bmatrix}^T, \\ \boldsymbol{E} &= \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \end{bmatrix}^T, \\ \boldsymbol{C} &= \begin{bmatrix} a_1 & b_1 & a_2 & b_2 & \dots & b_n \end{bmatrix} \\ D &= b_0 \end{aligned}$$

The unknown parameters exist only in the first row of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and also in matrix $\boldsymbol{C}$ and scalar D.

2.3 Cautious strategies of ARX model

From previous form of state space equation of ARX model it is obvious that state of the system is formed from delayed outputs and inputs. The state of such system is measurable and so it is not necessary to estimate it. Such situation is only possible in just described nonminimal realization of the ARX model.

Parameter vector $\boldsymbol{\theta}(t)$ equals $\boldsymbol{\theta}(t) = \begin{bmatrix} D & \boldsymbol{C} \end{bmatrix}^T$. Parameter estimation procedure described in the previous section results in parameter mean and covariance, so

$$\widehat{\boldsymbol{\theta}}(t) = \begin{bmatrix} \hat{b}_0(t) \\ \widehat{\boldsymbol{C}}^T(t) \end{bmatrix}, \qquad \boldsymbol{P}_\theta = \begin{bmatrix} \sigma^2_{b_0} & \boldsymbol{P}_{Cb_0} \\ \boldsymbol{P}_{b_0C} & \boldsymbol{P}_C \end{bmatrix}$$

Now the general results (4) and (7) for stochastic optimal control strategies are used. Means of the weighting matrices equal

$$\mathcal{E}\{\boldsymbol{Q}\} = Q_y\mathcal{E}\left\{\boldsymbol{C}^T\boldsymbol{C}\right\} = Q_y\left(\widehat{\boldsymbol{C}}^T\widehat{\boldsymbol{C}} + \boldsymbol{P}_C\right),$$

$$\begin{aligned}
\mathcal{E}\{\boldsymbol{Q}_u\} &= Q_y\mathcal{E}\left\{\boldsymbol{C}^T D\right\} = Q_y\left(\widehat{\boldsymbol{C}}^T\hat{b}_0 + \boldsymbol{P}_{b_0C}\right),\\
\mathcal{E}\{r\} &= r_u + Q_y\mathcal{E}\left\{D^T D\right\} = r_u + Q_y\left(\hat{b}_0^{\,2} + \sigma^2_{b_0}\right),\\
\mathcal{E}\{\boldsymbol{S}\} &= S_y\mathcal{E}\left\{\boldsymbol{C}^T\boldsymbol{C}\right\} = S_y\left(\widehat{\boldsymbol{C}}^T\widehat{\boldsymbol{C}} + \boldsymbol{P}_C\right),\\
\mathcal{E}\{S_u\} &= S_y\mathcal{E}\left\{D^T D\right\} = S_y\left(\hat{b}_0^{\,2} + \sigma^2_{b_0}\right),\\
\mathcal{E}\left\{D^T S_y\boldsymbol{C}\right\} &= S_y\mathcal{E}\{D\boldsymbol{C}\} = S_y\boldsymbol{P}_{Cb_0},
\end{aligned}$$

where r is used instead of weighting matrix $\boldsymbol{R}$ and r_u is used instead of $\boldsymbol{R}_u$ because of single input. Remaining products of matrices are computed in a similar way. Realize that unknown parameters are only in the first row of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. So

$$\begin{aligned}
\mathcal{E}\left\{\boldsymbol{B}^T\boldsymbol{G}(t+1)\boldsymbol{B}\right\} &= \mathcal{E}\left\{(b_0G_{11} + G_{21})\,b_0 + b_0G_{12} + G_{22}\right\} =\\
&\widehat{\boldsymbol{B}}^T\boldsymbol{G}(t+1)\widehat{\boldsymbol{B}} + G_{11}\sigma^2_{b_0} = G_{11}\left(\hat{b}_0^2 + \sigma^2_{b_0}\right) + 2\hat{b}_0G_{12} + G_{22},
\end{aligned}$$

where G_{11}, G_{12} and G_{22} are elements of the matrix $\boldsymbol{G}(t+1)$. And

$$\begin{aligned}
\mathcal{E}\left\{\boldsymbol{A}^T\boldsymbol{G}(t+1)\boldsymbol{B}\right\} &= \widehat{\boldsymbol{A}}^T\boldsymbol{G}(t+1)\widehat{\boldsymbol{B}} + G_{11}\boldsymbol{P}_{b_0C},\\
\mathcal{E}\left\{\boldsymbol{A}^T\boldsymbol{G}(t+1)\boldsymbol{A}\right\} &= \widehat{\boldsymbol{A}}^T\boldsymbol{G}(t+1)\widehat{\boldsymbol{A}} + G_{11}\boldsymbol{P}_C.
\end{aligned}$$

Final results of **quadratic optimal cautious control strategy** of an ARX model are:

- Optimal control according to (7) and (4) equals to

$$\begin{aligned}
\boldsymbol{u}^*(N) &= -\frac{1}{\hat{b}_0^2 + \sigma^2_{b_0}}\left(\widehat{\boldsymbol{C}}\hat{b}_0 + \boldsymbol{P}_{Cb_0}\right)\boldsymbol{x}(N),\\
\boldsymbol{u}^*(t) &= -\frac{\boldsymbol{z}^T(t)}{\alpha}\boldsymbol{x}(t),
\end{aligned}\tag{10}$$

  for time $t = 0, 1, \ldots, N-1$, where the vector $\boldsymbol{z}(t)$ equals

$$\boldsymbol{z}^T(t) = Q_y\left(\widehat{\boldsymbol{C}}\hat{b}_0 + \boldsymbol{P}_{Cb_0}\right) + \widehat{\boldsymbol{B}}^T\boldsymbol{G}(t+1)\widehat{\boldsymbol{A}} + \boldsymbol{P}_{Cb_0}G_{11}(t+1)$$

  and the constant $\alpha = r_u + Q_y\left(\hat{b}_0^{\,2} + \sigma^2_{b_0}\right) + G_{11}\left(\hat{b}_0^2 + \sigma^2_{b_0}\right) + 2\hat{b}_0G_{12} + G_{22}$.

- Matrix $\boldsymbol{G}(t)$ can be computed by recursive formula (5) which after the substitution has the form

$$\boldsymbol{G}(t) = Q_y\left(\widehat{\boldsymbol{C}}^T\widehat{\boldsymbol{C}} + \boldsymbol{P}_C\right) + \widehat{\boldsymbol{A}}^T\boldsymbol{G}(t+1)\widehat{\boldsymbol{A}} + G_{11}\boldsymbol{P}_C + \frac{\boldsymbol{z}(t)\boldsymbol{z}^T(t)}{\alpha}$$

  with end condition $\boldsymbol{G}(N) = \boldsymbol{O}$.

- The sequence $g(t)$ follows from the recursive relation (6), which after the substitution has the form
$$g(t) = g(t+1) + \sigma_e^2\left(1 + G_{11}(t+1)\right)$$
with end condition $g(N) = S_y\sigma_e^2$.

- Minimum of the criterion equals to
$$J^* = J_N^*(\boldsymbol{x}(0), 0) = \boldsymbol{x}^T(0)\boldsymbol{G}(0)\boldsymbol{x}(0) + g(0) \tag{11}$$

Certainty equivalent control strategy is obtained when only parameter mean values are considered. Here are the final results of certainty equivalent control:

- Optimal certainty equivalent control according to (7) and (4) equals[1]
$$\begin{aligned}\boldsymbol{u}^*(N) &= -\frac{\widehat{\boldsymbol{C}}}{\hat{b}_0}\boldsymbol{x}(N),\\ \boldsymbol{u}^*(t) &= -\frac{\boldsymbol{w}^T(t)}{\beta}\boldsymbol{x}(t),\end{aligned}$$
for time $t = 0, 1, \ldots, N-1$, where the vector $\boldsymbol{w}(t)$ equals
$$\boldsymbol{w}^T(t) = Q_y\widehat{\boldsymbol{C}}\hat{b}_0 + \widehat{\boldsymbol{B}}^T\boldsymbol{G}(t+1)\widehat{\boldsymbol{A}}$$
and the constant $\beta = r_u + Q_y\hat{b_0}^2 + G_{11}\hat{b}_0^2 + 2\hat{b}_0G_{12} + G_{22}$.

- Matrix $\boldsymbol{G}(t)$ for certainty equivalent strategy can be computed by the recursive formula (5) which after the substitution has the form
$$\boldsymbol{G}(t) = Q_y\widehat{\boldsymbol{C}}^T\widehat{\boldsymbol{C}} + \widehat{\boldsymbol{A}}^T\boldsymbol{G}(t+1)\widehat{\boldsymbol{A}} + \frac{\boldsymbol{w}(t)\boldsymbol{w}^T(t)}{\beta}$$
with end condition $\boldsymbol{G}(N) = \boldsymbol{O}$.

- The sequence $g(t)$ equals zero for certainty equivalent strategy.

- Minimum of the criterion is in (11) but the matrix $\boldsymbol{G}(t)$ has other form for certainty equivalent control.

[1]Note that for $\hat{b}_0 = 0$ (system with time delay) $u^*(N) = 0$ or any arbitrary input may be chosen because $u(N)$ does not influence the system in N-th step and subsequently the optimal control.

3 Simulation results

The concept of stochastic optimal control is quite abstract. That is why we felt useful and lucid to carry out some computer simulations to demonstrate how do the stochastic optimal control operate. Only the most interesting results are presented, for others see [8].

From previous section we can see that stochastic optimal control is a very complex task. To keep all simulations easy to understand and to explain we made in comparison with previous theoretical results following simplifications.

- Parameters uncertainty was modelled only with diagonal elements of matrix $\boldsymbol{P}_\Theta$.
- Quadratic control criterion

$$J = \mathcal{E}\left\{ y^T(N)\boldsymbol{S}y(N) + \sum_{t=0}^{N-1} y^T(t)\boldsymbol{Q}y(t) + ru^2(t) \right\} \tag{12}$$

  with tuning parameters r (control weight), $\boldsymbol{Q}$ (output weight) and $\boldsymbol{S}$ (final output weight) was taken into account.

- ARX model of the form

$$y(t) = \sum_{i=1}^{n} a_i y(t-i) + \sum_{i=0}^{n} b_i u(t-i) + e(t) \tag{13}$$

  was considered and measurement noise $e(t) \sim \mathcal{N}(0, \sigma_e^2)$ was supposed to be known. We feel necessary to notice that this is the main principal disadvantage of proposed approach. Considering a known model of the system we simplified very much the derivation of stochastic optimal control, but it is left to reader to find out a way how to estimate the variance σ_e^2 in each application.

In the remaining text unstable system

$$P(s) = \frac{1}{(10s-1)(10s+1)^2} \tag{14}$$

will be studied.[2] The choice of an unstable system was motivated by the fact, that we wanted to see how the stochastic optimal control treats the unstability.

In this point it is worthy to stress two important facts. Firstly the control was applied on finite control horizon, therefore the notion of stability and unstability is senseless. Secondly, even if the control law (10) is deterministic, it is applied to a stochastic process. Hence each realization is unique. This is why the interpretation of obtained results is very complicated and their generalization almost impossible. However we hope that following simulations will demonstrate the main features of stochastic optimal control.

[2]In [8] a stable system $P(s) = \frac{1}{(10s+1)^3}$ was studied as well to check whether the derived strategies behave correctly. Some of there obtained results will be mentioned in this text. We started from a continuous system to enable an easy comparison with previously obtained results.

3.1 Parameters setting

Considered unstable plant (14) was discretized with sampling frequency $T_s = 5$, which gives rise to the discrete model

$$P(d) = \frac{-0.0146d^3 - 0.0672d^2 - 0.0187d}{-0.6065d^3 + 2.3679d^2 - 2.8618d + 1},$$

where $d = z^{-1}$ is the delay operator.

To make clear the simulations procedure, parameters setting will be listed. Let us mention that a different choice of parameters set would lead to slightly different results. Having this in mind, we choose a parameters set, among many others, which demonstrates the studied properties of stochastic optimal control the best.

- Control horizon N equals to 20 time steps (sampling period $T_s = 5$). This time interval is long enough to observe the unstability of considered system.
- Unknown noise $e(t)$ in ARX model description (13) was supposed to be white, normally distributed $\mathcal{N}(0, \sigma_e^2)$ with standard deviation $\sigma_e = 0.01$. Let us mention that considering $\sigma_e \gg 0.01$ would lead to an *unstable*[3] behaviour of controlled system.
- Initial state $\boldsymbol{x}_0^T = \begin{bmatrix} -10 & -10 & -10 \end{bmatrix}$ was chosen.
- Parameter uncertainty was modelled by diagonal matrix $\boldsymbol{P}_\Theta$

$$\boldsymbol{P}_\theta = \begin{bmatrix} \sigma_b^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_a^2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \sigma_a^2 & 0 \\ 0 & 0 & \cdots & 0 & \sigma_b^2 \end{bmatrix}.$$

  The choice of different diagonal covariance elements permits to study the influence of uncertainty in parameters description on nominator (zeros) and denominator (poles) of the system separately. Precise values of standard deviations σ_a, σ_b will be given for each figure.

Tuning parameters r (control weight), Q (output weight) and S (final output weight) and auxiliary parameters will be specified in each paragraph.

3.2 Dependence on parameters uncertainty

The main contribution of this text consists in considering parameters uncertainty in system description. In comparison with classical LQG control this enables to anticipate

[3]Because stochastic optimal control strategies were derived on finite horizon, there is no reason to talk about stability and instability. The term of ***stability*** is used here only to mark a behaviour when for $t \to N$, $y(t) \to 0$. In this manner will be the notion of stability used in remaining text.

parameters change and deal with it. That is why in this paragraph we want to demonstrate how do the parameters uncertainty influence the behaviour of optimally controlled system. The increase of parameters uncertainty is modelled with the increase of standard deviations σ_a or/and σ_b in covariance matrix $\boldsymbol{P}_\Theta$.

In this section tuning parameters $\boldsymbol{Q} = 1$ (output weight), $\boldsymbol{S} = 1$ (final output weight) and $r = 1$ (control weight) are considered. Increasing parameters uncertainty was modelled with standard deviations $\sigma_a = \sigma_b = \begin{bmatrix} 0 & 0.01 & 0.02 & 0.033 \end{bmatrix}$. Other parameters are the same as in section (3.1)

Figures 1-3 show the influence of parameters uncertainty on optimal control $u(t)$ and output course $y(t)$. The difference between cautious and certainty equivalent control strategy can be seen. The bigger parameters uncertainty is considered (dotted, dash-dotted and dashed line), the smaller is the optimal control effect. This corresponds to our expectation and experience, that in presence of uncertainty our decisions are more restrained.

Figure 1 represents an exception. All optimal trajectories are the same. This is due to a very small value of noise σ_e. When bigger value of noise were considered, control strategy behaved in the same manner as in figures 2-3, but we wanted to keep all figures comparable.

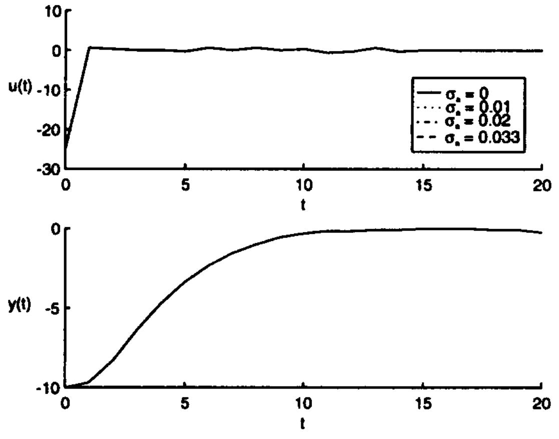


Figure 1: Optimal control $u(t)$ and output $y(t)$. Different line types represent amount of parameters a_i uncertainty (standard deviation σ_a).

Figure 4 shows the dependence of optimal value of the criterion $\mathcal{J}$ on the amount of parameters uncertainty. An interesting feature of cautious control can be observed. The optimal value $\mathcal{J}^*$ of the criterion decreases with parameters uncertainty. This is due to the fact that a smaller control effect corresponds to more cautious control. Consequently the contribution of the term $u^2(t)$ to $\mathcal{J}^*$ is smaller.

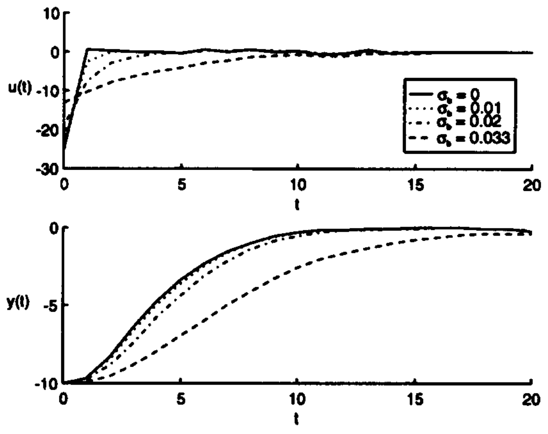


Figure 2: Optimal control $u(t)$ and output $y(t)$. Different line types represent amount of parameters b_i uncertainty (standard deviation σ_b).

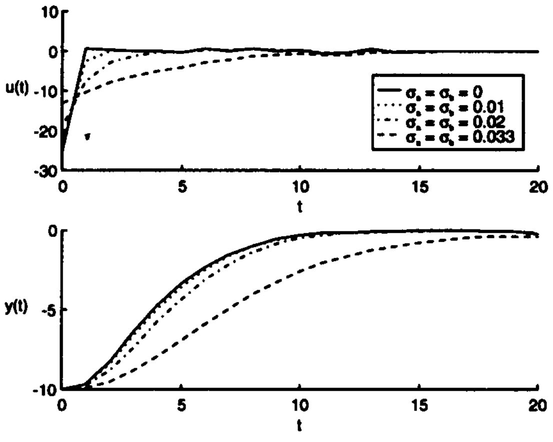


Figure 3: Optimal control $u(t)$ and output $y(t)$. Different line types represent amount of parameters a_i, b_i uncertainty (standard deviations $\sigma_a = \sigma_b$).

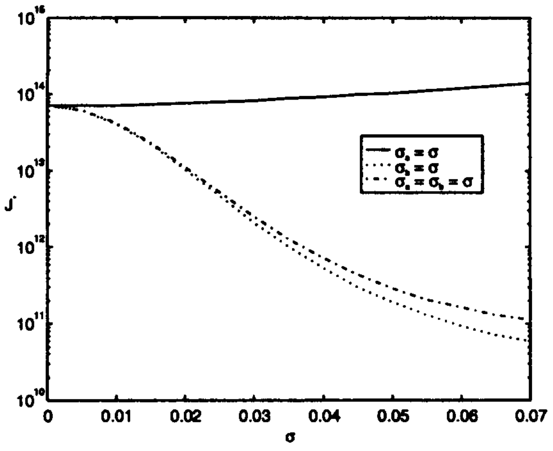


Figure 4: Criterion dependence on parameters uncertainty. Solid line corresponds to uncertainty only in a_i parameters ($\sigma_b = 0$), dotted line represents uncertainty in only b_i parameters ($\sigma_a = 0$) and the dash-dotted line in both a_i, b_i parameters.

3.3 Dependence on tuning parameters

The advantage of classical quadratic optimal control is that except of stabilizing the system it permits to influence the expended energy $u(t)$ and permits to control the settling course of $y(t)$ according to a chosen criterion. In this section we demonstrate that our approach maintains this property when parameters uncertainty is taken into account.

Chosen tuning parameters are given for each figure with increasing importance of output settling in the criterion: $r = 1$, $\boldsymbol{Q} = \begin{bmatrix} 0.01 & 0.1 & 1 & 10 & 100 \end{bmatrix}$. Uncertainty in both a_i and b_i is modelled in the covariance matrix $\boldsymbol{P}_\Theta$ with standard deviations $\sigma_a = \sigma_b = \begin{bmatrix} 0 & 0.01 & 0.02 & 0.033 \end{bmatrix}$ and represented with different line types.

Figures 5-6 show the influence of tuning parameters on optimal control law $u(t)$ and output course $y(t)$. It can be seen that expected behaviour was achieved. When $\boldsymbol{Q}$ increases, the output $y(t)$ even in presence of parameters uncertainty tends to 0 faster.

In all figures the typical difference between cautious and certainty equivalent control strategy is maintained. The bigger parameters uncertainty is considered, the smaller control effect is needed.[4]

Notice that in some cases the control strategy does not ensure the *stability* of the system. This confirms our supposition, that there should be a limit in parameters uncertainty, for which the system can be stabilized. Bigger parameters uncertainty will lead to an *unstable* behaviour even if optimal control is applied. Notice, however, that such a control still minimizes the criterion $\mathcal{J}$.

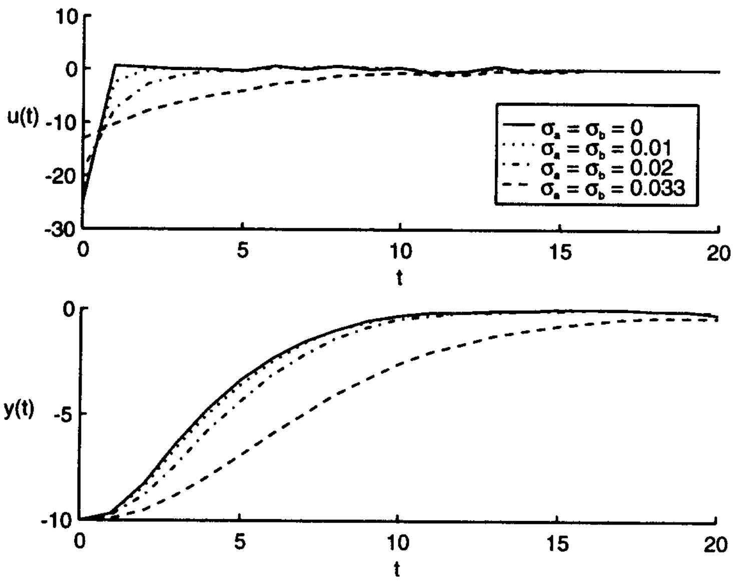
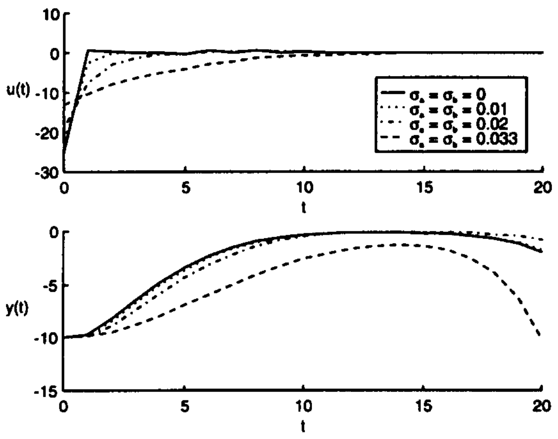
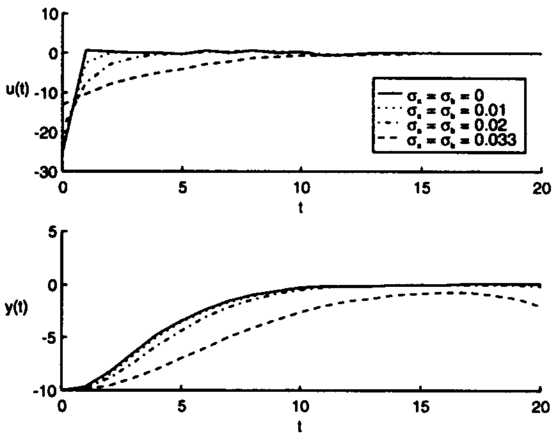


Figure 5: Optimal control $u(t)$ and output $y(t)$ for $Q = S = 0.01$ and $r = 1$.

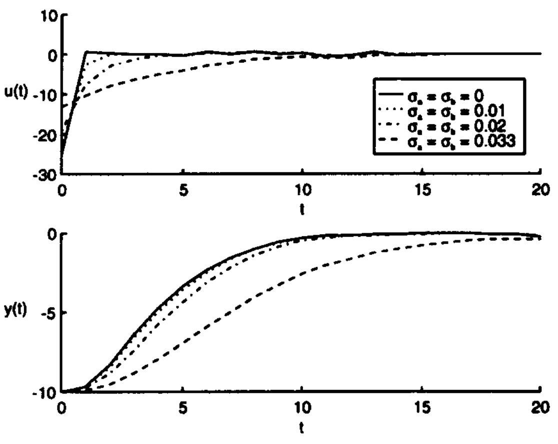
[4]Let us mention that this result was not achieved for stable system studied in [8].



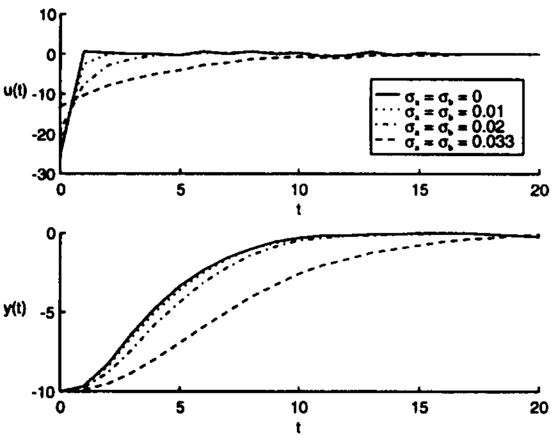
(a) $Q = S = 0.1,\ r = 1$



(b) $Q = S = 1,\ r = 1$



(c) $Q = S = 10,\ r = 1$



(d) $Q = S = 100,\ r = 1$

Figure 6: Optimal control $u(t)$ and output $y(t)$ for different tuning parameters.

Figure 7 shows the dependence of optimal value of the criterion $\mathcal{J}$ on tuning parameters $\boldsymbol{Q}$, $\boldsymbol{S}$, r. We can see that increase of $\boldsymbol{Q}$ (in comparison with r) leads to an increase of the optimal value of criterion $\mathcal{J}$ in a linear manner for all considered parameters uncertainty. Because from figures 5-6 we can see that good settling of $y(t)$ is achieved, we can conclude that more constrained control expends more energy.

Nevertheless an interesting fact has to be mentioned. The optimum of the criterion $\mathcal{J}^*$ is smaller when parameters are less certain. Thus, uncertainty in parameters make the control "easier". This might be explained by the fact that for increasing parameters uncertainty smaller control $u(t)$ is applied. Because the term $u^2(t)$ is summed in the criterion, less effective control leads to a smaller increase of the criterion.

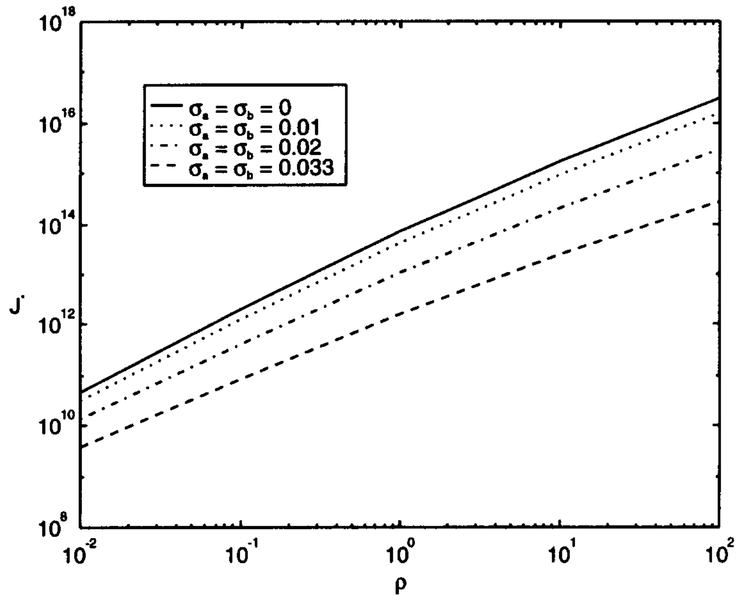


Figure 7: Dependence of the optimal value of the criterion $\mathcal{J}$ on tuning parameters Q, S, r $(\rho = \frac{Q}{r},\ S = Q)$.

3.4 Different optimal trajectories

As it was mentioned previously an important property of stochastic optimal control is that even if the control law is deterministic, it is applied to a stochastic process. So due to stochastic nature of the system, each realization of optimal control has to be different. But a reasonable control should in each case lead to a satisfactory result ($y(t)$ tends to 0 for $t \to N$).

To see the stochastic behaviour of the optimally controlled stochastic system, we carried out 100 simulation, each time with a different realization of random sequence $e(1), \cdots, e(N)$. Optimal control $u(t)$, trajectory and corresponding output course $y(t)$ were plotted for each simulation. Tuning parameters $\boldsymbol{Q}$, $\boldsymbol{S}$, r were assigned to $\boldsymbol{Q} = \boldsymbol{S} = r = 1$. Parameters uncertainty (standard deviations $\sigma_a = \sigma_b$) is given for each figure separately.

Figures 8-9 show that a satisfactory result was achieved for all considered parameters uncertainty. Due to stochastic properties of studied system, all studied trajectories $u(t)$ and $y(t)$ spread into a bounded band but remain *stable* in all cases. Notice that in figure 8 even for a CE control (deterministic) the resulting output course is stochastic.

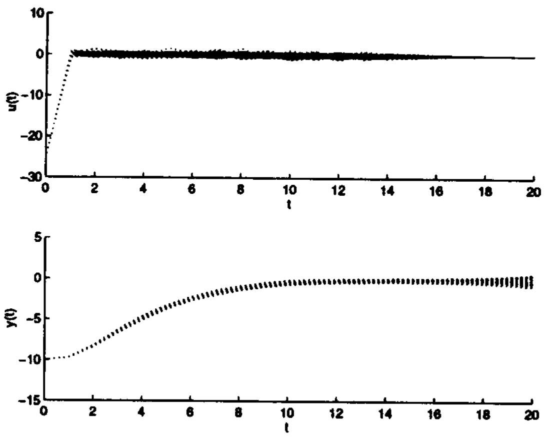
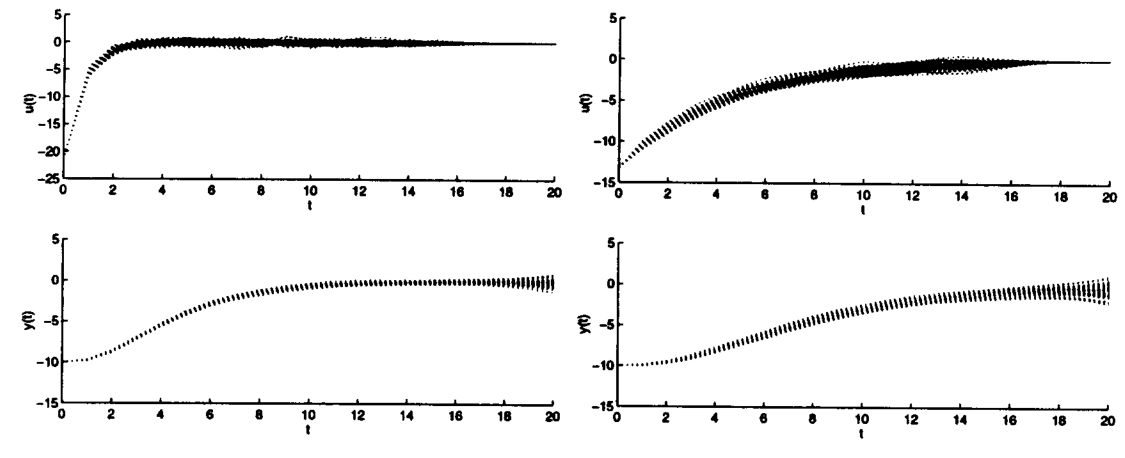


Figure 8: 100 courses of control strategy $u(t)$ and corresponding output $y(t)$ for different noise sequences $e(1) \cdots e(20)$ and parameters uncertainty $\sigma_a = \sigma_b = 0$.

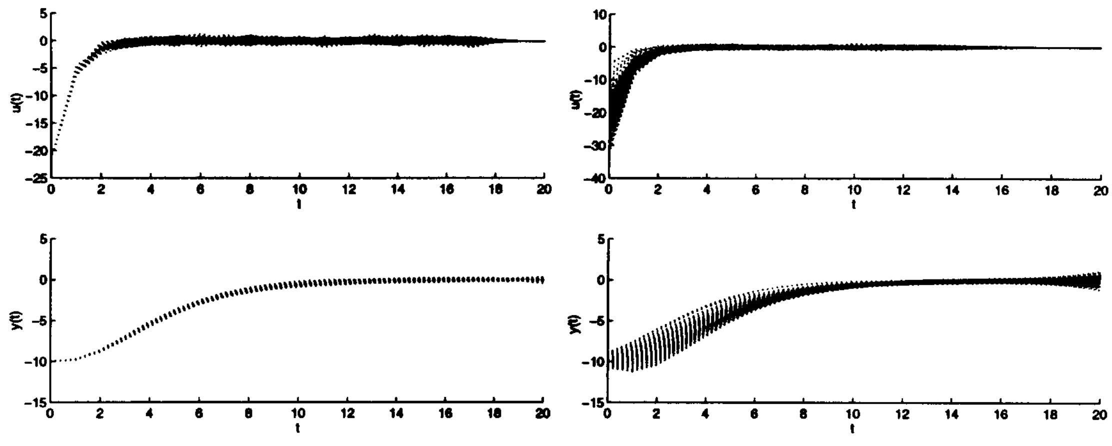


(a) $\sigma_a = \sigma_b = 0.015$ (b) $\sigma_a = \sigma_b = 0.033$

Figure 9: 100 courses of control strategy $u(t)$ and corresponding output $y(t)$ for different noise sequences $e(1) \cdots e(20)$ and parameters uncertainty.

Figure 10(a) shows that the same result as in figures 8-9 was achieved for different tuning parameters $\boldsymbol{Q} = \boldsymbol{S} = 100$, $r = 1$. Notice that the change in $\boldsymbol{Q}$ (output weight) led to a narrower band $y(t)$ in comparison with fig. 9(a).

Figure 10(b) shows that reasonable behaviour of stochastic optimal control is maintained when it is applied to a processes with random initial condition (real stochastic system) $x(0) = \hat{x}(0) + e_x$, where e_x is normally distributed white noise with zero mean and variance σ_x^2. The only difference in comparison with figure 9(a) is that the band $y(t)$ widened, especially for $t \approx 0$, but the behaviour remained *stable*.



(a) $Q = S = 100, r = 1,\ \sigma_a = \sigma_b = 0.015$ (b) $x(0) \sim \mathcal{N}(\hat{x}(0), \sigma_x^2),\ \sigma_x = 0.1,\ \sigma_a = \sigma_b = 0.015$

Figure 10: 100 courses of control strategy $u(t)$ and corresponding output $y(t)$ for different noise sequences $e(1) \cdots e(20)$: influence of tuning parameters (a) and random initial condition (b).

4 Conclusion

For ARX stochastic model it is possible to estimate its unknown parameters by Kalman filter. Conditional mean and covariance of parameters is obtained in recursive way from observed input and output data. Stochastic optimal control strategies of an ARX model are developed in the text. Cautious optimal control strategy respects parameter uncertainty represented with parameter covariance matrix. Certainty equivalent control strategy is based only on estimated parameters mean and neglect its uncertainty. Program in Matlab was realized to simulate the differences between both strategies and different simulation results were presented and discussed.

References

[1] D. P. Bertsekas. *Dynamic Programming.* Academic Press, New York, 1994.

[2] Vladimír Havlena. Simultaneous parameter tracking and state estimation in a linear system. *Automatica*, 29(4):1041–1052, 1993.

[3] F. L. Lewis. *Optimal Estimation.* J. Wiley, New York, 1986.

[4] Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw Hill, New York, 1965.

[5] V. Peterka. *Trends and Progress in System Identification*, chapter Bayesian Approach to System Identification. Pergamon Press, Oxford, 1981.

[6] Jan Štecha. Control, estimation, smoothing and system classification. Technical report, CTU FEL Prague, 1992.

[7] Jan Štecha and Rudolf Rázek. Cautious and certainty equivalent control strategies of an ARX model. *16th IASTED Int. Conference on Modelling, Identification and Control*, pages 384–387, 1997.

[8] Jan Štecha and Rudolf Rázek. Stochastic optimal control of ARX model. Technical report, CTU FEL Prague, 1997.

Implementation of a Real-Time Adaptive Controller for a SCARA Robot Based-on Digital Signal Processors

S. H. Han[1], Peter Kopacek[2] and M. H. Lee[3]

1 Dept. of Mechanical Engineering, Kyungnam Univ., Masan, Korea
2 Institute for Handling Devices and Robotics, Technical Univ., Viena, Austria
3 Dept. of Mechanical Engineering, Pusan National Univ., Pusan, Korea

Abstract

Real-time implementation of an adaptive controller for the robotic manipulator is presented in this paper. Digital signal processors are used in implementing real time adaptive control algorithms to provide an enhanced motion for robotic manipulators. In the proposed scheme, adaptation laws are derived from the improved Lyapunov second stability analysis based on the model reference adaptive control theory. The adaptive controller consists of an adaptive feedforward controller and feedback controller and time-varying auxiliary control elements to the nominal operating point. The proposed control scheme is simple in structure, fast in computation, and suitable for real-time control. Moreover, this scheme does not require any accurate dynamic modeling, nor values of manipulator parameters and payload. Performance of the adaptive controller is illustrated by simulation and experimental results for a SCARA robot.

Keywords *Adaptive Controller, Assembling Robotic Manipulator, Digital Signal Processor, Real-Time, Robust Control.*

1 Introduction

In general, industrial robotic manipulators consist of independent joint controllers which control joint angles separately through simple position servo loops (Ortega and Spong, 1989). This basic control system enables a manipulator to perform simple positioning tasks such as in the pick-and-place operation. However, joint controllers are severely limited in precise tracking of fast trajectories and in sustaining desirable dynamic performance for variations of payload and parameter uncertainties. Design of a high performance controller for robotic manipulators has been an active topic of research(Ortega and Spong, 1989; Tomei, 1989).

Today there are many advanced techniques that are suitable for servo control of a large class of nonlinear systems including robotic manipulators(Sadegh and Horowitz, 1990; Bortoff, 1994; Slotine and Li, 1987). Since the pioneering work of Dubowsky and DesForges (Sadegh and Horowitz, 1990), interest in adaptive control of robot manipulators has been growing steadily (Ortega and Spong, 1989; Tomei, 1989; Sadegh and Horowitz, 1990; Bortoff, 1994). This growth is largely due to the fact that adaptive control theory is particularly well-suited to robotic manipulators whose dynamic model is highly complex and may contain unknown parameters. However, implementation of these algorithms generally involves intensive numerical computations.

Digital signal processors(DSP's) are special purpose microprocessors that are particularly powerful for intensive numerical computations involving sums and products of variables (Ahmed, 1991). Digital version of most advanced control algorithms can be defined as sums and products of measured variables, thus can naturally be implemented by DSP's. In addition, DSP's are as fast in computation as most 32-bit microprocessors and yet at a fraction of their prices(Bortoff, 1994; Ahmed, 1991). These features make them a viable computational tool for digital implementation of advanced controllers.

In order to develop a digital servo controller one must carefully consider the effect of the sample and hold operation, the sampling frequency, the computational delay, and that of the quantization error on the stability of a closed-loop system(Ahmed, 1991; Parks, 1986; Dubowsky and DesForges, 1979; Choi, *et al*, 1986; Gavel and Hsia, 1987). Moreover, one must also consider the effect of disturbances on the transient variation of the tracking error as well as its steady-state value.

This paper presents a new approach to the design of an adaptive control system using DSP's, TMS320C30, for robotic manipulators to achieve trajectory tracking by the joint angles. This paper is organized as follows: in Section 2, the dynamic modeling of robotic manipulator and the adaptive control algorithm are derived. Adaptation laws are derived based on the model reference adaptive control theory using the improved Lyapunov second method. Section 3 represents simulation and experimental results is obtained for a SCARA robot. Finally, Section 4 discusses findings and draws some conclusions.

2 Adaptive Control Scheme

2.1 Dynamic Modeling

Let us consider a nonredundant-joint robotic manipulator in which the n×1 joint torque vector τ(t) is related to the n×1 joint angle vector q(t) by the following nonlinear dynamic equation of motion:

$$D(q)\ddot{q}+N(q,\dot{q})+G(q)=\tau(t) \tag{1}$$

where $D(q)$ is the n×n symmetric positive-definite inertia matrix, $N(q,\dot{q})$ is the n×1 Coriolis and centrifugal torque vector, and $G(q)$ is the n×1 gravitational load vector.

Equation (1) describes the manipulator dynamics without any payload. In order to consider payload on the manipulator dynamics, suppose that the manipulator end-effector is firmly grasping a payload represented by point mass *m*. For the payload to move with acceleration $\ddot{X}(t)$ in the gravity field, the end-effector must apply the n×1 force vector $T(t)$ given by

$$T(t) = \Delta P[\ddot{X}(t)+g] \tag{2}$$

where g is the n×1 gravitational acceleration vector and $X(t)$ is the n×1 end-effector position and orientation coordinates in a fixed task related Cartesian frame of reference.

The end-effector requires additional joint torque

$$\tau_f(t) = J(q)^T T(t) \tag{3}$$

where $J(q) = [\partial\lambda(q)/\partial q]$ is the n×n Jacobian matrix of the manipulator. Hence, the total joint torque vector can be obtained by combining equations (1) and (3) as

$$\tau(t) = J(q)^T T(t) + D(q)\ddot{q} + N(q, \dot{q}) + G(q). \tag{4}$$

Substituting equations (2) and (3) into equation (4) yields

$$\Delta P\, J(q)^T [J(q)\,\ddot{q} + \dot{J}(q, \dot{q})\dot{q} + g] + D(q)\,\ddot{q} + N(q, \dot{q}) + G(q) = \tau(t). \tag{5}$$

Equation (5) shows the explicit effect of payload m on the manipulator dynamics. This equation (5) can be defined as the implicit nonlinear dynamic equation

$$D^*(m, q, \dot{q})\ddot{q} + N^*(m, q, \dot{q})\dot{q} + G^*(m, q, \dot{q}) = \tau(t). \tag{6}$$

2.2 Adaptation Law

In order to cope with changes in operating point, gains are varied with the change in external working condition. This yields the adaptive control law

$$\begin{aligned} \tau(t) = &\ [\,P_A(t)\,\ddot{q}_r(t) + P_B(t)\,\dot{q}_r(t) + P_C(t)q_r(t)\,] \\ &+ [P_P(t)E(t) + P_V(t)\,\dot{E}(t) + P_I(t)] \end{aligned} \tag{7}$$

where $P_A(t)$, $P_B(t)$, $P_C(t)$ are feed forward time-varying adaptive gains, and $P_P(t)$ and $P_V(t)$ are feedback adaptive gains. And $P_I(t)$ denotes the time-varying control signal corresponding to the nominal operating point term, generated by a feedback controller driven by position backing error $E(t)$.

Fig. 1 represents the block diagram of adaptive control system for robotic manipulator.

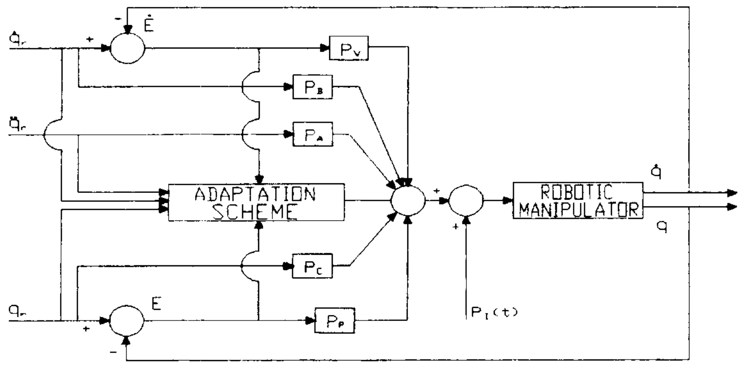


Fig. 1 : The block diagram of adaptive control scheme for robotic manipulator.

On applying adaptive control law (7) to the nonlinear robot dynamic equation (6), the error differential equation can be obtained as

$$\begin{aligned} &D^*\ddot{E}(t) + (N^* + P_V)\,\dot{E}(t) + (G^* + P_P)E(t) \\ &= P_I(t) + (D^* - P_A)\,\ddot{q}_r(t) + (N^* - P_B)\,\dot{q}_r(t) + (G^* - P_C)q_r(t). \end{aligned} \tag{8}$$

Defining the 2n×1 position-velocity error vector $\varepsilon(t) = [E(t), \dot{E}(t)]^T$, equation (8) can be written in the statespace form

$$\dot{\varepsilon}(t) = \begin{pmatrix} 0 & I_n \\ w_1 & w_2 \end{pmatrix} \varepsilon(t) + \begin{pmatrix} 0 \\ w_3 \end{pmatrix} q_r(t) + \begin{pmatrix} 0 \\ w_4 \end{pmatrix} \dot{q}_r(t)$$

$$+ \begin{pmatrix} 0 \\ w_5 \end{pmatrix} \ddot{q}_r(t) + \begin{pmatrix} 0 \\ w_6 \end{pmatrix} \qquad (9)$$

where $w_1 = [D^*]^{-1} [G^* + P_P]$ $\quad w_2 = [D^*]^{-1} [N^* + P_V]$

$w_3 = [D^*]^{-1} [G^* - P_C]$ $\quad w_4 = [D^*]^{-1} [N^* - P_B]$

$w_5 = [D^*]^{-1} [D^* - P_A]$ $\quad w_6 = -[D^*]^{-1} [P_I]$.

The adaptation laws are now derived by ensuring the stability of error dynamics. To this end, let us define a scalar positive definite Lyapunov function as

$$L = \varepsilon^T R \varepsilon + \mathit{trace}[\,Q_1^T K_1 Q_1] + \mathit{trace}[\,Q_2^T K_2 Q_2]$$
$$+ \mathit{trace}[\,Q_3^T K_3 Q_3] + \mathit{trace}[\,Q_4^T K_4 Q_4] \qquad (10)$$
$$+ \mathit{trace}[\,Q_5^T K_5 Q_5] + [Q_6 K_6 Q_6]^T$$

where $Q_1 = w_1 - S_1 - w_1^*$, $Q_2 = w_2 - S_2 - w_2^*$, $Q_3 = w_3 - w_3^*$, $Q_4 = w_4 - w_4^*$, $Q_5 = w_5 - w_5^*$, and $Q_6 = w_6 - w_6^*$. R is the solution of the Lyapunov equation for the reference model, $K_1, \ldots, K_6$ are arbitrary symmetric positive definite constant n×n matrices, and matrices $w_1^*, \ldots, w_6^*$ are functions of time which will be specified later.

From the stability analysis, required adaptive controller gains are obtained as

$$P_P(t) = p_1 [\,P_{p1}E + P_{p2}\dot{E}\,][\,E\,]^T + p_2 \int_0^t [\,P_{p1}E + P_{p2}\dot{E}\,][\,E\,]^T dt \qquad \text{(11-a)}$$

$$P_V(t) = v_1 [\,P_{v1}E + P_{v2}\dot{E}\,][\,\dot{E}\,]^T + v_2 \int_0^t [\,P_{v1}E + P_{v2}\dot{E}\,][\,\dot{E}\,]^T dt \qquad \text{(11-b)}$$

$$P_C(t) = c_1 [\,P_{c1}E + P_{c2}\dot{E}\,][\,q_r\,]^T + c_2 \int_0^t [\,P_{c1}E + P_{c2}\dot{E}\,][\,q_r\,]^T dt \qquad \text{(11-c)}$$

$$P_B(t) = b_1 [\,P_{b1}E + P_{b2}\dot{E}\,][\,\dot{q}_r\,]^T + b_2 \int_0^t [\,P_{b1}E + P_{b2}\dot{E}\,][\,\dot{q}_r\,]^T dt \qquad \text{(11-d)}$$

$$P_A(t) = a_1 [\,P_{a1}E + P_{a2}\dot{E}\,][\,\ddot{q}_r\,]^T + a_2 \int_0^t [\,P_{a1}E + P_{a2}\dot{E}\,][\,\ddot{q}_r\,]^T dt \qquad \text{(11-e)}$$

$$P_I(t) = \lambda_2 [\,P_{i2}E\,] + \lambda_1 \int_0^t [\,P_{i1}E\,]^T dt \qquad \text{(11-f)}$$

where [λ_1, p_{p1}, p_{v1}, p_{c1}, p_{b1}, p_{a1}]and [λ_2, p_{p2}, p_{v2}, p_{c2}, p_{b2}, p_{a2}] are positive and nonnegative scalar adaptation gains.

3 Experiment and Results

This section represents DSP's-based control results of the position and velocity control for a SCARA robot with four joint as shown in Fig. 2, and discusses the advantages of using DSP's for robotic motion control.

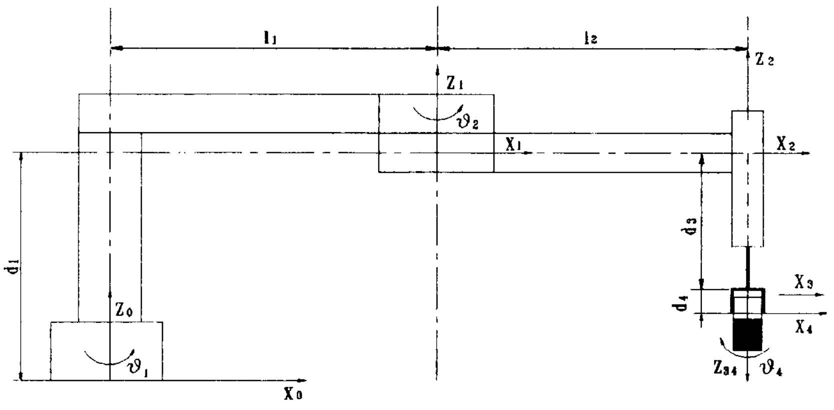


Fig. 2 : Link coordinate systems of a SCARA robot.

A set of experiments for the proposed adaptive controller was performed for four joints of the SCARA robot. To implement the proposed adaptive controller, we used our own TMS320C30 assembler software developed.

Fig. 3 shows the experimental set-up equipment. Also, a TMS320C30 emulator was used in experimental set-up as can be seen form Fig. 3. The TMS320C3x emulator is an application development tool which is based on the TI's TMS320C30 floating point DSP chip with an instruction cycle time 50ns. At each joint, a harmonic drive was used to transfer power from the motor, which has a resolver attached to its shaft for sensing angular velocity with a resolution of 8096 pulses/rev.

Fig. 4 represents the main hardware structure of control system of a SCARA robot.

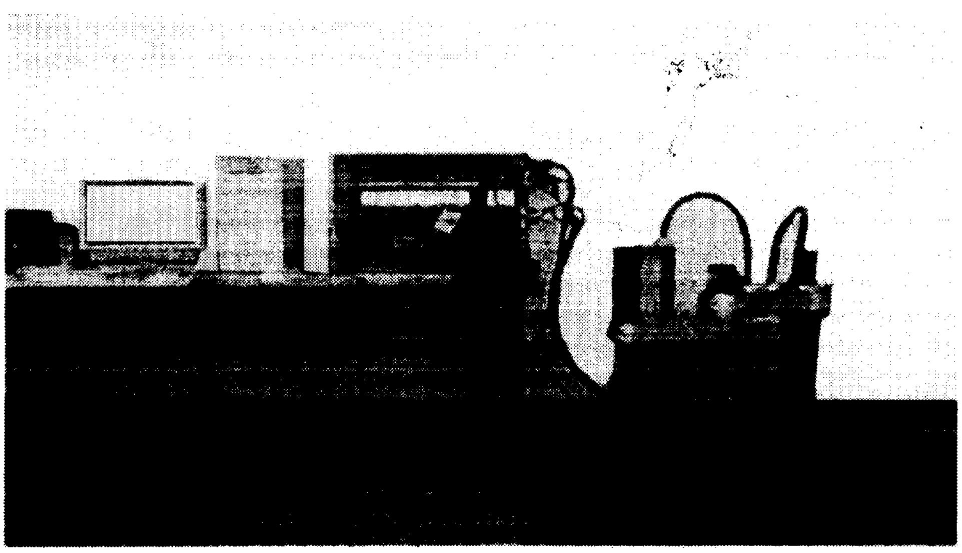


Fig. 3 : Experimental set-up.

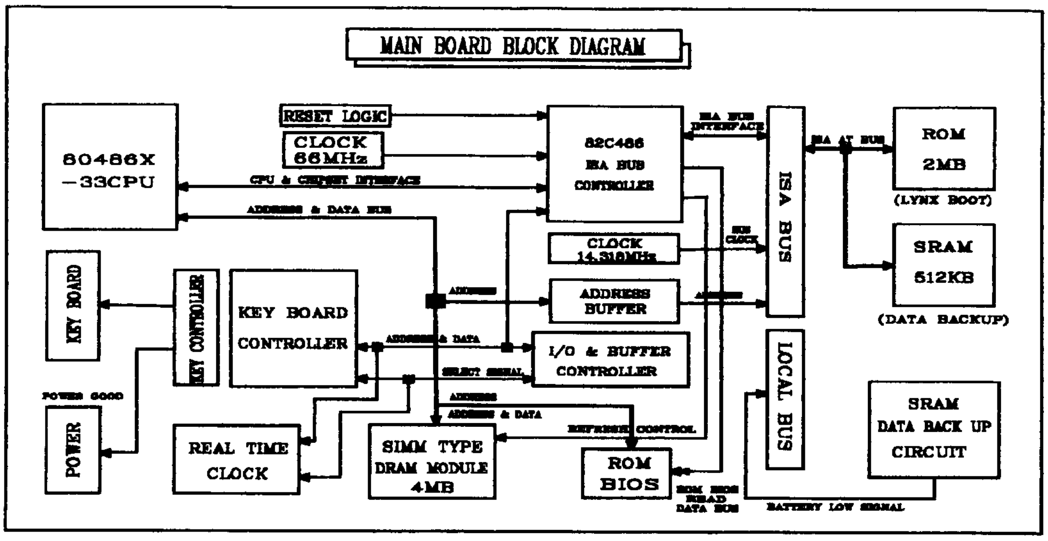


Fig. 4 : The block diagram of hardware structure of SCARA robot.

The performance evaluation of the proposed adaptive controller was performed in the joint space and cartesian space.

In the joint space, the experiment was carried out to evaluate the position and velocity control performance of the four joints for variation of payloads. Fig. 5 shows the results of the position and velocity tracking control for the first joint with 3.5 kg payload. In the joint space, as can be seen from results, the DSP-based adaptive controller shows extremely good tracking performance even with the added external disturbance. Fig. 6 shows the experimental results of the position and velocity tracking performance for the second joint with 3.5 kg payload in the joint space.

From experiment results, proposed adaptive controller shows very good control performance in the test for trajectory tracking of the velocity and position in the joint space.

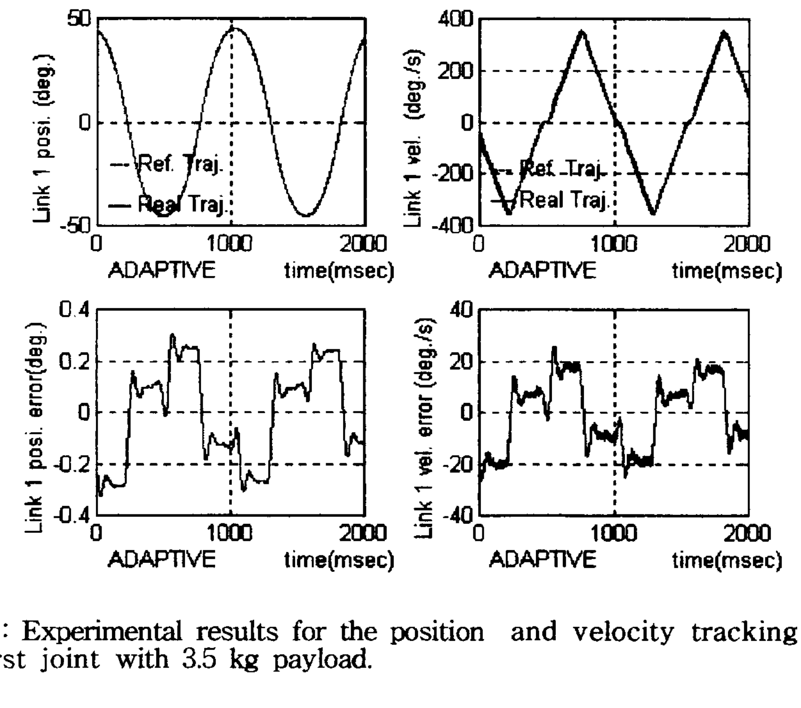


Fig. 5 : Experimental results for the position and velocity tracking at the first joint with 3.5 kg payload.

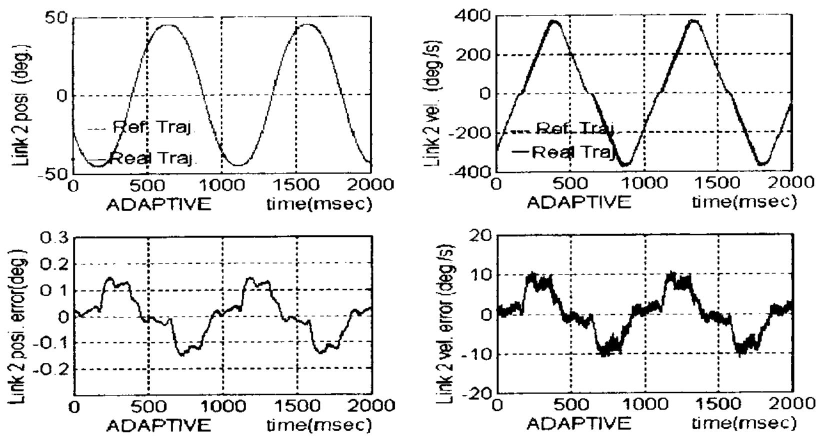


Fig. 6 : Experimental results for the position and velocity tracking at the second joint with 3.5 kg payload.

In the cartesian space, the adaptive controller was evaluated in a peg-in-hole task, and in a tracking task of B shaped reference trajectory.

Fig. 7 represents the B shaped reference trajectory in the cartesian space. Fig. 8 represents the experimental results of adaptive controller for the B shaped reference trajectory with 3.5 kg payload and maximum velocity (2.2 m/s) in the cartesian space. Fig. 9 shows the experimental results of PID controller for the B shaped reference trajectory with 3.5 kg payload. Fig. 10 represents the kinematic configuration of peg-in-hole task in the cartesian space. In Fig. 10, each length of linkl and link2 is 350㎜ and 260㎜ respectively.

Table 1 represents the experimental results for the peg-in-hole tasks with 3.5 kg payload and maximum velocity(2.2 m/s) during 8 hours running time in the cartesian space.

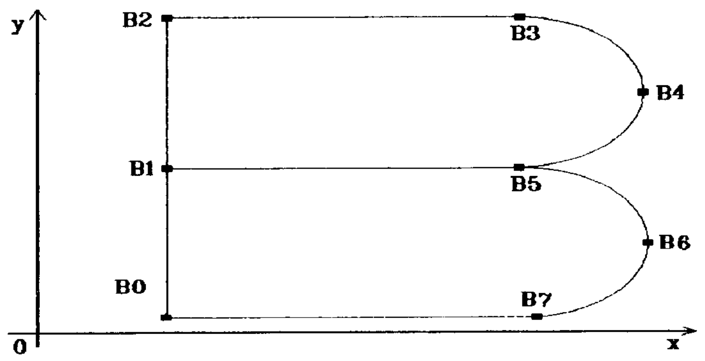


Fig. 7 : The B shaped reference trajectory in the cartesian space.

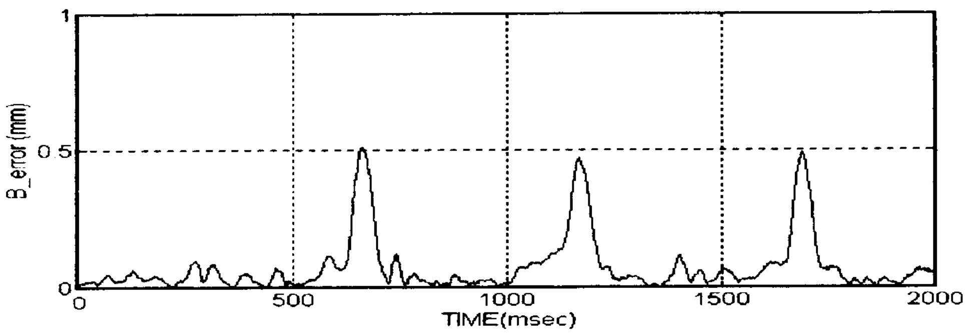


Fig. 8 : Experimental result of the adaptive controller for tracking of B reference trajectory with 3.5 kg payload.

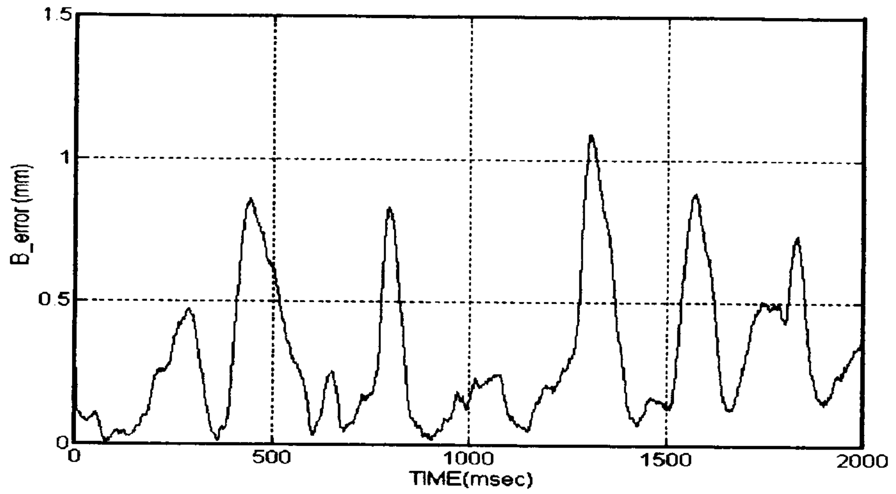


Fig. 9 : Experimental result of PID controller for tracking of B shaped reference trajectory with 3.5 kg payload

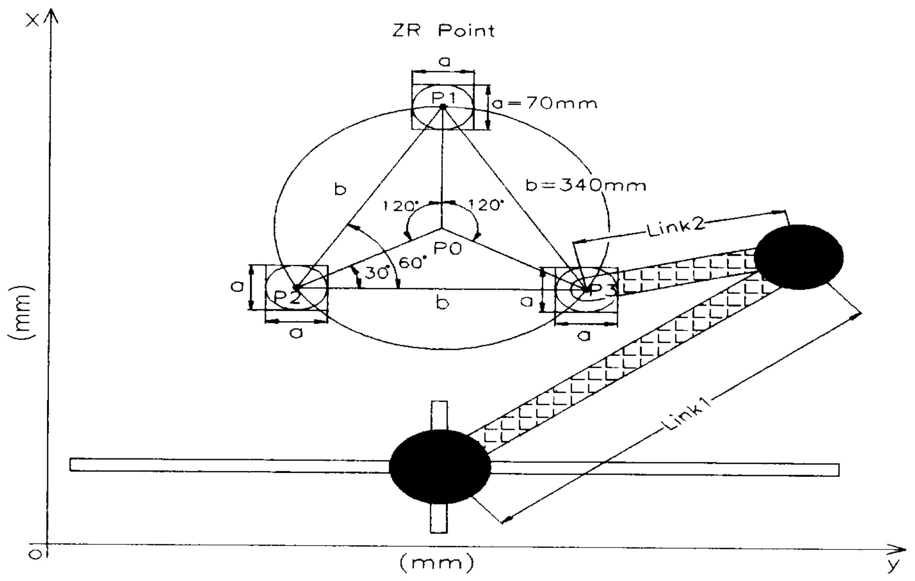


Fig. 10 : Kinematic configuration for peg-in-hole task in the cartesian space.

Table 1 : Comparision of the failure rate between the adaptive controller and PID controller in the peg-in-hole task.

Task speed	80.00 (%)	100 (%)
Failure (%) of adaptive controller	0.008 (%)	0.012 (%)
Failure (%) of PID controller	0.015 (%)	0.038 (%)

As can be seen from the experimental results (Table 1), the adaptive controller shows the better control performance and reliability than the existing PID controller in the higher speed.

4 Concusions

A new adaptive digital control scheme is described in this paper using the TMS320C30 chips for robotic manipulators. The adaptation laws are derived from the model reference adaptive theory using the improved direct Lyapunov method. The simulation and experimental results show that the proposed DSPs-adaptive controller is robust to the payload variation, inertia parameter uncertainty, and change of reference trajectory. This adaptive controller has been found to be suitable to the real-time control of robot system. A novel feature of the proposed scheme is the utilization of an adaptive feedforward controller, an adaptive feedback controller, and a PI type time-varying control signal to the nominal operating point which results in improved tracking performance.

Another attractive feature of this control scheme is that, to generate the control action, it neither requires a complex mathematical model of the manipulator dynamics nor any knowledge of the manipulator parameters and payload.

Control scheme uses only the information contained in the actual and reference trajectories which are directly available. Futhermore, the adaptation laws generate the controller gains by means of simple arithmetic operations. Hence, the calculation control action is extremely simple and fast. These features are suitable for implementation of on-line real time control for robotic manipulators with a high sampling rate, particularly when all physical parameters of the manipulator cannot be measured accurately and the mass of the payload can vary substantially.

References

Gavel, D.; Hsia, T.C., 1987, Decentralized Adaptive Control of Robot Manipulator, In Proceedings of the 1987 IEEE Conference on Robotics and Automation, Raleigh, NC.

Ahmed, I., 1991, Digital Control Applications with the TMS320 Family, Selected Application Notes, Texas Instruments Inc.

Slotine, J. J. E.; Li, W., 1987, Adaptive Manipulator Control - A Case Study, Proc. IEEE Conf. Robotics and Automation, pp. 1392-1400.

Sadegh, N.; Horowitz. R., 1990, An Exponentially Stable Adaptive Control Law for Robot Manipulators, IEEE Trans. Robotics and Automation.

Parks, P.C.V., 1986, Lyapunov Redesign of Model Reference Adaptive Control System, IEEE Trans. Auto. Contr. 11/3, pp. 362-267.

Tomei, P., 1991, Adaptive PD Controller for Robot Manipulators, IEEE Trans. Robotics and Automation.

Ortega, R.; Spong, M. W., 1989, Adaptive Motion Control of Rigid Robots: A Tutorial, Automatica. 25, pp. 877-888.

Bortoff, S. A., 1994, Advanced Nonlinear Robotic Control Using Digital Signal Processing, IEEE Trans. Indust. Elect.. 41, No. 1.

Dubowsky, S.; DesForges, D.T., 1979, The Application of Model Reference Adaptative Control to the Robot Manipulators, ASME J. Dyn. Syst., Meas., Contr. 101, pp. 193-200.

Choi, Y.K.; Chang, M.J.; Bien. Z., 1986, An Adaptive Control Scheme for Robot Manipulators, IEEE Trans. Auto. Contr. 44/4, pp. 1185-1191.

The Three Mass Model of Harmonic Transmission

Milan Balara, Dušan Balara, Alexander Balara, Ivan Gots
Technical University, Košice, Faculty of Mechanical Engineering, Institute of Automation and Robotics, Department of Automation Technics, Prešov, Slovakia

Abstract

The paper describes the dynamic properties of the harmonic transmission as a linear and non-linear three mass system. The paper contains the principle of operation, the equations and the block schema of harmonic transmission and describes simulated time responses of the torque and speed on the output shaft of harmonic transmission. Described harmonic transmission has damped oscillations of output torque and output speed.

1 Introduction

Harmonic transmission is used very often in the servosystems of robots, manipulators, special devices etc. Solution and function of harmonic transmission is described often (for example refer to Hudoba, M., 1989). Harmonic transmission employs three concentric components to produce high mechanical advantage and speed reduction. Basic components of harmonic transmission are pictured in Fig. 1.

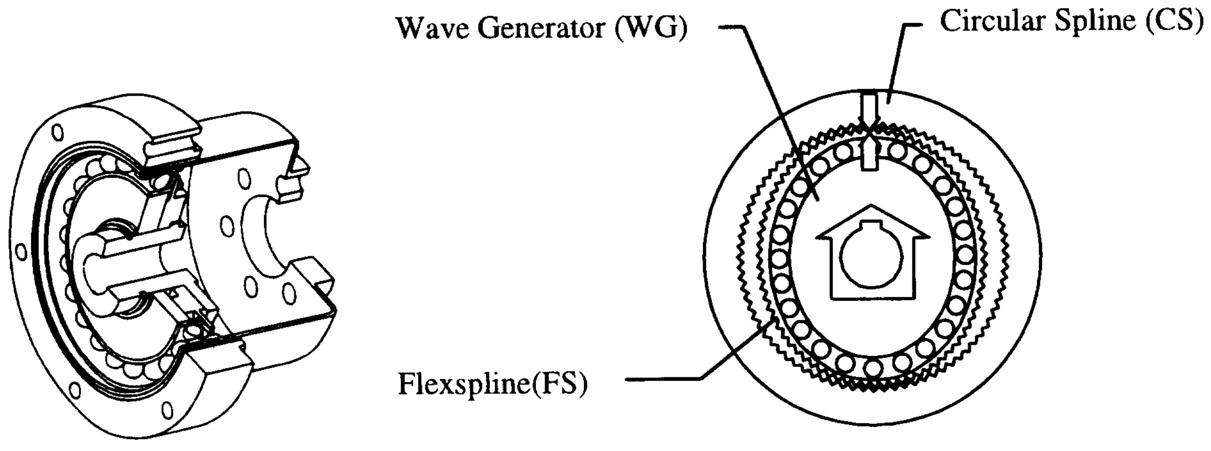


Fig. 1 Basic components of harmonic transmission

The Circular Spline (CS) is a rigid ring with internal teeth, engaging the teeth of the Flexspline accross the major axis of the Wave Generator. The Flexspline (FS) is a nonrigid, thin cylindrical cup with external teeth on a slightly smaller pitch diameter than the Circular

Spline, resulting in it having two fewer teeth on its outer circumference. It fits over and is held in an elliptical shape by the Wave Generator (WG). The Wave Generator is a thin raced ball bearing fitted onto an elliptical plug serving as a high efficiency torque converter.

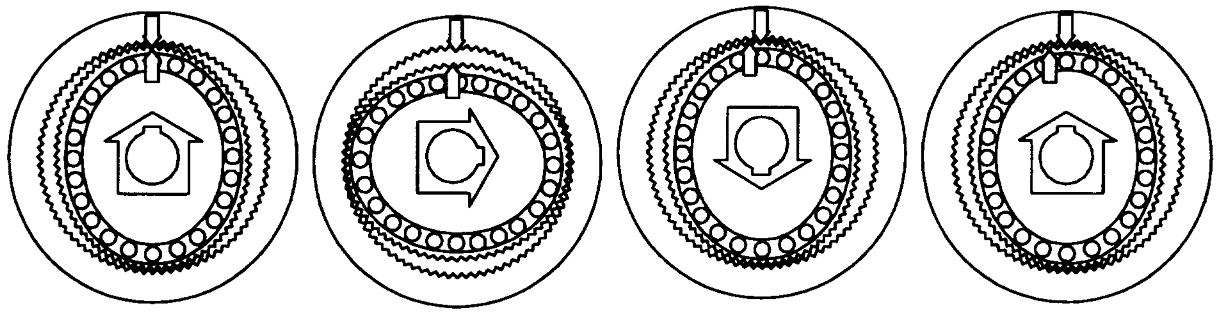


Fig. 2 The principle of operation of harmonic transmission

As soon as the Wave Generator starts to rotate clockwise, the zone of tooth engagement travels with the major elliptical axis.

When the Wave Generator has turned through 180 degrees clockwise the Flexspline has regressed by one tooth relative to the Circular Spline (see Fig. 2).

Each full turn of the Wave Generator thus causes relative motion between the Flexspline and Circular Spline equal to two teeth (Harmonic Drive Applications Handbook, 1992).

The dynamic properties of this type of reducer are not very simple. The elastic element of reducer with the masses of its some parts is the resource of unusual properties of harmonic transmission. The linear two mass mathematical model of harmonic transmission is noted from 1982 (Balara, D., 1982) together with computer simulations (Balara, M., 1987). The paper describes the dynamic properties of the harmonic transmission as a linear and non-linear three mass system (Balara, A.,1994). The paper contains the equations, block schema and describes the time characteristics of torque and speed on the output shaft of harmonic transmission as the results of the computer simulations. Described harmonic transmission has damped oscillations of output torque and output speed.

2 The Linear Three Mass Mathematical Model of Harmonic Transmission

The harmonic transmission is a non-linear three mass system with elastic coupling. The description of n-mass system with elastic coupling is possible by using dynamic equations. The linear mathematical model of harmonic transmission is derived upon next suppositions:

- we consider that the first mass J_1 is summation of inertia of servomotor's anchor, shaft and inertia of some input parts of the harmonic transmission. Those parts are connected by short shaft and we may consider them as a one mass, i. e.:

$$J_1 = J_m + J_{INP} \tag{1.1}$$

- we consider that the second mass J$_2$ is wave generator which is connected by elastic coupling and backlash with first mass (wave generator is connected with input part of harmonic transmission by connection which features backlash),
- we consider that the third mass J$_3$ is inertia of load (working mechanism) behind the harmonic transmission. It is elasticly connected to the J$_2$ mass by the flexspline, with backlash feature (see Fig. 3).

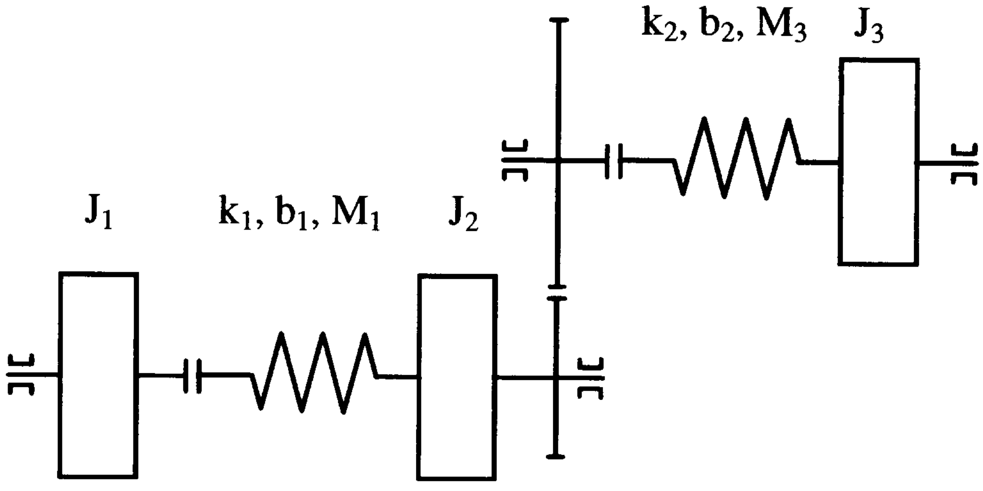


Fig. 3. Kinematic schema of harmonic transmission

$$J_1 \frac{d\omega_1}{dt} = M_1 - M_{t1} - M_2 \tag{1.2}$$

$$J_2 \frac{d\omega_2}{dt} = M_2 - M_{t2} - \frac{M_3}{i} \tag{1.3}$$

$$J_3 \frac{d\omega_3}{dt} = M_3 - M_{t3} \tag{1.4}$$

where $$M_2 = k_1(\phi_1 - \phi_2) + b_1(\omega_1 - \omega_2) \tag{1.5}$$

$$M_3 = k_2(\phi'_2 - \phi_3) + b_2(\omega'_2 - \omega_3) \tag{1.6}$$

$$M_{t1} = p_1.\omega_1 \tag{1.7}$$

$$M_{t2} = p_2.\omega_2 \tag{1.8}$$

$$M_{t3} = p_3.\omega_3 \tag{1.9}$$

$$\omega'_2 = \frac{\omega_2}{i} \tag{1.10}$$

$$\phi'_2 = \frac{\phi_2}{i} \tag{1.11}$$

where

M_1	- torque of servomotor, (input torque)
M_2	- torque of elastic connection between mass 1 and 2 (J_1 and J_2)
M_3	- torque of elastic connection between mass 2 and 3 (J_2 and J_3)
M_{t1}	- friction torque of servomotor's anchor and input shaft of harmonic transmission
M_{t2}	- friction torque of mechanisms inside the transmission
M_{t3}	- friction torque of mechanisms in output part of the harmonic transmission
i	- gear ratio
J_m	- inertia on the servomotor's shaft
J_{INP}	- inertia of the input part of harmonic transmission
J_1	- inertia which is summation of inertia of servomotor's anchor and inertia of input part of the harmonic transmission
J_2	- inertia of the wave generator
J_3	- inertia on the output part of harmonic transmission
ϕ_1	- angle of rotation of input shaft
ϕ_2	- angle of rotation of wave generator
ϕ'_2	- angle of rotation of wave generator reduced by gear ratio i
ϕ_3	- angle of rotation of output shaft
ω_1	- angle speed servomotor's anchor and input shaft
ω_2	- angle speed of wave generator
ω'_2	- angle speed of wave generator reduced by gear ratio i
ω_3	- angle speed of output shaft of harmonic transmission
k_1	- torsion stiffness coefficient of input shaft of harmonic transmission
k_2	- torsion stiffness coefficient of elastic cog-wheel
b_1	- dissipative damping coefficient on the input
b_2	- dissipative damping coefficient on the output

a_1	- friction coefficient on the input, (Coulomb friction)
a_2	- friction coefficient inside harmonic transmission's mechanism, (Coulomb friction)
a_3	- friction coefficient on the output, (Coulomb friction)
p_1	- viscous sliding friction coefficient on the input of harmonic transmission
p_2	- viscous sliding friction coefficient inside harmonic transmission
p_3	- viscous sliding friction coefficient on the output of harmonic transmission
α_1	- backlash on the input of harmonic transmission
α_2	- backlash on the output of harmonic transmission

3 The Non-linear Three Mass Mathematical Model of Harmonic Transmission

Non-linear model of harmonic transmission is created by using of two types of nonlinearities, which are represented by backlash and friction. The backlash in the kinematic connections is created by non-linear dependence of torque in elastic connections (see Fig. 4). Area of backlash is marked as an α_n. If $|\phi_1 - \phi_2| \leq \alpha_1/2$, and $|\phi'_2 - \phi_3| \leq \alpha_2/2$, then the shaft torques of elastic couples are equal to zero. Connections between mechanisms disappear. Damping coefficient and stiffness coefficient are equal to zero. Movement equations are:

For $|\phi_1 - \phi_2| \leq \alpha_1/2$ and $|\phi'_2 - \phi_3| \leq \alpha_2/2$ is $M_2 = M_3 = 0$

$$J_1 \frac{d\omega_1}{dt} = M_1 - M_{t1} \tag{2.1}$$

$$J_2 \frac{d\omega_2}{dt} = -M_{t2} \tag{2.2}$$

$$J_3 \frac{d\omega_3}{dt} = -M_{t3} \tag{2.3}$$

For $|\phi_1 - \phi_2| > \alpha_1/2$ and $|\phi'_2 - \phi_3| \leq \alpha_2/2$ is $M_3 = 0$

$$J_1 \frac{d\omega_1}{dt} = M_1 - M_{t1} - M_2 \tag{2.4}$$

$$J_2 \frac{d\omega_2}{dt} = M_2 - M_{t2} \tag{2.5}$$

$$J_3 \frac{d\omega_3}{dt} = -M_{t3} \tag{2.6}$$

For $|\phi_1 - \phi_2| \leq \alpha_1/2$ and $|\phi'_2 - \phi_3| > \alpha_2/2$ is $M_2 = 0$

$$J_1 \frac{d\omega_1}{dt} = M_1 - M_{t1} \tag{2.7}$$

$$J_2 \frac{d\omega_2}{dt} = -M_{t2} - \frac{M_3}{i} \tag{2.8}$$

$$J_3 \frac{d\omega_3}{dt} = M_3 - M_{t3} \tag{2.9}$$

For $|\phi_1 - \phi_2| > \alpha_1/2$ and $|\phi'_2 - \phi_3| > \alpha_2/2$ are valid equations for linear model (1.2) - (1.6), (1.10) and (1.11).

The terms for friction torques are as follows:

$$M_{t1} = p_1.\omega_1 + a_1.sign\omega_1 \tag{2.10}$$

$$M_{t2} = p_2.\omega_2 + a_2.sign\omega_2 \tag{2.11}$$

$$M_{t3} = p_3.\omega_3 + a_3.sign\omega_3 \tag{2.12}$$

4 Block Schema and Responses of The Harmonic Transmission

Block schema of non-linear model of the harmonic transmission is in Fig. 4. This schema is created as a system, which contains three masses, two elastic couples, two backlashes and a friction.

These assumptions are the base for solving the automatic control systems, featuring harmonic transmission. Simulated time responses of output angle speed and output torque of harmonic transmission (HP 60, ZTS Zvolen, Slovakia) are in Fig. 5. Responses are results of torque impulse M_1 on the input shaft of the harmonic transmission.

The parameters of the simulated harmonic transmission are as follows:

$J_{INP} = 4{,}4.10^{-5}$ kgm^2
$J_1 = 4{,}78.10^{-5}$ kgm^2
$J_2 = 1{,}3894.10^{-4}$ kgm^2
$J_3 = 20\ J_m.i^2 = 1{,}1684$ kgm^2
$i = 124$
$k_1 = 250\ 000$ Nm/rad
$k_2 = 33\ 000$ Nm/rad
$\alpha_1 = 23{,}58.10^{-4}$ rad
$\alpha_2 = 2{,}62.10^{-4}$ rad
$p_1 = 0{,}01$ Nms/rad
$p_2 = p_3 = 0{,}01$ Nms/rad
$a_1 = a_2 = a_3 = 0.00001$ Nm/rad
$b_1 = 16$ Nms/rad
$b_2 = 25$ Nms/rad

5 Conclusion

Described harmonic transmission has damped oscillations of output torque and output speed. It is the result of flexibility, backlash and friction within the harmonic transmission. This structure has not been excited by inputs with torrential time changes.

The running of the input torque has to be fluent. The designer of the servosystems should take into consideration special properties of this transmission and should to create the convenient control systems (for example refer to Balara, M., 1992; Hori, Y., Iseki, H., Sugiura, K., 1994).

Application of harmonic transmissions in servosystems requires the knowledge of dynamic properties of harmonic transmission and its behaviour, which are introduced in this paper.

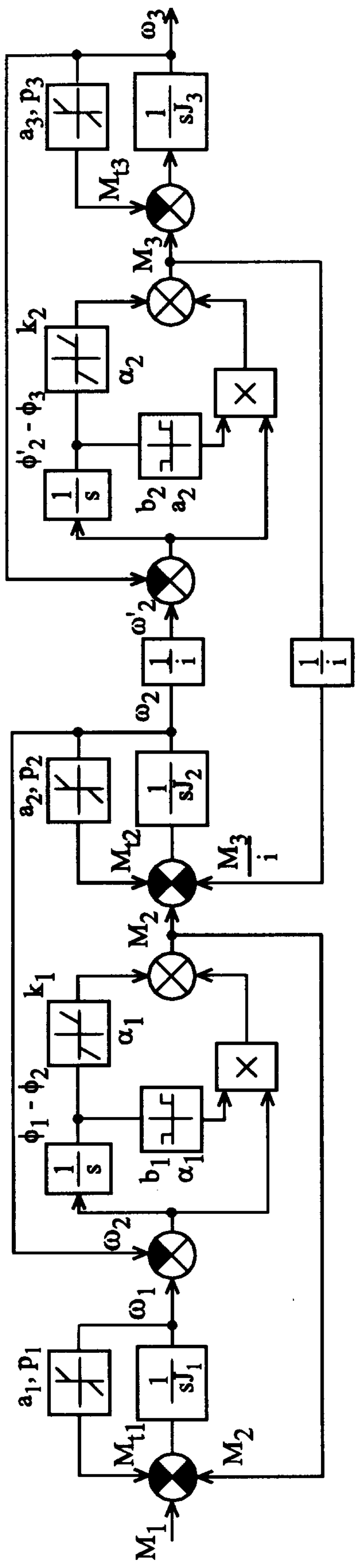


Fig. 4. Block schema of the harmonic transmission

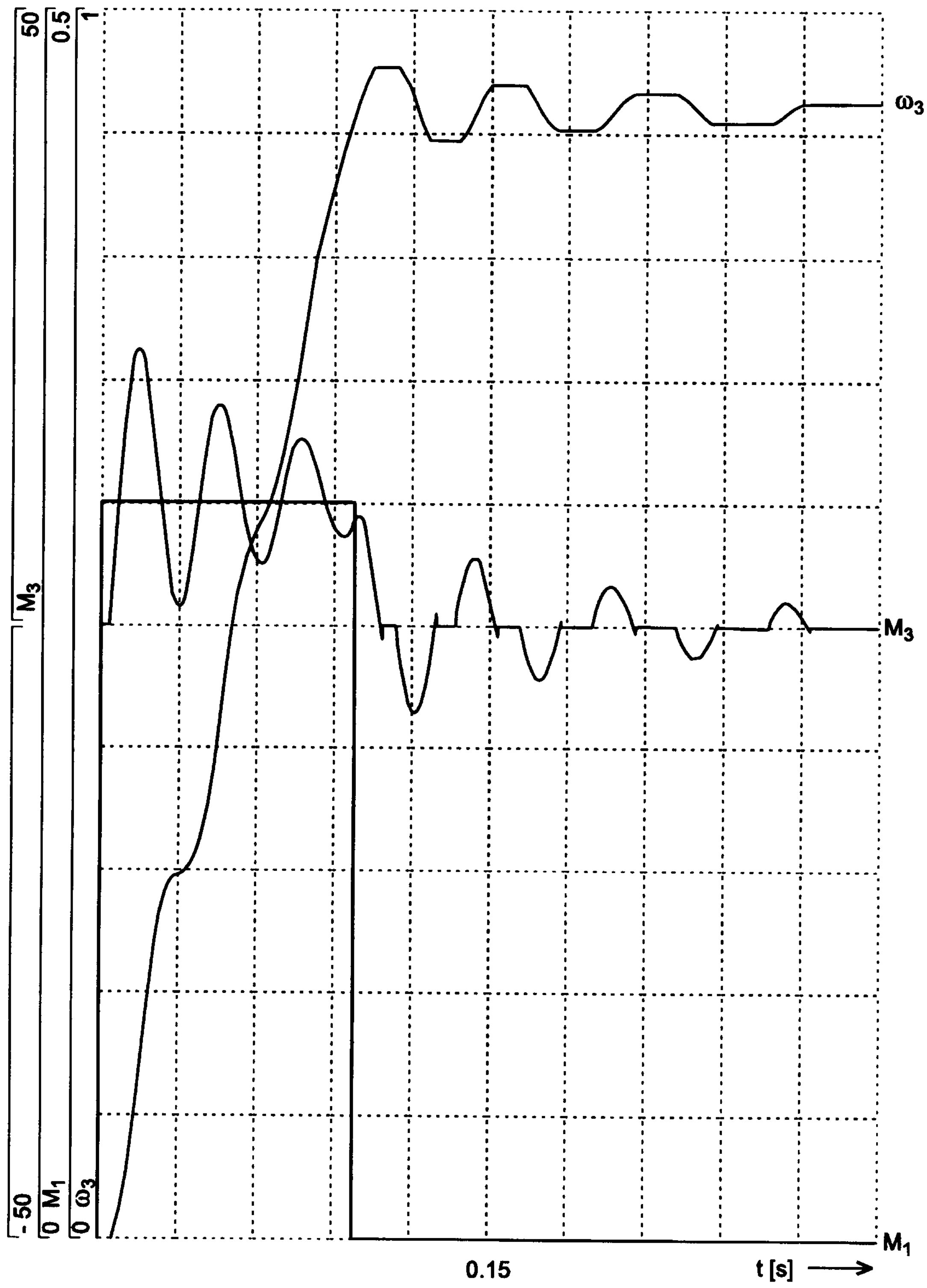


Fig. 5. Transient responses of output speed ω_3 [rad/s], output torque of harmonic transmission M_3 [Nm] to impulse torque excitation M_1 [Nm]

References

Balara, D., 1982, Matematický model harmonického prevodu, (Mathematical Model of Harmonic Transmission), Strojírenství, Praha, No. 2, pp. 85 - 86

Hudoba, M., 1989, Harmonické prevodovky pre servopohony, (Harmonic Transmissions for Servodrives), Automatizace, Praha, No. 2, pp. 44 - 47

Balara, M., 1987, Simulácia dynamických vlastností harmonického prevodu na číslicovom počítači, (Simulation of Dynamic Properties of Harmonic Transmission on PC), Automatizace, Praha, No. 1, pp. 24 - 25

Balara, M., 1992, A robust servosystem of an industrial robot, Transactions of the Technical University of Košice, Vol. 2, No 2, ISSN 0960 6076, Riecansky Science Publishing Co, Cambridge, CB1 6AZ, UK, pp. 251 - 257

Balara, A., 1994, June, MS Thesis

Hori, Y., Iseki, H., Sugiura, K., 1994, Basic Consideration of Vibration Suppression and Disturbance Rejection Control of Multi-inertia System using SFLAC (State Feedback and Load Acceleration Control), IEEE Tranactions on Industry Applications, VOL. 30, NO. 4, July/August 1994, pp. 889 to 896

Harmonic Drive Applications Handbook, 1992, Harmonic Drive Limited, West Sussex, England

Neural Network Architecture and Learning Algorithms

Alexander Weinmann, ÖVE, Senior Member IEEE*

September 17, 1998

Keywords: Neural network training, self-organizing learning, control engineering viewpoint

Abstract: *Emphasis is put on an overview of the architecture of several neural networks, especially from the viewpoint of control engineering. The concept of artificial neural networks and their algorithms are presented. Supervised and unsupervised learning algorithms are implemented in single-layer and multi-layer networks. The main properties of artificial neural networks are carried out by basic programms in MATLAB and by its more sophisticated tools to demonstrate the power of neural network applications. The algorithms and many examples of source programs are traced back to the original training operations.*

1 Introduction. Principles in Cybernetics

Artifical Neural Networks are composed of artificial neurons which are intercommunicating in parallel. Usually, artificial neural networks consist of a high number of neurons. They can learn to simulate biological systems by the behavior of their neurons, having learned either by being trained by a teacher or — which sounds very exciting — having learned in a self-organizing way.

Although the operation on a single neutron is very simple, due to the high number of neurons and due to the operation in parallel, the neural network has the ability to be trained for performing very complex functions, easily and quickly.

Artificial neural networks are designed to be trained by input and output signals. Before learning the network corresponds to a black or grey box. Several real-world actions and behavioral patterns primarily are only known as black or grey boxes. Having trained they can be considered as a white box. The knowledge obtained is stored by means of many coefficients in a memory. These coefficients are denoted as weighting factors.

A basic problem, e.g., is to determine the transfer characteristic in order to describe the functional behavior of the box and to change one's knowledge from black to white, provided input and output signals of the box (in time domain or frequency domain) are available and measurable.

The operating facilities of a neural network can be considered as a complex transformation from the input to the output of the network. The transformation is performed by computation in parallel by general purpose computers. Usually, all the neural network operations are simulated computationally.

*Head of the Institute of Control Engineering, Vienna University of Technology, Gußhausstraße 27, A-1040 Vienna, Tel. +43 1 58801*37500, Fax +43 1 58801*37599, email WEINMANN@IERT.TUWIEN.AC.AT

A typical operation of a neural net is its ability first to be trained in a training or learning phase and second to produce an adequate output information if an input is offered which is in the scope of the trained data.

Learning, training and natural evolution primarily are basic biological activities. They are used for computer-aided modelling of activities in various field of engineering and economics. Artificial neural networks are best suited for general modelling procedures since they can model any process irrespective if it is linear or nonlinear.

Unlike classical systems, analysis and design, which is based on the analytic relations of mass, energy or information flow and which is oriented to implement the relations in formulas and characteristics, artificial neural networks are only based on a specific neural structure and multitude of weighting factors (*Hafner, S., Geiger, H., Kreßel, U., 1992; Ritter, H., Schulten, K., und Marinez, T., 1989*).

Mathematical representations are used in neural network design because they provide a consistent overview and lead to simple numeric evaluation and computation. Aiming at implementing artificial neural networks on digital computers, mathematical representations are an excellent preparation.

Learning processes can be performed either by

- supervised learning, using a teacher,
- or by unsupervised learning, carried out without a teacher but by means of a self-organizing activity, i.e., the network itself separates, recognizes and analyzes similar input signals.

As outlined in Fig. 1, during the learning phase, consistent data of input and desired output are forwarded to the system in a proper sequence (presentation phase). Data of sufficient quantity have to be supplied. The more data the better the result will be, usually. But putting more data into work requires a longer training phase.

In the application (or recall) phase, new inputs u_r, which have not yet been supplied, are fed to the network. The network will separate and identify to which class the new information belongs presupposing that the new data are within the scope of the trained data. Yet unknown inputs, i.e., data beyond the trained orbit, are associated with that class which is optimally close. Probably they remain unidentified at all.

2 Neural Network Architecture

Referring to Fig. 2, there are neurons (or nodes, cells or processing elements) responsible for the transformation or processing of information from the input to the output. The node in an artificial neural network is the simplified model of a biological neuron. Biological nets are characterized by thousands and millions of nodes, artificial ones by dozens and hundreds, only.

Additionally, links provide the transportation of information inside the artificial neural network to and from the output and input. The direction is characterized by arrows. The vector-valued input $\mathbf{u}$ corresponds to the dendrites in biological neurons. The neuron produces an output (axon) $\mathbf{y}$ with entries y_i. Each output element y_i is modelled by a function of the sum of the weighted inputs; a function which is known as the activating function f_A. Learning is performed by adapting weighting factors w_{ij}, the so-called synapses, attaching the significance of the inputs to the nodes.

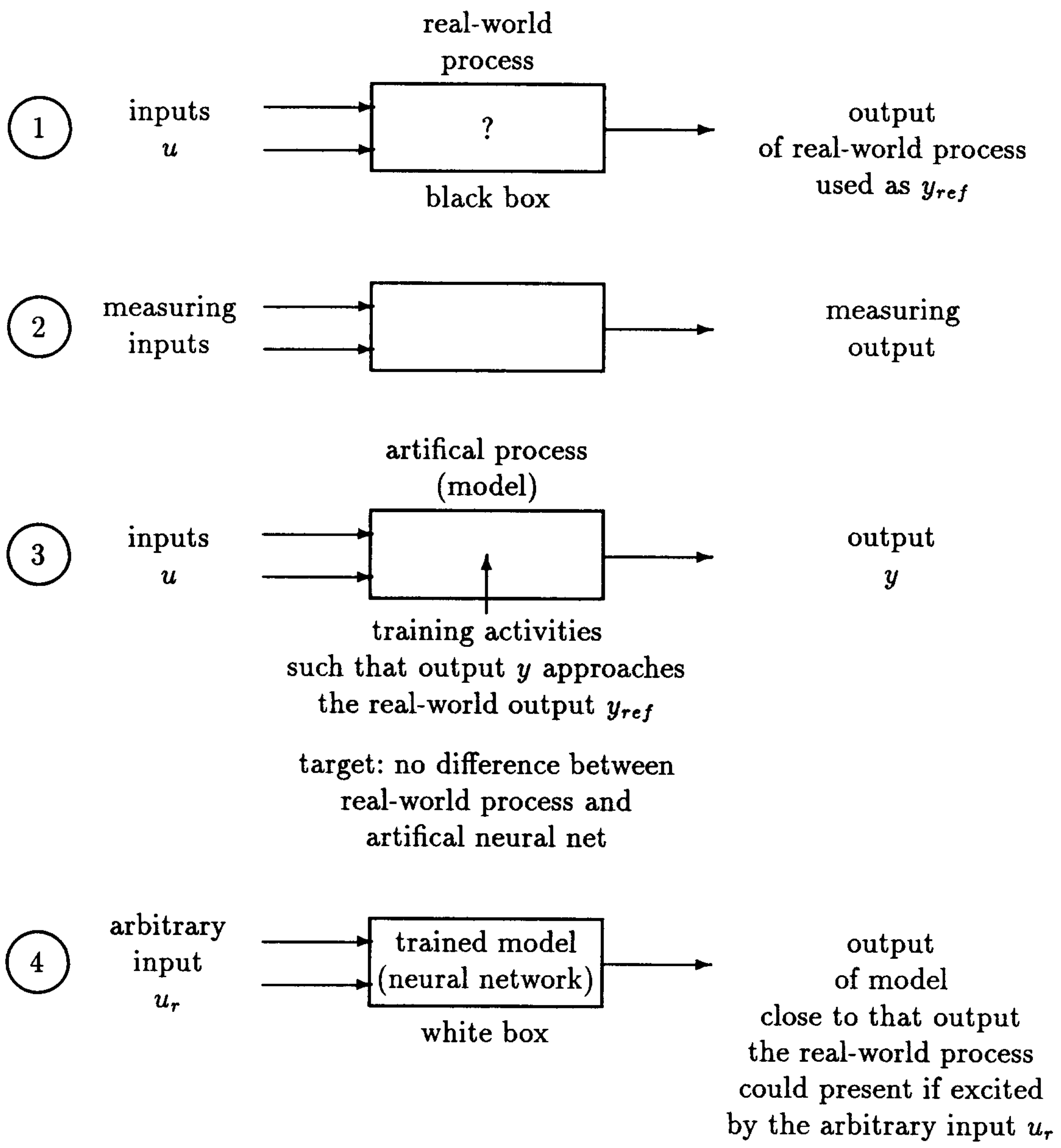

real-world process
1
inputs u
?
black box
output of real-world process used as y_{ref}
2
measuring inputs
measuring output
artifical process (model)
3
inputs u
output y
training activities such that output y approaches the real-world output y_{ref}
target: no difference between real-world process and artifical neural net
4
arbitrary input u_r
trained model (neural network)
white box
output of model close to that output the real-world process could present if excited by the arbitrary input u_r

Figure 1: Training and operation phase. Learning the behaviour of a real-world process by an artificial neural network

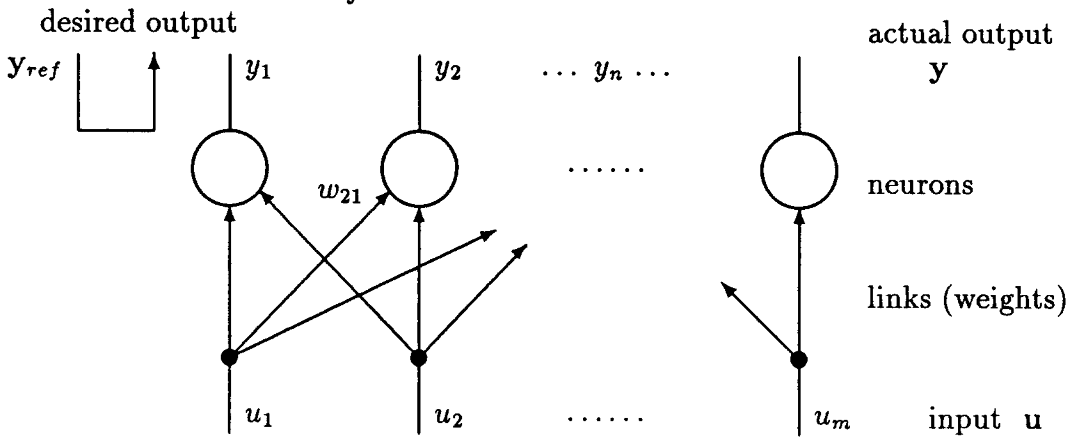


Figure 2: Linear associator neural network (without bias and f_A)

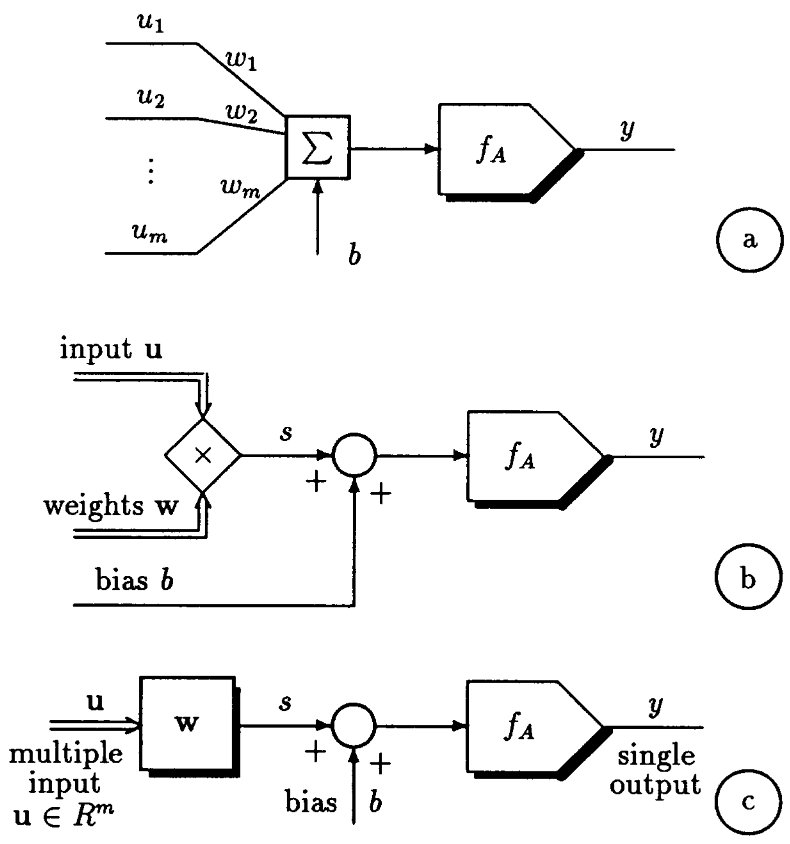


Figure 3: Layer of a single neuron (including weights and bias and f_A). Display with scalar data (a), with vector-valued data (b) and in a block diagram (c)

An input vector (an input time series) **u** with elements u_i is fed to the network. The actual output **y** is to approach the reference (or desired output) $\mathbf{y}_{ref}$ during the training phase.

2.1 Single Neuron with Multiple Input

A single neuron with m inputs $\mathbf{u} \in \mathcal{R}^m$ obeys

$$y = f_A(s + b) = f_A(\sum_{k=1}^{m} w_k u_k + b) \triangleq f_A(\mathbf{w}^T \mathbf{u} + b) \ , \tag{1}$$

see Fig. 3. In addition, a constant shift or bias b has been introduced. Besides, the Figs. 3, 4 provide the same information as Fig. 2 but they emphasize the mathematical interrelations.

2.2 Single Layer of n Neurons with Multiple Input

A layer of n neurons is characterized by the relations

$$\begin{aligned} s_1 &= W(1,1)u_1 + W(1,2)u_2 + \ldots W(1,m)u_m & (2)\\ s_2 &= W(2,1)u_1 + \ldots & (3)\\ &\vdots & (4)\\ s_n &= W(n,1)u_1 + \ldots & (5) \end{aligned}$$

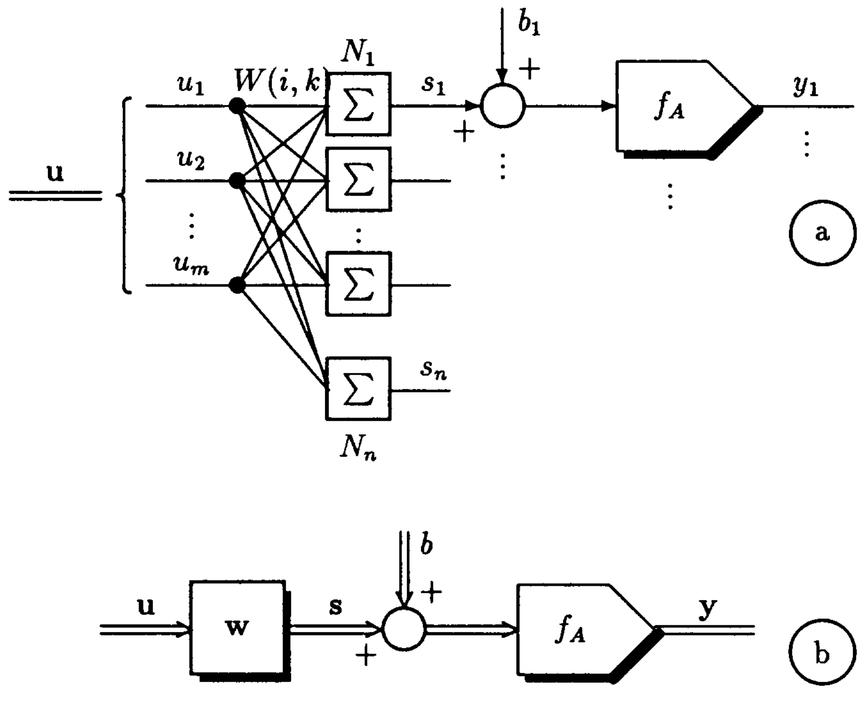


Figure 4: Layer network (layer of multiple neurons)

or in vector notation

$$\mathbf{s} = \mathbf{W}\mathbf{u} \qquad \mathbf{y},\ \mathbf{b},\ \mathbf{s} \in \mathcal{R}^n;\ \mathbf{W} \in \mathcal{R}^{n\times m} \tag{6}$$

$$\mathbf{y} = f_A(\mathbf{s}+\mathbf{b}) = f_A(\mathbf{W}\mathbf{u}+\mathbf{b})\ . \tag{7}$$

The relations are illustrated by Fig. 4.

The matrix entry $W(i,k)$ is the weight from the k-th input to the i-th neuron. For bias zero and $f_A = 1$, matrix multiplication $\mathbf{Wu}$ provides the i-th output $y(i)$ as the weighted sum of all the inputs $u(k)$

$$\mathbf{y} = \mathbf{W}\mathbf{u} \quad \text{or} \quad y(i) = \sum_{k=1}^{m} W(i,k)u(k)\ . \tag{8}$$

The m-vector $\mathbf{u}$ is transferred to the n-vector $\mathbf{y}$ by the weighting matrix $\mathbf{W} \in \mathcal{R}^{n\times m}$.

The input vector *element* is denoted by k from 1 to m. For *several* input vectors, l runs from 1 to q. The neurons are listed by i from 1 to n.

2.3 Recurrent Network

A sample and hold (with sampling period T) is assumed. The discrete-time variable is termed as ν. The output $y(\nu T)$ at νT is used as the next input $\mathbf{u}(\nu T + T)$

$$\mathbf{y}(\nu T) = \mathbf{u}(\nu T + T) \tag{9}$$

$$\mathbf{y}(\nu T) = f_A(\mathbf{W}\mathbf{u}(\nu T) + \mathbf{b})\ ; \tag{10}$$

or in combination

$$\mathbf{y}(\nu T + T) = f_A(\mathbf{W}\mathbf{y}(\nu T) + \mathbf{b})\ . \tag{11}$$

Recurrent networks can operate in a sequential behavior, i.e., according to a set of difference equations.

2.4 Input Waffle

Batching multiple input vectors $\mathbf{u}_l$ leads to

$$(\mathbf{u}_1 \vdots \mathbf{u}_2 \vdots \mathbf{u}_l \ldots \vdots \mathbf{u}_q) \triangleq \mathbf{U} \in \mathcal{R}^{m \times q} \, . \tag{12}$$

The presentation of a waffle of q input vectors $\mathbf{u}$ can be solved by extending a single input vector $\mathbf{u}_l$ to an input matrix $\mathbf{U}$. Simultaneous presentation is solved by sequential computation concerning each column of $\mathbf{U}$, by calculating one column after the other.

The l-th input vector (out of q vectors) obeys

$$\mathbf{s}_l = \mathbf{W}\mathbf{u}_l + \mathbf{b} \qquad \mathbf{b} \in \mathcal{R}^n \tag{13}$$

$$\mathbf{S} = \mathbf{W}\mathbf{U} + \mathbf{B} \qquad \mathbf{W} \in \mathcal{R}^{n \times m} \; ; \qquad \mathbf{S},\ \mathbf{B} \in \mathcal{R}^{n \times q} \tag{14}$$

$$\mathbf{B} = \mathbf{1}^T \otimes \mathbf{b} \triangleq (1 \;\, 1 \ldots 1) \otimes \mathbf{b} \qquad \mathbf{1} \in \mathcal{R}^q \tag{15}$$

$$\mathbf{Y} = f_A(\mathbf{W}\mathbf{U} + \mathbf{B}) \, . \tag{16}$$

The Kronecker symbol $\otimes$ is an abbreviation for $\mathbf{A} \otimes \mathbf{B} \triangleq \mathbf{matrix}_{i,j}[A_{ij}\mathbf{B}]$.

2.5 Multiple Layers of Neurons

In multiple layer networks, the output of a layer operates as the input of the subsequent layer

$$\mathbf{y}_1 = f_{A1}(\mathbf{W}_1\mathbf{u} + \mathbf{b}_1) \quad \text{(output of the input layer)} \tag{17}$$

$$\mathbf{y}_2 = f_{A2}(\mathbf{W}_2\mathbf{y}_1 + \mathbf{b}_2) \quad \text{(output of the final layer)} \, . \tag{18}$$

The input layer receives the incoming signal $\mathbf{u}$. The final layer, presenting the resulting output $\mathbf{y}_2$, is named the output layer. In between, the hidden layer (with index 1) is located.

For practical applications, two layers suffice; in order to be trained by almost any practical function.

3 Perceptron and Basics of its Training Operation

The perceptron, suggested by *Rosenblatt, F., 1961*, is characterized by a hard limit function f_A, i.e., $f_A(x) = 1$ if $x \geq 0$ and $f_A(x) = 0$ if $x < 0$. According to the hard limit activating function f_A, the perceptron operates as a classifier. More specifically, the set of q input m-vectors is separated into different regions.

In the training phase, the input vectors are presented and excite the neural network one after another. If the network output should correspond with the desired output (reference output), no changes in the weights and biases are needed. (This phase is denoted as check phase.) The learning algorithm has only to be started if the product of the initial weight and input plus bias does not match the reference. If the output does not correspond, the difference between reference y_{ref} and output y, i.e., the network error e, is used for changing weights $\mathbf{W}$ and biases $\mathbf{b}$, see Fig. 5.

In the operation phase, an unknown and arbitrary input vector, i.e., a previously unseen vector, which was not included in the training set, will cause the neural network to respond with an output that corresponds optimally to the training result, see Fig. 6.

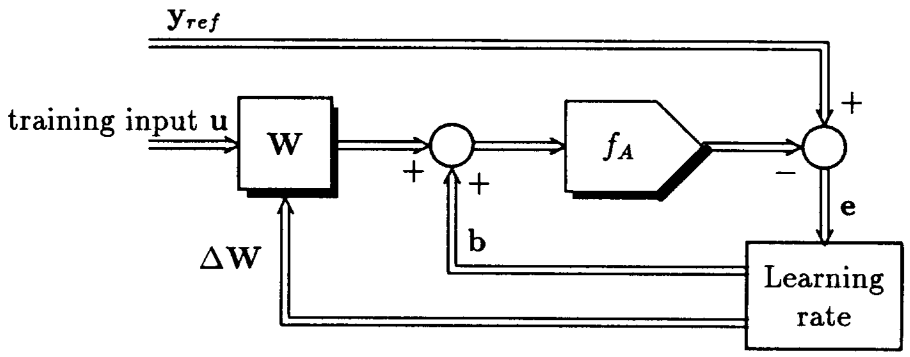


Figure 5: Neural network training phase

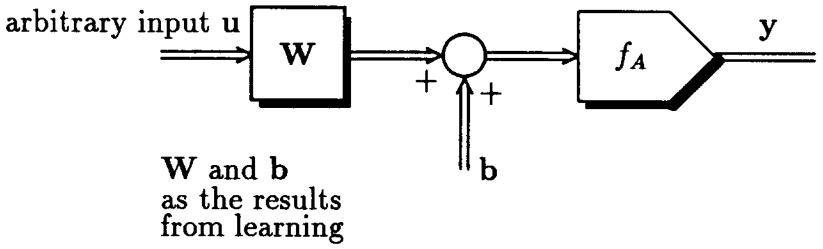


Figure 6: Neural network operation phase

4 Perceptron Training Calculation

The reference (target) outputs are presented in a matrix $\mathbf{Y}_{ref} \in \mathcal{R}^{n \times q}$. The error matrix definition is $\mathbf{E} \triangleq \mathbf{Y}_{ref} - \mathbf{Y} \in \mathcal{R}^{n \times q}$.

The number of biases equals the number of neurons. The weighting matrix and the bias are changed according to the error corresponding to $\mathbf{y}_{ref}$ and the last input vector $\mathbf{u}_l$, i.e. $\mathbf{E}(:,l)$

$$\mathbf{W}_{new} = \mathbf{W}_{old} + \mathbf{E}\mathbf{U}^T \tag{19}$$

$$b^{(i)}_{new} = b^{(i)}_{old} + E(i,l) \tag{20}$$

$$\begin{aligned} \mathbf{b}_{new} &= \mathbf{b}_{old} + \mathbf{E}(:,l) & (21)\\ \mathbf{B}_{new} &= \mathbf{1}^T \otimes \mathbf{b}_{new}\ . & (22) \end{aligned}$$

The outer (or dyadic) product plays an important role in perceptron learning, see Eq.(19). Hebb also suggested a linear learning law using outer product, see Eqs. (48), (49) and (68).

Example. Perceptron with one neuron, two inputs and three input signals:

$$\mathbf{u}_1 \text{ through } \mathbf{u}_q = \begin{pmatrix}-1\\-1\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix} \qquad m=2,\ q=3 \tag{23}$$

$$y_{ref\ 1} \text{ through } y_{ref\ 3} = 1\ \ 0\ \ 0 \qquad n=1 \tag{24}$$

$$\mathbf{U} = \begin{pmatrix} -1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \tag{25}$$

$$\mathbf{Y}_{ref} = (1\ \ 0\ \ 0) \tag{26}$$

$$\text{Initials}:\ \mathbf{W} = \mathbf{W}_o = (2\ \ 4)\ ,\ \ \mathbf{B} = \mathbf{B}_o = (1\ \ 1\ \ 1) \tag{27}$$

Epoch 1:

$$\mathbf{WU} = \mathbf{W}_o\mathbf{U} = (2\ \ 4)\begin{pmatrix} -1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} = (-6\ \ 2\ \ 6) \tag{28}$$

$$\mathbf{W}_o\mathbf{U} + \mathbf{B}_o = (-5\ \ 3\ \ 7) \tag{29}$$

$$\mathbf{Y}_1 = f_A(\mathbf{W}_o\mathbf{U} + \mathbf{B}_o) = \texttt{hardlim}\,(-5\ \ 3\ \ 7) = (0\ \ 1\ \ 1) \tag{30}$$

$$\mathbf{E}_1 = \mathbf{Y}_{ref} - \mathbf{Y}_1 = (1\ \ -1\ \ -1) \tag{31}$$

$$\mathbf{W}_1 = \mathbf{W}_o + \mathbf{E}_1\mathbf{U}^T = (1\ \ 1) \tag{32}$$

$$b_1 = b_o + \mathbf{E}_1(1\ \ 1\ \ 1)^T = 0 \qquad \mathbf{B}_1 = (0\ \ 0\ \ 0) \tag{33}$$

Epoch 2:

$$\mathbf{W}_1\mathbf{U} + \mathbf{B}_1 = (-2\ \ 0\ \ 2) \tag{34}$$

$$\mathbf{Y}_2 = f_A(\mathbf{W}_1\mathbf{U} + \mathbf{B}_1) = (0\ \ 1\ \ 1) \tag{35}$$

$$\mathbf{E}_2 = \mathbf{Y}_{ref} - \mathbf{Y}_2 = (1\ \ -1\ \ -1) \tag{36}$$

$$\mathbf{W}_2 = \mathbf{W}_1 + \mathbf{E}_2\mathbf{U}^T = (0\ \ -2) \tag{37}$$

$$b_2 = b_1 + \mathbf{E}_2(1\ \ 1\ \ 1)^T = -1 \qquad \mathbf{B}_2 = (-1\ \ -1\ \ -1) \tag{38}$$

Epoch 3:

$$\mathbf{W}_2\mathbf{U} + \mathbf{B}_2 = (1\ \ -3\ \ -3) \tag{39}$$

$$\mathbf{Y}_3 = (1\ \ 0\ \ 0) \tag{40}$$

$$\mathbf{E}_3 = \mathbf{Y}_{ref} - \mathbf{Y}_3 = (0\ \ 0\ \ 0)\,. \tag{41}$$

After two entire epochs the final state[1] is reached because $\mathbf{E}_3 = \mathbf{0}$. Thus, the results are $\mathbf{W}_2 = \mathbf{W}_3 = \mathbf{W}_f = (0\ \ -2)$ and $b_3 = b_2 = b_f = -1$.

A graphic interpretation of the classification in the one-neuron-perceptron can be found in Fig. 7. The borderline between the classes as executed by hardlimiter f_A is given by $\mathbf{W}_f\mathbf{u} + b_f = 0$

$$(w_{1f}\ w_{2f})\begin{pmatrix}u_1\\u_2\end{pmatrix} + b \;=\; 0 \tag{42}$$

$$w_{1f}u_1 + w_{2f}u_2 + b \;=\; 0\,. \tag{43}$$

The control flow diagram of the perceptron is depicted in Fig. 8. The appropriate MATLAB source program is

[1]with index f

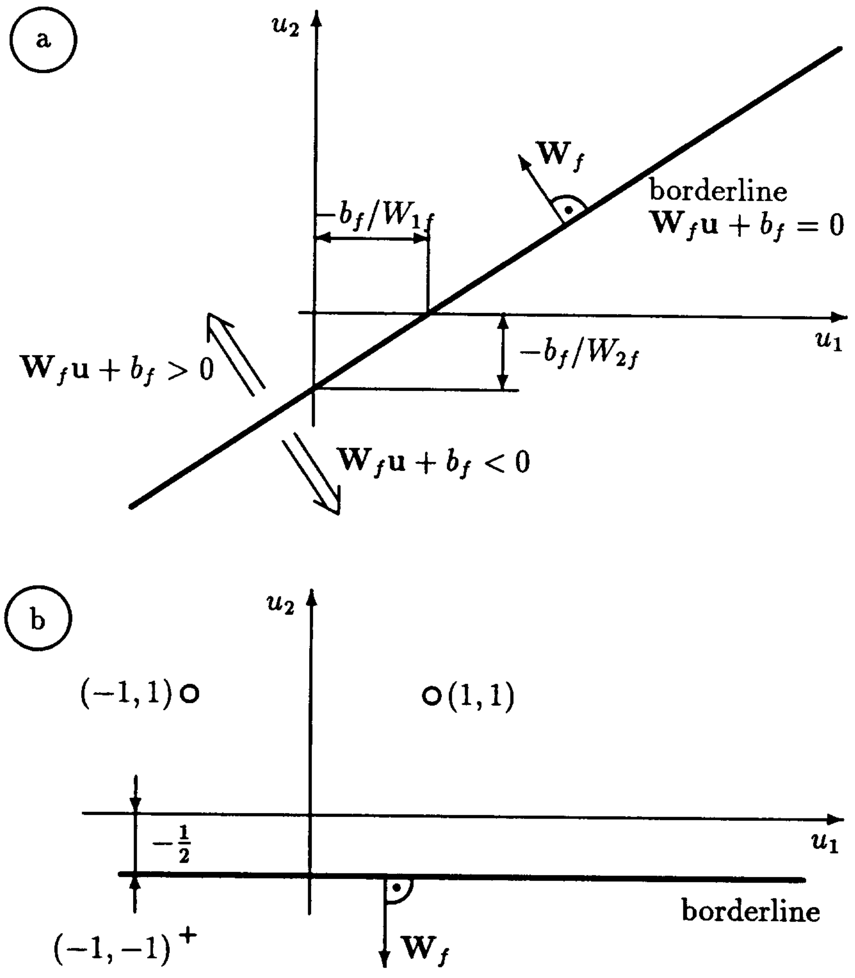


Figure 7: Input vector graph in the general case (a) and in the example (b) where $n = 1,\ \mathbf{W}_f = \mathbf{w}_f^T = \in \mathcal{R}^{1\times m}$

```
% Perceptron                                              % grz.m
TP=[1  4];  % number of epochs between displaying the current epoch
            % and maximum number of epochs to train
U=[-1 -1 1 ; -1  1 1];
Yref=[1 0 0];     % row matrix required
W=[2 4];
b=1;              % scalar required
for ii=1:5
    B=[b b b];
    W*U+B;
    Y=hardlim(W*U+B);
    W=W+(Yref-Y)*U';
    b=b+(Yref-Y)*[1 1 1]';
    pause
end
```

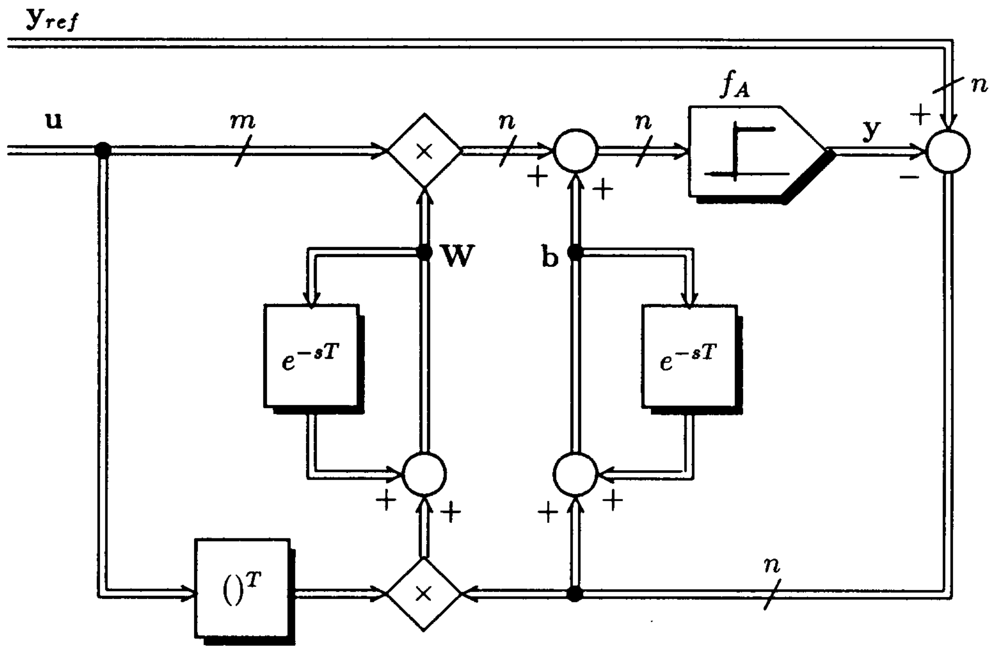


Figure 8: Control signal flow diagram respresenting the training algorithm for the perceptron

or equivalently

```
  :
b=1;
[W,B,epochs]=trainp(W,b,U,Yref,TP) .
```

The decision quality in this simple example versus training epochs can be viewed and assessed in Fig. 9.

Perceptrons are limited by being able only to classify linearly separable sets of input vectors. Linearly nonseparable data will cause the perceptron not to arrive at a final training result.

An outlier is an input vector with extraordinary magnitude. Such an outlier results in worse convergence properties.

5 Linear Learning Laws and Hebb's Outer-Product Algorithm

Denoting the input (entry) and output as $\mathbf{u} = \textbf{vec}\ \{u_i\} \in \mathcal{R}^m$ and $\mathbf{y} = \textbf{vec}\ \{y_i\} \in \mathcal{R}^n$, respectively, a weighting matrix is set

$$\mathbf{W} \triangleq (\mathbf{w}_1\ \mathbf{w}_2 \ldots \mathbf{w}_n)^T \in \mathcal{R}^{n \times m}, \tag{44}$$

$$\mathbf{w}_i \triangleq (w_{i1} \ldots w_{im})^T \ \ \forall\, i = 1 \ldots n\ . \tag{45}$$

The output $\mathbf{y} = \mathbf{W}\mathbf{u}$ is assumed to be a *linear* combination of the inputs u_k (see Fig. 2). The output $\mathbf{y}$ has to approach $\mathbf{y}_{ref}$, i.e., $\mathbf{y} \rightarrow \mathbf{y}_{ref}$ where $\mathbf{y}_{ref}$ is the training setpoint.

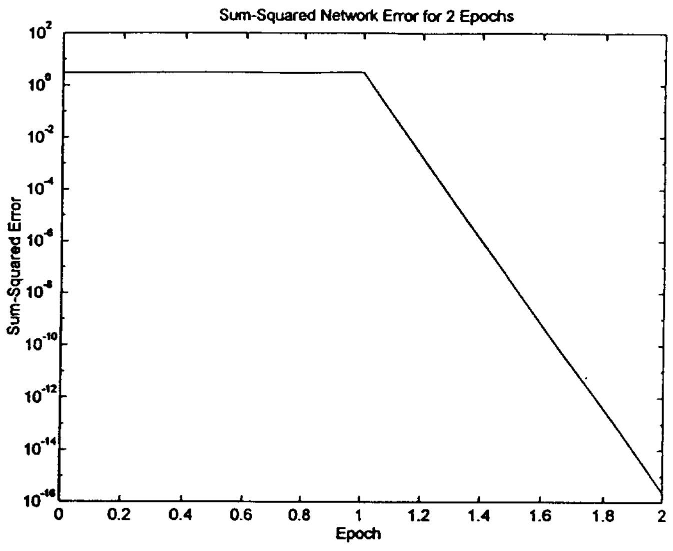


Figure 9: Decision quality for the Example grz.m

The question is if $\mathbf{W}$ exists and which $\mathbf{W}$ should be selected. If such a matrix $\mathbf{W}$ exists then the learning procedure turns out as an operation of simple matrix multiplication.

During the training phase, the linear associator neural network is to learn q pairs of input output vectors $\mathbf{u}_1$ $\mathbf{y}_1$ through $\mathbf{u}_q$ $\mathbf{y}_q$. If the vectors $\mathbf{u}_l$ are orthonormal, we claim that the outer-product sum is the solution, i.e.,

$$\mathbf{W} \triangleq \sum_{l=1}^{q} \mathbf{y}_{ref,l}\, \mathbf{u}_l^T \, . \tag{46}$$

Then, in fact, the result is $\mathbf{y}_{ref,l} = \mathbf{W}\mathbf{u}_l$, since by orthonormality $\mathbf{u}_i\mathbf{u}_j = \delta_{ij}$ (Kronecker symbol)

$$\mathbf{W}\mathbf{u}_k = (\sum_{l=1}^{q} \mathbf{y}_{ref,l}\, \mathbf{u}_l^T)\mathbf{u}_k = \mathbf{y}_{ref,k} \cdot 1 + \mathbf{0} = \mathbf{y}_{ref,k} \tag{47}$$

in the recall phase. The increment of the weighting matrix $\Delta\mathbf{W}$ is

$$\Delta\mathbf{W} = \mathbf{y}_{ref,l}\, \mathbf{u}_l^T \qquad \text{(Hebb's outer-product learning law),} \tag{48}$$

as long as $q \leq m$ (*Hecht-Nielsen, R., 1989*).

Example. Hebb's linear learning law:

$$\mathbf{u}_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{u}_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{u}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

$$\mathbf{y}_{ref,1} = \begin{pmatrix} 8 \\ 2 \\ 1 \end{pmatrix}, \quad \mathbf{y}_{ref,2} = \begin{pmatrix} 4 \\ 4 \\ 4 \end{pmatrix}, \quad \mathbf{y}_{ref,3} = \begin{pmatrix} -7 \\ 3 \\ 9 \end{pmatrix}$$

$$\rightsquigarrow \quad \mathbf{W} = \begin{pmatrix} 8 \\ 2 \\ 1 \end{pmatrix} (0 \ 1 \ 0) + \ .... \ = \begin{pmatrix} 4 & 8 & -7 \\ 4 & 2 & 3 \\ 4 & 1 & 9 \end{pmatrix}$$

and $\mathbf{Wu}_l = \mathbf{y}_l = \mathbf{y}_{ref,l}$ is satisfied.□

6 Widrow-Hoff Networks

Widrow-Hoff networks are characterized by performance orientation and by a linear activating function f_A. The learning algorithm is

$$\mathbf{W}_{new} = \mathbf{W}_{old} + \beta(\mathbf{Y}_{ref} - \mathbf{Y})\mathbf{U}^T \tag{49}$$

$$\mathbf{b}_{new} = \mathbf{b}_{old} + \beta[\mathbf{Y}_{ref}(:,l) - \mathbf{Y}(:,l)] \ . \tag{50}$$

The dimensions correspond to Eqs.(19) through (21). The parameter β is a learning rate.

This learning rule guarantees an adjustment of the weights and biases of least mean square, finally. Presupposing a sufficently small learning rate, convergence is guaranteed because the error surface is a multi-dimensional parabola, see Eq.(52). Learning rate of undue height causes an unstable search process.

A nonlinear function between input and output vectors is approximated linearly. Only single layer networks are sensible because any multi-layer linear network can be traced back to an equivalent single layer network.

Usually, the initials are selected as random numbers.

An important application is the pattern associator. Having trained the Widrow-Hoff network, the network will answer with the associated output precisely when an input out of the training set is presented to the network.

A single neuron ($n = 1$) excited by one input ($m = 1$) can only learn two different vectors since there are only two variables weight and bias. In such a case the training runs perfectly. If more input vectors are presented to be trained, this is an overdetermined case and the network can only minimize the error.

Whether or not input data cause overdetermination can be found out from the size of the final (minimum) error.

In the case of m inputs and n neurons the degree of freedom is $(m + 1)n$, given by m weights and one bias.

Example. Widrow-Hoff Learning: Assume $m = 1$, $q = 2$, $n = 1$. The MATLAB code for Widrow-Hoff search requires

```
% Widrow-Hoff Learning Example      % gse.m
W=-0.9;      B=-0.9;     % initial conditions
WW(1)=W;     BB(1)=B;
U=[1  -1.4];
Yref=[0.6  1];
EE(1)=Yref(1)-(W*U(1)+B);  % first component used as error for
                           % current recall phase
beta=0.4*maxlinlr(U);      % learning rate
for ii=1:35
     W=W+beta*(Yref-W*U-B)*U';
```

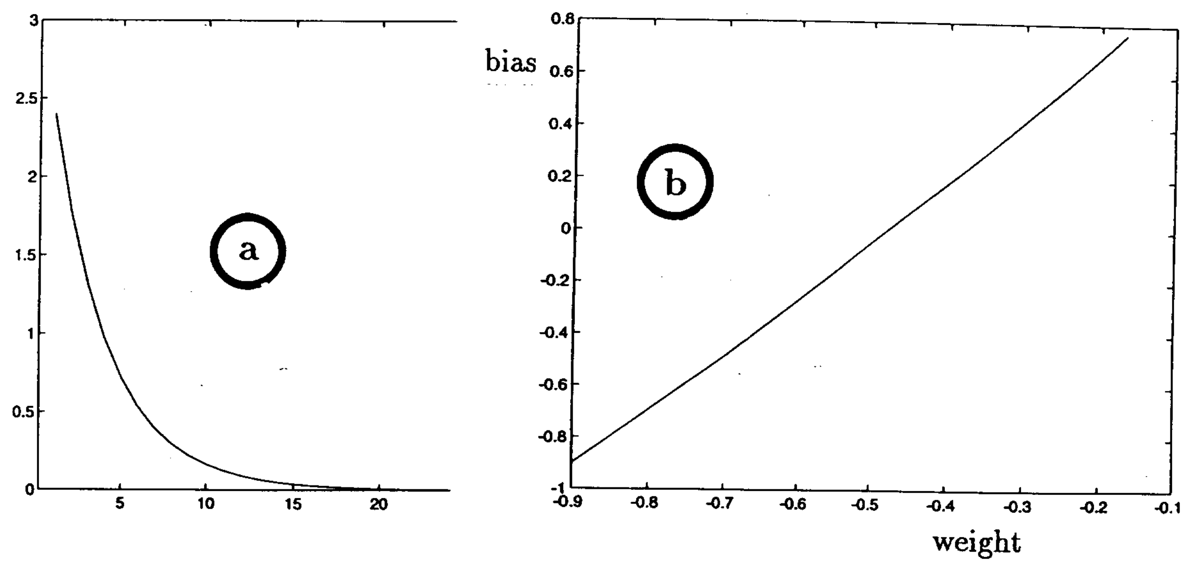


Figure 10: Error versus epochs for first component of U (a) and progress of W and B during training (b)

```
        B=B+beta*(Yref-W*U-B)*[1  1]';
        BB(ii+1)=B;        WW(ii+1)=W;
        EE(ii+1)=Yref(1)-(W*U(1)+B);
        axis([0.1 35 0 3])
        plot(EE)  % visualization of current error
        hold on
        pause2(0.2)
end
pause
hold off
plot(WW,BB)    % learning procedure in W and B
```

Numerical results after 35 epochs are $W = -0.1667$ and $B = 0.7666$. The results are shown in Fig. 10.

MATLAB Neural Network Toolbox comprises a specific Widrow-Hoff tool `trainwh`

```
TP=[disp_freq  max_epoch  err_goal beta];
[W,B,epochs]=trainwh(Wo,Bo,U,Yref,TP)
```

The learning rate is selected from `beta` $= 0.4$ `maxlinlr` (U) $= 0.1639$ where

$$\texttt{maxlinlr}(\mathrm{U}) \triangleq \frac{1}{\max \lambda[\mathrm{UU}^T]} \tag{51}$$

and $\lambda[\cdot]$ is the eigenvalue.

A three-dimensional diagram (error versus W and B) is a three-dimensional parabola. On this plane (represented in a mesh diagram) curves could be drawn guiding from any initial condition W_o, B_o into the final values W_f, B_f.

6.1 Performance Orientation of Widrow-Hoff Learning

Consider the linear combination given by the scalar or inner product $y = \mathbf{w}^T\mathbf{u}$, the signals y_{ref} and $\mathbf{u}$ are given, $\mathbf{w}$ is unknown. If a vector-valued output $\mathbf{y}$ should be achieved, the following derivations are rewritten to matrix-valued $\mathbf{W}$ and vector-valued $\mathbf{y}$ and $\mathbf{y}_{ref}$.

Assume $n = m$. Considering a sequence $\mathbf{u}_l$ of q vectors and the actual and the desired output y_l and $y_{ref,l}$, respectively, the mean squared error is characterized by a performance $g(\mathbf{w})$

$$g(\mathbf{w}) \triangleq \frac{1}{q}\sum_{l=1}^{q}(y_{ref,l} - y_l)^2 . \tag{52}$$

Taking a high number of vectors into account, the expectation operator $E\{\cdot\}$ can be applied

$$g(\mathbf{w}) \triangleq E\{(y_{ref,l} - y_l)^2\} = E\{(y_{ref,l} - \mathbf{w}^T\mathbf{u}_l)^2\} = \tag{53}$$

$$= E\{y^2_{ref,l}\} - 2\mathbf{w}^T E\{y_{ref,l}\mathbf{u}_l\} + \mathbf{w}^T E\{\mathbf{u}_l\mathbf{u}_l^T\}\mathbf{w} . \tag{54}$$

Searching the optimum, the gradient of the weighting function with respect to $\mathbf{w}$ is used and equated with zero, i.e.,

$$g(\mathbf{w}) \rightarrow \min_{\mathbf{w}} \quad \rightsquigarrow \quad \frac{\partial g(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial g(\mathbf{w})}{\partial \mathbf{w}} = -2E\{y_{ref,l}\mathbf{u}_l\} + 2E\{\mathbf{u}_l\mathbf{u}_l^T\}\mathbf{w} = 0 \tag{55}$$

which leads to $\mathbf{w} = \mathbf{w}_{\text{opt}}$

$$\mathbf{w}_{\text{opt}} = (E\{\mathbf{u}_l\mathbf{u}_l^T\})^{-1} \cdot E\{y_{ref,l}\mathbf{u}_l\} . \tag{56}$$

Moreover, referring to Eq.(52) and replacing q by N, the difference quotient is

$$\frac{\Delta g(\mathbf{w})}{\Delta \mathbf{w}} = \lim_{N\to\infty}\frac{1}{N}\sum_{l=1}^{N} 2(y_{ref,l} - y_l)(-\mathbf{u}_l) . \tag{57}$$

Considering the case that the input signals $\mathbf{u}_l$ are supplied stepwise, the increment $\Delta\mathbf{w}$ can be set proportional to the gradient in Eq.(57). *Widrow, B., and Hoff, M.E., 1960,* postulated a so-called least-mean-square learning law with an increment directly proportional to $(y_{ref,l} - y_l)\mathbf{u}_l$. Then, the resulting Widrow-Hoff learning law is

$$\Delta\mathbf{w} = \beta\,(y_{ref,l} - y_l)\mathbf{u}_l \qquad \text{(Widrow-Hoff learning law)} \tag{58}$$

which is always converging to $\mathbf{w}_{\text{opt}}$ from any $\mathbf{w}_o$. The learning rate β is a positive constant in order to obtain minimum g. Widrow-Hoff learning is also known as adaptive linear element learning. It is numerically simple.

The change of weights is proportional to the difference between the current output and reference multiplied by the input $\mathbf{u}_l$ and a learning rate β.

The result of Eq.(58) also provides a clear insight into the general linear neural learning rule. Consider the cell j being excited by cell i. The general Hebb's rule of learning (*Hebb, D., 1949*) is then given by $\Delta w_{ji} = \beta s_i\, s_j$ where the link from i to j is used, s_i is the output of the sourcing cell i and s_j is the activation of the consecutive cell j. If both the s_i and the s_j are high then Δw_{ji} is to be increased.

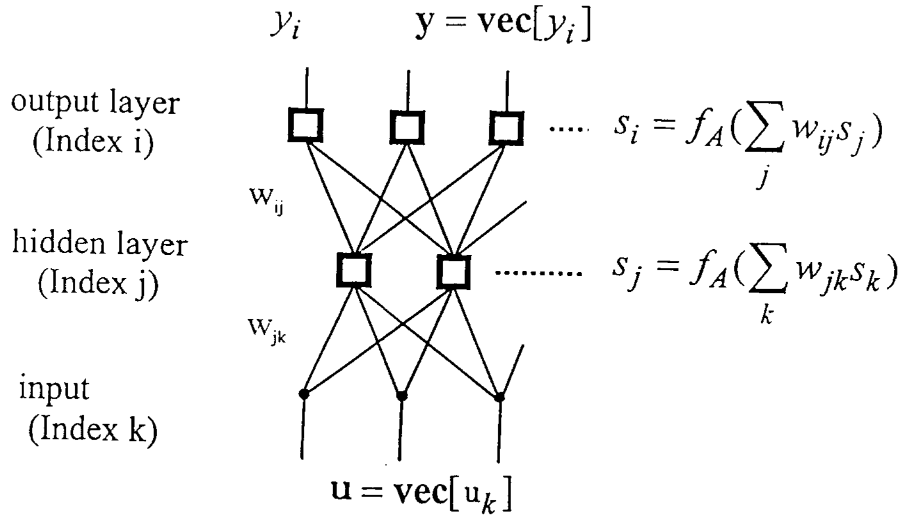


Figure 11: Neural net with two layers

7 Backpropagation Learning in a Two-Layer Network

Hitherto, neurons have been considered in a structure of being equally entitled in one single layer. In Fig. 11, a multilayer network is depicted. The layer organization stands for the property that within the layer there is no connection. All the connections are referred to another layer or to the input or output. A feedforward network is characterized by only one direction of data transfer, corresponding to the resulting direction from the input to the output.

The sensors are connected to the input u . The output layer with output y is connected to the actors. In between, there is one (or there are more) hidden layers (*Ritter, H., et al. 1990*). Referring to Kolmogorow, for the approximation of a continuous function only one hidden layer is required. Most of the neurons are not connected to input or output and, hence, are named hidden.

Unlike former index assignments, the activity of a neuron in each layer is denoted by s with an appropriate index, i.e., i, j, k for output layer and hidden layer and input, respectively. The input is u where $\mathbf{u} = \mathrm{vec}[u_k] = \mathrm{vec}[s_k]$. Analogously, the output is $\mathbf{y} = \mathrm{vec}[y_i] = \mathrm{vec}[s_i]$. Using the activation function f_A, the relations are

$$s_i = f_A(\sum_j w_{ij}s_j) \qquad \text{and} \qquad s_j = f_A(\sum_k w_{jk}s_k) \ . \tag{59}$$

In addition, a bias or offset input could be taken into account in order to valuate an overall displacement. The activating function f_A is a nonlinear function. The input is transferred to a (first) hidden layer with neurons characterized by a general LOG-SIGMOID or TAN-SIGMOID activation function. The SIGMOID function transfers an input between $-\infty$ and $+\infty$ to the output between 0 and +1; the TAN-SIGMOID function to -1 and $+1$. Alternative activating functions are identy, scaling (linear) or threshold (linear or hard limit). In order to emphasize the nonlinear activating function, the symbols of the nodes are squares in Fig. 11. The output layer is equipped with a linear f_A.

The aim of the neural network is to learn the weighting factors w of each path. Assuming a supervising performance g, a gradient method facilitates the adaptation of the weighting factors.

A wide-spead method is the backpropagation method by which the weighting factors are learned progressively, starting from the supervised output and its associated weighting factors, and proceeding to the input weighting factors. The approach operates in backward direction. Thus, the name "backpropagation learning" results from the fact that the derivative of an error of the hidden layer is calculated from the output layer derivative, reverse to the direction from input to output. Based on the available training data, a resulting performance is defined by a quadratic function g, see Eq.(60). The increments are derived from its gradient.

First of all, the relations are written in such a form as if the the output, input and weighting factors were given. But these relations are only utilized in order to find out the weighting factors, evaluating the given training input and output. The learning algorithm uses a multitude q of inputs characterized by superscript l. The purpose of the learning algorithm is to learn the weighting factors w_{ij} und w_{jk} supervised by a performance g to be minimized[2]. Learning is performed by comparison and incremental change in the direction of the gradient, and that in the following steps:

- input data are fed to the input,
- then initialize w_{ij} and w_{jk} and find y and $y_{ref} - y$,
- run the following gradient descent (backpropagation algorithm)

$$g = \sum_{l=1}^{q} \sum_{i} [y^{l}_{ref,i} - s_i(\mathbf{u}^l)]^2 \tag{60}$$

$$\begin{aligned}
\text{where the output activity is} \quad s_i &= f_A(s_j)\ , && (61)\\
\text{the hidden activity is} \quad s_j &= f_A(s_k) && (62)\\
\text{and the input (and its activity)} \quad \mathbf{u} &= \mathbf{vec}[s_k]\ . && (63)
\end{aligned}$$

The weighting factors w_{ij} are changed referring to

$$\Delta w_{ij} \propto -\frac{\partial g}{\partial w_{ij}} \quad \leadsto \quad \Delta g \propto \sum_{ij} \frac{\partial g}{\partial w_{ij}} \Delta w_{ij} \propto -\sum_{ij} (\frac{\partial g}{\partial w_{ij}})^2 \leq 0\ . \tag{64}$$

Differentiating the performance with respect to the weighting factors associated with the output, one has

$$\frac{\partial g}{\partial w_{ij}} = -2\sum_{l} [y_{ref,i} - s_i(\mathbf{u}^l)] \frac{\partial s_i(\mathbf{u}^l)}{\partial w_{ij}} = -\sum_{l} [y_{ref,i} - s_i(\mathbf{u}^l)] f'_A(\sum_{\nu} w_{i\nu} s_\nu) s_j \tag{65}$$

where $'$ is the derivation with respect to the argument. The activating function f_A is considered differentiable. Calculating the derivative with respect to the weighting factors associated with the input, the intermediate differentiation with respect to s_j has to be taken into account. Out of this and referring to the chain rule, the result is

$$\begin{aligned}
\frac{\partial g}{\partial w_{jk}} &= -2\sum_{l}\sum_{i} [y^{l}_{ref,i} - s_i(\mathbf{u}^l)] \left[f'_A(\sum_{\nu} w_{i\nu} s_\nu)\right] w_{ij} \frac{\partial s_j}{\partial w_{jk}} = && (66)\\
&= -2\sum_{l}\sum_{i} [y^{l}_{ref,i} - s_i(\mathbf{u}^l)] \left[f'_A(\sum_{\nu} w_{i\nu} s_\nu)\right] w_{ij} \left[f'_A(\sum_{\nu} w_{j\nu} s_\nu)\right] s_k\ . &&
\end{aligned}$$

[2]In the derivation, the bias is considered replaced by an additional constant input

From the foregoing analysis the gradients $\frac{\partial g}{\partial w}$ or $\frac{\Delta g}{\Delta w}$ result, yielding the basis for changing and learning the increments Δw in Eq.(64). The operations in Eqs.(65) and (66) demonstrate the numerical effort. Parallelling the computations is an urgent need as far as time for learning is concerned.

The elements of the vectors $\mathbf{y}_{ref}$ and $\mathbf{u}$ are given. It seems adequate to use a larger set of input and output vectors for better results. The topology of the network is predetermined according to the designer's experience. If the training procedure does not yield adequate results valuated by g, the topology of the pattern has to be changed.

```
% Neural network and backpropagation learning  % gsj.m
for t1=1:26
    t=t1-1;
    U(t1)=t;  % input variable corresponds to real-world time
    Yref(t1)=0.5* (1 - exp(-0.1*t) * cos(0.3*t));
    Time(t1)=t;
end
plot(Time,U,Time,20*Yref)
pause
[m,q]=size(U);  % 26 scalar input signals
n1=4;           % assumption of four neurons in the hidden layer
[n2,q]=size(Yref);
  W1o=[-0.5621  -0.9059  0.3577  0.3586]';  % initial conditions
  B1o=[ 0.8694  -0.2320  0.0388  0.6619]';  % initial conditions
  W2o=[-0.9309  -0.8931  0.0594  0.3423];   % initial conditions
  B2o=-0.9846;                              % initial conditions
    disp_freq=200;
    max_epoch=60000;   err_goal=0.01;
    beta=0.01;  % learning rate
TP=[disp_freq  max_epoch  err_goal  beta];
[W1,B1,W2,B2,epochs,TR]=...
     trainbp(W1o,B1o,'tansig',W2o,B2o,'purelin',U,Yref,TP)
```

During the operation phase, the output is

$\mathbf{y} = \mathtt{purelin}[\mathbf{W}_2\mathtt{tansig}(\mathbf{W}_1\mathbf{u}_r + \mathbf{b}_1) + \mathbf{b}_2]$

where $\mathbf{u}_r$ is an arbitrary input.

The result of the aformentioned programm is depicted in Fig. 12.

The two-layer network with backpropagation can learn any sensible nonlinear function. Problems arise from the problem of local minima and adequate learning rate. Inproper choice of the learning rate results in slow learning performance or instability. Slow convergence also results from the high number of neurons.

If there are not enough neurons available because the assumption was to low, the quality of learning is poor. This is called underfitting.

Techniques to overcome these difficulties are: Select multiple initial conditions or variable learning rate, reduce or increase the number of hidden neurons or choose the momentum method.

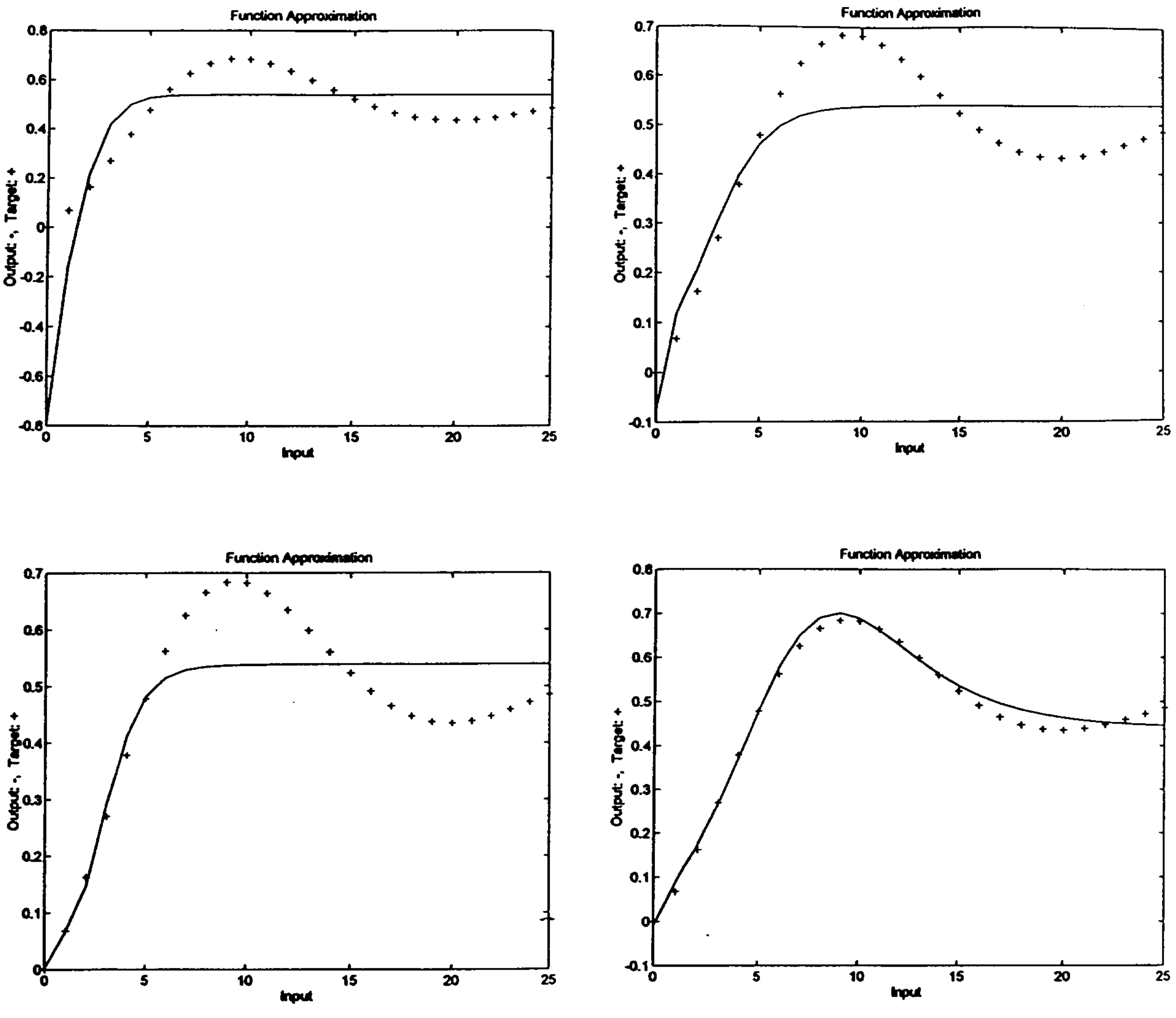


Figure 12: Backpropagations result after 20, 400, 4000 and 54382 epochs (after having satisfied the error goal)

The momentum method does not only follow the local gradient. The backpropagation algorithm is augmented by the gradient trend. Small minima are thus ignored and the algorithm cannot be stopped by a shallow minimum.

Moreover, the learning rate β can be chosen adaptively. As long as stability is guaranteed, the learning rate is increased. Reversely, the rate is reduced. The momentum algroithm requires a constant momentum (typically 0.95) and an error ratio (typically 1.04). The error ratio determines a fraction between new error and old error. If this fraction exceeds the error ratio, the new weights are rejected.

The algorithm including momentum contains the following essential part at the end of the aforementioned source programm

```
% Backpropagation learning with MOMENTUM  % gsk.m
:
max_epoch=60000;    err_goal=0.01;
err_ratio=1.04;
momentum=0.95;
beta=0.01;     % learning rate
TP=[disp_freq  max_epoch  err_goal  beta  momentum err_ratio];
[W1,B1,W2,B2,epochs,TR]=...
     trainbpm(W1o,B1o,'tansig',W2o,B2o,'purelin',U,Yref,TP)
```

The algorithm using adaptive learning rate changes (increases or decreases) the learning rate by predetermined factors if the new error exceeds the old by more than a pretermined error ratio. Adaptive learning rate usually works very well.

The essential part using adaptive learning rate comprises the following MATLAB toolbox demands

```
% Backpropagation learning with  ADAPTIVE LEARNING RATE % gsl.m
:
disp_freq=200;
max_epoch=60000;
err_goal=0.01;
beta=0.01;  % learning rate
beta_inc=1.05;
beta_dec=0.75;
err_ratio=1.045;
TP=[disp_freq  max_epoch  err_goal  beta beta_inc beta_dec...
    err_ratio];
[W1,B1,W2,B2,epochs,TR]=...
     trainbpa(W1o,B1o,'tansig',W2o,B2o,'purelin',U,Yref,TP)
```

The result is portrayed in Fig. 13. It is remarkably better than the result in Fig. 12.

Example. Positioning of robot arms: Consider robot arms fetching a screw in a plane. The positions are measured in two coordinates by a video camera. The neural network has to learn the ange of the robot arms which is required to position the screw definitively. Using classical methods, one has to implement calculation between positions and angles of the robot arm according to complicated trigonometry.

Example. Encoder Decoder Problem: Consider a neural net with 8 inputs, 8 outputs and 3 hidden neurons. The input and output are binary signals. (For better processing, the signal level 0.1 and 0.9 is selected.) The network has to learn to react with only one

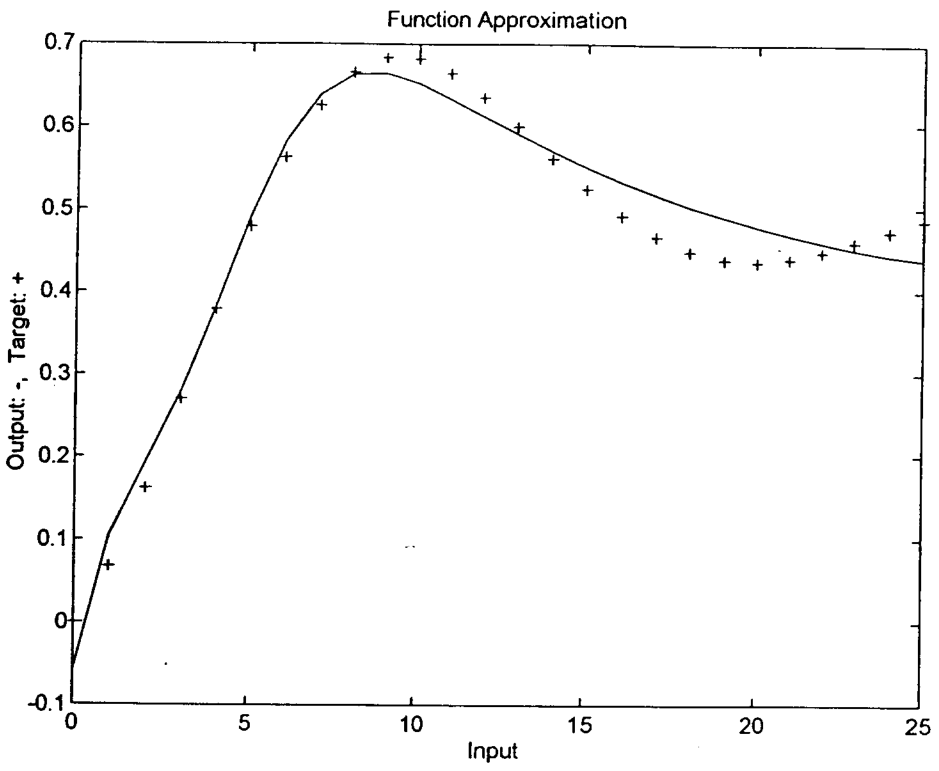


Figure 13: Backpropagation result with adaptive learning rate after 4001 epochs

output i if the input excitation is at position i, only. The problem is that the hidden layer has only 3 neurons, thus being a bottle neck.

Backpropagation algorithm and learning of the 48 weights can be performed without an external teacher if the system is trained to learn the identity map. The input and the output pattern are the same (autoencoder facility).

Example. Pattern Recognition: Backpropagation can be used for pattern recognition. Letters are quantized to 5×7 patterns. These 35 squares are fitted black or white for accurate presentation of the 26 letters of the alphabet. They are transferred to a vector of dimension 35. There are 26 reference values according to the alphabet and 26 output neurons.

In the recall phase, letters usually disturbed by noise are presented. Black and white is reduced to grey of different intensity depending on the amount of noise. Then, the vector $\mathbf{u}$ of a noisy input letter does not consist of 26 signals 0 or 1 but 26 elements *between* 0 and 1. More than one output neuron will respond due to input noise. Hence, in a post-processing procedure using f_A =`compet`, the result is selected and presented as the output.

In Fig. 14 the letters A without noise and N and B with 10 % noise are recognized correctly, the letter R with roughly 25 % noise is not classified correctly. Even under high noise, letters might happen to be recognized correctly, as Y in the figure.

Example. Identifiying a dynamical system: Consider a discrete-time system obeying

$$\alpha_2 x(\nu - 2) + \alpha_1 x(\nu - 1) + \alpha_o x(\nu) - v(\nu) = 0 \ . \tag{67}$$

Identification has to provide the parameters α_i of the difference equation, given input and output signals at consecutive instants ν.

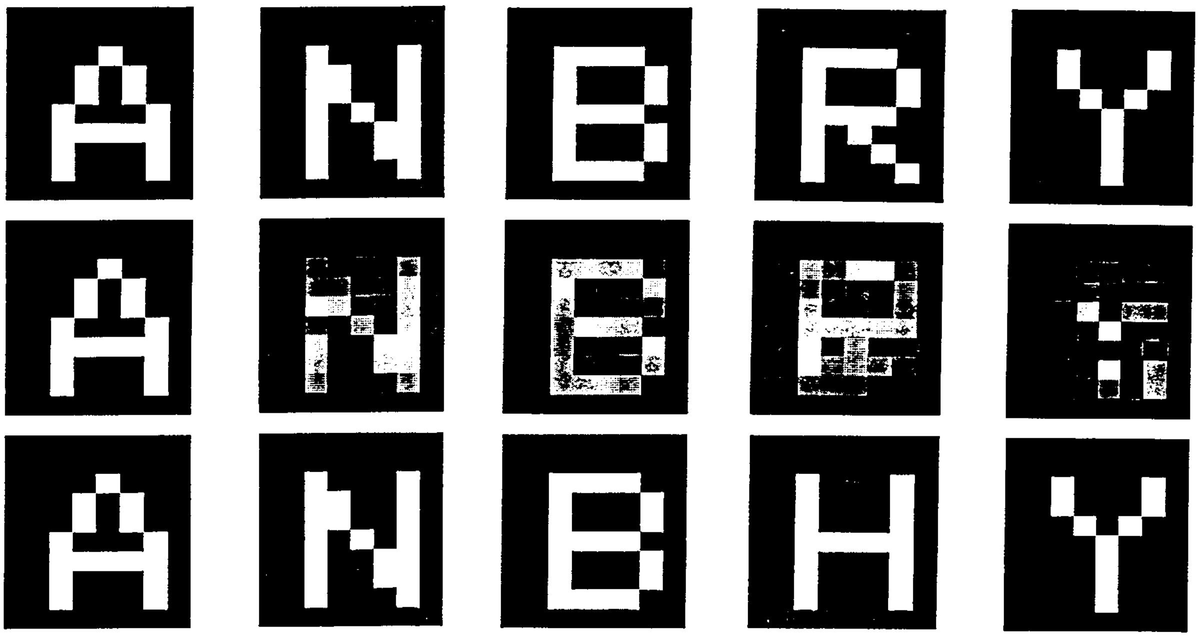


Figure 14: Letter recognition with noise of various standard deviation

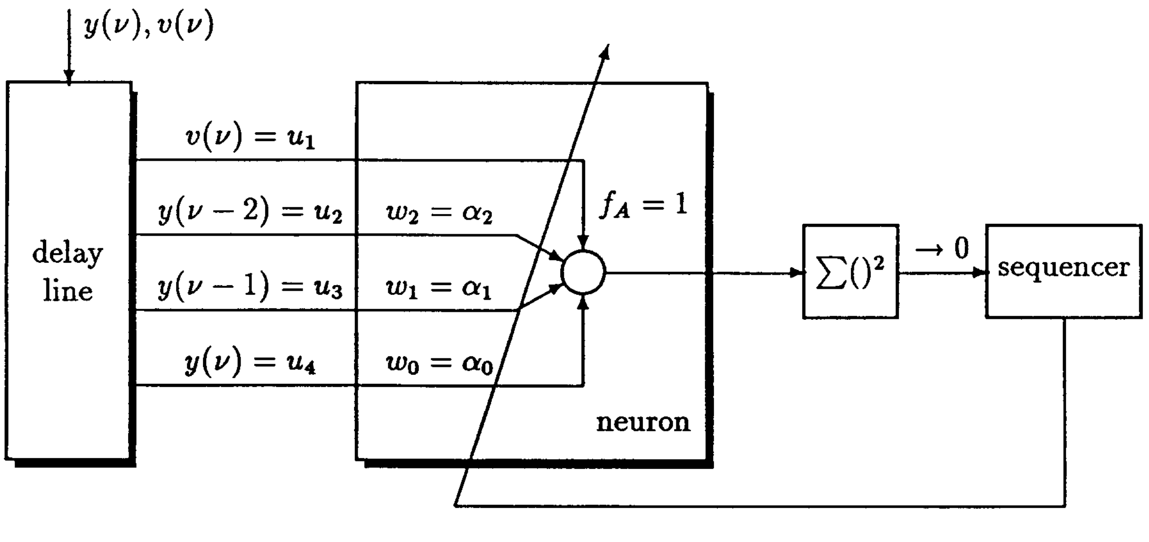


Figure 15: Identifying the difference equation of a discrete-time system

In Fig. 15, the input and outputs at ν are presented and stored in a delay line. Both the input and output signals are used as inputs u_l of the neuronal network. Activation function $f_A = 1$ is used. The sequencer at the neuronal network output demands changes in the weights w_i in order to approach input of the sequencer $\rightarrow 0$.

8 Hebb's Learning with Supervisor

Consider q input vectors $\mathbf{u} \in \mathcal{R}^m$, i.e., $\mathbf{u}_1 \ldots \mathbf{u}_l \ldots \mathbf{u}_q$, concatenated in a matrix $\mathbf{U} \in \mathcal{R}^{m \times q}$. A target (reference) is presented by $\mathbf{y}_{ref}$. The output is $\mathbf{y} \in \mathcal{R}^n$.

Find a weighting matrix $\mathbf{W}$ such that $\mathbf{y}_l = \mathbf{W}\mathbf{u}_l + \mathbf{b}$, $\forall l = 1 \ldots q$ approaches the reference. Hebb's learning rule for the weighting matrix increment is

$$\Delta \mathbf{W} = \beta \mathbf{y}_{ref} \mathbf{u}_l^T - \gamma \mathbf{W} \quad \text{where} \quad \mathbf{W} \in \mathcal{R}^{n \times m} \quad \text{and} \quad \gamma > 0 \ . \tag{68}$$

This computation is repeated for all l form 1 through q each epoch.

What follows is the consideration that both a strong input signal $\mathbf{u}$ and a strong reference should strengthen the corresponding weight. In order to avoid undue increase of $\Delta\mathbf{W}$, a negative term $\gamma\mathbf{W}$ is augmented to guarantee exponential decay, see Eq.(75). The parameter γ is the decay rate.

Usually, the bias $\mathbf{b} = -0,5\,\mathbf{I}_{n\times 1}$ is chosen.

The MATLAB code for Hebb's supervised learning is as follows

```
% Supervised Hebb learning   % gsm.m
Yref=[1 1 1 1 1 0 1];
U =  [1 1 1 1 1 0 0;
      0 0 0 0 0 1 0;
      0 0 0 0 0 0 1];
[n,q]=size(Yref);   [m,q]=size(U);
beta=0.1;           gamma=0.03;
Wo=zeros(n,m);      % initial conditions
W=Wo;
     for ii=1:20
         for l=1:7
             deltaW=beta*Yref(l)*(U(:,l))'-gamma*W;
             W=W+deltaW;
         end
     end
B=-0.5*ones(n,q);
Y=hardlim(W*U+B)
```

Starting with $\mathbf{W} = \mathbf{W}_o$ as zeros, after 20 epochs the result $\mathbf{W}$ suffices. The result equals the target

$$\mathbf{y} = \mathtt{hardlim}\,(\mathbf{W}\mathbf{u}_l + \mathbf{b}) = \mathbf{y}_{ref}\ \forall l \ . \tag{69}$$

The resulting weighting matrix is

$$\mathbf{W} = (2.2749 \quad 0 \quad 0.5135) \ . \tag{70}$$

8.1 Learning Dynamics. PT$_1$ Learning Algorithm

The exponential decay is derived as follows. From Eq.(68) and by $\mathbf{U}_I \triangleq \mathbf{y}\mathbf{u}^T$, one has an increment $\Delta\mathbf{W}$ per step, i.e.,

$$\frac{\Delta \mathbf{W}}{\Delta t} \doteq \dot{\mathbf{W}} = \mathbf{U}_I - \gamma \mathbf{W} \ . \tag{71}$$

Applying Laplace Transform to Eq.(71) and its time dependent variables $\mathbf{W}(t)$ and $\mathbf{U}_I(t)$,

$$\mathcal{L}\{\dot{\mathbf{W}}(t) + \gamma \mathbf{W}(t)\} = \mathbf{U}_I(t)\} \tag{72}$$

$$s\mathbf{W}(s) - \mathbf{W}_o + \gamma\mathbf{W}(s) = \mathbf{U}_I(s) \tag{73}$$

$$\mathbf{W}(s) = \frac{1}{s+\gamma}[\mathbf{U}_I(s) + \mathbf{W}_o] \tag{74}$$

$$\mathbf{W}(t) = \mathcal{L}^{-1}\{\mathbf{W}(s)\} = \mathbf{W}_o e^{-\gamma t} + \int_o^t \mathbf{U}_I(\tau)e^{-\gamma(t-\tau)}d\tau\ . \tag{75}$$

The initial condition decays exponentially. There is a PT$_1$ delay $\frac{1}{s+\gamma}$ from the input $\mathbf{U}_I(s)$ to the output $\mathbf{W}(s)$.

Omitting γ, i.e. $\gamma = 0$, yields an integral behaviour without stabilizing decay

$$\mathbf{W}(t) = \mathbf{W}_o + \int_o^t \mathbf{U}_I(\tau)d\tau\ . \tag{76}$$

8.2 Specific Case of Orthonormal u

In the case of orthonormal input vectors **u**, the desired final matrix $\mathbf{Y}_f$ should match $\mathbf{Y}_{ref}$ when **B**, **W** approach $\mathbf{B}_f$, $\mathbf{W}_f$. Due to orthonormality $\mathbf{U}^T\mathbf{U} = \mathbf{I}_q$

$$\mathbf{Y}_f = \mathbf{Y}_{ref} = \mathbf{W}_f\mathbf{U} + \mathbf{B}_f \tag{77}$$

and postulating

$$\mathbf{W}_f = (\mathbf{Y}_{ref} - \mathbf{B}_f)\mathbf{U}^T\ , \tag{78}$$

one finds by checking $\mathbf{W}_f\mathbf{U} + \mathbf{B}_f$ by combining Eqs.(77) and (78)

$$\mathbf{Y}_f = \mathbf{W}_f\mathbf{U} + \mathbf{B}_f = (\mathbf{Y}_{ref} - \mathbf{B}_f)\mathbf{U}^T\mathbf{U} + \mathbf{B}_f = \mathbf{Y}_{ref}\ . \tag{79}$$

9 Hebb's Unsupervised (Associative) Learning

Unsupervised learning results from learning without a reference (teacher) $\mathbf{Y}_{ref}$. Instead of presenting the reference, the output previously derived is used. The presentation phase simply is characterized by `y = hardlim(W*U(:,1)+b)`. Thus, the learning algorithm is

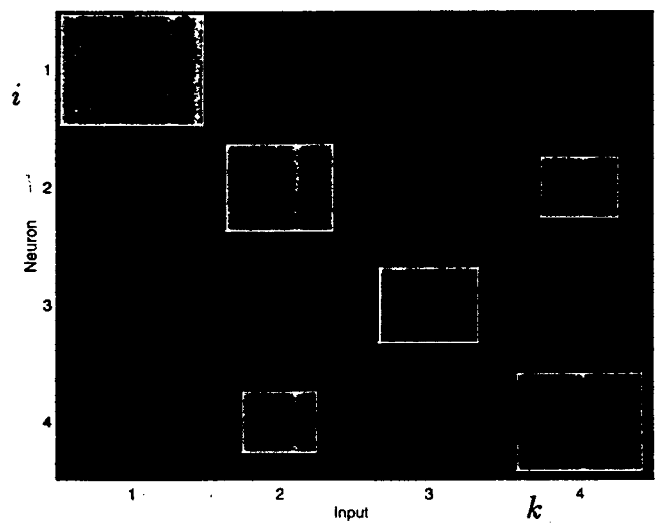


Figure 16: Visualization of the weighting matrix W as the result of unsupervised Hebb's learning

```
% Unsupervised Hebb Learning       % gsn.m
  U=[1 1 1 1 1 0 0 0 0;
     0 0 0 0 0 1 0 1 0;
     0 0 0 0 0 0 1 0 0;
     0 0 0 0 0 1 0 1 1];
[m,q]=size(U); n=4;     beta=0.1;     gamma=0.03;
Wo=eye(n,m);    % identity matrix with dimension n times m
W=Wo;           % initial conditions
b=-0.5*ones(n,1); % n-vector of entries -0.5
   for ii=1:20
       for l=1:q
           y=hardlim(W*U(:,l)+b);
           deltaW=beta*y*(U(:,l))'-gamma*W;
           W=W+deltaW;
       end
       hintonw(W)
       pause2(0.8)
   end
utest=[0;0;0;1];
y=hardlim( W*utest+b )
```

The algorithm is capable of evaluating associative properties in the input signals.

The network has learned that input 4, which is a 1 in `utest`, is associated with neuron 2, which is a 1 because of the two-times-pairing inputs 2 and 4; although `utest` itself is an input (the last in **U**) but only presented once.

The resulting **W** shows that its entries (4,2) and (2,4) have increased reasonably. The command `hintonw(W)` provides a visualization of the 4×4 weighting matrix **W**, separated by a pause of 0.8 seconds. The graphical representation of **W** by `hintonw(W)` enables the reader to watch and observe the learning procedure, see Fig. 16.

10 Instar Learning Algorithm

For comparison purpose only, the supervised and unsupervised Hebbian learning algorithms are repeated

$$\Delta W(i,k) = \beta y_{ref}(i)u(k) - \gamma W(i,k) \tag{80}$$

$$\Delta W(i,k) = \beta y(i)u(k) - \gamma W(i,k) \ , \tag{81}$$

respectively. Slightly different, Instar Learning procedure is defined.

Instar learning (*Grossberg, S., 1982*) obeys the product of output y on the one hand and the difference between input u and weight W on the other hand. The input vectors are normalized before operating, then the bias is set to $b = -0.05$ `ones` $(n,1)$. With the initials $W_o =$ `zeros` (n,m) the algorithm runs as follows

$$\Delta W(i,k) = \beta y(i)[u(k) - W(i,k)] \ . \tag{82}$$

The MATLAB source program is

```
% Instar learning          % gsy.m
U=(  randnr(6,4)  )'       %  4 x 6 matrix of four-element
                           %  normalized random vectors
Yref=[0 0 0 0 1 0];
[m,q]=size(U);      [n,q]=size(Yref);    beta=0.25;
Wo=zeros(n,m);      b=-0.95*ones(n,1);   W=Wo;
for ii=1:20
    for l=1:q
        deltaW=beta*Yref(l)*( ( U(:,l) )'-W);  % 1 x n
        W=W+deltaW;
    end
    hintonw(W)
    pause2(0.4)
end
```

The systems precisely learns those column in **U** for which it is trained with the element 1 in $\mathbf{Y}_{ref}$.

11 Outstar Learning Algorithm

For comparison purpose only, the outstar algorithm is briefly outlined. This algorithm learns a vector output of a neuron layer being informed about the reference and an input. The learning rule is

$$\Delta W(i,k) = \beta[y_{ref}(k) - W(i,k)]u(k) \tag{83}$$

$$\mathbf{y} = f_A(\mathbf{Wu}) \qquad f_A = \texttt{purelin}\ . \tag{84}$$

When the input is set to 1 then the reference will occur. Training only occurs if there is a reasonable input **u**.

12 Self-Organizing (Kohonen) Learning

Consider Eq.(82) of Instar Learning and a specific f_A = `hardlim`, only an output $y = 1$ contributes to changes ΔW, $y = 0$ causes no change. Kohonen algorithm searches for output $y = 1$ and only those are included into the Instar Learning calculus

$$\Delta W(i,k) = \beta[u(k) - W(i,k)] \quad \forall i \ \text{ where } y(i) = 1\ . \tag{85}$$

Kohonen corresponds closely to instar, but by preselecting $y(i) = 1$ computation effort is reduced.

Competitive networks of neurons are capable of detecting associative relations within the input vectors without being instructed by a teacher.

Competive learning is derived from instar learning, see Fig. 17. Based on input vectors normalized in length, i.e., Frobenius (Euler) norm $||\mathbf{u}||_F = 1$, and based on random and normalized rows of the weighting matrix **W**

$$\mathbf{y} = \texttt{compet}(\mathbf{Wu}) \qquad f_A = f_c = \texttt{compet} \tag{86}$$

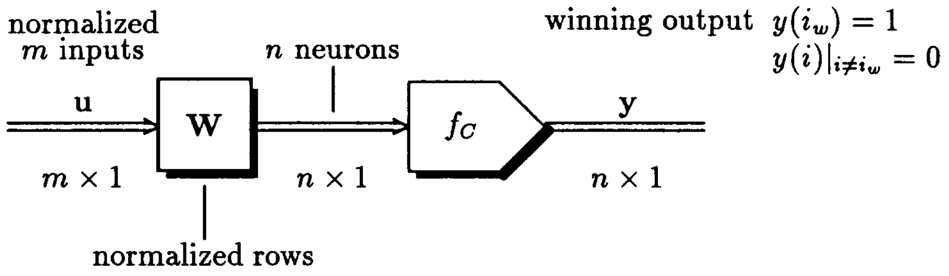


Figure 17: Architecture of a competive network

$$\Delta W(i_w, k) = \beta[u(k) - W(i_w, k)] \quad \text{for winning output } i_w \text{ only} . \tag{87}$$

Due to the dot product of normalized vectors, the product equals to the cosine between the vectors. The maximum cosine determines the winner neuron

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_n^T \end{pmatrix} \tag{88}$$

$$\mathbf{Wu} = \begin{pmatrix} \mathbf{w}_1^T\mathbf{u} \\ \mathbf{w}_2^T\mathbf{u} \\ \vdots \\ \mathbf{w}_n^T\mathbf{u} \end{pmatrix} = \begin{pmatrix} \cos\alpha_1 \\ \cos\alpha_2 \\ \vdots \\ \cos\alpha_n \end{pmatrix} \tag{89}$$

$$f_c(\mathbf{Wu}) = f_A \begin{pmatrix} 0 \\ 0 \\ (\cos\alpha_i)_{\text{winning}} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{90}$$

There is only a single 1 at the i_w position (winning position).

With the help of the toolbox function `trainc`

```
TP=[disp_freq   max_cycle   beta]
W=trainc(Wo,U,TP)
```

and a three-dimensional training parameter vector `TP` the training operation is performed. The matrix `Wo` contains the initials, `U` is the matrix of input vectors **u**. The training parameters in `TP` are the frequency for displaying the current epoch number, the maximum number of training cycles and the learning rate β.

During the learning phase the input vector **u** is fed to all the neurons. There will be a neuron the sensitivity of which is closest to the input, i.e., there is an output node which is activated most. The difference between the input **u** and the weighting vector **w** is minimum and is detected by their difference. The vector-valued difference is treated by any norm, e.g. the Frobenius norm.

This optimum reaction can also be transferred to neurons in some local neighbourhood, denoted feature map, thus effecting their adaptation to the property of the optimum neuron.

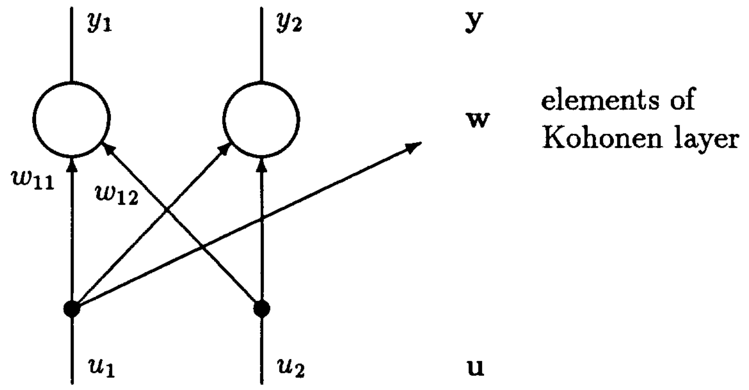


Figure 18: Kohonen layer

If another input is supplied to the Kohonen network for teaching purpose, another group will react optimally and store its specific property. In such a way, by persistent teaching each significant property of the input signal is put into a specified group of neurons. Without supervision the neurons adapt to the input (*Kohonen, T., 1989*).

The processing elements compete at any input $\mathbf{u}$ entering. Which of them has its weight vector $\mathbf{w}_i$ closest to $\mathbf{u}$ (vicinity measured by a certain scalar distance)? The closest element i_w is selected, in order to perform learning. An algorithm can be stated as follows: The winner takes all, i.e., the winning neuron i_w emits a signal $y_{i_w} = 1$, the others $y_{j(\neq i_w)} = 0$, see Fig. 18.

Unlike other learning algorithms, the weight $\mathbf{w}_i$ is to approach $\mathbf{u}$.

A distance metric performance produced by each element of the Kohonen layer is

$$g_i = \|\mathbf{w}_i - \mathbf{u}\|_F \ . \tag{91}$$

The Kohonen learning law results from

$$\mathbf{w}_{i\ new} - \mathbf{w}_{i\ old} = \Delta\mathbf{w}_i = \beta(\mathbf{u} - \mathbf{w}_{i\ old})y_{i_w} \qquad (0 < \beta \leq 1) \ . \tag{92}$$

For the winning element one has $y_{i_w} = 1$ and $\mathbf{w}_{i\ new} = (1-\beta)\mathbf{w}_{i\ old} + \beta\mathbf{u}$. For all the loosing elements $y_{i(\neq i_w)} = 0$ is achieved and the weights are not altered, i.e., $\mathbf{w}_{j\ new} = \mathbf{w}_{j\ old}$. Only the nearest weight vector is attracted.

In the beginning of the training phase, the learning factor β is chosen near 1 in order to guarantee quick training start-up, in advancing the training procedure, β is reduced to a small number (choice of receding learning rate).

Example: Consider four neurons with initial sensitivities $\mathbf{w}_1$ through $\mathbf{w}_4$. Only the closest neuron 2 is changed from $\mathbf{w}_{2\ old}$ to $\mathbf{w}_{2\ new}$, according to Eqs.(91) and (92), see Fig. 19. No additional relation to the neighbouring neurons is taken into consideration. □

Example. Balancing a Rod Using Kohonen Network: Consider the simple problem of balancing a rod, Fig. 20. On the upper end of the rod of length l, a mass of m is attached, the lower end of the rod is linked to a crab (of the mass M) and influenced by the

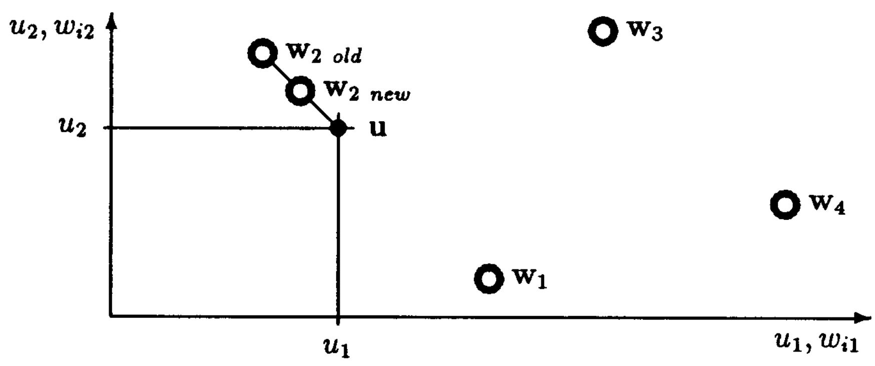


Figure 19: Weight plane and adaptation of the weights of neuron 1 through 4 when a single **u** is supplied and used for learning with $\beta = 0.5$

force f. The deviation of the rod with respect to the vertical direction is denoted as ϑ. The rod is only capable of tilting in a plane of vertical orientation. Assume $l = m = M = 1$.

Out of a classical control design, a controller could stabilize the rod, when determining that the force

$$f_T = 5\sin\vartheta + \dot{\vartheta} \doteq 5\vartheta + \dot{\vartheta} \tag{93}$$

is the actuating variable since the inverted pendulum obeys

$$G_1(s) = \frac{\vartheta(s)}{f_T(s)} = \frac{-\frac{1}{M\,l}}{s^2 - \frac{g(m+M)}{M\,l}} = \tag{94}$$

$$= \frac{-\frac{1}{M\,l}}{(s + \sqrt{\frac{g(m+M)}{M\,l}})(s - \sqrt{\frac{g(m+M)}{M\,l}})} \tag{95}$$

In the equations above, g is the gravitation constant and f_T is the output of the controller (see, e.g., *Weinmann, A., 1995*, Eq.(1.9))

$$K(s) = \frac{f_T(s)}{\vartheta(s)} = 5 + s. \tag{96}$$

Root locus theory requires the zero at -5 in order to stabilize the plant with poles at $\pm\sqrt{\frac{m+M}{M}\frac{g}{l}}$. For the purpose of distinction, this force of a classical controller is denoted as f_T. This force is intended as a force operating as a teaching variable.

Applying the philosophy of neural controllers in the Kohonen model, a system of e.g. 400 neurons is selected. Thus, the input to the neural net is reduced and quantized to 20 discrete values of ϑ and $\dot{\vartheta}$. Each neuron is characterized by a certain ϑ and $\dot{\vartheta}$ and by an output information f in the third dimension (Fig. 20). Only one of the 400 neurons is excited according to that discrete value which is closest to the input ϑ and $\dot{\vartheta}$, i.e., the "winner takes all". The output of each neuron is characterized by the force f_N where N stands for neural net.

The algorithm for the learning phase is then expressed by

$$f_N^{new} = f_N^{old} + \beta(f_T - f_N^{old}) \,, \tag{97}$$

operating at consecutive instants, usually equidistant sampling instants. The variable f_N^{old} is the previous force presented by the neural network. The input of f_T depends on the measured instantaneous value of ϑ and $\dot{\vartheta}$ according to Eq.(93), f_N is the output of that neuron

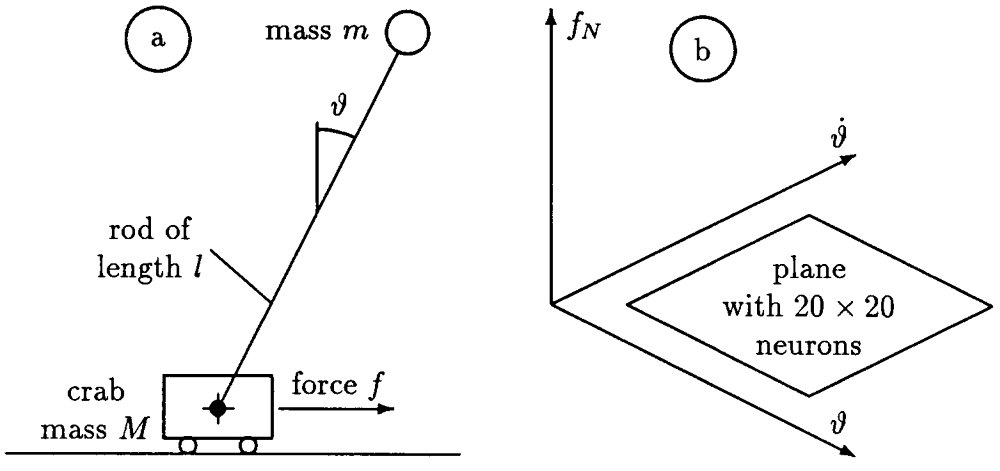


Figure 20: Sketch of the rod (a) and plane of 20 × 20 neurons with force f on the vertical axis (b)

located best, the superscripts ^{old} and ^{new} correspond to the trained value before and after the training step, β is the learning rate determining the training speed. During learning, β is reduced presupposing that the result of learning was improved. The complement of β with 1 can be considered as a confidence variable.

Initially, the values f_N are arbitrary. After sufficient training operations, f_N^{old} will coincide with f_T for any ϑ and $\dot{\vartheta}$, hence $f_T - f_N^{old}$ will not contribute to f_N^{new} any more.

With respect to the insight into the physical relations, f has to be positively high if both ϑ and $\dot{\vartheta}$ are positive, i.e., the inverted pendulum is tilted with ϑ and the angle ϑ is also increasing. For ϑ positive and $\dot{\vartheta}$ negative the inverted pendulum is already moving towards the steady-state upright position, hence f may be small. The analogous condition occurs for the opposite directions of ϑ and $\dot{\vartheta}$.

Figuring out the feature map $f_N(\vartheta, \dot{\vartheta})$, the result is a surface similar to a plane as already can be expected from the linear dependence f_T versus ϑ and $\dot{\vartheta}$ in Eq.(93).

Instead of Eq.(93), a person could operate as a teacher, as well.

Applying unsupervised learning, this could operate as follows: An increment of the difference $f_N^{new} - f_N^{old}$ is only learned if it is evident that an unmistakable improvement of a performance index $\vartheta^2 \Delta t$ was infered from it during the interval Δt.

Example. Travelling salesman problem, solved by Kohonen Network:

A salesman intends to travel a closed-loop tour passing through n towns. The tour should be of minimum distance in sum. Kohonen network philosophy is applied by choosing n neurons at the map of the district of the n towns. The network is initialized with a "circle" of n neurons, or a polygon with n corners.The change in the n corners infered from an iterations step is based on considering one of the n towns. Irrespective of the initial location of the n neurons, the Kohonen iteration will cause an approximation of the neuron locations to that of the towns.

A shortest tour can not be guaranteed but it is very probable. Anyway, it is a closed tour.

Fig. 21 presents a simple example and one iterations step. The initial choice is characterized by a close vicinity of corner 2,0 and town T2 but a long distance 3,0 and T3. The

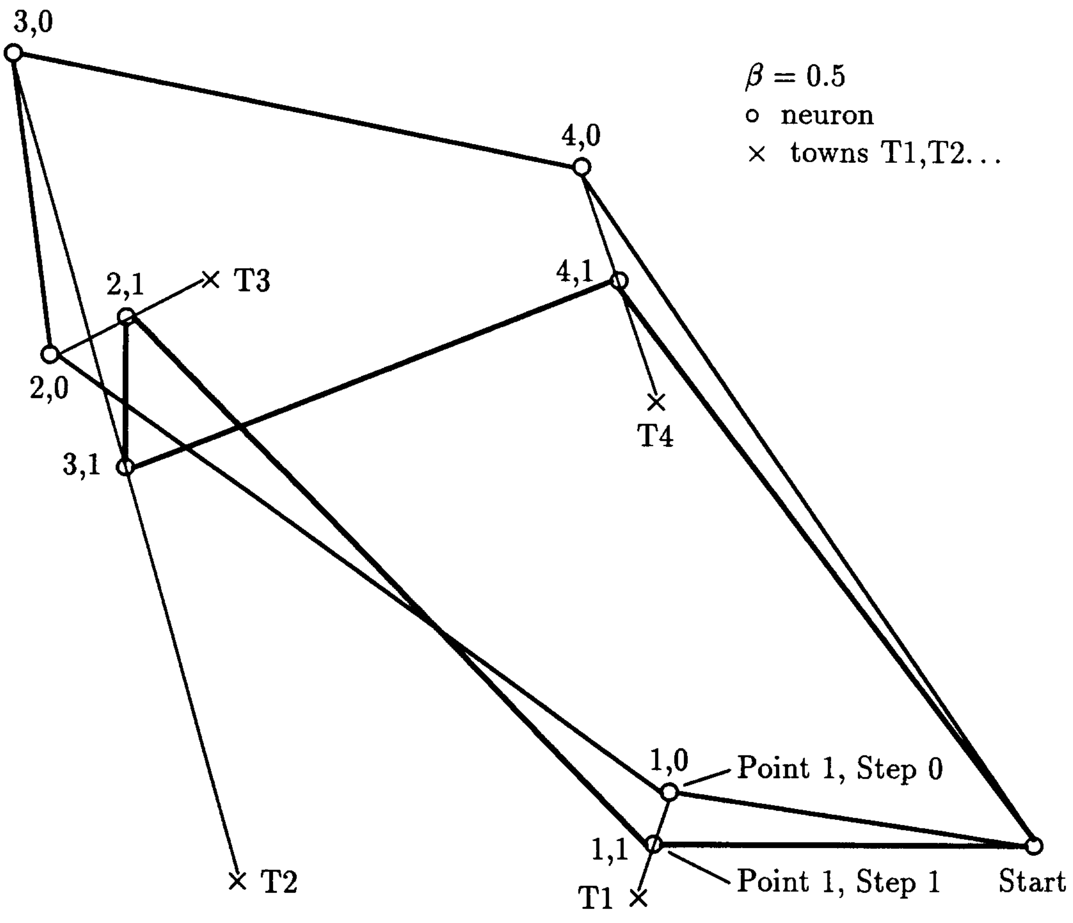


Figure 21: One Kohonen step in a 5 corner tour

distance 3,0 to T3 dominates the sum of the entire tour. Thus, the system should try to replace the connection 3,0 to T3 by the connenction 3,0 to T2 and to force neuron 2,0 to approach T3 which yields a shorter tour without intersection.

Another choice can be triggered by the fact that the neighbouring neurons 2 and 3 are involved in the intersection and should change their enumeration to cancel the intersection.

Classical solution of systematically checking all travel tour opportunities will fail. The number of variation equals $0.5(n-1)!$, which might be a very large number.

13 One-Dimensional and Two-Dimensional Feature Maps

Augment the neurons of a competitive layer network by a group of neighbours. The winning neuron *and* its neighbours are updated during learning.

Example. Two-dimensional feature map: Consider a high number of random input vectors in second dimension and the coordinates selected with uniform distribution density. Choosing a scheme of 25 neurons at random points and operating Kohonen algorithm leads to feature maps shown in Figs. 22. Referring to the uniform distribution, the neural net learns the topology of a geometric pattern of orderly rectangular shape.

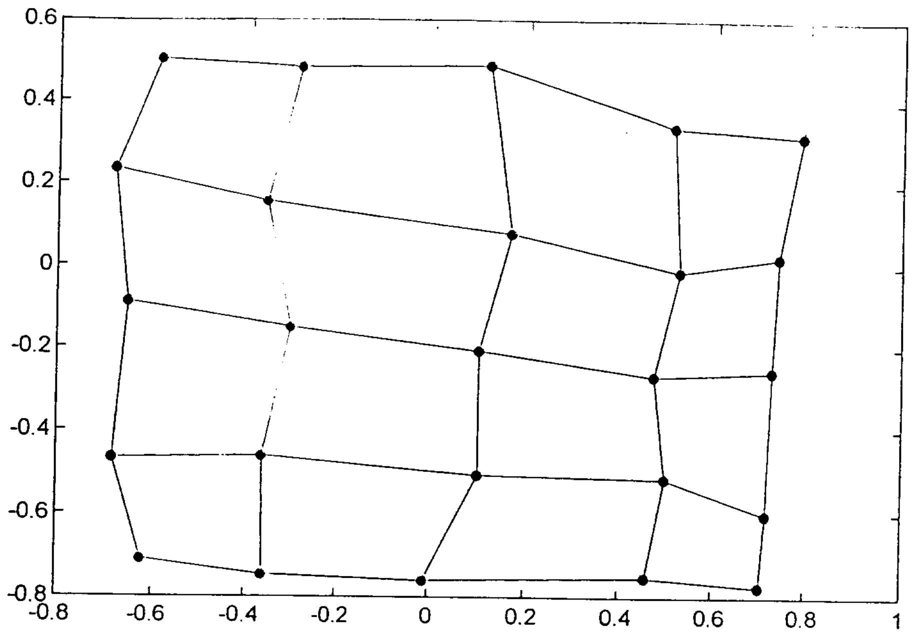


Figure 22: Geometrical pattern for Kohonen learning of uniformly distributed vectors after 800 epochs

14 Hopfield Layer and its Learning

Hopfield network can be considered as a content-addressable and hence associative memory. The basic ideas are:

- Feeding back the output information to the input can contribute to the input information.
- If the input to the memory is incomplete or noisy, the output is incomplete but can augment, nevertheless, the input and improve the input even if it is incomplete.

The network is characterized by a state vector $\mathbf{y} = (y_1 \;\; y_2 \;\; y_3)^T \in \mathcal{R}^n$ where y_i is $+1$ or -1, only. The number of vectors $\mathbf{m}_j$ to be memorized is p. Then, without derivation, the so-called synaptic weight matrix of the Hopfield network is

$$\mathbf{W} \triangleq \sum_{j=1}^{p} \mathbf{m}_j \mathbf{m}_j^T - \frac{p}{n}\mathbf{I}\,. \tag{98}$$

The formula in Eq.(98) is only valid for uncorrelated binary clusters. With the use of a bias vector $\mathbf{b}$ the memorized vectors $\mathbf{m}_j$ satisfy the so-called alignment condition of stability, i.e.,

$$\mathbf{m}_j = \text{sign}\{\mathbf{W}\mathbf{m}_j - \mathbf{b}\} \tag{99}$$

where "sign" is the signum function. For $\gamma > 0$ or < 0, one has sign $\gamma = +1$ or -1 ; γ can also symbolize a vector $\gamma = \mathbf{W}\mathbf{m}_j - \mathbf{b}$; for $\gamma_k = 0$, as a component of the vector γ, the corresponding component m_{jk} of $\mathbf{m}_j$ is not altered.

Any other vector $\mathbf{h}_j$ does not satisfy Eq.(99) if $\mathbf{m}_j$ was substituted by $\mathbf{h}_j$. Thus, $\mathbf{h}_j$ is denoted unstable.

The stable memorized vectors $\mathbf{m}_j$ illustrate the stable corners of a cube of dimension n, the vectors $\mathbf{h}_j$ are arbitrary (and unstable) corners.

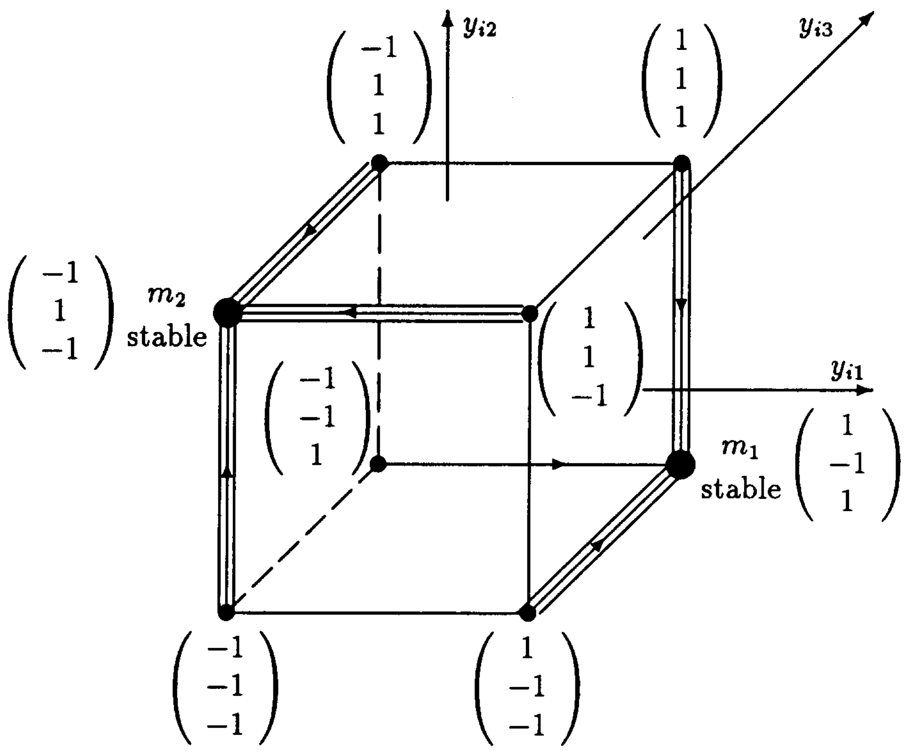


Figure 23: Hopfield network, architectural graph

Applying Eq.(99) to any input information $\mathbf{u}_r$ during the recall phase, the recall algorithm runs as follows ($\mathbf{u}_r$ is considered as an information contaminated with noise or errors):

$$\begin{aligned}
\mathbf{y}^{old} &= \mathbf{u}_r \text{ (input)} \\
\mathbf{y}^{new} &= \text{sign}\,\{\mathbf{W}\mathbf{y}^{old} - \mathbf{b}\} && (100)\\
\text{repeat Eq.(100), i.e.,}\quad \mathbf{y}^{old} &:= \mathbf{y}^{new} && (101)\\
\text{stop if}\quad \mathbf{y}^{new} &= \mathbf{y}^{old} && (102)\\
\mathbf{y} &= \mathbf{y}^{new} \text{ (output)} . && (103)
\end{aligned}$$

Eq.(100) should be calculated randomly as far as components of $\mathbf{y}^{old}$ are concerned.

Example. Hopfield Network: Assume $n = 3,\ p = 2, \mathbf{b} = \mathbf{0}$ and

$$\mathbf{m}_1 = (+1 \quad -1 \quad +1)^T \qquad \mathbf{m}_2 = (-1 \quad +1 \quad -1)^T \tag{104}$$

Learning phase:

$$\text{Eq.(98)} \leadsto \quad \mathbf{W} = \frac{1}{3}\begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix} . \tag{105}$$

The cube is three-dimensional, see Fig.23.

Recall phase: Choosing an arbitrary input $\mathbf{u} = (-1 \quad -1 \quad 1)^T$, then $\mathbf{y}_{old} = \mathbf{u}$ and

$$\mathbf{W}\mathbf{u} = \frac{1}{3}\begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix} \qquad \text{sign}\ \mathbf{W}\mathbf{u} = \begin{pmatrix} +1 \\ \text{“0”} \\ \text{“0”} \end{pmatrix} . \tag{106}$$

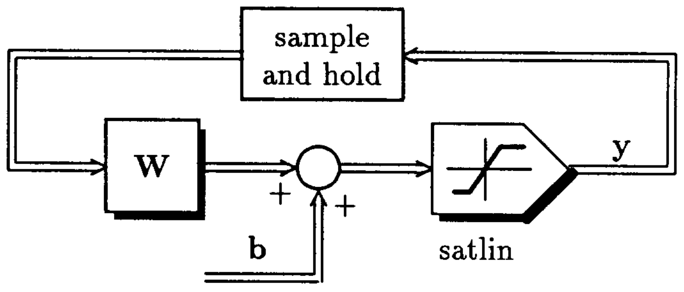


Figure 24: Hopfield network

This requires that u_1 is changed from -1 to $+1$, but u_2 and u_3 keep unchanged

$$\mathbf{y}^{new} = (1 \quad -1 \quad 1)^T \ . \tag{107}$$

This is already the stable output $\mathbf{m}_1$. For $n = 3$, there are 2^3 various $(-1,\ +1)$-combinations in $\mathbf{y}$, corresponding to the corners of the cube in Fig. 23.

The edge vertical above $\mathbf{m}_1, \forall\ a : -1 < a < 1$, leads to

$$\frac{1}{3}\begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ a \\ 1 \end{pmatrix} = \frac{1}{3}\begin{pmatrix} -2a+2 \\ -4 \\ 2-2a \end{pmatrix} = \begin{pmatrix} +1 \\ -1 \\ +1 \end{pmatrix} \ . \tag{108}$$

Hence, for any a, the point $\mathbf{m}_1$ is reached. In the recall phase, the information has to be sufficient only in such a way that the edge is allocated. □

The architecture of a Hopfield network can also be portrayed in a different way, see Fig. 24. The output of the sample and hold in the feedback is weighted by a square weighting matrix $\mathbf{W} \in \mathcal{R}^{n \times m}$ where $m \equiv n$ and augmented by the bias vector $\mathbf{b} \in \mathcal{R}^n$. The result is fed to a nonlinear element with `satlin` characteristic. The resulting output $\mathbf{y}$ equals its input in the linear part of the characteristic, outside the output $\mathbf{y}$ is restricted to $+1$ and -1, respectively,

$$\mathbf{y}[\nu+1)T] = \mathtt{satlin}[\mathbf{W}\mathbf{y}(\nu T) + \mathbf{b}] \ . \tag{109}$$

The sampling period is T.

The operation is subdivided into the following steps

- System is presented several input vectors as initial conditions.
- Then, the network output is fed back to operate as an input.
- The former process is repeated until an equilibrium point is reached. In most cases, the system is stable at the equilibrium point.
- The equilibrium operates as a reference vector.
- The system is presented arbitrary input vectors (as columns of a matrix $\mathbf{U}$). After sufficient training cycles, the network equals to those equilibrium points which are located close to the input.

15 Training Algorithms Compared

An overview of various networks and the associated learning algorithms for the bias and weighting matrix during the training phase is presented in the table below.

Network	f_A	Bias	Weighting matrix increment $\Delta W(i,k)$
Perceptron	hardlim	$\Delta b(i) = y_{ref}(i) - y(i)$	$[y_{ref}(i) - y(i)]u(k)$
Widrow-Hoff	purelin	$\Delta b(i) = \beta[y_{ref}(i) - y(i)]$	$\beta[y_{ref}(i) - y(i)]u(k)$
Backpropagation	log-sigmoid tan-sigmoid purelin	$\Delta b(i) = \beta \Delta y(i)$	$\beta \Delta y(i) u(k)$
Hebb with supervisor	hardlim	$\mathbf{b} = -0.5\mathbf{I}_{n\times 1}$	$\beta y_{ref}(i)u(k) - \gamma W(i,k)$
Hebb unsupervised	hardlim	$\mathbf{b} = 0.5\mathbf{I}_{n\times 1}$	$\beta y(i)u(k) - \gamma W(i,k)$
Instar	hardlim	$\mathbf{b} = -0.95\mathbf{I}_{n\times 1}$	$\beta y(i)[u(j) - W(i,j)]$
Outstar	purelin	0	$\beta[y_{ref}(i) - W(i,k)]u(k)$
Kohonen	compet	0	$\beta[u(i_w) - W(i_{w,j})]\ \forall i = i_w$ where $y(i_w) = 1$
Hopfield	satlin		see Eq.(100)

16 Alternative Algorithms

16.1 Radial Basis Functions

Radial basis functions are feedforward networks with only one hidden layer. Each neuron of this layer is characterized by a radial symmetric activating function. The number of neurons n corresponds to the number of training samples. The activating cluster is characterized by

$$y = \sum_{i=1}^{n} h_i \, e^{-\gamma \|\mathbf{u}-\mathbf{u}_i\|_F} \qquad \mathbf{u} \in \mathcal{R}^n \tag{110}$$

where the $\mathbf{u}_i$'s are the given and supporting input samples. The setting γ determines the shape of the plane of the radial basis function, the h_i's are unknown. The network produces a transformation from $\mathcal{R}^n$ to $\mathcal{R}$ by portraying $\mathbf{u}_l$ to y. During the training phase for given y_l and $\mathbf{u}_l$

$$y = y_l \quad \rightsquigarrow \quad y_l = h_l + \sum_{i\ (i \neq l)}^{n} h_i \, e^{-\gamma \|\mathbf{u}_l-\mathbf{u}_i\|_F} \quad \forall\ l = 1, 2 \ldots n\ . \tag{111}$$

The coefficients h_l are achieved in a direct (non-iterative) way, since Eq.(111) is a system with n unknowns in n linear equations. For a given data more than n, the pseudo inverse formula of Eq.(112) is adapted with respect to h_i. The exponential activity function provides remarkable contribution only if $\mathbf{u}$ is close to $\mathbf{u}_i$ which can be considered as an advantage in order to guarantee good approximation properties.

$$\mathbf{W}_{\text{opt}} = \mathbf{Y}_{ref}\mathbf{U}^{\sharp}\ . \tag{112}$$

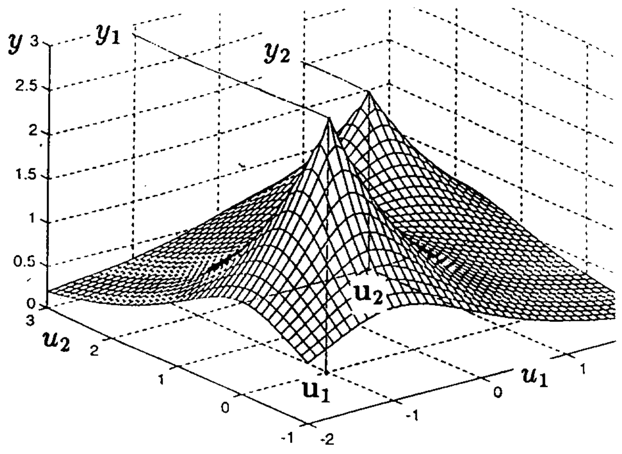


Figure 25: Radial basis function network

Example. Radial basis functions: Select $n = 2$:

$$\mathrm{u}_l: \quad \mathrm{u}_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad \mathrm{u}_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \qquad \gamma = 1 \ . \tag{113}$$

Presentation phase:

$$\mathrm{u}_l: \quad l = 1,\ \mathrm{u}_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad y_1 = 2.8 \ ; \qquad l = 2,\ \mathrm{u}_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad y_2 = 2 \ . \tag{114}$$

Calculation for the training phase:

$$l = 1: \quad y_1 = 2.8 = h_1 + h_2 e^{-\|\left(\begin{smallmatrix} -1 \\ 0 \end{smallmatrix}\right) - \left(\begin{smallmatrix} 1 \\ 2 \end{smallmatrix}\right)\|_F} = h_1 + h_2 \cdot 0.059 \ . \tag{115}$$

Similarly, $y_2 = 2 = h_2 + h_1 \cdot 0.059$. The result is $h_1 = 2.69$ and $h_2 = 1.84$. The resulting plane $y(\mathrm{u})$ is depicted in Fig. 25. The training samples correspond to the peaks of the plane of the radial basis functions. □

16.2 Genetic Algorithms

Genetic algorithms are based on transposing parameters of technical systems into clusters of genes. Starting an optimization problem, a group of parameter constellations is selected and defined by genes. This corresponds to a group of individua. The individua are subject to an evolution by computer simulation. Natural evolution is simulated by the transfer of properties from generation to generation, by natural heredity from parents to children. Both the natural selection and the natural mutation are simulated. Each individuum is characterized by a fitness function. Selection is based on this fitness function, in order to find the probability of survival and to decide which individuum should survive. The principle is "survival of the fittest" (see, e.g., *Grünberger, T., 1995*).

17 Applications

17.1 Optimization

For optimization purposes an arrangement of Fig. 27 can be applied. First, an optimum actuating variable u_{opt} has to be found by using an artificial neural network of the process

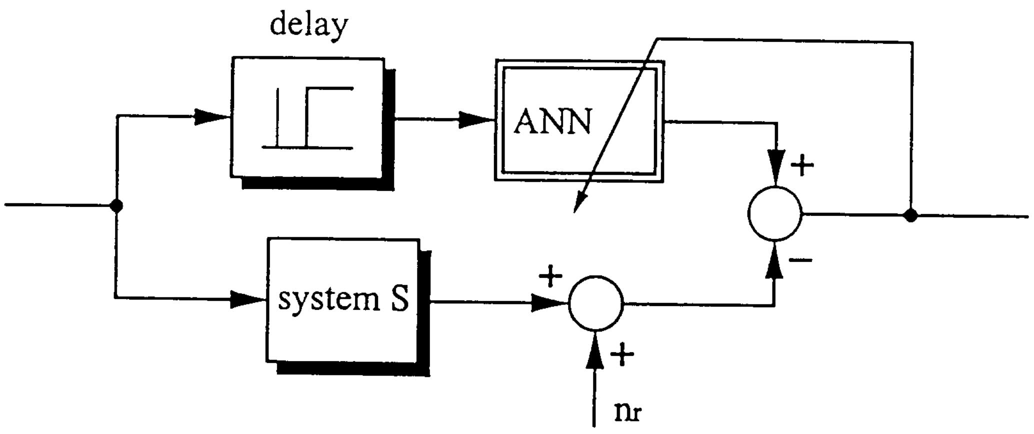


Figure 26: Predictive System

G, see Fig. 27a. This is performed by time-lapse in most cases. In parallel, the deviation $e(k)$ is evaluated which actually results from G being excited by u_{opt}, see Fig. 27b. This value $e(k)$ is utilized as the error $e(k-1)$ for the next optimization step.

Referring to Fig. 27a, based on the model of G in ANN_G and on the deviation $e(k-1)$ of the past, u_{opt} is calculated in order to optimally obey y_{ref} at presented. Fig. 27b points out, when ANN_G is used to identify G. During this identification, ANN_G approaches a memory state that best approximates G in the subsequent interval.

17.2 Predictive Control

Consider Fig. 26 and set the delay equal to αT where T is the sampling interval characterizing two consecutive computations and α is a positive integer. According to the fact that the input of the artificial neural network is delayed artificially but the output is compared to the real system output, the artificial neural network dynamics will try to compensate for the delay as long as the system is not confronted with unpredictable situations. Modern applications are *Wang, Y., et. al., 1998; Kührer, M., and Reinisch, W., 1998; Luo, F.L., et al., 1998; Reinisch, W., et al., 1998.*

When transferring the artificial neural network parameters to an additional network ANN_P, which is not depicted in Fig. 26, excited by the undelayed input of the system S, then the output of ANN_P is a α-step-ahead predictor of S.

17.3 Further Applications Areas

In the following fields, neural network applications have become important:

- Fault detection (*Schöneburg, E., 1992; Boehme, T.J., and Fletcher, I., 1998*),
- Guidance and transportation systems (*Maier, K.D., et al., 1998; Forio, L., and Mussone, L., 1998*)
- Noise suppression in signal processing (*Sincak, P., et al., 1998*)
- Market forecasting (*Finzi, G., et al., 1998*)
- Stabilization (*Dudnikov, E.E., and Rybashov, M.V., 1998*)
- Self-Organizing Maps (*Haese, K., 1998*)

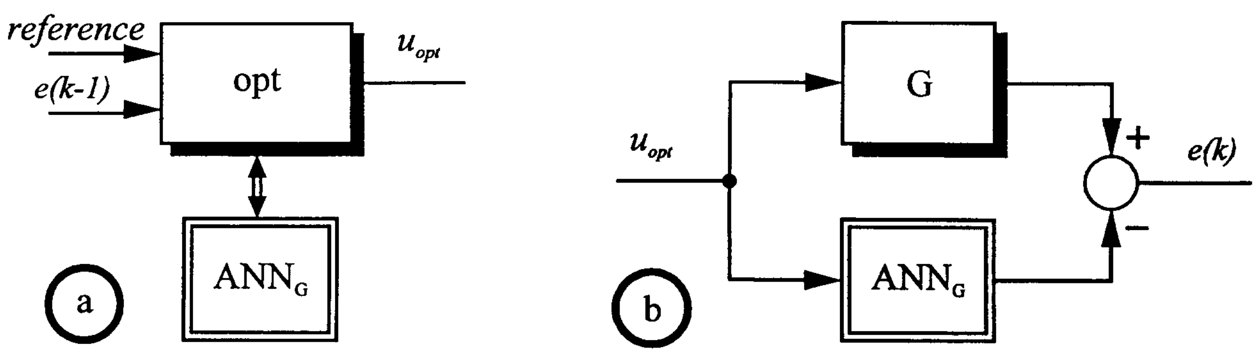


Figure 27: Optimization carried out by an artificial neural network ANN

- Pattern Recognition (*Kröner, S., and Burkhardt, H., 1998; Seager, J., and Marsh, B., 1998*)
- Neuro-Fuzzy Systems (*Bothe, H.H., 1998; Frattale Mascioli, F.M., et al., 1998*)
- Identification and Verification (*Haese, K., and Meier, C., 1998; Nagano, T., and Hirai, Y., 1998*)
- Speech recognition and image compression (*Rabi, G., and Si Wei Lu, 1998; Cho Y-H., 1998*)
- Robust Control (*Nakanishi, H., and Inoue, K., 1998*).

18 Special Features of Artificial Neural Networks

Above all, artificial neural networks are characterized by the following properties:

- Application is favourable if the process is not known sufficiently or if the mathematical model is incomplete as far as the nonlinear transformation function, the coefficients or order of the differential equation are concerned.
- Many input and output data of the process are available in order to train the artificial neural network.
- Evaluation by an artificial neural network is performed by highly parallel computation, hence the results, e.g. for image detection or pattern recognition, are found out promptly even if each computational operation is relatively slow.
- Knowledge about the process need not be put into formulas when initiating the system, e.g., pattern recognition and selection becomes feasible. On the other hand, it is difficult to insert preknowledge by appropriate formulas.
- Combination with fuzzy control is advantageous, e.g., learning or improving the membership functions via an artificial neural network, or demonstrating and illustrating the artificial neural network results by fuzzy methods (via membership functions or inference) (*Preuß, H.-P., und Tresp, V., 1994*).

- Increase of knowledge can only be performed by learning. The analysis of the knowledge obtained is hardly possible. Deductions are difficult to be drawn.
- Learning may turn out as slow. By means of particular strategies, e.g., momentum terms, convergence and learning can be accelerated.
- Danger of passing over the limits gained by the training process (risk of undue extrapolation).
- Danger of overtraining: An overtraining effect arises by too extensive training. Then, the network happens to learn details of the input data which are not essential, e.g., measurement or process noise. Overtraining can be detected by applying test data to the artificial neural network during or after the training phase. Training is no more useful and should be interrupted before the sum of squares errors referring to the test data might increase as a consequence of more training. More neurons support the overtraining effect.

19 Conclusion

Algorithms characterizing the basic types of artificial neural networks have been presented, pointing out the advantages and computational facilites for simulation purposes.

The algorithms are the basis of several software tools and strategies combining the real-world process and the artificial neural network.

The particular advantages and the individual characteristics concerning the application of artificial neural networks are overviewed in order to estimate or predict the opportunities of exceptionally applying artificial neural networks in various fields.

References

Boehme, T.J., and Fletcher, I., 1998, Sensor failure detection and signal reconstruction using autoassociative neural networks, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 220-225

Bothe, H.H., 1998, A tutorial on neuro-fuzzy methods, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 43-60

Cho Y.H., 1998, An efficient compression of image data using neural networks of hybrid learning algorithm, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp.798-801

Demuth, H., and Beale, M., 1992, Neural Network Toolbox for Use with MATLAB, User's Guide (The MATHWORKS, Natick)

Dudnikov, E. E., and Rybashov, M. V., 1998, Stabilization of single-layer neural network with feedbacks, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 954-959

Finzi, G., et al., 1998, Real-time ozone episode forecast: a comparison between neural network and grey box models, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 854-860

Florio, L., and Mussone, L., 1998, Applications of feedforward neural networks to transportation research, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 820-826

Frattale Mascioli, F.M., et al., 1998, Approximation with noisy training data using FBF neural networks, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 900-907

Grünberger, T., 1995, Optimierung neuronaler Regler mit Hilfe genetischer Algorithmen, *e&i (Elektrotechnik und Informationstechnik)* **112**, H. 7/8, S. 338-344

Haese, K., 1998, Fast self-growing and self-organizing feature map using automatic learning parameters, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 123-129

Haese, K., and Meier, C., 1998, Object identification on airports using radial basis function networks, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 298-303

Hafner, S., Geiger, H., Kreßel, U., 1992, Anwendungsstand Künstlicher Neuronaler Netze in der Automatisierungstechnik, Teil 1, Einführung, *Automatisierungstechnische Praxis* **34**, S. 592-599

Hebb, D., 1949, The Organization of Behavior (Wiley, New York)

Hecht-Nielsen, R., 1989, Neurocomputing (Addison-Wesley, Reading)

Hunt, K.J., Sbarbaro, D., Zbikowski, R., and Gawthrop, P.J., 1992, Neural Networks for Control Systems — A Survey, *Automatica* **28**, pp. 1083 - 1112

Kohonen, T., 1989, Self-Organization and Associative Memory (Springer-Verlag, Berlin)

Kröner, S., and Burkhardt, H., 1998, A structured neural network for shift and rotation invariant pattern recognition, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 476-482

Kührer, M., and Reinisch, W., 1998, Forecasting financial markets utilizing artificial neural networks and genetic algorithms, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 347-350

Lippmann, R.P., 1987, An introduction to computing with neural nets, *IEEE ASSP Mag.*, pp. 4-22

Luo, F. L., et al., 1998, Analog neural networks for linear predictive coding, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp.451-455

Meier, K.D., 1998, Controlling one-legged dynamic movement with MLPs, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 784-790

Miller, W.T., Sutton, R.S., and Werbos, P.J., 1990, Neural Networks for Control (MIT Press, Cambridge Mass.)

Mistry, S.I., and Nair, S.S., 1994, Identification and control experiments using neural designs, *IEEE Control Systems* **14**, pp. 48-57

Nagano, T., and Hirai, Y., 1998, Personal verification with palm print and hand shape by utilizing neural network techniques, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 398-404

Nakashi, H., and Inoue, K., 1998, Design methods of robust feedback controller by use of neural networks, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 731-743

Narendra, K.S., and Parthasharathy, K., 1990, Identification and control of dynamical systems using neural networks, *IEEE Trans. on Neural Networks* **1**, pp. 4-27

Preuß, H.-P., und Tresp, V., 1994, Neuro-Fuzzy, *Automatisierungstechnische Praxis* **36**, S. 10-24

Rabi, G., and Lu, S.W., 1998, Automatic lipreading using neural networks, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 391-397

Reinisch, W., and Roche, G., 1998, Prediction of the daily tax declaration submissions by neural networks, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 811-815

Ritter, H., Schulten, K., und Marinez, T., 1989, Eine Einführung in die Neuroinformatik (Springer, Berlin)

Schöneburg, E., 1992, Anwendungsstand Künstlicher Neuronaler Netze in der Automatisierungstechnik, Teil 3: Diagnose mit Neuronalen Netzen, *Automatisierungstechnische Praxis* **35**, S. 161-166

Seager, J., and Marsh, B., 1998, Electronic fruit grading using neural networks for real time pattern recognition, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 280-286

Sincak, P., et al., 1998, Experience with voting neural networks for multi-spectral image classification, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 226-232

Wang, Y., et al., 1998, A distributed predictive expert system, *Proc. Int. ICSC/IFAC Symposium on Neural Computation, Vienna*, pp. 848-853

Weinmann, A., 1991, Uncertain Models and Robust Control (Springer, New York and Vienna)

Widrow, B., and Hoff, M.E., 1960, Adaptive switching circuits, *IRE Wescon Convention Record*, pp. 96-104

Zell, A., 1994, Simulation Neuronaler Netze (Addison-Wesley Deutschland)

1998 American Control Conference ACC'98

Philadelphia, USA

24.-26.6.1998

H. Peter Jörgl

Die vom *American Automatic Control Council* (AACC), der *United States National Member Organization* der IFAC, organisierte American Control Conference 1998 (ACC'98) fand vom 24. bis 26.6.1998 im Adam's Mark Hotel in Philadelphia statt. Die IFAC fungierte dabei als *cooperating organization* der ACC. Das technische Programm wurde in 3 Tagen in jweils 16 parallenen Sektionen für *contibited papers* und 1 Sektion für 5 *Tutorial Sessions* abgewickelt. Insgesamt wurden 887 Beiträge präsentiert, wovon ca. ein Drittel von Nichtamerikanern stammte. Dies bedeutet eine doch erstaunlich starke Internationalität dieser ursprünglich rein US-amerikanischen Konferenz. An jedem der 3 Tage begann das Programm mit einer Plenarsitzung. 8 *tutorial workshops* vervollständigten das Programm, wobei 6 vor der eigentlichen Konferenz und 2 nach deren Beendingung stattfanden.

Das technische Programm reflektierte das schnelle Wachstum und die immense Breite der Regelungstechnik im weitesten Sinn. Es gelang den Organisatoren jedoch eine gute Ausgewogenheit zwischen Theorie und Anwendung im Programm sicherzustellen. Den industriellen bzw. industrienahen Anwendungsbeiträgen wurde durch die Einführung von 5 über alle 3 Tage laufende *special tracks* ein genügend großer Rahmen eingeräumt. Die Tutorial Workshops begannen jeweils mit einem einstündigen Einführungsreferat und wurden durch 4 *state of the art* Präsentationen industrieller Anwendungen vervollständigt. Dadurch sollten Konferenzteilnehmer aus der Industrie ganz speziell angesprochen werden.

Die Plenarvorträge befaßten sich ebenfalls mit sehr anwendungsnahen Themen. Für den Berichterstatter von besonderem Interesse war dabei der von Prof.J.S. Shamma gehaltene Vortrag zum Thema „Gain scheduling", einer bereits seit vielen Jahren angewandten Entwurfsmethode für nichtlineare Regelungen. Der Vortragende gab einen Überblick über die systemtheoretischen Grundlagen dieser oft nur intuitiv angewandten, sehr populären Methode. Hervorzuheben ist auch der Plenarvortrag von Dr.B.Ogunnaike zum Thema „Controlling Industrial Chemical Processes", in dem es ihm in hervorragender Weise gelang, die Verbindungen zwischen akademischer Forschung und deren Umsetzung in der Praxis darzustellen.

Die Sitzungen für *Contibuted Papers* waren, wie nicht anders zu erwarten, auch bei dieser Konferenz in der Mehrheit klassischen regelungstechnischen Themenkreisen wie robuste, adaptive und prädiktive Regelung, Fuzzy und neuronale Regelung, nichtlineare Regelung, Identifikation, stochastische Systeme etc. gewidmet. Auf der Anwendungsseite standen Roboter, Verkehrssysteme sowie die Prozeßindustrie im Vordergrund. Hervorzuheben sind zwei Sitzung, die sich mit *Discrete Event Systems* und *Hybrid Systems* befaßten, zwei Themen, denen in der Zukunft sehr viel Aufmerksamkeit geschenkt werden dürfte. Auf dem Gebiet der regelungstechnischen Ausbildung erscheint dem Berichterstatter eine Sitzung von besonderer Bedeutung, nämlich jene zum Thema *Control Education on the Web*.

Im Rahmen der ACC'98 fand auch das jährlich abzuhaltene Meeting des IFAC Technical Committee on Control Education (EDCOM) statt. Der Berichterstatter hatte die Ehre, in Vertretung des verhinderten EDCOM-Chairman Prof.K.H.Fasol als Chairman dieses Meeting zu leiten.

1st IFAC Workshop on „Intelligent Assembly and Disassembly – IAD'98“

May 21 – 23, 1998
Bled, Slovenia

This first Workshop in this rapidly growing field was organized by the „Laboratory for Handling and Assembly“, Faculty of Mechanical Engineering, University Ljubljana, in cooperation with the Institute of „Handling Devices and Robotics“, Vienna University of Technology. The 50 participants had the possibility to attend 3 Plenary sessions, 9 Technical sessions with 28 papers and a round table discussion.

In the first plenary session Feldmann et al. described innovative disassembly strategies based on flexible partial destructive tools. After an analysis of the characteristics of disassembly an approach for computer aided disassembly planning as well as some innovative tools were presented. D. Noe tried to give an overview on sensors necessary for intelligent assembly and disassembly. A special field of disassembly – disassembly of electronic equipment – was discussed by Kopacek et al. in their survey paper.

The technical papers arranged in 9 sessions covered the whole field of assembly and disassembly. Starting with planning systems and methods via Petri Nets, Fuzzy and Neural Networks, Logistics, Robot Vision until Components and Systems.

The main results also from the round table were: Assembly is an established field and nearly all problems can be solved – implementation is only a question of economics. Disassembly is a rapidely growing field of automation with several not yet solved problems.

The Workshop was extremely well organized and the atmosphere of Bled stimulated the „IAD-family“ to exchange ideas and discuss problems. Because of the importance of this subject it was decided to have the next event on this topic in the year 2000.

D. Noe
NOC Chairperson

P. Kopacek
IPC chairman

IFAC Conference on Supplemental Ways for Improving International Stability SWIIS'98

May 14 – 16, 1998
Sinaia, Romania

SWIIS'98 was held in Sinaia. It was the 7th in the triennial cycle and mainly sponsored by the TC on SWIIS.

38 participants from 14 countries attended 2 survey papers and 21 technical papers. The 1st survey paper presented by F. Kile (USA) discussed the political and social factors which have driven leadership in the past. The paper suggested that leadership in the future must include sustainability as one of the driving factors.

The Democratic People Republic of (north) Korea concentrated on national self-sufficiency for five decades. Its quest for nuclear energy in the 1980s led to a conflict with the United Nations – now slowly resolving itself. Pointed out by J. Richardson in the 2nd survey paper.

The 21 technical papers were arranged under topics like:
- Methodologies
- Stability
- Modelling and
- International Policy Cooperation.

Methodology papers dealt with decision making in a reconfigered multipolar world. Issues included like labour decision making and analysis of social interactions. Stability was scheduled under the rubrics of economics, cooperation and social restructions following the solution of the centraly planned economics.

Modelling sessions were chaired by J. Holubiec (PL) and focussed on cumulative games, prediction techniques and as a new topic "Virtual society". The policy cooperation was described in terms of makroeconomic modells as well as linear programming.

The conference was very well organized by the Romanian NMO and stimulated intensive and sucessful discussions. The next event will be in 2001.

I. Dumitrache
NOC chairman

K. Kile
IPC chairman

9th Symposium Information and Control in Manufacturing

24. – 26. 6. 1998
Nancy, Frankreich

Das Hauptereignis des TC „Advanced Manufacturing Technologies – AMT“ fand diesmal in Nancy (1. und 3. Tag) und Metz (2. Tag) statt. Zum Unterschied von anderen IFAC Veranstaltungen legten die Organisatoren größten Wert auf verstärkte Industriepräsenz. Dies wurde dadurch erreicht, daß eine neue Struktur gewählt wurde. Statt der bisher üblichen Übersichtsvorträge und Round Table Diskussionen waren die 6 Plenary Sessions so organisiert, daß zu einem Themenbereich zunächst 5 – 6 Vorträge gehalten wurden, und anschließend Fragen an die Vortragenden sowie zusätzliche Fachleute gestellt werden konnten.

Die Themen dieser Plenary Sessions:
- Advanced Automation Engineering
- Engineering Technologies for Advanced Manufacturing
- Information Technologies for Integration in Manufacturing
- Intelligent Manufacturing & Process Systems Engineering
- Management of Advanced Industrial Systems
- Industrial Safety, Dependability and Quality

spiegeln die Entwicklungsrichtungen dieses Fachgebietes wieder. Die ungefähr 150 gehaltenen technischen Vorträge waren ebenfalls diesen Themenkreisen zuzuordnen.

Als Entwicklungsrichtungen zeichnen sich eine verstärkte Integration; autonome intelligente „Agenten“; die Erstellung von Modellen für das dynamische Verhalten von Produktionsausrichtungen ab. Diese Tendenzen können als logische Weiterführung der „ims“ Test-cases aus dem Jahre 1993 gesehen werden.

Zusammenfassend kann festgestellt werden, daß sich die neue Struktur bewährt hat – allerdings war die Dauer der Plenary Session (2,5 Stunden ohne Pause) etwas zu lang. Durch Einwerbung von Sponsorgeldern war es den Veranstaltern möglich den 250 Teilnehmern ein exzellentes Begleitprogramm zu bieten.

Gemäß dem üblichen drei Jahreszyklus findet INCOM 2001 im September 2001 in Wien statt.

Peter Kopacek

Taschenbuch Versuchsplanung
Produkte und Prozesse optimieren

Wilhelm Kleppmann

Praxisreihe Qualitätswissen, herausgegeben von Franz J. Brunner

283 Seiten, broschiert; ISBN 3-446-19271-9
Carl Hanser Verlag 1998

Produkte werden entwickelt, verbessert, produziert, geprüft, getestet und später auch repariert. Diese Tätigkeiten sind oft sehr komplex, weil sehr viele Anforderungen und Einflußfaktoren eine Rolle spielen und aufeinander abgestimmt werden müssen. Für die Ingenieure in Entwicklung, Konstruktion, Fertigung und Qualitätssicherung ist dies im Rahmen der gegebenen Termine und Kosten nicht immer einfach, manchmal unmöglich. Ein probates Mittel diesem Problem beizukommen, ist der Versuch. Der Autor bietet erstmals einen anwendungsorientierten Überblick zu diesem Thema, der auch Ingenieure überzeugen wird, die bisher bei dem Begriff Versuch eher an statistische Zahlenakrobatik dachten.

Welche Probleme können durch gezielte Versuche am besten gelöst werden? Wie legt man einen Versuch an, damit die Ergebnisse wirklich aussagekräftig sind? Auf diese Fragen geht das Buch mit Hilfe von vielen Beispielen ein, die das Thema greifbar und umsetzbar werden lassen. Oft müssen mehrere Zielgrößen unter einen Hut gebracht werden, aber wenn man die eine verbessert, verschlechtert sich die andere. Eine Lösung ist, zunächst die Zusammenhänge einzeln genau zu bestimmen und mit der Wunschfunktion einen optimalen Kompromiß zu suchen. Weitere Beispiele sind: Mit dem „Multi-Vari-Bild“ lassen sich die Ursachen für die Streuung von Produkteigenschaften ermitteln; ein „paarweiser Vergleich“ hilft hartnäckige Fehler aufzuspüren; die Ausbeute bestimmter Fertigungsabläufe kann durch „Variablenvergleich“ erhöht werden und vieles mehr.

Das Buch kann sowohl in der Praxis stehenden Ingenieuren sowie Studierenden wärmstens empfohlen werden.

P. Kopacek

Das virtuelle Produkt

Management der CAD-Technik

G. Spur, F-L. Krause

738 Seiten, ISBN 3-446-19176-3
Carl Hanser Verlag München Wien, 1997

Mit diesem Lehr- und Arbeitsbuch ist eine fundierte und umfassende Darstellung der modernen rechnerintegrierten Produktentwicklung erschienen. Es faßt die dazu erforderlichen Techniken und Methoden zur Entwicklung des „virtuellen Produktes“ zusammen.

Die Anwendung von Rechnern in der Konstruktion unterlag in den letzten Jahren einem großem Wandel. Firmenstrukturen haben sich verändert, neue Paradigmen der Vorgehensweisen wurden erprobt. Das Buch macht deutlich, wie sich die Anforderungen an die Produktentwicklung und Produktentstehung gewandelt haben. Es kann wohl zu Recht prognostiziert werden, daß zukünftige in der praktischen Anwendung komplexe Produkte auf virtuelle Weise antizipiert werden. Das virtuelle Produktverhalten zu erkennen, d.h. die Simulation aller Phasen des Produktlebenszyklus von der Produktplanung, Design, Konstruktion und Arbeitsvorbereitung über die Fertigung bis zu Service und Recycling, wird zunehmend ein strategisches Ziel produzierender Unternehmen.

Systeme zur Entwicklung eines virtuellen Prototypen dienen der besseren Gestaltung von Produkten und Prozessen sowie der Kommunikation und schnelleren Entscheidungsfindung. Dabei ist die komplexe rechnerunterstützte Produktmodellierung, die neben Funktion, Geometrie und Technologie auch Gebrauchsverhalten und Umwelteigenschaften umfaßt, durch produktspezifische Entwicklungslogiken zu ergänzen.

Dieses Handbuch stellt den aktuellen Stand von Forschung, Technik und Praxis dar. Es dient gleichermaßen als Lehrbuch für Studierende sowie als Nachschlagwerk für alle Anwender in den verschiedenen Bereichen der Industrie und spricht mit den dargestellten Strategien zudem das Management an.

Harald Zebedin

Fertigungsregelung – Logistische Beherrschung von Fertigungsabläufen auf Basis des Trichtermodells

H.-P. Wiendahl

382 Seiten, ISBN 3-446-19084-8
Hanser Verlag, 1997

Die logistische Beherrschung von Fertigungs-, Lagerhaltungs- und Beschaffungsprozessen in der variantenreichen Einzel- und Serienfertigung ist ein Kernproblem der Fertigungsindustrie. Das Buch Fertigungsregelung von Hans-Peter Wiendahl baut auf dem mittlerweile etablierten Hannoverschen Trichtermodell auf. Dabei werden zunächst die logistischen Kennlinien zur formelmäßigen Beschreibung der Wechselwirkungen zwischen Bestand, Reichweite, Durchlaufzeit und Auslastung von Arbeitssystemen behandelt. Ebenso wird die logistische Durchlauf- und Engpaßanalyse ganzer Fertigungsbereiche anhand praktischer Beispiele erläutert.

Die vielfach bewährte belastungsorientierte Fertigungssteuerung (BOA), die über die drei Komponenten Auftragsfreigabe, Monitorsystem und Kapazitätsmodul verfügt, wird in diesem Buch durch einen neuen Ansatz der durchlauforientierten Losgrößenbestimmung und der Durchlaufterminierung erweitert und in allen Einzelheiten vorgestellt. Ein Konfigurationsmodul zur widerspruchsfreien Einstellung der Verfahrensparameter baut es zu einem geschlossenen Konzept der Fertigungsregelung aus. Die Anwendung des Trichtermodells beschränkt sich dabei nicht mehr auf die Fertigung alleine, sondern schließt wegen der wachsenden Bedeutung von Zukaufteilen jetzt auch die Beschaffung und Lagerung mit ein, und umfaßt die logistische Analyse, Überwachung, Diagnose und Regelung der Fertigung sowie die Analyse und Überwachung der Beschaffung.

Um trotz des inhaltlichen Umfanges die Lesbarkeit und die rasche Umsetzung in die betriebliche Praxis zu erleichtern, wurde der Stoff in theoretische Grundlagen und in Anwendungen gegliedert.

Das Buch wendet sich hauptsächlich an Praktiker in Produktion und Logistik sowie an Wissenschaftler und Studierende der Betriebswirtschafts- und Produktionstechnik. Zahlreiche Bilder und detaillierte Rechenbeispiele bereiten den Inhalt auf. Die Erkenntnisse können insbesondere mit Tabellenkalkulationsprogrammen unmittelbar nachvollzogen und praktisch genutzt werden.

Gernot Kronreif

Dynamische Niederschlags-Abflußsimulation von Donauzubringern

Dr. Christoph Fessel, 1997

Begutachter: O.Univ.Prof. Dr.Dr.h.c.mult. P. Kopacek
O.Univ.Prof. Dr. H.-B. Matthias

Das gegenständliche Projekt befaßt sich mit der Entwicklung eines Softwarepaketes zur koordinierten Steuerung von mehreren Laufkraftwerken an der österreichischen Donau. Die Modellformulierung erfolgt dabei mit dem Ziel der Ermittlung und Beurteilung von Handlungsalternativen im Hochwasserfall. Im Sinne einer Pilotstudie werden dabei die Donaukraftwerke Melk, Ybbs, Wallsee und Abwinden betrachtet. Eingangsgrößen der durch getrennte Staustufen modellierten Kraftwerkskette stellen neben dem, im allgemeinen bekannten Staustufenzufluß des Kraftwerkes Abwinden vor allem der nicht bekannte Zubringerabfluß dar. Auf dessen Kenntnis kann aber bei einer koordinierten Steuerung der Kraftwerkskette dennoch nicht verzichtet werden.

Diese Arbeit beschäftigt sich daher mit der Analyse, Beurteilung und Simulation des Donauzubringerabflusses in die genannten Staustufen. Basierend auf einer Untersuchung der in der Hydrologie bestehenden Modellvorstellungen zur Modellierung des Abflußvorganges, wird eine für die Beschreibung der Einzugsgebiete der Donauzubringer geeignete Modellkonzeption erarbeitet und erläutert.
Eine angeschlossene detaillierte Analyse der Abflußsituation der Einzugsgebiete zeigt die Notwendigkeit der Berücksichtigung des aktuellen Feuchtezustandes der Einzugsgebiete. Aus diesem Grund ergibt sich als zentrales Element des sich in den Abflußbildungs- und Konzentrationsprozeß gliedernden Modellkonzeptes ein Speicherelement. Die erforderliche Funktion dieses Speicherelements hinsichtlich aller im Gebiet ablaufenden hydrologischen Prozesse wird detailliert behandelt, womit letztendlich eine Formulierung für die Systemgleichung des Speichers dargestellt werden kann.

Diese Modellformulierung stellt die theoretische Basis des im Rahmen dieser Arbeit erstellten Simulations-Softwarepakets *DoNAb* dar, womit ein „Werkzeug" zur dynamischen Modellierung der Zubringereinzugsgebiete verfügbar ist.

DoNAb ist dabei auf die Ziele einer möglichst guten Anpassungsfähigkeit an die erarbeiteten Einzugsgebietscharakteristika und einer gleichzeitigen Forderung nach einer möglichst guten Übertragbarkeit auf alle Einzugsgebiete der Staustufen ausgerichtet. Zur Anpassung des Programms an die unterschiedlichen Einzugsgebiete der Staustufen, deren Zubringer nach Lage, Größe Bedeutung der Abflußspende und Verfügbarkeit von Meßdaten untersucht werden, ist dem Simulationsprogramm ein Modul zur Parameterberechnung angeschlossen.

Optimierung neuronaler Regler mittels Genetischer Algorithmen

Dr. techn. Thomas Grünberger, 1997

Begutachter: O.Univ.Prof.Dr.techn.A.Weinmann
O.Univ.Prof.Dr.techn.H.P.Jörgl

Die Motivation dieser Arbeit besteht im Einsatz von Künstlichen Neuronalen Netzen für regelungstechnische Aufgabenstellungen als Regler für sowie zur Modellbildung von nichtlinearen Prozessen.

Für den Reglerentwurf bietet sich aufgrund der unbekannten Stellgröße, die ein überwachtes Lernverfahren ausschließt, ein bestrafendes Lernen an, d.h. eine zufällige Änderung im neuronalen Regler wird qualitativ oder quantitativ als gut oder schlecht bewertet. Die Funktion des Lehrers übernimmt ein Gütekriterium für den geschlossenen Regelkreis. Dieses Verfahren wird dahingehend erweitert, daß mit einer Population von neuronalen Reglern fixer Struktur gearbeitet wird, die einem Evolutionsprozeß unterzogen wird, wobei jedes dieser Netze ein Individuum darstellt, dessen Gene durch die Gewichte des Neuronalen Netzes bestimmt werden. Die einzelnen Neuronalen Netze können durch genetische Operatoren wie Mutation ihr Verhalten verbessern und durch Kreuzung der Gene voneinander Lernen, zusätzlich wird die Gesamtpopulation einem Selektionsdruck unterzogen. Als Verfahren wird hierfür ein Genetischer Algorithmus eingesetzt.
Für den neuronalen Regler wird ein Feedforwardnetz verwendet, welches eine rein stationäre nichtlineare Abbildung der Eingangsgrößen auf die Ausgangsgröße darstellt. Um dem Regler ein dynamisches Verhalten zu geben, müssen dynamische Elemente vor- oder nachgeschaltet werden. Als Vorwärtsreglerstruktur wird ein neuronaler PI-Regler untersucht, es wird gezeigt, daß mit diesem Regler bessere Ergebnisse als mit einem linearen PI-Regler für lineare Regelstrecken erzielt werden können. Weiters wird ein *two degree of freedom* Regleransatz vorgestellt, der einen kombinierten Führungs- und Störungsentwurf ermöglicht. Ein Vergleich dieser Reglerstruktur mit dem neuronalen PI-Regler anhand verschiedener Gütekriterien zeigt, daß sich damit besseres dynamisches Verhalten erzielen läßt.
Für die Modellbildung mittels Neuronaler Netze wird einleitend ein nichtlinearer Differenzengleichungsansatz unter Verwendung eines Feedforwardnetzes hinsichtlich der Verwendung als Parallelmodell untersucht. Da sich dieses Verfahren als nicht tauglich erweist, werden teilweise rückgekoppelte Netze (Elman- bzw. Jordan-Netze) untersucht. Die Anwendung auf lineare Regelstrecken zeigt, daß eine Zustandsraumdarstellung des Systems gelernt wird.

Abschließend wird eine spezielle Netzwerkstruktur, eine Serienschaltung eines nichtlinearen Feedforwardnetzes mit einem linearen Elman-Netz, für die Identifikation von Hammersteinmodellen vorgestellt. Für diese Netzwerkstruktur kann als Lernverfahren der Backpropagationalgorithmus eingesetzt werden, Simulationsbeispiele ergeben, daß die statische Nichtlinearität durch das Feedforwardnetz abgebildet wird, das Elman-Netz erlernt wiederum eine Zustandsraumdarstellung des linearen Systemteils.

Kombinationsfehlermaß und kompakte Anregungssignale für optimale parametrische Prozeßidentifikation

Dr. techn. Herbert Swaton, 1997

Begutachter: O.Univ.Prof.Dr.techn.A.Weinmann
titl.A.o.Prof.Dr.techn.R.Noisser

Bei der Identifikation von Prozessen stehen üblicherweise nur gestörte Meßdaten zur Verfügung. Damit eine gewisse Modellgüte garantiert werden kann, ist dieser Meßdatensatz ausreichend informativ zu gestalten. Daher kommt der Gestaltung der experimentellen Rahmenbedingungen bei der Datenaufnahme im Rahmen des Identifikationsvorganges eine besondere Bedeutung zu.

In der zugehörigen Fachliteratur finden sich zu dieser Fragestellung zwei Lösungsansätze. Einerseits lassen sich die Rahmenbedingungen durch Betrachtung geeigneter Optimalitätskriterien analytisch festlegen. Dieser Ansatz ist aber wegen des hohen mathematischen Aufwandes nur für einfache Ausnahmefälle tatsächlich durchzuführen; als weitere Erschwernis kommt hinzu, daß zur exakten Lösung des Optimierungsproblems die Kenntnis des zu identifizierenden Prozesses notwendig wäre. Andererseits existieren zahlreiche Faustformeln und Abschätzungen, deren Anwendung eine geeignete Gestaltung des Identifikationsexperimentes ohne großen Aufwand ermöglichen soll. Letztere Vorgangsweise berücksichtigt allerdings keine Abhängigkeiten zwischen diversen Einflußgrößen. Aus diesem Grund können damit auch keine Optimalitätsanforderungen erfüllt werden.
In der vorliegenden Arbeit wird eine neue Methode zur (sub)optimalen Festlegung der Rahmenbedingungen für das Identifikationsexperiment von linearen Eingrößenystemen vorgestellt. Der Vorteil dieser Methode, die auf Simulationsbetrachtungen basiert, liegt darin, daß ihre Anwendung bedeutend einfacher ist als die analytische Vorgangsweise (und auch nicht auf die Identifikation einfacher Systeme beschränkt bleiben muß), andererseits aber dennoch eine Berücksichtigung der gegenseitigen Abhängigkeiten zwischen den Einflußgrößen erlaubt. Der Grundgedanke, der die Anwendung dieses Ansatzes nicht nur zur Untersuchung bekannter Systeme, sondern auch zur Betrachtung unbekannter Prozesse nahelegt, ist jener, daß Systeme, deren Klemmenverhalten hinreichend ähnlich sind, unter vergleichbaren Voraussetzungen auch ähnliche optimale Experimentrahmenbedingungen für die Identifikation aufweisen. Daher sollten diese (sub)optimalen Bedingungen für unbekannte Systeme unter günstigen Umständen durch Simulationsbetrachtungen an voridentifizierten Modellen ('Vergleichssystemen') zu ermitteln sein. Dies wird an einigen Beispielen demonstriert.
Um die Auswirkungen diverser Maßnahmen bei der Experimentgestaltung beurteilen zu können, erweist sich in einem ersten Schritt die Beurteilung der resultierenden Modelle mit skalaren Fehlermaßen als günstig. Allerdings erlaubt keines der bekannten Fehlermaße für sich alleine eine hinreichende Beurteilung des Modellverhaltens; dies kann allenfalls durch eine gemeinsame Betrachtung mehrerer dieser Kenngrößen erfolgen. Daher wird als neue Kenngröße das *Formfehlermaß* definiert, welches die Aussagen einiger Standardfehlermaße geeignet kombiniert und dessen Anwendung somit eine einfachere Bewertung von Modellen ermöglicht.
Die Wahl der Prozeßanregung ist für das Ergebnis eines Identifikationsvorganges von besonderer

Bedeutung. Signale, die durch Überlagerung von harmonischen Schwingungen entstehen (Multifrequenzsignale), bieten die Möglichkeit zu einer nahezu beliebigen Vorgabe des Anregungsspektrums. Allerdings reicht die Betrachtung des spektralen Verhaltens für sich alleine nicht aus. Mindestens ebenso wichtig ist es, den Zeitverlauf derartiger Signale geeignet vorzugeben. Dies hat durch eine Kompression der Amplitude mit dem Ziel zu erfolgen, möglichst kompakte Zeitverläufe zu erzeugen, wobei der Crestfaktor eine Kenngröße für den Erfolg dieser Maßnahme darstellt.
Für dieses Problem existiert derzeit noch keine geschlossene analytische Lösung. In der vorliegenden Arbeit wird das *Verfahren der Maximalwertkompensation* eingeführt. Dieses ermöglicht unter gewissen Voraussetzungen, welche die Gestalt des Signalspektrums betreffen, eine bessere Kompression derartiger Multifrequenzsignale als die aus der Literatur bekannten Verfahren.

Stabilitätsanalyse von Fuzzy-Regelungen mit Hilfe der Hyperstabilitätstheorie

Dr. techn. Ernst Bodenstorfer, 1997

Begutachter: O.Univ.Prof.Dr.techn.A.Weinmann
titl.A.o.Prof.Dr.techn.R.Noisser

Die vorliegende Arbeit beschäftigt sich mit der Stabilitätsanalyse von Fuzzy-Regelungen, wobei davon ausgegangen wird, daß ein mathematisches Modell der Regelstrecke vorliegt. Es wird ein bekanntes Verfahren zur Stabilitätsanalyse von nichtlinearen Regelkreisen, das auf der Hyperstabilitätstheorie von V. M. Popov basiert, erweitert. Voraussetzung für die Anwendung der Hyperstabilitätsmethode ist, daß der betrachtete Regelkreis auf die Form des sogenannten nichtlinearen Standardregelkreises, bestehend aus einem linearen dynamischen und einem nichtlinearen dynamikfreien Teilsystem, gebracht werden kann. Der Standardregelkreis muß fiktiv so umgeformt werden, daß bestimmte - im Sinne der Hyperstabilitätstheorie für Stabilität hinreichende - Bedingungen womöglich erfüllt sind.

Die Hyperstabilitätsmethode liefert (wie praktisch alle Verfahren zur Stabilitätsbeurteilung nichtlinearer Regelkreise) hinreichende, aber nicht notwendige Bedingungen für die Stabilität. Jede von ihr ausgehende Stabilitätsanalyse ergibt daher mehr oder weniger konservative Resultate. Daher wird in der vorliegenden Arbeit eine neue Art der Umformung des Standardregelkreises eingeführt und mit den aus der Literatur bekannten Umformungen kombiniert. Es wird gezeigt, wie die aus den Umformungen resultierenden freien Parameter und Freiheitsgrade teils analytisch, teils numerisch (mit Hilfe eines Optimierungsalgorithmus) optimiert werden können. Durch die Erweiterung der Anzahl an optimierbaren Parametern, die sich aus der neu eingeführten Regelkreisumformung ergibt, soll die Konservativität des Verfahrens gegenüber dem bekannten Grundverfahren verringert werden. Abschließend wird die Brauchbarkeit des Verfahrens zur Stabilitätsbeurteilung an Hand zweier konkreter Fuzzy-Regelkreise demonstriert.

Datum	***Veranstaltung***	***Ort***	***Weitere Informationen erhältlich bei:***
9.-11.11.1998	5th IFAC Workshop on Intelligent Manufacturing Systems	*Gramado Brazil*	Prof. Carlos E. Pereira Rua Siqueira Campos 341/304 CEP92010-230, Canoas RS, Brazil FAX: +55/51/316 3129 e-mail: cpereira@iee.ufrgs.br http://www.iee.ufrgs.br/iee.cepport.htm
13.-18.6.1999	15th IMEKO WORLD CONGRESS	*Osaka Japan*	Society of Instrument and Control Eng. 35-28-303, 1-Chome Hongo, Bunkyu-ku Tokyo 113, Japan FAX: +81/3/3814 4699
5.-9.7.1999	14th IFAC WORLD CONGRESS	*Beijing PRC*	IFAC'99 IPC Secretariat Institute of Systems Science Chinese Academy of Sciences Beijing 100080, PR China FAX: +86/10/6258 7343 e-mail: ifac99@iss03.iss.ac.cn http://www.ia.ac.cn/ifac99/ifac99.html
16.-20.8.1999	15th IFORS WORLD CONGRESS	*Beijing PRC*	IFORS XV Conference Secretariat Institute of Applied Mathematics Chinese Academy of Sciences Beijing 100080, PR China FAX: ++86/10/6254 1689 e-mail: orchina@publ.east.cn.net http://www.ifors.org/leaflet/triennial.html
29.9.-2.10.1999	7th International Workshop on Computer Aided Systems Theory and Technology – EUROCAST'99	*Vienna Austria*	Prof. Dr. Peter Kopacek Institute for Handling Devices and Robotics Vienna University of Technology Floragasse 7a A-1040 Vienna, Austria FAX: +43/1/504 18 359 e-mail ecast99@ihrt1.ihrt.tuwien.ac.at http://www.ihrt.tuwien.ac.at/ECAST99
2.-4.12.1999	1st Intern. Workshop on Multi-Agent Systems in Production – MAS'99	*Vienna Austria*	Prof. Dr. Peter Kopacek Institute for Handling Devices and Robotics Vienna University of Technology Floragasse 7a A-1040 Vienna, Austria FAX: +43/1/504 18 359 e-mail mas99@ihrt1.ihrt.tuwien.ac.at http://www.ihrt.tuwien.ac.at/MAS99/