Chapter 3

State space adaptive control

A regulator is a feedback mechanism by which a system is guided along a desired behaviour. The design of this regulator is commonly based upon a mathematical model of the system to be controlled. Such a model as derived in the previous chapter is treated as a true description of the system and is expected not to change in terms of its structure. The plant parameters, however, can be slowly time-variant or may be uncertain from the beginning. In these cases, adaptive control is necessary. A state space approach to adaptive control is presented in this chapter.

> "*... adaptive control is a special type of nonlinear feedback control in which the states of the process are separated in two categories, which change at different rates. The slowly changing states are viewed as parameters.*", [Åstrøm and Wittenmark, 1988].

This means that there is a fast feedback loop with an ordinary controller calculating the control variable from the fast states like the system output. A slow feedback loop is then wrapped around the first one adapting the regulator parameters which are the slow states according to [Åstrøm and Wittenmark, 1988] as *meta-regulator* in order to guarantee a desired closed loop behaviour if the controlled system changes (see Fig. 3.1). The controller parameters can be estimated directly from the states of the first feedback loop or indirectly via a separate controller design. In this work an indirect approach for adaptive control has been chosen, i.e. the adaptation process is separated into two parts, the identification of the system parameters and its states and the calculation of the controller parameters based upon the estimated plant parameters. For both parts the MATLAB program codes are included in Appendix D.2.

3.1 Parameter and state estimation

Identification is the extraction of parameters according to a model from measured data of a system under investigation. This process can be done on-line or off-line. Since parameters of a system can change during operation, the estimation of these parameters has to be done on-line, i.e. at each sample time when new information is available. Only

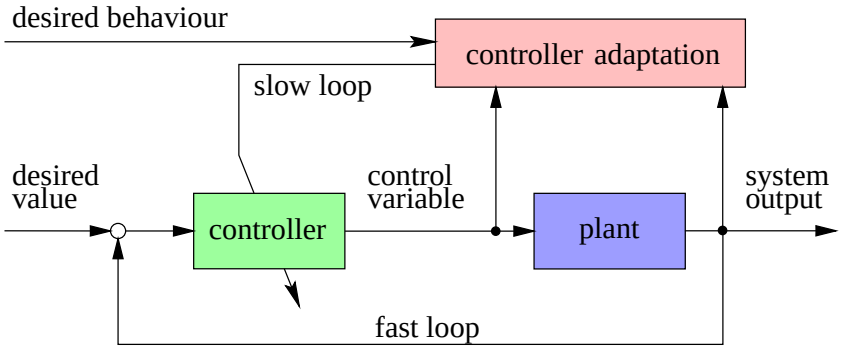

Figure 3.1: Block diagram of an adaptive control loop.

recursive algorithms can be used for that task, because its computation time must be constant. Such algorithms calculate a new estimate of the system parameters from a previous one, the current data and eventually from auxiliary variables based on a model for systems with stochastic disturbances. A basic requirement for an estimation algorithm is a predictor which predicts future outputs of a system under the assumption of known system parameters.

3.1.1 Innovations model

Based upon the stochastic state space model defined by Eqn.(2.99) and Eqn.(2.100) derived in Chapter 2, a predictor for the output $\mathbf{y}(k)$ including the Kalman matrix $\mathbf{K}$ is introduced as [Ljung and Søderstrøm, 1983]

$$\hat{\mathbf{x}}(k+1) = [\mathbf{A} - \mathbf{K}\,\mathbf{C}]\,\hat{\mathbf{x}}(k) + \mathbf{B}\,\mathbf{u}(k) + \mathbf{K}\,\mathbf{y}(k), \tag{3.1}$$

$$\hat{\mathbf{y}}(k) = \mathbf{C}\,\hat{\mathbf{x}}(k). \tag{3.2}$$

At the moment the system matrices are assumed to be known, and the time invariant Kalman matrix $\mathbf{K}$ is calculated via the Riccati equation [Ljung and Søderstrøm, 1983] from the covariance matrices. The usual solution in adaptive control using the Kalman filter is to estimate only the parameters of the system matrix and the covariances $\mathbf{R}_1$,$\mathbf{R}_2$ and $\mathbf{R}_{12}$ as defined in Eqn.(2.103), Eqn.(2.104) and Eqn.(2.105), and hereafter calculate the Kalman matrix via the time variant Riccati equation. A different approach is to directly estimate the parameters of the Kalman matrix $\mathbf{K}$. Then the matrices $\mathbf{A}$, $\mathbf{K}$ and $\mathbf{C}$ depend on their parameters $\mathbf{p}$. Eqn.(3.1) and Eqn.(3.2) can be rewritten as

$$\hat{\mathbf{x}}(k+1,\mathbf{p}) = \mathbf{A}(\mathbf{p})\,\hat{\mathbf{x}}(k,\mathbf{p}) + \mathbf{B}\,\mathbf{u}(k) + \mathbf{K}(\mathbf{p})\,\boldsymbol{\varepsilon}(k), \tag{3.3}$$

$$\mathbf{y}(k) = \mathbf{C}(\mathbf{p})\,\hat{\mathbf{x}}(k,\mathbf{p}) + \boldsymbol{\varepsilon}(k), \tag{3.4}$$

where $\boldsymbol{\varepsilon}(k)$ corresponds to the prediction error or *innovation* with

$$\boldsymbol{\varepsilon}(k,\mathbf{p}) = \mathbf{y}(k) - \hat{\mathbf{y}}(k|\mathbf{p}), \tag{3.5}$$

and $\hat{\mathbf{y}}(k|\mathbf{p})$ is the estimated output of the system under the assumption of the parameter vector $\mathbf{p}$. For this reason the model described by Eqn.(3.3) and Eqn.(3.4) is known as *innovations model*. The Kalman matrix is parameterised as $\mathbf{K}(\mathbf{p}) = \{\mathbf{k}^{(ij)}\}$ with $i, j = 1, 2, 3, 4$ and

$$\mathbf{k}^{(ij)} = \left[\begin{array}{cc} k_2^{(ij)} & k_1^{(ij)} \end{array} \right]^T . \tag{3.6}$$

All parameters of this innovations model defined by the matrices $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{K}$ are summarised within the n_p-dimensional parameter vector

$$\mathbf{p} = \left[-a_2^{(11)}, -a_1^{(11)}, \ldots, c_2^{(11)}, c_1^{(11)}, \ldots, k_2^{(11)}, k_1^{(11)}, \ldots \right] . \tag{3.7}$$

The Kalman matrix introduces $n_s\, n_o = 32$ more parameters to the entire estimation algorithm, which means that there are $n_p = 96$ parameters to be estimated. Note, that the control-input matrix $\mathbf{B}$, as defined in Eqn.(2.96) and Eqn.(2.97), does not depend on the parameter vector, if the system is described in controller canonical form.

Based upon the innovations model a recursive algorithm has to be implemented to estimate the system parameters and the Kalman matrix under on-line conditions for a controlled system with stochastic disturbances.

3.1.2 The recursive prediction error method

The *recursive prediction error method* (RPEM, [Ljung and Søderstrøm, 1983]) applied to the innovations model is used in this work. Figure 3.2 shows the principle of the algorithm. This algorithm can estimate both system parameters and the system states. An innovations model is used to predict the system states $\hat{\mathbf{x}}(k)$ and the output $\hat{\mathbf{y}}(k)$ based upon the system inputs $\mathbf{u}(k)$ and its parameters $\hat{\mathbf{p}}(k)$. The difference between model output $\hat{\mathbf{y}}(k)$ and the measured system output $\mathbf{y}(k)$, the prediction error $\boldsymbol{\varepsilon}(k)$, is then used for updating the system parameters $\hat{\mathbf{p}}(k)$. These parameters are later on used for the controller design. Independent inputs are the measurement noise $\mathbf{v}(k)$ and the set point $\mathbf{w}(k)$. In the following this algorithm is explained in detail.

3.1.3 Derivation of the recursive algorithm

The goal of any estimation algorithm is to keep the prediction error $\mathbf{y} - \hat{\mathbf{y}}(k|\mathbf{p})$ as small as possible in terms of its statistical properties. Thus, it is necessary to define a criterion functional to be minimised as for example

$$V(\mathbf{p}) = \mathcal{E}\left\{ H(\mathbf{p}, \boldsymbol{\varepsilon}) \right\} \rightarrow \text{Min.} \tag{3.8}$$

with the quadratic function

$$H(\mathbf{p}, \boldsymbol{\varepsilon}) = \boldsymbol{\varepsilon}^T(k, \mathbf{p})\, \boldsymbol{\Lambda}^{-1} \boldsymbol{\varepsilon}(k, \mathbf{p}), \tag{3.9}$$

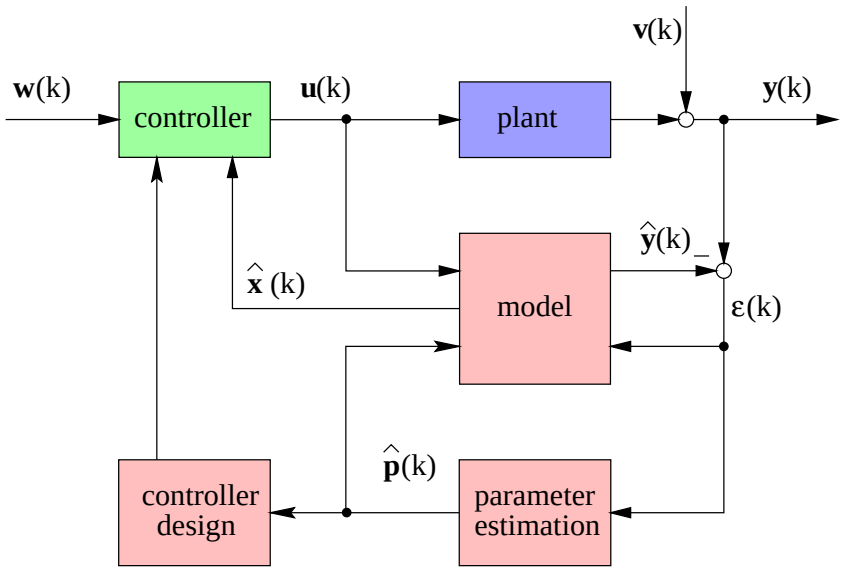

Figure 3.2: Sketch of the recursive predictive error method.

where $\mathbf{\Lambda}$ is a weighting matrix. The functional $V(\mathbf{p})$ can be minimised with respect to the parameter vector $\mathbf{p}$ by setting its gradient equal to zero as

$$-\left[\frac{\partial}{\partial \mathbf{p}} V(\mathbf{p})\right]^T = \mathbf{0}. \tag{3.10}$$

Under the assumption that the gradient

$$-\frac{\partial}{\partial \mathbf{p}} H(\mathbf{p}, \boldsymbol{\varepsilon}(k)) = \mathbf{q}^T(\mathbf{p}, \boldsymbol{\varepsilon}(k)) \tag{3.11}$$

can be calculated for any $\mathbf{p}$ exchanging the expectation and the derivative operator, Eqn.(3.10) can be solved in the form

$$\mathcal{E}\left\{\, \mathbf{q}(\mathbf{p}, \boldsymbol{\varepsilon}(k)) \,\right\} = \mathbf{0}. \tag{3.12}$$

The problem involved in the solution of Eqn.(3.12) is that the expectation cannot be evaluated, because the distributions of $\mathbf{p}$ and $\boldsymbol{\varepsilon}$ are not known *a priori*. Of course, there would be a solution by replacing the expectation with a mean value from an experimental sequence such that for a given $\mathbf{p}$, the function $\mathbf{q}(\mathbf{p}, \boldsymbol{\varepsilon}(k))$ is evaluated for a series of $\boldsymbol{\varepsilon}(k)$. If this procedure is carried out for different $\mathbf{p}$, a solution can be found numerically, but off-line.

A pragmatic solution is the recursive approximation [Robbins and Monroe, 1951] with

$$\hat{\mathbf{p}}(k) = \hat{\mathbf{p}}(k-1) - \gamma(k)\, \mathbf{q}(\hat{\mathbf{p}}, \boldsymbol{\varepsilon}(k)), \tag{3.13}$$

where $\gamma(k)$ is a sequence of positive scalars. It has been proved that the sequence $\hat{\mathbf{p}}(k)$ converges to a solution, if certain assumptions are met as, for example, $\boldsymbol{\varepsilon}(k)$ has to be a sequence of independent random vectors [Robbins and Monroe, 1951].

A deterministic approach to a solution is the method of the steepest descent given by

$$\hat{\mathbf{p}}(k) = \hat{\mathbf{p}}(k-1) - \gamma(k) \left[\frac{\partial}{\partial \mathbf{p}} V(\mathbf{p})\right]^T \Bigg|_{\mathbf{p}=\hat{\mathbf{p}}(k-1)}, \tag{3.14}$$

where $\gamma(k)$ is a positive scalar, too. As it is well known, this algorithm does not converge very well to the minimum, because the first derivative becomes very small in the vicinity of the solution. Therefore, it is extended to the so-called Newton method [Ljung and Søderstrøm, 1983], where the gradient is modified such that the algorithm results in

$$\hat{\mathbf{p}}(k) = \hat{\mathbf{p}}(k-1) - \left[\frac{\partial^2}{\partial \mathbf{p}^2} V(\mathbf{p})\right]^{-1} \cdot \left[\frac{\partial}{\partial \mathbf{p}} V(\mathbf{p})\right]^T \Bigg|_{\mathbf{p}=\hat{\mathbf{p}}(k-1)}, \tag{3.15}$$

replacing the weighted first derivative by the product of the inverse second derivative and the first derivative of the criterion functional. This gives a better approximation in the vicinity of the solution, especially if the criterion is quadratic. The same solution can be obtained, if the criterion functional $V(\mathbf{p})$ is approximated by a second order Taylor series, and differentiated with respect to $\mathbf{p}$ afterwards in order to solve Eqn.(3.10) for the estimated vector $\hat{\mathbf{p}}$.

The stochastic Newton method

Merging the Newton method and the scheme for the recursive calculation of the expectation in Eqn.(3.13), Eqn.(3.15) can be rewritten in the form

$$\hat{\mathbf{p}}(k) = \hat{\mathbf{p}}(k-1) + \gamma(k) \left[\bar{V}''(\hat{\mathbf{p}}(k-1), \boldsymbol{\varepsilon}(k))\right]^{-1} \cdot \mathbf{q}(\hat{\mathbf{p}}(k-1), \boldsymbol{\varepsilon}(k)) \tag{3.16}$$

where $\bar{V}''(\hat{\mathbf{p}}(k-1), \boldsymbol{\varepsilon}(k))$ is the approximation of the second derivative of $V''(\mathbf{p})$ with respect to the parameter vector $\mathbf{p}$ based upon all observations up to k. The vector for the given criterion can be written for k as

$$\mathbf{q}(\hat{\mathbf{p}}, \boldsymbol{\varepsilon}(k)) = -\left[\frac{\partial}{\partial \mathbf{p}} H(\mathbf{p}, \boldsymbol{\varepsilon}(k))\right]^T \Bigg|_{\mathbf{p}=\hat{\mathbf{p}}(k)} = -\boldsymbol{\Psi}(k, \hat{\mathbf{p}})\, \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\varepsilon}(k). \tag{3.17}$$

with the gradient matrix $\boldsymbol{\Psi}$ as the partial derivative of the estimation error as

$$\boldsymbol{\Psi}(k, \hat{\mathbf{p}}) = -\left(\frac{\partial}{\partial \mathbf{p}}\, \boldsymbol{\varepsilon}(k)\right)^T \Bigg|_{\mathbf{p}=\hat{\mathbf{p}}(k)} = \left(\frac{\partial}{\partial \mathbf{p}}\, \hat{\mathbf{y}}(k, \mathbf{p})\right)^T \Bigg|_{\mathbf{p}=\hat{\mathbf{p}}(k)}. \tag{3.18}$$

The Newton search direction

The matrix $\bar{V}''(\hat{\mathbf{p}}(k-1), \boldsymbol{\varepsilon}(k))$ in Eqn.(3.16) has to be computed for each sample time interval and has to be positive definite to guarantee a stable convergence. Therefore,

the second derivative $V''(\mathbf{p})$ with respect to the parameter vector $\mathbf{p}$ is approximated by

$$\begin{aligned}\frac{\partial^2}{\partial \mathbf{p}^2} V(\mathbf{p}) &= \mathcal{E}\left\{ \boldsymbol{\Psi}(k,\mathbf{p})\, \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\Psi}^T(k,\mathbf{p}) \right\} \\ &+ \mathcal{E}\left\{ \left[\frac{\partial^2}{\partial \mathbf{p}^2} \boldsymbol{\varepsilon}^T(k,\mathbf{p}) \right] \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\varepsilon}(k,\mathbf{p}) \right\} \\ &\approx \mathcal{E}\left\{ \boldsymbol{\Psi}(k,\mathbf{p})\, \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\Psi}^T(k,\mathbf{p}) \right\} \end{aligned} \tag{3.19}$$

with the assumption that the last term in Eqn.(3.19) is equal to zero at the true solution. Replacing the expectation in this equation with a weighted sample mean and evaluating this sample mean with the knowledge up to k based upon the estimated parameter vector $\mathbf{p} = \hat{\mathbf{p}}(k-1)$, the approximation for the second derivative of $V(\mathbf{p})$ with respect to the parameter vector $\mathbf{p}$ is according to [Ljung and Søderstrøm, 1983]

$$\mathbf{R}(k) = \frac{1}{k} \sum_{h=1}^{k} \gamma_1(h,k)\, \boldsymbol{\Psi}(h,\hat{\mathbf{p}})\, \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\Psi}^T(h,\hat{\mathbf{p}}) + \gamma_2(k)\, \mathbf{R}_0, \tag{3.20}$$

with an initial matrix $\mathbf{R}_0$ and $\gamma_1(h,k)$ and $\gamma_2(k)$ being weighting sequences, respectively. If these weighting coefficients are chosen properly, the matrix $\mathbf{R}(k)$ can be computed recursively such that

$$\mathbf{R}(k) = \mathbf{R}(k-1) + \gamma(k)\, \left[\boldsymbol{\Psi}(k)\, \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\Psi}^T(k) - \mathbf{R}(k-1) \right], \tag{3.21}$$

Using this algorithm for the estimation of $\bar{V}''(\hat{\mathbf{p}}(k-1), \boldsymbol{\varepsilon}(k))$, the recursive parameter estimation algorithm can be carried out in the form

$$\hat{\mathbf{p}}(k) = \hat{\mathbf{p}}(k-1) + \gamma(k)\, \mathbf{R}^{-1}(k)\, \boldsymbol{\Psi}(k,\hat{\mathbf{p}})\, \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\varepsilon}(k,\hat{\mathbf{p}}(k-1)). \tag{3.22}$$

The weighting sequence $\gamma(k)$ finally turns out to be a function depending on the so-called forgetting factor $\rho(k)$. This factor and the weighting sequence $\gamma(k)$ determine the speed of convergence.

Simplification

The matrix $\boldsymbol{\Lambda}$ is chosen as the identity matrix $\mathbf{I}$, i.e. it is assumed to be constant and therefore only scales the algorithm. Since the inverse matrix $\mathbf{R}^{-1}(k)$ has to be calculated for the parameter update, the matrix

$$\mathbf{P}(k) = \gamma(k)\, \mathbf{R}^{-1}(k) \tag{3.23}$$

is introduced. Using Eqn.(3.21) the previous equation can then be rewritten as

$$\mathbf{P}(k) = \gamma(k) \left[\mathbf{R}(k-1) + \gamma(k)\, \left[\boldsymbol{\Psi}(k)\, \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\Psi}^T(k) - \mathbf{R}(k-1) \right] \right]^{-1}. \tag{3.24}$$

Applying Eqn.(3.23) for $k-1$ and using the matrix inversion Lemma for the sum of matrices [Ljung and Søderstrøm, 1983], the recursive calculation of the parameter vector and the matrix $\mathbf{P}$ yields

$$\mathbf{P}(k) = \frac{1}{\rho(k)} \left[\mathbf{P}(k-1) - \mathbf{L}(k)\, \mathbf{\Psi}^T(k)\mathbf{P}(k-1)\right], \tag{3.25}$$

$$\hat{\mathbf{p}}(k) = \hat{\mathbf{p}}(k-1) + \mathbf{L}(k)\, \boldsymbol{\varepsilon}(k), \tag{3.26}$$

with the auxiliary matrix

$$\mathbf{L}(k) = \mathbf{P}(k-1)\, \mathbf{\Psi}(k)\, \left[\mathbf{I} + \mathbf{\Psi}^T(k)\, \mathbf{P}(k-1)\, \mathbf{\Psi}(k)\right]^{-1}, \tag{3.27}$$

and the so-called forgetting factor

$$\rho(k) = \frac{\gamma(k-1)}{\gamma(k)}\Big[1 - \gamma(k)\Big]. \tag{3.28}$$

The calculation of the factor $\rho(k)$ is treated Subsection 3.1.5 in detail. The matrix $\mathbf{P}$ is later on referred to as covariance matrix, because it corresponds to the covariance of the estimated parameters. The choice of the starting value for this matrix is considered in Subsection 3.1.6.

Derivation of the gradient matrix

The only expression which is left to be determined is the gradient matrix $\mathbf{\Psi}$ (see also [Nazaruddin, 1994]). Utilising the definition for $\hat{\mathbf{y}}(k, \mathbf{p})$ the gradient matrix can be obtained for a parameter vector $\mathbf{p}$ as

$$\mathbf{\Psi}^T(k, \mathbf{p}) = \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{C}(\mathbf{p})\, \hat{\mathbf{x}}(k, \mathbf{p})\Big]. \tag{3.29}$$

In order to calculate the gradient matrix, the matrices

$$\mathbf{W}(k, \mathbf{p}) = \frac{\partial}{\partial \mathbf{p}} \hat{\mathbf{x}}(k, \mathbf{p}), \tag{3.30}$$

$$\mathbf{V}_k(\mathbf{p}, \hat{\mathbf{x}}) = \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{C}(\mathbf{p})\, \hat{\mathbf{x}}\Big] \tag{3.31}$$

are introduced. Then the partial derivative using the product derivative theorem yields

$$\begin{aligned} \mathbf{\Psi}^T(k, \mathbf{p}) &= \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{C}(\mathbf{p})\, \hat{\mathbf{x}}(k, \mathbf{p})\Big]. \\ &= \mathbf{C}(\mathbf{p})\, \frac{\partial}{\partial \mathbf{p}}\left[\hat{\mathbf{x}}(k, \mathbf{p})\right] + \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{C}(\mathbf{p})\, \hat{\mathbf{x}}\Big], \\ &= \mathbf{C}(\mathbf{p})\, \mathbf{W}(k, \mathbf{p}) + \mathbf{V}_k(\mathbf{p}, \hat{\mathbf{x}}(k, \mathbf{p})). \end{aligned} \tag{3.32}$$

Applying Eqn.(3.30) to $k+1$ and using the innovations model output for $\hat{\mathbf{x}}(k+1,\mathbf{p})$ from Eqn.(3.3) the auxiliary matrix results in

$$\begin{aligned}\mathbf{W}(k+1,\mathbf{p}) &= \frac{\partial}{\partial \mathbf{p}}\hat{\mathbf{x}}(k+1,\mathbf{p}) \\ &= \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{A}(\mathbf{p})\,\hat{\mathbf{x}}(k,\mathbf{p}) + \mathbf{B}\,\mathbf{u}(k) + \mathbf{K}(\mathbf{p})\,\boldsymbol{\varepsilon}(k)\Big]. \end{aligned} \tag{3.33}$$

The previous equation can be expressed as

$$\begin{aligned}\mathbf{W}(k+1,\mathbf{p}) &= \mathbf{A}(\mathbf{p})\,\frac{\partial}{\partial \mathbf{p}}\,\hat{\mathbf{x}}(k,\mathbf{p}) + \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{A}(\mathbf{p})\,\hat{\mathbf{x}}\Big] \\ &\quad + \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{B}\,\mathbf{u}\Big] \\ &\quad + \mathbf{K}(\mathbf{p})\,\frac{\partial}{\partial \mathbf{p}}\,\boldsymbol{\varepsilon}(k,\mathbf{p}) + \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{K}(\mathbf{p})\,\boldsymbol{\varepsilon}\Big]. \end{aligned} \tag{3.34}$$

Using the definition for the gradient matrix $\boldsymbol{\Psi}$ in Eqn.(3.18) the fourth term of the previous equation can be written as

$$\begin{aligned}\mathbf{K}(\mathbf{p})\,\frac{\partial}{\partial \mathbf{p}}\,\boldsymbol{\varepsilon}(k,\mathbf{p}) &= \mathbf{K}(\mathbf{p})\,[-\boldsymbol{\Psi}] \\ &= \mathbf{K}(\mathbf{p})\Big[-\mathbf{C}(\mathbf{p})\,\mathbf{W}(k,\mathbf{p}) - \mathbf{V}_k(\mathbf{p},\hat{\mathbf{x}}(k,\mathbf{p}))\Big] \\ &= -\mathbf{K}(\mathbf{p})\,\mathbf{C}(\mathbf{p})\,\mathbf{W}(k,\mathbf{p}) - \mathbf{K}(\mathbf{p})\,\mathbf{V}_k(\mathbf{p},\hat{\mathbf{x}}(k,\mathbf{p})). \end{aligned} \tag{3.35}$$

Inserting this solution into Eqn.(3.34) yields

$$\begin{aligned}\mathbf{W}(k+1,\mathbf{p}) &= \mathbf{A}(\mathbf{p})\,\frac{\partial}{\partial \mathbf{p}}\,\hat{\mathbf{x}}(k,\mathbf{p}) \\ &\quad + \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{A}(\mathbf{p})\,\hat{\mathbf{x}}\Big] + \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{B}\,\mathbf{u}\Big] + \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{K}(\mathbf{p})\,\boldsymbol{\varepsilon}\Big] \\ &\quad - \mathbf{K}(\mathbf{p})\,\mathbf{C}(\mathbf{p})\,\mathbf{W}(k,\mathbf{p}) - \mathbf{K}(\mathbf{p})\,\mathbf{V}_k(\mathbf{p},\hat{\mathbf{x}}(k,\mathbf{p})) \end{aligned} \tag{3.36}$$

and finally with some simplifications and the definition for $\mathbf{W}(k,\mathbf{p})$, the result is

$$\begin{aligned}\mathbf{W}(k+1,\mathbf{p}) &= \Big[\mathbf{A}(\mathbf{p}(k)) - \mathbf{K}(\mathbf{p}(k))\,\mathbf{C}(\mathbf{p}(k))\Big]\,\mathbf{W}(k,\mathbf{p}) \\ &\quad + \mathbf{M}_k - \mathbf{K}(\mathbf{p}(k))\,\mathbf{V}_k. \end{aligned} \tag{3.37}$$

with the matrix

$$\mathbf{M}_k(\hat{\mathbf{p}},\mathbf{x},\mathbf{u},\boldsymbol{\varepsilon}) = \frac{\partial}{\partial \mathbf{p}}\Big[\mathbf{A}(\mathbf{p})\,\mathbf{x} + \mathbf{B}\,\mathbf{u} + \mathbf{K}(\mathbf{p})\,\boldsymbol{\varepsilon}\Big]. \tag{3.38}$$

Finally, the gradient matrix is calculated according to Eqn.(3.32) for $k+1$ as

$$\boldsymbol{\Psi}(k+1) = \mathbf{W}^T(k+1,\mathbf{p})\,\mathbf{C}^T(\mathbf{p}(k)) + \mathbf{V}_k^T(\mathbf{p}(k),\hat{\mathbf{x}}(k+1)). \tag{3.39}$$

The matrices $\mathbf{M}_k = \mathbf{M}_k(\mathbf{p},\hat{\mathbf{x}}(k),\boldsymbol{\varepsilon}(k))$ and $\mathbf{V}_k = \mathbf{V}_k(\mathbf{p},\hat{\mathbf{x}}(k))$ are model specific and have to be calculated only once in terms of their structure. A derivation for the magnetic bearing system is given in Appendix B.

3.1.4 Summary of the recursive algorithm

The implemented algorithm can be summarised by the following equations with the estimated parameter vector $\hat{\mathbf{p}}$. These equations have to be evaluated at each sample time interval.

1. Compute the estimation error $\boldsymbol{\varepsilon}$ from the measured output and the previous estimate such that

$$\boldsymbol{\varepsilon}(k) = \mathbf{y} - \hat{\mathbf{y}}(k|\hat{\mathbf{p}}(k-1)). \tag{3.40}$$

2. Compute the auxiliary matrix

$$\mathbf{L}(k) = \mathbf{P}(k-1)\,\boldsymbol{\Psi}(k)\left(\mathbf{I} + \boldsymbol{\Psi}^T(k)\,\mathbf{P}(k-1)\,\boldsymbol{\Psi}(k)\right)^{-1}. \tag{3.41}$$

3. Then compute the new parameter estimate $\hat{\mathbf{p}}$ such that

$$\hat{\mathbf{p}}(k) = \hat{\mathbf{p}}(k-1) + \mathbf{L}(k)\,\boldsymbol{\varepsilon}(k), \tag{3.42}$$

4. and update the covariance matrix

$$\mathbf{P}(k) = \frac{1}{\rho(k)}\left(\mathbf{P}(k-1) - \mathbf{L}(k)\,\boldsymbol{\Psi}^T(k)\mathbf{P}(k-1)\right). \tag{3.43}$$

5. For the use with the state space controller compute the estimated states

$$\hat{\mathbf{x}}(k+1) = \mathbf{A}(\hat{\mathbf{p}}(k))\,\hat{\mathbf{x}}(k) + \mathbf{B}\,\mathbf{u}(k) + \mathbf{K}(\hat{\mathbf{p}}(k))\,\boldsymbol{\varepsilon}(k), \tag{3.44}$$

6. and the estimated outputs $\hat{\mathbf{y}}$ for the next cycle

$$\hat{\mathbf{y}}(k+1) = \mathbf{C}(\hat{\mathbf{p}}(k))\,\hat{\mathbf{x}}(k+1). \tag{3.45}$$

7. The computation of the auxiliary matrix

$$\begin{aligned}\mathbf{W}(k+1, \hat{\mathbf{p}}(k)) \;=\; & \left(\mathbf{A}(\hat{\mathbf{p}}(k)) - \mathbf{K}(\hat{\mathbf{p}}(k))\mathbf{C}(\hat{\mathbf{p}}(k))\right)\mathbf{W}(k, \hat{\mathbf{p}}(k-1)) \\ & + \mathbf{M}_k - \mathbf{K}(\hat{\mathbf{p}}(k))\,\mathbf{V}_k,\end{aligned} \tag{3.46}$$

8. yields the gradient matrix

$$\boldsymbol{\Psi}(k+1) = \mathbf{W}^T(k+1, \hat{\mathbf{p}})\,\mathbf{C}^T(\hat{\mathbf{p}}(k)) + \mathbf{V}_k^T(k+1). \tag{3.47}$$

The initialisation of this recursive algorithm is discussed in Section 3.1.7 and in Section 4.5.1 with respect to its numerical implementation.

3.1.5 Forgetting factor

Special attention has to be paid to the so-called forgetting factor $\rho(k)$ in Eqn.(3.43). This factor determines the rate of change for the covariance matrix and can be compared to a gain for an ordinary control loop. If this factor is too small ($\rho(k) \ll 1$), the covariance matrix increases too fast, if it is too large (close to 1), the algorithm reacts too slowly to parameter changes. Therefore, the forgetting factor needs to be controlled separately by a statistical value, which gives the change of the variance of the estimation error. Similarly to [Basseville and Beneviste, 1984] and [Basseville, 1983] the parameter

$$\delta(k) = \frac{\hat{\sigma}_\varepsilon^2(N_1, k) - \hat{\sigma}_\varepsilon^2(N_2, k)}{\hat{\sigma}_\varepsilon^2(N_2, k)}, \tag{3.48}$$

is introduced with $N_1 \leq N_2$ and $\hat{\sigma}_\varepsilon^2(N_i, k)$ as estimate for the prediction error variance with a variable memory N_i. The estimates for the variances are calculated recursively as

$$\hat{\sigma}_\varepsilon^2(N_i, k) = \frac{N_i - 2}{N_i - 1}\,\hat{\sigma}_\varepsilon(N_i, k-1) + \frac{1}{n_o\, N_i}\,\boldsymbol{\varepsilon}^T(k)\,\boldsymbol{\varepsilon}(k) \tag{3.49}$$

with n_o being the number of outputs. The estimated value for $\delta(k)$ is filtered recursively by

$$\hat{\delta}(k) = k_\delta\,\hat{\delta}(k-1) + (1 - k_\delta)\,\delta(k) \tag{3.50}$$

with k_δ determining the rate of change.

The idea behind this algorithm is, that under the assumption of a stationary process, the variance of the estimation error in a steady state should be stationary as well. This assumption is called the hypothesis H_0 [Nazaruddin, 1994]. The alternate hypothesis H_1 is that parameters of the controlled system have changed which causes higher estimation errors. Statistically speaking, H_0 assumes $\delta(k)$ to have zero mean with a certain variance. Under the assumption of an uniform distribution, this probability density function maps $\delta(k)$ to the likelihood of H_0. A threshold δ_H decides between both hypotheses by [Basseville and Beneviste, 1984]

$$\hat{\delta}(k) \underset{H_1}{\overset{H_0}{\lessgtr}} \delta_H. \tag{3.51}$$

If $\hat{\delta}(k)$ triggers the threshold δ_H, e.g. 20% of the mean variance, then the hypothesis H_1 is more likely than H_0. As long as hypothesis H_1 is more likely than H_0, the consequences are:

1. Reset the forgetting factor $\rho(k)$ to a smaller value ρ_0,
2. Add a white noise signal of certain deflection to the set point.

The first action causes the increase of the covariance matrix on the one hand, and weight the estimated values in its memory less than the innovation on the other hand.

This makes the algorithm converge to the true parameters faster. With the second action the closed loop system is excited by an uncorrelated signal. Therefore, the signal to noise ratio increases and accelerate the convergence.

If $\hat{\delta}(k)$ falls beyond the threshold δ_H again, $\rho(k)$ follows a law

$$\rho(k) = k_\rho \, \rho(k-1) + (1 - k_\rho) \, \rho_\infty, \tag{3.52}$$

with ρ_∞ the final value and k_ρ determining the rate of change. Then, assuming that the new parameters are already found, the covariance decreases to a minimum value.

3.1.6 Parameter covariance

Since one has to guarantee that the covariance matrix in Eqn.(3.43) remains positive definite, a special algorithm for the covariance update has been used, based upon the factorisation of the covariance matrix $\mathbf{P} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ with $\mathbf{U}$ as upper triangular matrix with all diagonal elements equal to 1 and $\mathbf{D}$ a diagonal matrix [Ljung and Søderstrøm, 1983]. Since the system under investigation is a MIMO-system and the proposed factorisation algorithm is only applicable to SISO-systems, all prediction errors are considered to be independent observations [Bierman, 1988]. The matrices $\mathbf{U}$ and $\mathbf{D}$ are updated separately by the following algorithm for all outputs $h = 1 \ldots n_o$, which yields a column $\mathbf{L}_h$ for the auxiliary matrix $\mathbf{L}$.

For $h = 1 \ldots n_o$ outputs do the following [Bierman, 1988]:

1. Compute the vectors

$$\begin{aligned} \mathbf{f} &= \mathbf{U}^T(k-1) \, \mathbf{\Psi}_h(k), && (3.53)\\ \mathbf{g} &= \mathbf{D}(k-1) \, \mathbf{f}, && (3.54)\\ \boldsymbol{\beta}_0 &= \rho(k). && (3.55) \end{aligned}$$

2. For $j = 1 \ldots n_p$, with n_p being the number of parameters in the parameter vector $\mathbf{p}$ do

   (a) Compute the vectors

$$\begin{aligned} \boldsymbol{\beta}_j &= \boldsymbol{\beta}_{j-1} + \mathbf{f}_j \, \mathbf{g}_j, && (3.56)\\ \mathbf{D}_{jj}(k) &= \frac{\boldsymbol{\beta}_{j-1}}{\boldsymbol{\beta}_j \, \rho(k)} \, \mathbf{D}_{jj}(k-1), && (3.57)\\ \boldsymbol{\nu}_j &= \mathbf{g}_j, && (3.58)\\ \boldsymbol{\mu}_j &= -\frac{1}{\boldsymbol{\beta}_{j-1}} \mathbf{f}_j. && (3.59) \end{aligned}$$

   (b) For $i = 1 \ldots j-1$ compute

$$\begin{aligned} \mathbf{U}_{ij}(k) &= \mathbf{U}_{ij}(k-1) + \boldsymbol{\nu}_i \boldsymbol{\mu}_j, && (3.60)\\ \boldsymbol{\nu}_i &= \boldsymbol{\nu}_i + \mathbf{U}_{ij}(k-1) \, \boldsymbol{\nu}_j. && (3.61) \end{aligned}$$

3. Calculate one column for the auxiliary matrix

$$\mathbf{L}_h = \frac{1}{\boldsymbol{\beta}_{n_p}} \left[\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_{n_p}\right]^T. \tag{3.62}$$

In the preceding equations $\boldsymbol{\Psi}_h$ symbolises the h^{th} column of the gradient matrix $\boldsymbol{\Psi}$. $\mathbf{f}_j$ symbolises the j^{th} entry of the column vector $\mathbf{f}$ as example for $\mathbf{g}$, $\boldsymbol{\beta}$, $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$. $\mathbf{U}_{ij}$ is the ij^{th} entry of the upper triangular matrix $\mathbf{U}$, and $\mathbf{D}_{jj}$ the diagonal entry at index jj of the diagonal matrix $\mathbf{D}$.

Eqn.(3.43) and Eqn.(3.41) can therefore be replaced by the above equations. This algorithm keeps the covariance matrix positive definite. If the covariance becomes small anyhow, the diagonal matrix $\mathbf{D}$ is increased by $c\,\mathbf{I}$ with $c > 0$.

3.1.7 Analysis of the recursive algorithm

The prediction error method algorithm was derived in the previous section for a discrete time state space model in controller canonical form. In the following a short analysis of that algorithm is given.

Initial conditions

All recursively computed matrices and vectors have to be initialised once. In most off-line applications, these elements are set equal to zero at the beginning, except for the parameter covariance matrix. This matrix should be large, e.g. $\mathbf{P}(0) = 1000\,\mathbf{I}$, in order to guarantee a large variance of the parameter vector when the algorithm is started. This makes the estimates converge very fast to the true parameters. For the rotor bearing system, however, which is open loop unstable, the choice is more complex and the initial values have to be found by means of trial and error as shown in Section 4.5.1.

Stability of the estimation algorithm

The most interesting problem about the estimation algorithm is its stability. As proved in [Ljung and Søderstrøm, 1983], the stability of the estimation algorithm is given, if

$$\left|\text{eig}\left(\mathbf{A}(\mathbf{p}) - \mathbf{K}(\mathbf{p})\,\mathbf{C}(\mathbf{p})\right)\right| < 1, \tag{3.63}$$

i.e. if the predictor based upon the innovations model defined by Eqn.(3.3) is stable.

Therefore, the eigenvalues of the predictor have to be calculated each time step, when a new estimate has been calculated with a projection algorithm described in the following.

1. Choose a factor $0 \leq \kappa \leq 1$,
2. Compute the differential update of the parameter vector $\hat{\mathbf{p}}$ as

$$\Delta\hat{\mathbf{p}}(k) = \mathbf{L}(k)\,\boldsymbol{\varepsilon}(k), \tag{3.64}$$

3. As long as the predictor is unstable do with $i = 0, 1, \ldots$

$$\hat{\mathbf{p}}(k) = \hat{\mathbf{p}}(k-1) + \kappa^i \Delta\hat{\mathbf{p}}(k), \tag{3.65}$$

until the predictor is stable. Since κ remains as design parameter and the aforementioned algorithm does not guarantee a constant computation time, κ can be chosen equal to zero as well. This means that an observation is not used if it causes the predictor to be unstable and the parameter vector is only updated if the resulting predictor is stable.

Convergence of the estimation algorithm

In Section 3.1.3, the prediction error method was derived as a recursive computation of the system parameter vector based upon a criterion functional of the prediction error. Basically, a negative gradient multiplied by a strictly positive matrix adjusts the parameter vector until a minimum is reached, i.e.: "*The estimate will converge with probability one to a local minimum of* $V(\mathbf{p})$ *as* t *approaches infinity*", [Ljung and Søderstrøm, 1983]. An extensive derivation for the convergence of the algorithm presented can be found in [Ljung and Søderstrøm, 1983].

3.2 Controller design

Once the parameters of the system are estimated, the state space controller can be computed based upon the identified model with the assumption that the system parameters are estimated properly (*certainty equivalence principle*, [Goodwin and Sin, 1984, Isermann et al., 1992]).

In this work a pole placement controller is chosen, because stability has to be guaranteed from the beginning based upon a precalculated model. Another reason for pole placement is that active magnetic bearing systems are viewed as mechanical systems with eigenfrequencies and a certain bandwidth. By means of pole placement specific dynamic characteristics can be applied to the controlled system.

There are two possibilities for a state space controller, simple state feedback with feedforward gain, or state feedback with additional integrative feedback. Both concepts are presented in the following sections.

3.2.1 State space controller

The control law for a state space controller is generally defined as

$$\mathbf{u}(k) = \mathbf{K}_w\, \mathbf{w}(k) - \mathbf{K}_x\, \mathbf{x}(k). \tag{3.66}$$

The controller gain $\mathbf{K}_x = \{\mathbf{k}_x^{T\,(ij)}\}$ is parameterised with $i, j = 1, 2, 3, 4$ by

$$\mathbf{k}_x^{T\,(ij)} = \begin{bmatrix} k_{x_2}^{(ij)} & k_{x_1}^{(ij)} \end{bmatrix}, \tag{3.67}$$

and the feed forward gain matrix is parameterised as $\mathbf{K}_w = \{k_w^{(ij)}\}$ with $i, j = 1, 2, 3, 4$. A block diagram of the presented controller is shown in Fig. 3.3.

The poles of the closed loop system are determined by the closed loop system matrix $\mathbf{A}_{cl} = \mathbf{A} - \mathbf{B}\mathbf{K}_x = \{\mathbf{A}_{cl}^{(ij)}\}$ with its sub-matrices

$$\mathbf{A}_{cl}^{(ij)} = \begin{bmatrix} 0 & 1 \\ a_2^{(ij)} - k_{x_2}^{(ij)} & a_1^{(ij)} - k_{x_1}^{(ij)} \end{bmatrix}, \tag{3.68}$$

$$\mathbf{A}_{cl}^{(ii)} = \begin{bmatrix} 0 & 0 \\ a_2^{(ii)} - k_{x_2}^{(ii)} & a_1^{(ii)} - k_{x_1}^{(ii)} \end{bmatrix}, \tag{3.69}$$

and the system matrices $\mathbf{A}$ and $\mathbf{B}$ being in controller canonical form. The coefficients of the control matrix can be easily calculated in the form [Nazaruddin, 1994]

$$k_{x_m}^{(ij)} = a_m^{(ij)}, \tag{3.70}$$

$$k_{x_m}^{(ii)} = a_m^{(ii)} + p_m^{(ii)}, \tag{3.71}$$

with $m = 1, 2$ and $i, j = 1, 2, 3, 4$. The coefficients $p_m^{(ii)}$ are derived from the desired polynomial for each subsystem of order $\nu_i = 2$ by

$$P^{(ii)}(z) = z^2 + p_1^{(ii)} z + p_2^{(ii)}. \tag{3.72}$$

Here, an advantage of a model description in controller canonical form becomes obvious. All parameters of the state space controller can be calculated directly from the state space model using simple algebra.

The output of the controlled system is

$$\mathbf{Y}(z) = \mathbf{G}_{cl}(z)\, \mathbf{W}(z), \tag{3.73}$$

with the reference transfer function matrix

$$\mathbf{G}_{cl}(z) = \mathbf{C}\, (\mathbf{I}z - \mathbf{A}_{cl})^{-1}\, \mathbf{B}\, \mathbf{K}_w. \tag{3.74}$$

This transfer function matrix contains the closed loop system matrix $\mathbf{A}_{cl} = \{\mathbf{A}_{cl}^{(ij)}\}$ with $i, j = 1, 2, 3, 4$ and its diagonal sub-matrices using Eqn.(3.70) to Eqn.(3.71)

$$\mathbf{A}_{cl}^{(ii)} = \begin{bmatrix} 0 & 1 \\ -p_2^{(ii)} & -p_1^{(ii)} \end{bmatrix}. \tag{3.75}$$

The feed forward matrix $\mathbf{K}_w$ can then be calculated from the steady state condition $\mathbf{G}_{cl}(z)|_{z=1} = \mathbf{I}$ by

$$\mathbf{K}_w = \Big(\mathbf{C}\, (\mathbf{I}z - \mathbf{A}_{cl})^{-1}\, \mathbf{B}\Big)^{-1}\Big|_{z=1} \tag{3.76}$$

The inverse of $(\mathbf{I} - \mathbf{A}_{cl})$ can be calculated in advance. Only Eqn.(3.76) has to be evaluated at every sample time interval. If the set point is equal to zero, the feed forward gain can be omitted at all.

3.2.2 State space controller with integrative feedback

The control law for a state space controller with integrative feedback is defined as

$$\begin{aligned} \mathbf{u}(k) &= \mathbf{u}_I(k) - \mathbf{K}_x\,\mathbf{x}(k), && (3.77)\\ \mathbf{u}_I(k+1) &= \mathbf{u}_I(k) + \mathbf{K}_I\,\mathbf{e}(k), && (3.78)\\ \mathbf{e}(k) &= \mathbf{w}(k) - \mathbf{y}(k). && (3.79) \end{aligned}$$

with the integrative feedback gain matrix $\mathbf{K}_I = \{k_I^{(ij)}\}$ and $i, j = 1, 2, 3, 4$.

Usually, the set point error is integrated first, and the resulting integral is weighted by an integrative gain [Jörgl, 1994]. In the presented control concept, an additional control variable is integrated with a the weighted set point error as input. The implementation of the integrator using the backwards difference method [Ogata, 1987] in Eqn.(3.78) is used for the sake simplicity. A block diagram of the presented controller is shown in Fig. 3.4.

Using the transformation into the z-domain, Eqn.(3.78) yields

$$\mathbf{U}_I(z) = \frac{1}{z-1}\mathbf{K}_I\,\mathbf{E}(z). \tag{3.80}$$

With the assumption that all system parameters and all states are estimated properly, the deterministic state space model used for controller design is

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\,\mathbf{x}(k) + \mathbf{B}\,\mathbf{u}(k), && (3.81)\\ \mathbf{y}(k) &= \mathbf{C}\,\mathbf{x}(k). && (3.82) \end{aligned}$$

Applying the z-transformation for Eqn.(3.77), Eqn.(3.81) and Eqn.(3.82), the transformed output of the controlled system with simple state feedback results in

$$\mathbf{Y}(z) = \mathbf{C}\Big(z\,\mathbf{I} - \mathbf{A} + \mathbf{B}\,\mathbf{K}_x\Big)^{-1}\mathbf{B}\,\mathbf{U}_I(z). \tag{3.83}$$

Inserting the z-transformed Eqn.(3.79) and Eqn.(3.80), the output remains as function depending on $\mathbf{W}(z)$ and $\mathbf{Y}(z)$ such that

$$\mathbf{Y}(z) = \mathbf{N}\,\mathbf{D}^{-1}\,\frac{1}{z-1}\,\mathbf{K}_I\,\left(\mathbf{W}(z) - \mathbf{Y}(z)\right), \tag{3.84}$$

with the convention making use of the controller canonical form [Keuchel, 1988]

$$\mathbf{N}\,\mathbf{D}^{-1} = \mathbf{C}\Big(z\,\mathbf{I} - \mathbf{A} + \mathbf{B}\,\mathbf{K}_x\Big)^{-1}\mathbf{B}. \tag{3.85}$$

A derivation of the above convention is given in Appendix C.1.

Matrix $\mathbf{D}$ is a diagonal matrix and can be interpreted as the characteristic polynomial matrix of the closed loop system without integrative feedback

$$\mathbf{D} = \operatorname{diag} P^{(ii)}(z) \quad \text{with} \quad i = 1, 2, 3, 4, \tag{3.86}$$

with the same polynomials as defined in Eqn.(3.72). The matrix $\mathbf{N}$ depends on the measurement matrix $\mathbf{C}$ such that $\mathbf{N} = \{\mathbf{N}^{(ii)}\}$ with

$$\mathbf{N}^{(ij)} = \left[\begin{array}{cc} c_2^{(ij)} & c_1^{(ij)}\, z \end{array} \right]. \tag{3.87}$$

Solving Eqn.(3.84) for $\mathbf{Y}(z)$ yields

$$\mathbf{Y}(z) = \mathbf{G}_{cl_I}(z)\mathbf{W}(z) \tag{3.88}$$

with the reference transfer function matrix

$$\mathbf{G}_{cl_I}(z) = \mathbf{N}\,\mathbf{D}_I^{-1}\,\mathbf{K}_I \tag{3.89}$$

and with the closed loop polynomial matrix

$$\mathbf{D}_I = (z-1)\,\mathbf{D} + \mathbf{K}_I\,\mathbf{N}. \tag{3.90}$$

A derivation for the preceding equation is given in Appendix C.2.

The essence of the controller design is that the matrix $\mathbf{D}_I$ is set equal to a desired polynomial matrix $\mathbf{P}_I$, which is a diagonal matrix with the elements $P_I^{(ii)}(z)$ of order $\nu_i + 1$ such as

$$\mathbf{P}_I = \operatorname{diag} P_I^{(ii)}(z) \quad \text{with} \quad i = 1,2,3,4 \tag{3.91}$$

with the desired polynomials for the closed loop system with integrative feedback

$$P_I^{(ii)}(z) = z^3 + p_{I_1}^{(ii)} z^2 + p_{I_2}^{(ii)} z + p_{I_3}^{(ii)}. \tag{3.92}$$

This matrix has to be chosen in advance. The gain for the integrative feedback path $\mathbf{K}_I$ can be determined from Eqn.(3.89) for the stationary case with the desired closed loop polynomial matrix $\mathbf{P}_I$ from the steady state condition $\mathbf{G}_{cl_I}(z)|_{z=1} = \mathbf{I}$ by

$$\mathbf{K}_I = \mathbf{P}_I(z)\,\mathbf{N}^{-1}(z)\Big|_{z=1}. \tag{3.93}$$

What remains is the computation of the polynomials in $\mathbf{D}$, or the desired polynomials $P^{(ii)}(z)$ of the closed loop system without integrative feedback, i.e. heuristically speaking, how to chose the polynomials $P^{(ii)}(z)$ in order to achieve the closed loop polynomials $P_I^{(ii)}(z)$ based upon the integrative feed back gain $\mathbf{K}_I$. Eqn.(3.90) contains the necessary conditions.

Without claim on generality,the following solution derived in Appendix C.3 is proposed

$$p_1^{ii} = p_{I_1}^{ii} + 1, \tag{3.94}$$

$$p_2^{ii} = \sum_{j=1}^{4} c_2^{ji}\, k_w^{(ij)} - p_{I_3}^{ii}. \tag{3.95}$$

With the thus computed polynomials for the state space controller without integrative feedback, the matrix $\mathbf{K}_x$ can be determined according to Eqn.(3.70) and Eqn.(3.71).

3.3 Closed loop eigenvalues

Using the *certainty equivalence principle* the estimated parameters are assumed to be true [Goodwin and Sin, 1984, Isermann et al., 1992]. Therefore, the estimation algorithm, which is the *meta-controller* with the slow states, is omitted in this subsection. Only the fast states are objective for the stability analysis for each simple state feedback and state space control with additional integrative feedback.

3.3.1 P-structure

The recursive equations for the controlled system, the predictor and the controller are given by the following equations

$$\begin{aligned}
\mathbf{x}(k+1) &= \mathbf{A}\,\mathbf{x}(k) + \mathbf{B}\,\mathbf{u}(k), && (3.96)\\
\mathbf{y}(k) &= \mathbf{C}\,\mathbf{x}(k) + \mathbf{v}(k), && (3.97)\\
\hat{\mathbf{x}}(k+1) &= \hat{\mathbf{A}}\,\hat{\mathbf{x}}(k) + \mathbf{B}\,\mathbf{u}(k) + \hat{\mathbf{K}}\,\boldsymbol{\varepsilon}(k), && (3.98)\\
\hat{\mathbf{y}}(k) &= \hat{\mathbf{C}}\,\hat{\mathbf{x}}(k), && (3.99)\\
\mathbf{u}(k) &= \mathbf{K}_w\,\mathbf{w}(k) - \mathbf{K}_x\,\hat{\mathbf{x}}(k), && (3.100)\\
\boldsymbol{\varepsilon}(k) &= \mathbf{y}(k) - \hat{\mathbf{y}}(k). && (3.101)
\end{aligned}$$

Fig. 3.3 shows a block diagram of the entire linear control system.

The above equations can be rewritten in the form

$$\bar{\mathbf{x}}(k+1) = \bar{\mathbf{A}}\,\bar{\mathbf{x}}(k) + \bar{\mathbf{B}}\,\bar{\mathbf{u}}(k), \tag{3.102}$$

with the new system states and inputs

$$\bar{\mathbf{x}}(k) = \left[\begin{array}{c} \mathbf{x}(k) \\ \hline \hat{\mathbf{x}}(k) \end{array}\right] \quad \text{and} \quad \bar{\mathbf{u}}(k) = \left[\begin{array}{c} \mathbf{w}(k) \\ \hline \mathbf{v}(k) \end{array}\right], \tag{3.103}$$

and the system matrices

$$\bar{\mathbf{A}} = \left[\begin{array}{c|c} \mathbf{A} & -\mathbf{B}\,\mathbf{K}_x \\ \hline \hat{\mathbf{K}}\,\mathbf{C} & \hat{\mathbf{A}} - \mathbf{B}\,\mathbf{K}_x - \hat{\mathbf{K}}\,\hat{\mathbf{C}} \end{array}\right], \tag{3.104}$$

$$\bar{\mathbf{B}} = \left[\begin{array}{c|c} \mathbf{B}\,\mathbf{K}_w & \mathbf{0} \\ \hline \mathbf{B}\,\mathbf{K}_w & \hat{\mathbf{K}} \end{array}\right]. \tag{3.105}$$

Utilising the equivalence of the "hatted" matrices and the true ones, ($\hat{\mathbf{A}} = \mathbf{A}$, $\hat{\mathbf{C}} = \mathbf{C}$) the separation principle [Jörgl, 1994] holds and

$$\det(z\,\mathbf{I} - \bar{\mathbf{A}}) = \underbrace{\det(z\,\mathbf{I} - \mathbf{A} + \mathbf{B}\,\mathbf{K}_x)}_{\text{controller}} \cdot \underbrace{\det(z\,\mathbf{I} - \mathbf{A} + \mathbf{K}\,\mathbf{C})}_{\text{predictor}}. \tag{3.106}$$

The eigenvalues then are the ones from the predictor which are inside the unit circle, and the eigenvalues determined by the control design.

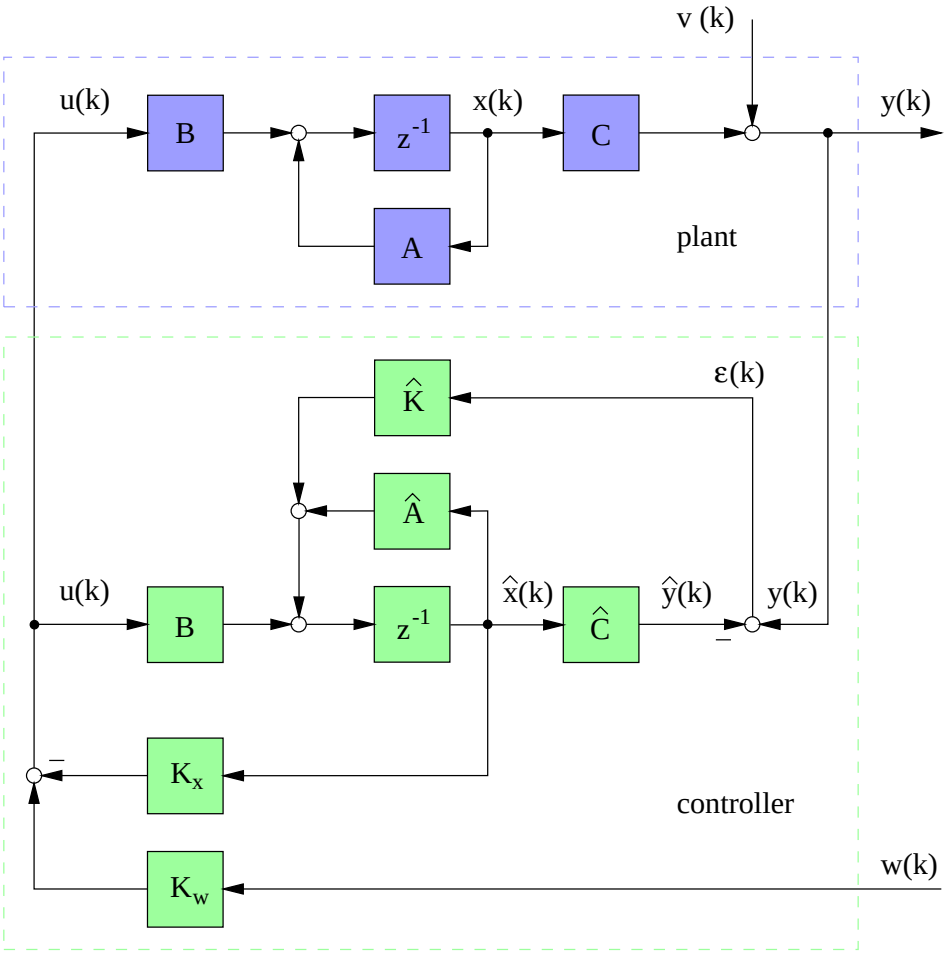

Figure 3.3: The block diagram of the system controlled by a conventional state space controller.

3.3.2 PI-structure

The recursive equations for the controlled system, the predictor and the controller with integrative feedback are given by the following equations as

$$\begin{aligned}
\mathbf{x}(k+1) &= \mathbf{A}\,\mathbf{x}(k) + \mathbf{B}\,\mathbf{u}(k), && (3.107)\\
\mathbf{y}(k) &= \mathbf{C}\,\mathbf{x}(k) + \mathbf{v}(k), && (3.108)\\
\hat{\mathbf{x}}(k+1) &= \hat{\mathbf{A}}\,\hat{\mathbf{x}}(k) + \mathbf{B}\,\mathbf{u}(k) + \hat{\mathbf{K}}\,\boldsymbol{\varepsilon}(k), && (3.109)\\
\hat{\mathbf{y}}(k) &= \hat{\mathbf{C}}\,\hat{\mathbf{x}}(k), && (3.110)\\
\mathbf{u}(k) &= \mathbf{u}_I(k) - \mathbf{K}_x\,\hat{\mathbf{x}}(k), && (3.111)\\
\mathbf{u}_I(k+1) &= \mathbf{u}_I(k) + \mathbf{K}_I\,\mathbf{e}(k), && (3.112)\\
\mathbf{e}(k) &= \mathbf{w}(k) - \mathbf{y}(k), && (3.113)\\
\boldsymbol{\varepsilon}(k) &= \mathbf{y}(k) - \hat{\mathbf{y}}(k). && (3.114)
\end{aligned}$$

Fig. 3.4 shows a block diagram of the entire linear control system with integrative feedback.

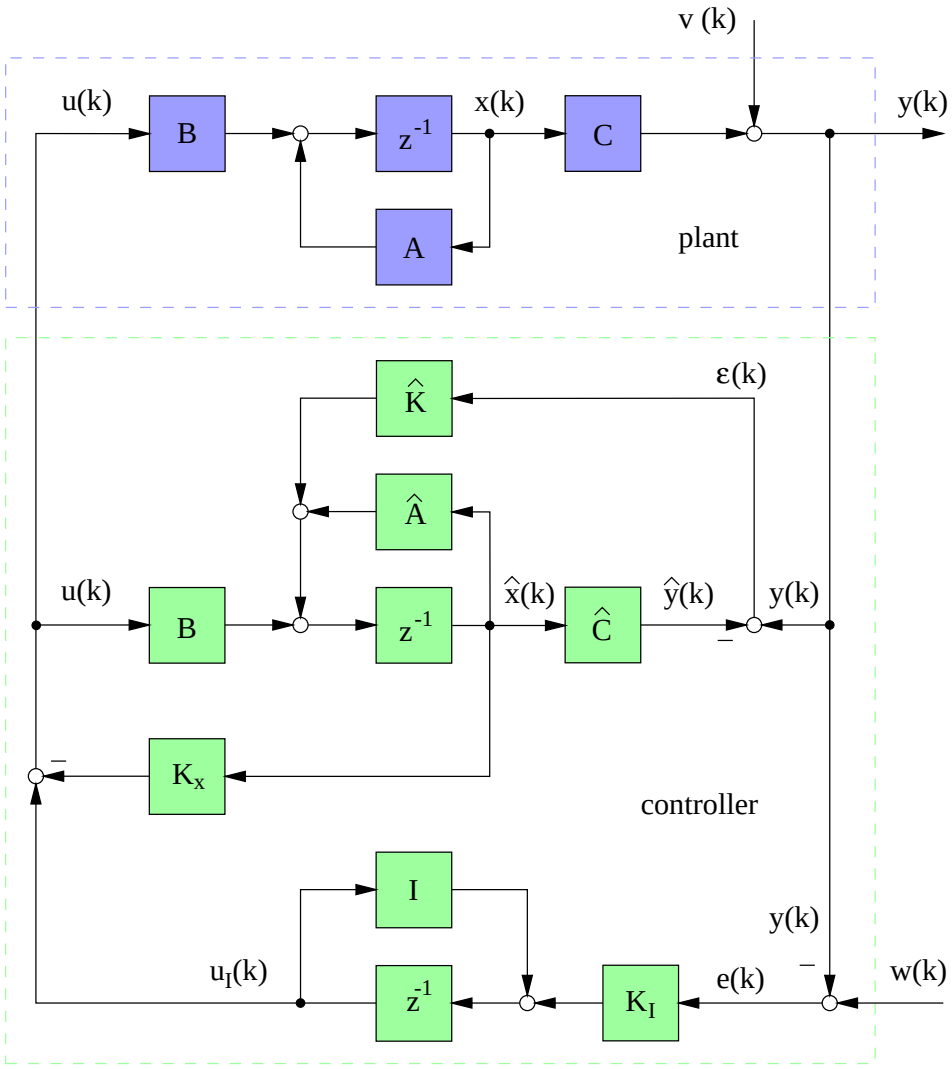

Figure 3.4: The block diagram of the system controlled by a state space controller with additional integrative feedback.

The above equations can be rewritten in the form

$$\bar{\mathbf{x}}(k+1) = \bar{\mathbf{A}}\,\bar{\mathbf{x}}(k) + \bar{\mathbf{B}}\,\bar{\mathbf{u}}(k), \tag{3.115}$$

with the new system states and inputs

$$\bar{\mathbf{x}}(k) = \left[\begin{array}{c} \mathbf{x}(k) \\ \hline \mathbf{u}_I(k) \\ \hline \hat{\mathbf{x}}(k) \end{array}\right] \quad \text{and} \quad \bar{\mathbf{u}}(k) = \left[\begin{array}{c} \mathbf{w}(k) \\ \hline \hat{\mathbf{v}}(k) \end{array}\right], \tag{3.116}$$

and the system matrices

$$\bar{\mathbf{A}} = \left[\begin{array}{c|c|c} \mathbf{A} & \mathbf{B} & -\mathbf{B}\,\mathbf{K}_x \\ \hline -\mathbf{K}_I\,\mathbf{C} & \mathbf{I} & \mathbf{0} \\ \hline \hat{\mathbf{K}}\,\mathbf{C} & \mathbf{B} & \hat{\mathbf{A}} - \mathbf{B}\,\mathbf{K}_x - \hat{\mathbf{K}}\,\hat{\mathbf{C}} \end{array}\right], \tag{3.117}$$

$$\bar{\mathbf{B}} = \left[\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline \mathbf{K}_I & -\mathbf{K}_I \\ \hline \mathbf{0} & \hat{\mathbf{K}} \end{array}\right]. \tag{3.118}$$

Utilising the equivalence of the "hatted" matrices and the true ones, ($\hat{\mathbf{A}} = \mathbf{A}$, $\hat{\mathbf{C}} = \mathbf{C}$) the separation principle [Jörgl, 1994] holds and

$$\det(z\,\mathbf{I} - \bar{\mathbf{A}}) = \underbrace{\det(z\,\mathbf{I} - \tilde{\mathbf{A}})}_{\text{controller}} \cdot \underbrace{\det(z\,\mathbf{I} - \mathbf{A} + \mathbf{K}\,\mathbf{C})}_{\text{predictor}}, \tag{3.119}$$

and the extended system matrix with integrative feedback

$$\tilde{\mathbf{A}} = \left[\begin{array}{c|c} \mathbf{A} - \mathbf{B}\,\mathbf{K}_x & \mathbf{B} \\ \hline -\mathbf{K}_I\,\mathbf{C} & \mathbf{I} \end{array}\right]. \tag{3.120}$$

The eigenvalues of the entire system are the eigenvalues of the predictor, and the eigenvalues for the state space controlled system with additional integrative feedback.