# PREDICTION ERROR METHODS

David Di Ruscio
Telemark University College
N-3414 Porsgrunn, Norway

Revised: March 22, 2024

March 22, 2024

## Contents

# 1 Introduction

An overview of the Prediction Error Method (PEM) for system identification is given. Furthermore, a description of the common discrete time input and output polynomial model structures, which are frequently used in prediction error system identification methods, is presented in this note. The relationship between the state space model and the input and output model is in particular pointed out.

# 2 Overview of the prediction error methods

Given the state space model on innovations form.

$$\begin{aligned} \bar{x}_{k+1} &= A\bar{x}_k + Bu_k + Ke_k, && (1)\\ y_k &= \underbrace{D\bar{x}_k + Eu_k}_{\bar{y}_k} + e_k, && (2) \end{aligned}$$

where $\Delta = \mathrm{E}(e_k e_k^T)$ is the covariance matrix of the innovations process and $\bar{x}_1$ is the initial predicted state. Suppose now that the model is parameterized, i.e. so that the free parameters in the model matrices $(A, B, K, D, E, x_1)$ are organized into a parameter vector $\theta$. The problem is to identify the "best" parameter vector from known output and input data matrices $(Y, U)$. The optimal predictor, i.e. the optimal prediction, $\bar{y}_k$, for the output $y_k$, is then of the form

$$\begin{aligned} \bar{x}_{k+1} &= (A - KD)\bar{x}_k + (B - KE)u_k + Ky_k, && (3)\\ \bar{y}_k(\theta) &= D\bar{x}_k + Eu_k, && (4) \end{aligned}$$

with initial predicted state $\bar{x}_1$. $\bar{y}_k(\theta)$ is the prediction of the output $y_k$ given inputs $u$ up to time $k$, outputs $y$ up to time $k-1$ and the parameter vector $\theta$. Note that if $E = 0_{m\times r}$ then inputs $u$ only up to $k-1$ are needed. The free

parameters in the system matrices are mapped into the parameter vector (or visa versa). Note that the predictor $\bar{y}_k(\theta)$ is (only) optimal for the parameter vector $\theta$ which minimize some specified criterion. This criterion is usually a function of the *prediction errors*. Note also that it is common to use the notation $\bar{y}_{k|\theta}$ for the prediction. Hence, $\bar{y}_{k|\theta} = \bar{y}_k(\theta)$.

Define the Prediction Error (PE)

$$\epsilon_k(\theta) = y_k - \bar{y}_k(\theta). \tag{5}$$

A good model is a model for which the model parameters $\theta$ results in a "small" PE. Hence, it make sense to use a PE criterion which measure the size of the PE. A PE criterion is usually always in one ore another way defined as a scalar function of the following important expression and definition

$$R_\epsilon(\theta) = \frac{1}{N}\sum_{k=1}^{N} \epsilon_k(\theta)\epsilon_k(\theta)^T \ \in \ \mathbb{R}^{m\times m}, \tag{6}$$

which is the sample covariance matrix of the PE. This means that we want to find the parameter vector which make $R_\epsilon$ as small as possible. Hence, it make sense to use a PE criterion which measure the size of the sample covariance matrix of the PE. A common scalar PE criterion for multivariable output systems is thus

$$V_N(\theta) = \mathrm{tr}(\frac{1}{N}\sum_{k=1}^{N} \epsilon_k(\theta)\epsilon_k(\theta)^T) = \frac{1}{N}\sum_{k=1}^{N} \epsilon_k(\theta)^T\epsilon_k(\theta). \tag{7}$$

Note that the trace of a matrix is equal to the sum of its diagonal elements and that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ of two matrices $A$ and $B$ of appropriate dimensions. We will in the following give a discussion of the PE criterion as well as some variants of it. Define a PE criterion as follows

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N} \ell(\epsilon_k(\theta)) \tag{8}$$

where $\ell(\cdot)$ is a scalar valued function, e.g. the Euclidean $l_2$ norm, i.e.

$$\ell(\epsilon_k(\theta)) = \| \ \epsilon_k(\theta) \ \|_2^2 = \epsilon_k(\theta)^T\epsilon_k(\theta), \tag{9}$$

or a quadratic function

$$\ell(\epsilon_k(\theta)) = \epsilon_k(\theta)^T\Lambda\epsilon_k(\theta) = \mathrm{tr}(\Lambda\epsilon_k(\theta)\epsilon_k(\theta)^T), \tag{10}$$

for some weighting matrix $\Lambda$. A common criterion for multiple output systems (with weights) is thus also

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N} \epsilon_k(\theta)^T\Lambda\epsilon_k(\theta) = \mathrm{tr}(\Lambda(\frac{1}{N}\sum_{k=1}^{N} \epsilon_k(\theta)\epsilon_k(\theta)^T)), \tag{11}$$

where we usually simply are using $\Lambda = I$. The weighting matrix $\Lambda$ is usually a diagonal and positive matrix.

One should note that there exist an optimal weighting matrix $\Lambda$, but that this matrix is difficult to define a-priori. The optimal weighting (when the number of observations $N$ is large) is defined from the knowledge of the innovations noise covariance matrix of the system, i.e., $\Lambda = \Delta^{-1}$ where $\Delta = \mathrm{E}(e_k e_k^T)$. An interesting solution to this would be to define the weighting matrix from the subspace system identification method DSR or DSR_e and use $\Delta = FF^T$ where $F$ is computed by the DSR algorithm.

The sample covariance matrix of the PE is positive semi-definite, i.e. $R_\epsilon \geq 0$. The PE may be zero for deterministic systems however for combined deterministic and stochastic systems we usually have that $R_\epsilon > 0$, i.e. positive definite. This means off-course in any case that $R_\epsilon$ is a symmetric matrix. The eigenvalues of a symmetric matrix are all real. Define $\lambda_1, \ldots, \lambda_m$ as the eigenvalues of $R_\epsilon(\theta)$ for use in the following discussion.

Hence, a good parameter vector, $\theta$, is such that the sample covariance matrix $R_\epsilon$ is small. The trace operator in (11), i.e. the PE criterion

$$V_N(\theta) = \mathrm{tr}(R_\epsilon(\theta)) = \lambda_1 + \ldots + \lambda_m, \tag{12}$$

is a measure of the size of the matrix $R_\epsilon(\theta)$. Hence, the trace of a matrix is equal to the sum of the diagonal elements of the matrix, but the trace is also equal to the sum of the eigenvalues of a symmetric matrix.

An alternative PE criterion which often is used is the determinant, i.e.,

$$V_N(\theta) = \det(R_\epsilon(\theta)) = \lambda_1 \lambda_2 \ldots \lambda_m. \tag{13}$$

Hence, the determinant of the matrix is equal to the product of its eigenvalues. This lead us to a third alternative, which is to use the maximum eigenvalue of $R_\epsilon(\theta)$ as a measure of its size, i.e., we may use the following PE criterion

$$V_N(\theta) = \lambda_{\max}(R_\epsilon(\theta)), \tag{14}$$

where $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of a symmetric matrix.

Note the special case for a single output system, then we have

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N} \epsilon_k(\theta)^2 = \frac{1}{N}\sum_{k=1}^{N} (y_k - \bar{y}_k(\theta))^2. \tag{15}$$

The minimizing parameter vector is defined by

$$\hat{\theta}_N = \arg\min_{\theta \in D_{\mathcal{M}}} V_N(\theta) \tag{16}$$

where arg min denotes the operator which returns the argument that minimizes the function. The subscript $N$ is often omitted, hence $\hat{\theta} = \hat{\theta}_N$ and $V(\theta) = V_N(\theta)$.

The definition (16) is a standard optimization problem. A simple solution is then to use a software optimization algorithm which is dependent only on function evaluations, i.e., where the user only have to define the PE criterion $V(\theta)$.

We will in the following give a discussion of how this may be done. The minimizing parameter vector $\hat{\theta} = \hat{\theta}_N \in \mathbb{R}^p$ has to be searched for in the parameter space $D_{\mathcal{M}}$ by some iterative non-linear optimization method. Optimization methods are usually constructed as variations of the Gauss-Newton method and the Newton-Raphson method, i.e.,

$$\theta_{i+1} = \theta_i - \alpha H_i^{-1}(\theta_i) g_i(\theta_i) \tag{17}$$

where $\alpha$ is defined as a line search (ore step length) scalar parameter chosen to ensure convergence (i.e. chosen to ensure that $V(\theta_{i+1} < V(\theta_i))$, and where the gradient, $g_i(\theta_i)$, is

$$g_i(\theta_i) = \frac{dV(\theta_i)}{d\theta_i} \quad \in \quad \mathbb{R}^p, \tag{18}$$

and

$$H_i(\theta_i) = \frac{dg_i(\theta_i)}{d\theta_i^T} = \frac{d}{d\theta_i^T}(\frac{dV(\theta_i)}{d\theta_i}) = \frac{d^2V(\theta_i)}{d\theta_i^T d\theta_i} \quad \in \quad \mathbb{R}^{p\times p}, \tag{19}$$

is the Hessian matrix. Remark that the Hessian is a symmetric matrix and that it is positive definite in the minimum, i.e.,

$$H(\hat{\theta}) = \frac{d^2V(\hat{\theta})}{d\hat{\theta}^T d\hat{\theta}} > 0, \tag{20}$$

where $\hat{\theta} = \hat{\theta}_N$ is the minimizing parameter vector. Note that the iteration scheme (17) is identical to the Newton-Raphson method when $\alpha = 1$. In practice one often have to use a variable step length parameter $\alpha$, both in order to stabilize the algorithm and to improve the rate of convergence far from the minimum. Once the gradient, $g_i(\theta_i)$ and the Hessian matrix $H_i(\theta_i)$ (or an approximation of the Hessian) have been computed, we can chose the line search parameter, $\alpha$, as

$$\alpha = \arg\min_{\alpha} V_N(\theta_{i+1}(\alpha)) = \arg\min_{\alpha}(\theta_i - \alpha H_i^{-1}(\theta_i) g_i(\theta_i)). \tag{21}$$

Equation (21) is an optimization problem for the line search parameter $\alpha$. Once $\alpha$ have been determined from the scalar optimization problem (21), the new parameter vector $\theta_{i+1}$ is determined from (17).

The iteration process (17) must be initialized with an initial parameter vector, $\theta_1$. This was earlier (before the subspace methods) a problem. However, a good solution is to use the parameters from a subspace identification method. Equation (17) can then be implemented in a **while** or **for** loop. That is to iterate Equation

(17) for $i = 1, \ldots,$ until convergence, i.e., until the gradient is sufficiently zero, i.e., until $g_i(\theta_i) \approx 0$ for some $i \geq 1$. This, and only such a parameter vector is our estimate, i.e. $\hat{\theta} = \hat{\theta}_N = \theta_i$ when $g(\theta_i) \approx 0$.

The iteration equation (17) can be deduced from the fact that in the minimum we have that the gradient, $g$, is zero, i.e.,

$$g(\hat{\theta}) = \frac{dV(\hat{\theta})}{d\hat{\theta}} = 0. \tag{22}$$

We can now use the Newton-Raphson method, which can be deduced as follows. An expression of $g(\hat{\theta})$ can be defined from a Taylor series expansion of $g(\theta)$ around $\theta$, i.e.,

$$0 = g(\hat{\theta}) \approx g(\theta) + \left. \frac{dg(\theta)}{d\theta^T} \right|_{\theta} (\hat{\theta} - \theta). \tag{23}$$

Using this and that $g(\hat{\theta}) = 0$ gives

$$\hat{\theta} = \theta - \left( \left. \frac{dg(\theta)}{d\theta^T} \right|_{\theta} \right)^{-1} g(\theta). \tag{24}$$

This equation is the background for the iteration scheme (17), i.e., putting $\theta := \theta_i$ and $\hat{\theta} := \theta_{i+1}$. Hence,

$$\theta_{i+1} = \theta_i - \left( \left. \frac{dg(\theta)}{d\theta^T} \right|_{\theta_i} \right)^{-1} g(\theta_i). \tag{25}$$

Note that we in (17) has used the shorthand notation

$$\frac{dg(\theta_i)}{d\theta_i^T} = \left. \frac{dg(\theta)}{d\theta^T} \right|_{\theta_i}. \tag{26}$$

Note that the parameter vector, $\theta$, the gradient, $g$, and the Hessian matrix have structures as follows

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix} \in \mathbb{R}^p, \tag{27}$$

$$g(\theta) = \frac{dV(\theta)}{d\theta} = \begin{bmatrix} g_1 \\ \vdots \\ g_p \end{bmatrix} = \begin{bmatrix} \frac{dV}{d\theta_1} \\ \vdots \\ \frac{dV}{d\theta_p} \end{bmatrix} \in \mathbb{R}^p, \tag{28}$$

$$H = \frac{dg(\theta)}{d\theta^T} = \frac{d^2V(\theta)}{d\theta^T d\theta} = \begin{bmatrix} \frac{dg_1}{d\theta_1} & \cdots & \frac{dg_1}{d\theta_p} \\ \vdots & \ddots & \vdots \\ \frac{dg_p}{d\theta_1} & \cdots & \frac{dg_p}{d\theta_p} \end{bmatrix} = \begin{bmatrix} \frac{d^2V}{d\theta_1^2} & \cdots & \frac{d^2V}{d\theta_p d\theta_1} \\ \vdots & \ddots & \vdots \\ \frac{d^2V}{d\theta_1 d\theta_p} & \cdots & \frac{d^2V}{d\theta_p^2} \end{bmatrix} \in \mathbb{R}^{p \times p}. \tag{29}$$

The gradient and the Hessian can be computed numerically. However, it is usually more efficient by an analytically computation if possible. Remark that a common notation of the Hessian matrix is

$$H = \frac{d^2V(\theta)}{d\theta^2}, \tag{30}$$

and that the elements in the Hessian is given by

$$h_{ij} = \frac{\partial^2 V(\theta)}{\partial\theta_i \partial\theta_j}, \tag{31}$$

where $\theta_i$ and $\theta_j$ are parameter number $i$ and $j$, respectively, in the parameter vector $\theta$.

The iteration process (17) is guaranteed to converge to a local minimum, at least theoretically. There may exist many local minima. However, our experience is that the parameters from an estimated model from a subspace method is very close to the minimum. Hence, this initial choice for $\theta_1$ should always be considered first. For systems with many outputs, many inputs and many states there may be a huge number of parameters. The optimization problem may in some circumstances be so complicated that the process (17) diverges even when the initial parameter $\theta_1$ is close to minimum, due to numerical problems. Another problem with PEM is the model parameterization (canonical form) for systems with many outputs. One need to specify a canonical form of the state space model, i.e. a state space realization where there are as few free parameters as possibile in the model matrices $(A, B, D, E, K, x_1)$. A problem for multiple output systems is that there may not even exist such a canonical form. Another problem is that the PE criterion, $V_N(\theta)$, may be almost in-sensitive to perturbations in the parameter vector, $\theta$, for the specified canonical form. Hence, the optimization problem may be ill-conditioned.

An advantage of the PE methods is that it does not matter if the data ($Y$, $U$) is collected from closed loop or open loop process operation. It is however necessary that for open loop experiments, that the inputs, $u_k$, are rich enough for the specified model structure (usually the model order, $n$). For closed loop experiments we must typically require that the feedback is not to simple, e.g. not a simple proportional controller $u_k = K_p y_k$. However, feedback of the type $u_k = K_p(r_k - y_k)$ where the reference, $r_k$, is perturbed gives data which are informative enough.

## 2.1 Further remarks on the PEM

Note also that in the multivariable case we have that the parameter estimate

$$\hat{\theta}_N = \arg\min_{\theta} \operatorname{tr}(\Lambda R_\epsilon(\theta)), \tag{32}$$

with $\Lambda = \Delta^{-1}$ where $\Delta = (\mathrm{E}(e_k e_k^T))$, has the same asymptotic covariance matrix as the parameter estimate

$$\hat{\theta}_N = \arg\min_{\theta} \det(R_\epsilon(\theta)). \tag{33}$$

It can also be shown that the PEM parameter estimate for Gaussian distributed disturbances, $e_k$, (33), or equivalently the PEM estimate (32) with the optimal weighting, is identical to the Maximum Likelihood (ML) parameter estimate. The PEM estimates (32) or (33) are both statistically optimal. The only drawback by using (33), i.e., minimizing the determinant PE criterion $V_N(\theta) = \det(R_\epsilon(\theta))$, is that it requires more numerical calculations than the trace criterion. However, on the other side the evaluation of the PE criterion (32) with the optimal weighting matrix $\Lambda = \Delta^{-1}$ is not (directly) realistic since the exact covariance matrix $\Delta$ is not known in advance. However, note that an estimate of $\Delta$ can be built up during the iteration (optimization) process.

The parameter estimate $\hat{\theta}_N$ is a random vector, i.e., Gaussian distributed when $e_k$ is Gaussian. This means that $\hat{\theta}_N$ has a mean and a variance. We want the mean to be as close to the true parameter vector as possibile and the variance to be as small as possible. We can show that the PEM estimate $\hat{\theta}_N$ is consistent, i.e., the mean of the parameter estimate, $\hat{\theta}_N$, converges to the true parameter vector, $\theta_0$, as $N$ tends to infinity. In other words we have that $\mathrm{E}(\hat{\theta}_N) = \theta_0$. Consider $g(\hat{\theta}_N) = 0$ expressed as a Taylor series expansion of $g(\theta_0)$ around the true parameter vector $\theta_0$, i.e.

$$0 = g(\hat{\theta}_N) \approx g(\theta_0) + H(\theta_0)(\hat{\theta}_N - \theta_0). \tag{34}$$

where the Hessian matrix

$$H(\theta_0) = \left. \frac{dg(\theta)}{d\theta} \right|_{\theta_0}, \tag{35}$$

is a deterministic (constant) matrix. However, the gradient, $g(\theta_0)$, is a random vector with zero mean and covariance matrix $P_0$. From this we have that the difference between our estimate, $\hat{\theta}_N$ and the true vector $\theta_0$ is given by

$$\hat{\theta}_N - \theta_0 = -H^{-1}(\theta_0) g(\theta_0). \tag{36}$$

Hence,

$$\mathrm{E}(\hat{\theta}_N - \theta_0) = -H^{-1}(\theta_0)\mathrm{E}(g(\theta_0)) = 0. \tag{37}$$

This shows consistency of the parameter estimate, i.e.

$$\mathrm{E}(\hat{\theta}_N) = \theta_0, \tag{38}$$

because $\theta_0$ is deterministic.

The parameter estimates (33) and (32) with optimal weighting are efficient, i.e., they ensures that the parameter covariance matrix

$$P = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = H^{-1}(\theta_0)\mathrm{E}(g(\theta_0)g(\theta_0)^T)H^{-1}(\theta_0) \tag{39}$$

is minimized. The covariance matrix $P$ may be estimated from the data by evaluating (39) numerically by using the approximation $\theta_0 \approx \hat{\theta}_N$.

For single output systems ($m = 1$) we have that the parameter covariance matrix, $P$, can be expressed as

$$P = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = \Delta(\mathrm{E}(\psi_k(\theta_0)\psi_k^T(\theta_0)))^{-1}, \tag{40}$$

where $\Delta = \mathrm{E}(e_k e_k^T) = \mathrm{E}(e_k^2)$ in the single output case, and with

$$\psi_k(\theta_0) = \frac{d\bar{y}_k(\theta)}{d\theta}\mid_{\theta_0} \in \mathbb{R}^{p\times m} \tag{41}$$

Loosely spoken, Equation (40) states that the variance, $P$, of the parameter estimate, $\hat{\theta}_N$, is "small" if the covariance matrix of $\psi_k(\theta_0)$ is large. This covariance matrix is large if the predictor $\bar{y}_k(\theta)$ is "very" sensitive to (perturbations in) the parameter vector $\theta$.

## 2.2 Derivatives of the prediction error criterion

Consider the PE criterion (8) with (10), i.e.,

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N}\ell(\epsilon_k(\theta)) = \frac{1}{N}\sum_{k=1}^{N}\overbrace{\epsilon_k^T(\theta)\Lambda\epsilon_k(\theta)}^{\ell(\epsilon_k(\theta))}. \tag{42}$$

The derivative of the scalar valued function $\ell = \ell(\epsilon_k(\theta))$ with respect to the parameter vector $\theta$ can be expressed from the chain rule

$$\frac{\partial\ell(\epsilon_k(\theta))}{\partial\theta} = \frac{\partial\epsilon_k}{\partial\theta}\frac{\partial\ell(\epsilon_k(\theta))}{\partial\epsilon_k} = -\frac{\partial\bar{y}_k(\theta)}{\partial\theta}2\Lambda\epsilon_k. \tag{43}$$

Define the gradient matrix of the predictor $\bar{y}_k \in \mathbb{R}^{\times m}$ with respect to the parameter vector $\theta \in \mathbb{R}^{\times p}$ as

$$\psi_k(\theta) = \frac{\partial\bar{y}_k(\theta)}{\partial\theta} \in \mathbb{R}^{p\times m}. \tag{44}$$

This gives the following general expression for the gradient, i.e.,

$$g(\theta) = \frac{\partial V_N(\theta)}{\partial\theta} = \frac{1}{N}\sum_{k=1}^{N}\frac{\partial\ell(\epsilon_k(\theta))}{\partial\theta} = -2\frac{1}{N}\sum_{k=1}^{N}\psi_k(\theta)\Lambda\epsilon_k \tag{45}$$

## 2.3 Least Squares and the prediction error method

The optimal predictor, $\bar{y}_k(\theta)$, is generally a non-linear function of of the unknown parameter vector, $\theta$. The PEM estimate can in this case in generally not be solved analytically. However, in some simple and special cases we have that the predictor is a linear function of the parameter vector. This problem have an analytical solution and the corresponding PEM is known as the Least Squares (LS) method. we will in this section give a short description of the solution to this problem.

### 2.3.1 Linear regression models

Consider the simple special case of the general linear system (1) and (2) described by

$$y_k = Eu_k + e_k, \tag{46}$$

where $y_k \in \mathbb{R}^m$, $E \in \mathbb{R}^{m\times r}$, $u_k \in \mathbb{R}^r$ and $e_k \in \mathbb{R}^m$ is white with covariance matrix $\Delta = \mathrm{E}(e_k e_k^T)$. The model (46) can be written as a linear regression

$$y_k = \varphi_k^T \theta + e_k, \tag{47}$$

where

$$\varphi_k^T = u_k^T \otimes I_m \ \in \ \mathbb{R}^{m\times rm} \tag{48}$$

and the true parameters in the system, $\theta_0$, is related to those in $E$ as

$$\theta = \mathrm{vec}(E) \ \in \ \mathbb{R}^{mr}. \tag{49}$$

Note that the number of parameters in this case is $p = mr$. Furthermore, note that (47) is a standard notation of a linear regression equation used in the identification literature. In order to deduce (47) from (46) we have used that $\mathrm{vec}(AXB) = (B^T \otimes A)\mathrm{vec}(X)$. Using this and the fact that $Eu_k = I_m Eu_k$ gives (47).

Also note that dynamic systems described by ARX models can be written as a linear regression of the form (47).

### 2.3.2 The least squares method

Given a linear regression of the form

$$y_k = \varphi_k^T \theta_0 + e_k, \tag{50}$$

where $y_k \in \mathbb{R}^m$, $\varphi_k \in \mathbb{R}^{p\times m}$, $e_k \in \mathbb{R}^m$ is white with covariance matrix $\Delta = \mathrm{E}(e_k e_k^T)$ $\in \mathbb{R}^{m\times m}$ and where $\theta_0 \in \mathbb{R}^p$ is the true parameter vector.

Here $\varphi_k$ is a vector of known variables (or quantities) and these variables is often called regression variables or regressors. The output variables $y_k$ is also called the regressed variables.

A natural predictor is as usual

$$\bar{y}_k(\theta) = \varphi_k^T \theta. \tag{51}$$

We will in the following find the parameter estimate, $\hat{\theta}_N$, which minimizes the PE criterion

$$V_N(\theta) = \frac{1}{N} \sum_{k=1}^{N} \epsilon_k^T(\theta) \Lambda \epsilon_k(\theta), \tag{52}$$

where $\epsilon_k = y_k - \varphi_k^T \theta$ is the Prediction Error (PE). The gradient matrix, $\psi_k(\theta)$, defined in (44) is in this case given by

$$\psi_k(\theta) = \frac{\partial \bar{y}_k(\theta)}{\partial \theta} = \varphi_k \ \in \ \mathbb{R}^{p \times m}. \tag{53}$$

Using this and the expression (45) for the gradient, $g(\theta)$, of the PE criterion, $V_N(\theta)$, gives

$$g(\theta) = \frac{\partial V_N(\theta)}{\partial \theta} = -2 \frac{1}{N} \sum_{k=1}^{N} \varphi_k \Lambda (y_k - \varphi_k^T \theta) = -2 \frac{1}{N} \sum_{k=1}^{N} (\varphi_k \Lambda y_k - \varphi_k \Lambda \varphi_k^T \theta)). \tag{54}$$

The OLS and the PEM estimate is here simply given by solving $g(\theta) = 0$, i.e.,

$$\hat{\theta}_N = (\sum_{k=1}^{N} \varphi_k \Lambda \varphi_k^T)^{-1} \sum_{k=1}^{N} \varphi_k \Lambda y_k. \tag{55}$$

if the indicated inverse (of the Hessian) exists. Note that the Hessian matrix in this case is simply given by

$$H(\theta) = \frac{\partial g(\theta)}{\partial \theta^T} = 2 \frac{1}{N} \sum_{k=1}^{N} \varphi_k \Lambda \varphi_k^T. \tag{56}$$

The Hessian (symmetric) matrix should be positive definite, $H(\theta) > 0$, for Eq. (55) to be a minimum. This indicates a positive weighting matrix $\Lambda > 0$.

The gradient vector in Eq. (54) may also be derived using the chain rule as follows

$$g(\theta) = \frac{\partial (V_N(\theta))}{\partial \theta} = \frac{1}{N} \sum_{k=1}^{N} \frac{\partial \epsilon_k}{\partial \theta} \frac{\partial (\epsilon_k(\theta)^T \Lambda \epsilon_k(\theta))}{\partial \epsilon_k} = \frac{-2}{N} \sum_{k=1}^{N} \varphi \Lambda (y_k - \varphi^T \theta). \tag{57}$$

Putting $g(\theta) = 0$ gives the optimal solution in Eq. (55).

### 2.3.3 Matrix derivation of the least squares method

For practical reasons when computing the least squares solution as well as for the purpose of analyzing the statistical properties of the estimate it may be convenient to write the linear regression (47) in vector/matrix form as follows

$$Y = \Phi\theta_0 + e, \tag{58}$$

where $Y \in \mathbb{R}^{mN}$, $\Phi \in \mathbb{R}^{mN\times p}$, $\theta_0 \in \mathbb{R}^p$, $p = mr$ and $e \in \mathbb{R}^{mN}$ and given by

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^{mN}, \quad \Phi = \begin{bmatrix} \varphi_1^T \\ \varphi_2^T \\ \vdots \\ \varphi_N^T \end{bmatrix} \in \mathbb{R}^{mN\times p}, \quad e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix} \in \mathbb{R}^{mN}. \tag{59}$$

Furthermore, $e$, is zero mean with covariance matrix $\tilde{\Delta} = \mathrm{E}(ee^T)$. This covariance matrix will be discussed later in this section.. Define

$$\varepsilon = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_k \\ \vdots \\ \epsilon_N \end{bmatrix} = Y - \Phi\theta, \tag{60}$$

as the matrix of prediction errors. Hence we have that the PE criterion is given by

$$V_N(\theta) = \frac{1}{N}\sum_{k=1}^{N} \epsilon_k^T(\theta)\Lambda\epsilon_k(\theta) = \frac{1}{N}\varepsilon^T\tilde{\Lambda}\varepsilon, \tag{61}$$

where $\tilde{\Lambda}$ is a block diagonal matrix with $\Lambda$ on the block diagonals, i.e.,

$$\tilde{\Lambda} = \begin{bmatrix} \Lambda & 0 & \dots & 0 \\ 0 & \Lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Lambda \end{bmatrix} \in \mathbb{R}^{mN\times mN}. \tag{62}$$

The nice thing about (61) is that the summation is not present in the last expression of the PE criterion. Hence we can obtain a more direct derivation of the parameter estimate. The gradient in (54) can in this case simply be expressed as

$$g(\theta) = \frac{\partial V_N(\theta)}{\partial\theta} = \frac{\partial\epsilon}{\partial\theta}\frac{\partial V_N(\theta)}{\partial\epsilon} = \frac{1}{N}(-\Phi^T)2\tilde{\Lambda}(Y - \Phi\theta). \tag{63}$$

The Hessian is given by

$$H(\theta) = \frac{\partial g(\theta)}{\partial \theta^T} = 2\frac{1}{N}\Phi^T\tilde{\Lambda}\Phi. \tag{64}$$

Solving $g(\theta) = 0$ gives the following expression for the estimate

$$\hat{\theta}_N = (\Phi^T\tilde{\Lambda}\Phi)^{-1}\Phi^T\tilde{\Lambda}Y, \tag{65}$$

which is identical to (55). It can be shown that the optimal weighting is given by $\Lambda = \Delta^{-1}$. The solution (55) or (65) with the optimal weighting matrix $\Lambda = \Delta^{-1}$ is known in the literature as the Best Linear Unbiased Estimate (BLUE). Choosing $\Lambda = I_m$ in (55) or (65) gives the OLS solution.

A short analysis of the parameter estimate is given in the following. Substituting (58) into (65) gives an expression of the difference between the parameter estimate and the true parameter vector, i.e.,

$$\hat{\theta}_N - \theta_0 = (\Phi^T\tilde{\Lambda}\Phi)^{-1}\Phi^T\tilde{\Lambda}e. \tag{66}$$

The parameter estimate (65) is an unbiased estimate since

$$\mathrm{E}(\hat{\theta}_N) = \theta_0 + (\Phi^T\tilde{\Lambda}\Phi)^{-1}\Phi^T\tilde{\Lambda}\mathrm{E}(e) = \theta_0. \tag{67}$$

The covariance matrix of the parameter estimate is given by

$$P = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = (\Phi^T\tilde{\Lambda}\Phi)^{-1}\Phi^T\tilde{\Lambda}\mathrm{E}(ee^T)\tilde{\Lambda}\Phi(\Phi^T\tilde{\Lambda}\Phi)^{-1}. \tag{68}$$

Suppose first that we are choosing the weighting matrix $\tilde{\Lambda} = \tilde{\Delta}^{-1}$ where $\tilde{\Delta} = \mathrm{E}(ee^T)$. It should be noted that $\tilde{\Delta}$ also is a block diagonal matrix with $\Delta = \mathrm{E}(e_i e_i^T)$ on the block diagonals. Then we have

$$P_{\mathrm{BLUE}} = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = \sum_{k=1}^{N}\varphi_k\Delta^{-1}\varphi_k^T = (\Phi^T\tilde{\Delta}^{-1}\Phi)^{-1}. \tag{69}$$

The Ordinary Least squares (OLS) estimate is obtained by choosing $\Lambda = I_m$, i.e., no weighting. For the OLS estimate we have that

$$P_{\mathrm{OLS}} = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{E}(ee^T)\Phi(\Phi^T\Phi)^{-1}. \tag{70}$$

In the single output case we have that $\tilde{\Delta} = \mathrm{E}(ee^T) = \delta_0 I_N$. Using this in (70) gives the standard result for univariate ($m = 1$) data which is presented in the literature, i.e.,

$$P_{\mathrm{OLS}} = \mathrm{E}((\hat{\theta}_N - \theta_0)(\hat{\theta}_N - \theta_0)^T) = \delta_0(\Phi^T\Phi)^{-1}. \tag{71}$$

An important result is that

$$P_{\mathrm{BLUE}} \leq P_{\mathrm{OLS}}. \tag{72}$$

In fact, all other symmetric and positive definite weighting matrices gives a larger parameter covariance matrix than the covariance matrix of the BLUE estimate.

### 2.3.4 Alternative matrix derivation of the least squares method

Another linear regression model formulation which is frequently used, e.g., in the Chemometrics literature, is given by

$$Y = XB + E, \tag{73}$$

where $Y \in \mathbb{R}^{N\times m}$ is a matrix of dependent variables (outputs), $X \in \mathbb{R}^{N\times r}$ is a matrix of the independent variables (regressors or inputs), $B \in \mathbb{R}^{m\times r}$ is a matrix of system parameters (regression coefficients). $E \in \mathbb{R}^{N\times m}$ is a matrix of white noise.

The OLS solution is simply obtained by solving the normal equations $X^TY = X^TXB$ for $B$, i.e,

$$B_{\text{OLS}} = (X^TX)^{-1}X^TY, \tag{74}$$

where $X^TX$ is non-singular.

Alternative approximations to the inverse $X' * X$ is obtained by the Partial Least Squares (PLS) method solved by reduced rank analysis and the Singular Value Decomposition (SVD). Furthermore a popular approximation is the Partial Least Squares Regression (PLS) method. Here in the next subsection we mention the Ridge regression method.

### 2.3.5 Ridge regularization method

The Ridge regression method is a method to overcome possible problems by calculation the inverse and ill-conditioned problems with the matrix $X^TX$ in (74) by simply adding a ridge on the diagonal, i.e. instead work with the inverse of a regularized matrix $X^TX + \delta I$ for some positive parameter $\delta > 0$.

$$B_{\text{OLS\_Ridge}} = (X^TX + \delta I)^{-1}X^TY, \tag{75}$$

fore some nonzero scalar parameter $\delta > 0$.

Adding a ridge on the diagonal of $X^TX$ will include a bias in the solution but the variance may be smaller. This is a trade of between bias and variance.

**Example 2.1 (Regularization and system identification)** *Given a simple linear dynamic model*

$$\begin{aligned} x_{k+1} &= ax_k + bu_k + Ke_k, & (76)\\ y_k &= x_k + e_k & (77) \end{aligned}$$

*where we fix the true parameters as* $a = 0.9$ *and* $b = 0.5$.

*When $K = a$ this gives an ARX or linear regression model of the form*

$$y_k = \varphi_k^T \theta + e_k, \tag{78}$$

*where*

$$\varphi_k^T = \begin{bmatrix} y_{k-1} & u_{k-1} \end{bmatrix}, \ \theta = \begin{bmatrix} a \\ b \end{bmatrix}. \tag{79}$$

*This may also be written as a linear regression model as in Eq. (74) with*

$$Y = \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix}, \ X = \begin{bmatrix} y_1 & u_1 \\ y_2 & u_2 \\ \vdots & \vdots \\ y_{N-1} & u_{N-1} \end{bmatrix}, \ B = \begin{bmatrix} a \\ b \end{bmatrix}. \tag{80}$$

# 3 Input and output model structures

Input and output discrete time model structures are frequently used in connection with the prediction error methods and software. We will in this section give a description of these model structures and the connection with the general state space model structure.

## 3.1 ARMAX model structure

An ARMAX model structure is defined by the (polynomial) model

$$A(q)y_k = B(q)u_k + C(q)e_k. \tag{81}$$

The acronym ARMAX comes from the fact that the term $A(q)y_k$ is defined as an Auto Regressive (AR) part, the term $C(q)e_k$ is defined as an Moving Average (MA) part and that the part $B(q)u_k$ represents eXogenous (X) inputs. Note in connection with this that an Auto Regressive (AR) model is of the form $A(q)y_k = e_k$, i.e. with additive equation noise. The noise term, $C(q)e_k$, in a so called ARMA model $A(q)y_k = C(q)e_k$ represents a moving average of the white noise $e_k$.

In general we have that the polynomials in Eq. (81) may be expressed as

$$\begin{aligned} A(q) &= 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na}, & (82) \\ B(q) &= b_1 q^{-1} + \ldots + b_{nb} q^{-nb}, & (83) \\ C(q) &= 1 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc}, & (84) \end{aligned}$$

when $y_k$, $u_k$ and $e_k$ are scalar signals and where $na$, $nb$ and $nc$ are the order of the $A(q)$, $B(q)$ and the $C(q)$ polynomials, respectively. Note also that the

polynomials also may be written as $A(q) = A(q^{-1})$, $B(q) = B(q^{-1})$ and $C(q) = C(q^{-1})$ where $q^{-1}$ is the shift operator such that

$$q^{-1}y_k = y_{k-1}. \tag{85}$$

The ARMAX model structure can be deduced from a general linear SISO state space model. MIMO systems is not considered in this Section. This will be illustrated in the following example.

**Example 3.1 (Estimator canonical form to polynomial form)** *Consider the following single input and single output discrete time state space model on estimator canonical form*

$$x_{k+1} = \begin{bmatrix} -a_1 & 1 \\ -a_0 & 0 \end{bmatrix} x_k + \begin{bmatrix} b_1 \\ b_0 \end{bmatrix} u_k + \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} v_k \tag{86}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + eu_k + fv_k \tag{87}$$

*where $u_k$ is a known deterministic input signal, $v_k$ is an unknown white noise process and $x_k^T = \begin{bmatrix} x_{k+1}^1 & x_{k+1}^2 \end{bmatrix}$ is the state vector.*

*An input output model formulation can be derived as follows*

$$x_{k+1}^1 = -a_1 x_k^1 + x_k^2 + b_1 u_k + c_1 v_k \tag{88}$$

$$x_{k+1}^2 = -a_0 x_k^1 + b_0 u_k + c_0 v_k \tag{89}$$

$$y_k = x_k^1 + eu_k + fv_k \tag{90}$$

*Express Equation (88) with $k =: k+1$ and substitute for $x_{k+1}^2$ defined by Equation (89). This gives an equation in terms of the 1st state $x_k^1$. Finaly, eliminate $x_k^1$ by using Equation (90). This gives*

$$\begin{bmatrix} 1 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} y_k \\ y_{k-1} \\ y_{k-2} \end{bmatrix}$$

$$= \begin{bmatrix} e & b_1 + a_1 e & b_0 + a_0 e \end{bmatrix} \begin{bmatrix} u_k \\ u_{k-1} \\ u_{k-2} \end{bmatrix} + \begin{bmatrix} f & c_1 + a_1 f & c_0 + a_0 f \end{bmatrix} \begin{bmatrix} v_k \\ v_{k-1} \\ v_{k-2} \end{bmatrix} \tag{91}$$

*Let us introduce the backward shift operator $q^{-1}$ such that $q^{-1}u_k = u_{k-1}$. This gives the following polynomial (or transfer function) model*

$$\overbrace{(1 + a_1 q^{-1} + a_0 q^{-2})}^{A(q)} y_k$$

$$= \overbrace{(e + (b_1 + a_1 e)q^{-1} + (b_0 + a_0 e)q^{-2})}^{B(q)} u_k + \overbrace{(f + (c_1 + a_1 f)q^{-1} + (c_0 + a_0 f)q^{-2})}^{C(q)} v_k \tag{92}$$

*which is equal to an ARMAX model structure.*

**Example 3.2 (Observability canonical form to polynomial form)**
*Consider the following single input and single output discrete time state space model on observability canonical form*

$$x_{k+1} = \begin{bmatrix} 0 & 1 \\ -a_1 & -a_0 \end{bmatrix} x_k + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} u_k + \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} v_k, \tag{93}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + eu_k + fv_k, \tag{94}$$

*where $u_k$ is a known deterministic input signal, $v_k$ is an unknown white noise process and $x_k^T = \begin{bmatrix} x_{k+1}^1 & x_{k+1}^2 \end{bmatrix}$ is the state vector. An input and output model can be derived by eliminating the states in (93) and (94). We have*

$$x_{k+1}^1 = x_k^2 + b_0 u_k + c_0 v_k, \tag{95}$$
$$x_{k+1}^2 = -a_1 x_k^1 - a_0 x_k^2 + b_1 u_k + c_1 v_k, \tag{96}$$
$$y_k = x_k^1 + eu_k + fv_k. \tag{97}$$

*Solve (95) for $x_k^2$ and substitute into (96). This gives an equation in $x_k^1$, i.e.,*

$$x_{k+2}^1 - b_0 u_{k+1} - c_0 v_{k+1} = -a_1 x_k^1 - a_0 (x_{k+1}^1 - b_0 u_k - c_0 v_k) + b_1 u_k + c_1 v_k. \tag{98}$$

*Solve (97) for $x_k^1$ and substitute into (98). This gives*

$$\begin{aligned} y_{k+2} - eu_{k+2} - fv_{k+2} - b_0 u_{k+1} - c_0 v_{k+1} &= -a_1 (y_k - eu_k - fv_k) \\ -a_0 (y_{k+1} - eu_{k+1} - fv_{k+1}) + a_0 b_0 u_k + a_0 c_0 v_k + b_1 u_k + c_1 v_k. \end{aligned} \tag{99}$$

*This gives the input and output equation*

$$\begin{aligned} \begin{bmatrix} 1 & a_0 & a_1 \end{bmatrix} \begin{bmatrix} y_{k+2} \\ y_{k+1} \\ y_k \end{bmatrix} &= \begin{bmatrix} e & b_0 + a_0 e & b_1 + a_0 b_0 + a_1 e \end{bmatrix} \begin{bmatrix} u_{k+2} \\ u_{k+1} \\ u_k \end{bmatrix} \\ &+ \begin{bmatrix} f & c_0 + a_0 f & c_1 + a_0 c_0 + a_1 f \end{bmatrix} \begin{bmatrix} v_{k+2} \\ v_{k+1} \\ v_k \end{bmatrix}. \end{aligned} \tag{100}$$

*Hence, we can write (100) as an ARMAX polynomial model*

$$A(q) y_k = B(q) u_k + C(q) v_k, \tag{101}$$
$$A(q) = 1 + a_0 q^{-1} + a_1 q^{-2}, \tag{102}$$
$$B(q) = e + (b_0 + a_0 e) q^{-1} + (b_1 + a_0 b_0 + a_1 e) q^{-2}, \tag{103}$$
$$C(q) = f + (c_0 + a_0 f) q^{-1} + (c_1 + a_0 c_0 + a_1 f) q^{-2}. \tag{104}$$

## 3.2 ARX model structure

An Auto Regression (AR) model $A(q)y_k = e_k$ with eXtra (or eXogenous) inputs (ARX) model can be expressed as follows

$$A(q)y_k = B(q)u_k + e_k \tag{105}$$

where the term $A(q)y_k$ is the Auto Regressive (AR) part and the term $B(q)u_k$ represents the part with the eXtra (X) inputs. The eXtra variables $u_k$ are called eXogenous in econometrics. Note also that the ARX model also is known as an *equation error model* because of the additive error or white noise term $e_k$.

It is important to note that the parameters in an ARX model, e.g. as defined in Equation (105) where $e_k$ is a white noise process, can be identified directly by the Ordinary Least Squares (OLS) method, if the inputs, $u_k$, are informative enough. A problem with the ARX structure is off-course that the noise model (additive noise) is to simple in many practical cases.

**Example 3.3 (ARX and State Space model structure)** *Comparing the ARX model structure (105) with the more general model structure given by (92) we find that the state space model (86) and (87) is equivalent to an ARX model if*

$$c_1 = -a_1 f, \quad c_0 = -a_0 f, \quad e_k = f v_k \tag{106}$$

*This gives the following ARX model*

$$\overbrace{(1 + a_1 q^{-1} + a_0 q^{-2})}^{A(q)} y_k = \overbrace{(e + (b_1 + a_1 e)q^{-1} + (b_0 + a_0 e)q^{-2})}^{B(q)} u_k + f v_k \tag{107}$$

*which, from the above discussion, have the following state space model equivalent*

$$x_{k+1} = \begin{bmatrix} -a_1 & 1 \\ -a_0 & 0 \end{bmatrix} x_k + \begin{bmatrix} b_1 \\ b_0 \end{bmatrix} u_k + \begin{bmatrix} -a_1 \\ -a_0 \end{bmatrix} f v_k \tag{108}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + e u_k + f v_k \tag{109}$$

*It is at first sight not easy to see that this state space model is equivalent to an ARX model. It is also important to note that the ARX model has a state space model equivalent. The noise term $e_k = f v_k$ is white noise if $v_k$ is withe noise. The noise $e_k$ appears as both process (state equation) noise and measurements (output equation) noise. The noise is therefore filtred throug the process dynamics.*

## 3.3 OE model structure

An Output Error (OE) model structure can be represented as the following polynomial model

$$A(q)y_k = B(q)u_k + A(q)e_k \tag{110}$$

or equivalently for single output plants

$$y_k = \frac{B(q)}{A(q)}u_k + e_k \tag{111}$$

**Example 3.4 (OE and State Space model structure)** *From (92) we find that the state space model (86) and (87) is equivalent to an OE model if*

$$c_1 = 0, \quad c_0 = 0, \quad f = 1, \quad v_k = e_k \tag{112}$$

*Hence, the following OE model*

$$\begin{aligned} \overbrace{(1 + a_1q^{-1} + a_0q^{-2})}^{A(q)} y_k &= \overbrace{(e + (b_1 + a_1e)q^{-1} + (b_0 + a_0e)q^{-2})}^{B(q)} u_k \\ &+ \overbrace{(1 + a_1q^{-1} + a_0q^{-2})}^{A(q)} e_k \end{aligned} \tag{113}$$

*have the following state space model equivalent*

$$x_{k+1} = \begin{bmatrix} -a_1 & 1 \\ -a_0 & 0 \end{bmatrix} x_k + \begin{bmatrix} b_1 \\ b_0 \end{bmatrix} u_k \tag{114}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + eu_k + e_k \tag{115}$$

*Note that the noise term $e_k$ appears in the state space model as an equivalent measurements noise term or output error term.*

## 3.4 BJ model structure

In an ARMAX model structure the dynamics in the path from the inputs $u_k$ to the output $y_k$ is the same as the dynamics in the path from the process noise $e_k$ to the output. In practice it is quite realistic that some or all of the dynamics in the two paths are different. This can be represented by the Box Jenkins (BJ) model structure

$$F(q)D(q)y_k = D(q)B(q)u_k + F(q)C(q)e_k \tag{116}$$

or for single output systems

$$y_k = \frac{B(q)}{F(q)}u_k + \frac{C(q)}{D(q)}e_k \tag{117}$$

In the BJ model the moving average noise (coloured noise) term $C(q)e_k$ is filtred throug the dynamics represented by the polynomial $D(q)$. Similarly, the dynamics in the path from the inputs $u_k$ are represented by the polynomial $A(q)$.

However, it is important to note that the BJ model structure can be represented by an equivalent state space model structure.

## 3.5 Summary

The family of polynomial model structures

$$\begin{array}{lcl} BJ & : & \text{Box Jenkins} \\ ARMAX & : & \text{Auto Regressive Moving Average with eXtra inputs} \\ ARX & : & \text{Auto Regressive with eXtra inputs} \\ ARIMAX & : & \text{Auto Regressive Integrating Moving Average with eXtra inputs} \\ OE & : & \text{Output Error} \end{array} \tag{118}$$

can all be represented by a state space model of the form

$$y_k = Dx_k + Eu_k + Fe_k, \tag{119}$$

$$x_{k+1} = Ax_k + Bu_k + Ce_k. \tag{120}$$

When using the prediction error methods for system identification the model structure and the order of the polynomial need to be specified. It is important that the prediction error which is to be minimized is a function of as few unknown parameters as possible. Note also that all of the model structures discussed above are linear models. However, the optimization problem of computing the unknown parameters are highly non-linear. Hence, we can run into numerical problems. This is especially the case for multiple output and MIMO systems.

The subspace identification methods, e.g. **DSR**, is based on the general state space model defined by (119) and (120. The subspace identification methods is therefore flexibile enough to identify systems described by all the polynomial model structures (118). Note also that not even the system order $n$ need to be specified beforehand when using the subspace identification method.

**Remark 3.1 (Other notations frequently used)**
*Another notation for the backward shift operator $q^{-1}$ (defined such that $q^{-1}y_k = y_{k-1}$) is the $z-$ operator, ($z^{-1}$ such that $z^{-1}y_k = y_{k-1}$. The notation $A(q)$, $B(q)$ and so on are used by Ljung (1999). In Söderström and Stoica (1989) the notation $A(q^{-1})$, $B(q^{-1})$ are used for the same polynomials.*

# 4 Optimal one-step-ahead predictions

## 4.1 State Space Model

Consider the innovations model

$$\begin{aligned} \bar{x}_{k+1} &= A\bar{x}_k + Bu_k + Ke_k, && (121)\\ y_k &= D\bar{x}_k + Eu_k + e_k, && (122) \end{aligned}$$

where $K$ is the Kalman filter gain matrix. A predictor $\bar{y}_k$ for $y_k$ can be defined as the two first terms on the right hand side of (122), i.e.

$$\bar{y}_k = D\bar{x}_k + Eu_k. \tag{123}$$

The equations for the (optimal) predictor (Kalman filter) is therefore given by

$$\begin{aligned} \bar{x}_{k+1} &= A\bar{x}_k + Bu_k + K(y_k - \bar{y}_k), && (124)\\ \bar{y}_k &= D\bar{x}_k + Eu_k, && (125) \end{aligned}$$

which can be written as

$$\begin{aligned} \bar{x}_{k+1} &= (A - KD)\bar{x}_k + (B - KE)u_k + Ky_k, && (126)\\ \bar{y}_k &= D\bar{x}_k + Eu_k. && (127) \end{aligned}$$

Hence, the optimal prediction, $\bar{y}_k$, of the output $y_k$ can simply be obtained by simulating (126) and (127) with a specified initial predicted state $\bar{x}_1$. The results is known as the one-step ahead predictions. The name one-step ahead predictor comes from the fact that the prediction of $y_{k+1}$ is based upon all outputs up to time $k$ as well as all relevant inputs. This can be seen by writing (126) and (127) as

$$\bar{y}_{k+1} = D(A - KD)\bar{x}_k + D(B - KE)u_k + Eu_{k+1} + DKy_k, \tag{128}$$

which is the one-step ahead prediction.

Note that the optimal prediction (126) and (127) can be written as the following transfer function model

$$\bar{y}_k(\theta) = H_e^d(q)u_k + H_e^s(q)y_k, \tag{129}$$

where

$$\begin{aligned} H_e^d(q) &= D(qI - (A - KD))^{-1}(B - KE) + E, && (130)\\ H_e^s(q) &= D(qI - (A - KD))^{-1}K. && (131) \end{aligned}$$

The derivation of the one-step-ahead predictor for polynomial models is further discussed in the next section.

## 4.2 Input-output model

The linear system can be expressed as the following input and output polynomial model

$$y_k = G(q)u_k + H(q)e_k. \tag{132}$$

The noise term $e_k$ can be expressed as

$$H^{-1}(q)y_k = H^{-1}(q)G(q)u_k + e_k. \tag{133}$$

Adding $y_k$ on both sides gives

$$y_k = (I - H^{-1}(q))y_k + H^{-1}(q)G(q)u_k + e_k. \tag{134}$$

The prediction of the output is given by the first two terms on the right hand side of (134) since $e_k$ is white and therefore cannot be predicted, i.e.,

$$\bar{y}_k(\theta) = (I - H^{-1}(q))y_k + H^{-1}(q)G(q)u_k. \tag{135}$$

Loosely spoken, the optimal prediction of $y_k$ is given by the predictor, $\bar{y}_k$, so that a measure of the difference $e_k = y_k - \bar{y}_k(\theta)$, e.g., the variance, is minimized with respect to the parameter vector $\theta$, over the data horizon. We also assume that a sufficient model structure is used, and that the unknown parameters is parameterized in $\theta$. Ideally, the prediction error $e_k$ will be white.

# 5 Optimal M-step-ahead prediction

## 5.1 State space models

The aim of this section is to develop the optimal $j$-step-ahead predictions $\bar{y}_{k+j}$ $\forall\ j = 1, ..., M$. The optimal one-steap-ahead predictor, i.e., for $j = 1$, is defined in (128). We will in the following derivation assume that only outputs up to time $k$ is available. Furthermore, it is assumed that all inputs which are needed in the derivation are available. This is realistic in practice. Only, past outputs $\ldots, y_{k-2}, y_{k-1}, y_k$ are known. Note that we can assume values for the future inputs, or the future inputs can be computed in an optimization strategy as in Model Predictive Control (MPC). The Kalman filter on innovations form is

$$\begin{aligned} \bar{x}_{k+1} &= A\bar{x}_k + Bu_k + Ke_k, & (136)\\ y_k &= D\bar{x}_k + Eu_k + e_k. & (137) \end{aligned}$$

The prediction for $j = 1$, i.e., the one-step-ahead prediction, is given in (128). The other predictions are derived as follows. The output at time $k := k + 2$ is then defined by

$$\begin{aligned} \bar{x}_{k+2} &= A\bar{x}_{k+1} + Bu_{k+1} + Ke_{k+1}, & (138)\\ y_{k+2} &= D\bar{x}_{k+2} + Eu_{k+2} + e_{k+2}. & (139) \end{aligned}$$

The (white) noise vectors, $e_{k+1}$ and $e_{k+2}$ are all in the future, and they can not be predicted because they are white. The best prediction of $y_{k+2}$ must then be to put $e_{k+1} = 0$ and $e_{k+2} = 0$. Hence, the predictor for $j = 2$ is

$$\begin{aligned} x_{k+1} &= \bar{x}_{k+1}, && (140)\\ x_{k+2} &= Ax_{k+1} + Bu_{k+1}, && (141)\\ \bar{y}_{k+2} &= Dx_{k+2} + Eu_{k+2}. && (142) \end{aligned}$$

This can simply be generalized for $j > 2$ as presented in the following lemma.

**Lemma 5.1 (Optimal $j$-step-ahead predictions)**
*The optimal $j$-step-ahead predictions, $j = 1, 2, \ldots, M$, can be obtained by a pure simulation of the system $(A, B, D, E)$ with the optimal predicted Kalman filter state $\bar{x}_{k+1}$ as the initial state.*

$$\begin{aligned} x_{k+j+1} &= Ax_{k+j} + Bu_{k+j}, && (143)\\ y_{k+j} &= Dx_{k+j} + Eu_{k+j}, \ \forall\ j = 1, \ldots, M && (144) \end{aligned}$$

*where the initial state $x_{k+1} = \bar{x}_{k+1}$ is given from the Kalman filter state equation*

$$\bar{x}_{k+1} = (A - KD)\bar{x}_k + (B - KE)u_k + Ky_k, \qquad (145)$$

*where the initial state $\bar{x}_1$ is known and specified. This means that all outputs up to time, $k$, and all relevant inputs, are used to predict the output at time $k + 1$, ..., $k + M$.*

**Example 5.1 (Prediction model on matrix form (proper system))**
*Given the Kalman filter matrices $(A, B, D, E, K)$ of an only proper process, and the initial predicted state $\bar{x}_k$. The predictions $\bar{y}_{k+1}$, $\bar{y}_{k+2}$ and $\bar{y}_{k+3}$ can be written in compact form as follows*

$$\begin{aligned} \begin{bmatrix} \bar{y}_{k+1} \\ \bar{y}_{k+2} \\ \bar{y}_{k+3} \end{bmatrix} &= \begin{bmatrix} D \\ DA \\ DA^2 \end{bmatrix} (A - KD)\bar{x}_k + \begin{bmatrix} D \\ DA \\ DA^2 \end{bmatrix} Ky_k \\ &+ \begin{bmatrix} D(B - KE) & E & 0 & 0 \\ DA(B - KE) & DB & E & 0 \\ DA^2(B - KE) & DAB & DB & E \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ u_{k+3} \end{bmatrix}. \qquad (146) \end{aligned}$$

*This formulation may be useful in e.g., model predictive control.*

**Example 5.2 (Prediction model on matrix form (strictly proper system))**

*Given the Kalman filter matrices $(A, B, D, K)$ of an strictly proper system, and the initial predicted state $\bar{x}_k$. The predictions $\bar{y}_{k+1}$, $\bar{y}_{k+2}$ and $\bar{y}_{k+3}$ can be written in compact form as follows*

$$\begin{bmatrix} \bar{y}_{k+1} \\ \bar{y}_{k+2} \\ \bar{y}_{k+3} \end{bmatrix} = \begin{bmatrix} D \\ DA \\ DA^2 \end{bmatrix} (A - KD)\bar{x}_k + \begin{bmatrix} D \\ DA \\ DA^2 \end{bmatrix} K y_k + \begin{bmatrix} DB & 0 & 0 \\ DAB & DB & 0 \\ DA^2B & DAB & DB \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \end{bmatrix}. \quad (147)$$

*This formulation is more realistic for control considerations, where we almost always have a delay of one sample between the input and the output.*

# 6 Matlab implementation

We will in this section illustrate a simple but general MATLAB implementation of a Prediction Error Method (PEM).

## 6.1 Tutorial: SS-PEM Toolbox for MATLAB

The MATLAB files in this work is built up as a small toolbox, i.e., the SS-PEM toolbox for MATLAB. This toolbox should be used in connection with the D-SR Toolbox for MATLAB. An overview of the functions in the toolbox is given by the command

```
>> help ss-pem
```

The main prediction error method is implemented in the function **sspem.m**. An initial parameter vector, $\theta_1$, is computed by first using the DSR algorithm to compute an initial state space Kalman filter model, i.e. the corresponding matrices $(A, B, D, E, K, x1)$. This model is then transformed to observability canonical form by the D-SR Toolbox function **ss2cf.m**. The initial parameter vector, $\theta_1$, is then taken as the free parameters in these canonical state space model matrices. The minimizing parameter vector, $\hat{\theta}$, is then computed by the Optimization toolbox function **fminunc.m**. Other optimization algorithms can off-course be used.

A tutorial and overview is given in the following. Assume that output and input data matrices, $Y \in \mathbb{R}^{N \times m}$ and $U \in \mathbb{R}^{N \times r}$ are given. A state space model (Kalman filter) can then simply be identified by the PEM function **sspem.m**, in the MATLAB command window as follows.

```
>> [A,B,D,E,K,x1]=sspem(Y,U,n);
```

A typical drawback with PEMs is that the system order, $n$, has to be specified beforehand. A good solution is to analyze and identify the system order, $n$, by the subspace algorithm DSR. The identified model is represented in observability canonical form. That is a state space realization with as few free parameters as possibile. This realization can be illustrated as follows. Consider a system with $m = 2$ outputs, $r = 2$ inputs and $n = 3$ states, which can be represented by a linear model. The resulting model from **sspem.m** will be on Kalman filter innovations form, i.e.,

$$x_{k+1} = Ax_k + Bu_k + Ke_k, \tag{148}$$
$$y_k = Dx_k + Eu_k + e_k, \tag{149}$$

with initial predicted state $x_1$ given, or on prediction form, i.e.,

$$x_{k+1} = Ax_k + Bu_k + K(y_k - \bar{y}_k), \tag{150}$$
$$\bar{y}_k(\theta) = Dx_k + Eu_k, \tag{151}$$

and where the model matrices $(A, B, D, E, K)$ and the initial predicted state $x_1$ are given by

$$A = \begin{bmatrix} 0 & 0 & 1 \\ \theta_1 & \theta_3 & \theta_5 \\ \theta_2 & \theta_4 & \theta_6 \end{bmatrix}, \; B = \begin{bmatrix} \theta_7 & \theta_{10} \\ \theta_8 & \theta_{11} \\ \theta_9 & \theta_{12} \end{bmatrix}, \; K = \begin{bmatrix} \theta_{13} & \theta_{16} \\ \theta_{14} & \theta_{17} \\ \theta_{15} & \theta_{18} \end{bmatrix}, \tag{152}$$

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \; E = \begin{bmatrix} \theta_{19} & \theta_{21} \\ \theta_{20} & \theta_{22} \end{bmatrix}, \; x_1 \begin{bmatrix} \theta_{23} \\ \theta_{24} \\ \theta_{25} \end{bmatrix}. \tag{153}$$

This model is on a so called observability canonical form, i.e. the third order model has as few free parameters as possibile. The number of free parameters are in this example 25. The observability canonical form is such that the $D$ matrix is filled with ones and zeros. We have that $D = \text{ones}(m, n)$. Furthermore, we have that a concatenation of the $D$ matrix and the first $n - m$ rows of the $A$ matrix is the identity matrix, i.e. $\begin{bmatrix} D \\ A(1 : n - m, :) \end{bmatrix} = I_n$. This may be used as a rule when writing up the $n - m$ first rows of the $A$ matrix. The rest of the $A$ matrix as well as the matrices $B$, $K$, $E$ and $x_1$ is filled with parameters. However, the user does not have to deal with the construction of the canonical form, when using the SSPEM toolbox.

The total number of free parameters in a linear state space model and in the parameter vector $\theta \in \mathbb{R}^p$ are in general

$$p = mn + nr + nm + mr + n = (2m + r + 1)n + mr \tag{154}$$

An existing DSR model can also be refined as follows. Suppose now that a state space model, i.e. the matrices $(A, B, D, E, K, x1)$ are identified by the DSR algorithm as follows

```
>> [A,B,D,E,C,F,x1]=dsr(Y,U,L);
>> K=C*inv(F);
```

This model can be transformed to observability canonical form and the free parameters in this model can be mapped into the parameter vector $\theta_1$ by the function **ss2thp.m**, as follows

```
>> th_1=ss2thp(A,B,D,E,K,x1);
```

The model parameter vector $\theta_1$ can be further refined by using these parameters as initial values to the PEM method **sspem.m**. E.g. as follows

```
>> [A,B,D,E,K,x1,V,th]=sspem(Y,U,n,th_1);
```

The value of the prediction error criterion, $V(\theta)$, is returned in $V$. The new and possibly better (more optimal) parameter vector is returned in $th$.

Note that for a given parameter vector, then the state space model matrices can be constructed by

```
>> (A,B,D,E,K,x1)=thp2ss(th,n,m,r);
```

It is also worth to note that the value, $V(\theta)$, of the prediction error criterion is evaluated by the MATLAB function **vfun_mo.m**. The data matrices $Y$ and $U$ and the system order, $n$, must first be defined as global variables, i.e.

```
>> global Y U n
```

The PE criterion is then evaluated as

```
>> V=vfun_mo(th);
```

where $th$ is the parameter vector. Note that the parameter vector must be of length $p$ as explained above.

See also the function **ss2cf.m** in the D-SR Toolbox for MATLAB which returns an observability form state space realization from a state space model.

# 7 Recursive ordinary least squares method

We start this section by a simple example of how the mean of a variable, say $y_k$, may be recursively estimated.

**Example 7.1**
*The mean of a variable $y_k$ at present time $t$ may be expressed as*

$$\bar{y}_t = \frac{1}{t}\sum_{k=1}^{t} y_k. \tag{155}$$

*In recursive identification we want to use new information at present time, i.e., $y_t$, to update an estimate at the previous time instant. The sum in Eq. (155) may be divided into two parts and expressed as follows*

$$\bar{y}_t = \frac{1}{t}(\sum_{k=1}^{t-1} y_k + y_t) = \frac{t-1}{t}\overbrace{\frac{1}{t-1}\sum_{k=1}^{t-1} y_k}^{\bar{y}_{t-1}} + \frac{1}{t}y_t. \tag{156}$$

*This may be written as*

$$\bar{y}_t = \bar{y}_{t-1} + \frac{1}{t}(y_t - \bar{y}_{t-1}), \tag{157}$$

*where*

$$\bar{y}_{t-1} = \frac{1}{t-1}\sum_{k=1}^{t-1} y_k, \tag{158}$$

*is the mean at the previous time instant.*

*Eq. (157) is quite appealing since the mean is estimated very similarly as a Kalman filter estimate, i.e., the new estimate is obtained as the sum of the previous estimate plus a correction term.*

The OLS estimate (55) can be written as

$$\hat{\theta}_t = (\sum_{k=1}^{t} \varphi_k \Lambda \varphi_k^T)^{-1} \sum_{k=1}^{t} \varphi_k \Lambda y_k. \tag{159}$$

We have simply replaced $N$ in (55) by $t$ in order to stress the dependence of $\hat{\theta}$ on time, $t$. Let us define the indicated inverse in (159) by

$$P_t = (\sum_{k=1}^{t} \varphi_k \Lambda \varphi_k^T)^{-1}. \tag{160}$$

From this definition we have that

$$P_t = (\sum_{k=1}^{t-1} \varphi_k \Lambda \varphi_k^T + \varphi_t \Lambda \varphi_t^T)^{-1}. \tag{161}$$

which gives.

$$P_t^{-1} = P_{t-1}^{-1} + \varphi_t \Lambda \varphi_t^T \tag{162}$$

$P_t$ can be viewed as a covariance matrix which can be recursively computed at each time step by (162).

The idea is now to write (159) as

$$\begin{aligned} \hat{\theta}_t &= P_t \sum_{k=1}^{t} \varphi_k \Lambda y_k \\ &= P_t (\sum_{k=1}^{t-1} \varphi_k \Lambda y_k + \varphi_t \Lambda y_t) \end{aligned} \tag{163}$$

From (159) we have that $\hat{\theta}_{t-1}$ is given by

$$\hat{\theta}_{t-1} = P_{t-1} \sum_{k=1}^{t-1} \varphi_k \Lambda y_k \tag{164}$$

Substituting this into (163) gives

$$\hat{\theta}_t = P_t (P_{t-1}^{-1} \hat{\theta}_{t-1} + \varphi_t \Lambda y_t) \tag{165}$$

From (162) we have that

$$P_{t-1}^{-1} = P_t^{-1} - \varphi_t \Lambda \varphi_t^T \tag{166}$$

Substituting this into (165) gives

$$\hat{\theta}_t = P_t ((P_t^{-1} - \varphi_t \Lambda \varphi_t^T) \hat{\theta}_{t-1} + \varphi_t \Lambda y_t) \tag{167}$$

Rearranging this we get

$$\hat{\theta}_t = \hat{\theta}_{t-1} + P_t \varphi_t \Lambda (y_t - \varphi_t^T \theta_{t-1}) \tag{168}$$

We then have the following ROLS algorithm

**Algorithm 7.1 (Recursive OLS algorithm)**
*The algorithm is summarized as follows:*

*Step 1 Initial values for $P_{t=0}$ and $\hat{\theta}_{t=0}$. It is common practice to take $P_{t=0} = \rho I_p$ with $\rho$ a "large" constant and $\hat{\theta}_{t=0} = 0$ without any a-priori information.*

*Step 2 Updating the covariance matrix, $P_t$, and the parameter estimate, $\theta_t$, as follows*

$$\begin{aligned} P_t^{-1} &= P_{t-1}^{-1} + \varphi_t \Lambda \varphi_t^T, & (169) \\ K_t &= P_t \varphi_t \Lambda, & (170) \end{aligned}$$

*and*

$$\hat{\theta}_t = \hat{\theta}_{t-1} + K_t(y_t - \varphi_t^T \hat{\theta}_{t-1}). \quad (171)$$

At each time step we need to compute the inverse

$$P_t = (P_{t-1}^{-1} + \varphi_t \Lambda \varphi_t^T)^{-1}. \quad (172)$$

Using the following matrix inversion lemma

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}, \quad (173)$$

we have that (172) can be written as

$$P_t = P_{t-1} - P_{t-1}\varphi_t(\Lambda^{-1} + \varphi_t^T P_{t-1}\varphi_t)^{-1}\varphi_t^T P_{t-1}. \quad (174)$$

**Example 7.2 (Recursive OLS algorithm)**
*Given a system described by a state space model*

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ce_k & (175) \\ y_k &= Dx_k & (176) \end{aligned}$$

*where*

$$A = \begin{bmatrix} 0 & 1 \\ a_1 & a_2 \end{bmatrix}, \; B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \; C = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \; D = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad (177)$$

This gives the input and output (ARMAX) model

$$\begin{aligned} y_{k+2} &= a_2 y_{k+1} + a_1 y_k \\ &+ b_1 u_{k+1} + (b_2 - a_2 b_1) u_k \\ &+ e_{k+2} + (c_1 - a_2) e_{k+1} + (c_2 - a_2 c_1 - a_1) e_k \quad (178) \end{aligned}$$

Putting $c_1 = a_2$ and $c_2 = a_2^2 + a_1$ and changeing the time index with $t = k + 2$ gives the ARX model

$$y_t = a_2 y_{t-1} + a_1 y_{t-2} + b_1 u_{t-1} + (b_2 - a_2 b_1) u_{t-2} + e_t \quad (179)$$

This can be written as a linear regression model

$$y_t = \overbrace{\begin{bmatrix} y_{t-1} & y_{t-2} & u_{t-1} & u_{t-2} \end{bmatrix}}^{\varphi_t^T} \overbrace{\begin{bmatrix} a_2 \\ a_1 \\ b_1 \\ b_2 - a_2 b_1 \end{bmatrix}}^{\theta} + e_t. \tag{180}$$

An ROLS algorithm for the estimation of the parameter vector is implemented in the following MATLAB script.

```
% File name: main_rols_ex.m
% Example:
% Implementation of a Reqursive Ordinary Least Squares (ROLS) algorithm
% for ARX model parameter estimation.
a1=-0.7; a2=1.5; b1=0.25; b2=0.625;
A=[0 1;a1 a2]; B=[b1;b2]; D=[1,0];

N=200;
rand('state',0), randn('seed',0)
u=randn(N,1);
u=prbs1(N,10,40);

randn('seed',0)

for i=1:5                                % Simulate M=5 different data set..
y=dlsim(A,B,D,0,u);
y=y+randn(N,1)*0.01;

Th=zeros(N,4);                            % Array to hold parameter estimates
P=10000*eye(4); th=[0;0;0;0]; Lam=0.998;   % Initial values
%Lam=1;
lf=1;
for t=3:N
    Phi=[y(t-1);y(t-2);u(t-1);u(t-2)]; % Varphi_t matrix at time t.

    ybar=Phi'*th;                         % Predikted output at time t.

    Pi=inv(P);                            % Updating the covariance matrix, P_t.
    Pi=lf*Pi+Phi*Lam*Phi'; P=inv(Pi);

    K=P*Phi*Lam;                          % Kalman gain, K_t.

    th=th+K*(y(t)-Phi'*th);               % Parameter estimates at time, t.
```

```
    Th(t,:)=th';                          % Store the parameter estimates.
end
Tm(i,:)=th';
end

th
th0=[a2;a1;b1;b2-a2*b1]

figure(1)
subplot(411), plot(Th(:,1)), ylabel('a_2'), title('ROLS example')
subplot(412), plot(Th(:,2)), ylabel('a_1')
subplot(413), plot(Th(:,3)), ylabel('b_1')
subplot(414), plot(Th(:,4)), ylabel('b_2-a_2b_1')
xlabel('Diskret tid: t')

figure(2)
subplot(211), plot(u), ylabel('u_k'), title('ROLS_example')
subplot(212), plot(y), ylabel('y_k')
xlabel('Diskret tid: t')
```

## 7.1 Comparing ROLS and the Kalman filter

Let us compare the ROLS algorithm with the Kalman filter on the following model.

$$\begin{aligned} \theta_{t+1} &= \theta_t, && (181)\\ y_t &= \varphi_t^T\theta_t + w_t, && (182) \end{aligned}$$

with given noise covariance matrix $W = E(w_t w_t^T)$.

The kalman filter gives

$$\begin{aligned} \bar{y}_t &= D_t\bar{\theta}_t && (183)\\ K_t &= \hat{X}_t D_t^T W^{-1} = \bar{X}_t D_t^T (D_t\bar{X}_t D_t^T + W)^{-1} && (184)\\ \hat{\theta}_t &= \bar{\theta}_t + K_t(y_t - \bar{y}_t) && (185)\\ \bar{\theta}_{t+1} &= \hat{\theta}_t && (186)\\ \hat{X}_t &= \bar{X}_t - \bar{X}_t D_t^T (W + D_t\bar{X}_t D_t^T)^{-1} D_t\bar{X}_t && (187)\\ \bar{X}_{t+1} &= \hat{X}_t && (188) \end{aligned}$$

From this we have that $\bar{X}_t = \hat{X}_{t-1}$ which gives with $D_t = \varphi_t^T$ that

$$\hat{X}_t = \hat{X}_{t-1} - \hat{X}_{t-1}\varphi_t(W + \varphi_t^T\hat{X}_{t-1}\varphi_t)^{-1}\varphi_t^T\hat{X}_{t-1} \quad (189)$$

Comparing this with the ROLS algorithm, Equation (174) shows that

$$P_t = \hat{X}_t = \mathrm{E}((\theta_t - \hat{\theta}_t)(\theta_t - \hat{\theta}_t)^T) \quad (190)$$

Using (186) in (185) gives

$$\hat{\theta}_t = \hat{\theta}_{t-1} + K_t(y_t - \varphi_t^T \hat{\theta}_{t-1}). \tag{191}$$

Hence, $\bar{\theta}_t = \hat{\theta}_{t-1}$. Finally, the comparisons with ROLS and the Kalman filter also shows that

$$\Lambda = W^{-1}. \tag{192}$$

Hence, the optimal weighting matrix in the OLS algorithm is the inverse of the measurements noise covariance matrix.

The formulation of the Kalman filter gain presented in (184) is slightly different from the more common formulation, viz.

$$K_t = \bar{X}_t D_t^T (W + D_t \bar{X} D_t^T)^{-1}. \tag{193}$$

In order to prove that (184) are equivalent we substititute $\hat{X}_t$ into the first expression in (184), i.e.,

$$\begin{aligned}
K_t &= \hat{X}_t D_t^T W^{-1} \\
&= (\bar{X}_t - \bar{X}_t D_t^T (W + D_t \bar{X}_t D_t^T)^{-1} D_t \bar{X}_t) D_t^T W^{-1} \\
&= (\bar{X}_t D_t^T W^{-1} - \bar{X}_t D_t^T (W + D_t \bar{X}_t D_t^T)^{-1} D_t \bar{X}_t D_t^T W^{-1}) \\
&= \bar{X}_t D_t^T (W^{-1} - (W + D_t \bar{X}_t D_t^T)^{-1} D_t \bar{X}_t D_t^T W^{-1}) \\
&= \bar{X}_t D_t^T (W + D_t \bar{X} D_t^T)^{-1} ((W + D_t \bar{X} D_t^T) W^{-1} - D_t \bar{X}_t D_t^T W^{-1}) \\
&= \bar{X}_t D_t^T (W + D_t \bar{X} D_t^T)^{-1}.
\end{aligned} \tag{194}$$

## 7.2 ROLS with forgetting factor

Consider the following modification of the objective in (52), i.e.,

$$V_t(\theta) = \sum_{k=1}^{t} \lambda^{t-k} \epsilon_k^T(\theta) \Lambda \epsilon_k(\theta). \tag{195}$$

where we simply have set $N = t$ and omitted the mean $\frac{1}{t}$ and included a forgetting factor $\lambda$ in order to be able to weighting the newest data more than old data. We have typically that $0 < \lambda \leq 1$, often $\lambda = 0.99$.

The least squares estimate is given by

$$\hat{\theta}_t = P_t \sum_{k=1}^{t} \lambda^{t-k} \varphi_k \Lambda y_k. \tag{196}$$

where

$$P_t = (\sum_{k=1}^{t} \lambda^{t-k} \varphi_k \Lambda \varphi_k^T)^{-1}. \tag{197}$$

A recursive formulation is deduced as follows.

### 7.2.1 Recursive computation of $P_t$

Let us derive a recursive formulation for the covariance matrix $P_t$. We have from the definition of $P_t$ that

$$\begin{aligned} P_t &= (\sum_{k=1}^{t} \lambda^{t-k} \varphi_k \Lambda \varphi_k^T)^{-1} \\ &= (\sum_{k=1}^{t-1} \lambda^{t-k} \varphi_k \Lambda \varphi_k^T + \varphi_t \Lambda \varphi_t^T)^{-1} \\ &= (\underbrace{\lambda \sum_{k=1}^{t-1} \lambda^{t-1-k} \varphi_k \Lambda \varphi_k^T}_{P_{t-1}^{-1}} + \varphi_t \Lambda \varphi_t^T)^{-1} \end{aligned} \tag{198}$$

Using that

$$P_{t-1} = (\sum_{k=1}^{t-1} \lambda^{t-1-k} \varphi_k \Lambda \varphi_k^T)^{-1} \tag{199}$$

gives

$$P_t = (\lambda P_{t-1}^{-1} + \varphi_t \Lambda \varphi_t^T)^{-1} \tag{200}$$

Using the matrix inversion lemma (173) we have the more common covariance update equation

$$\lambda P_t = P_{t-1} - P_{t-1}\varphi_t(\lambda \Lambda^{-1} + \varphi_t^T P_{t-1} \varphi_t)^{-1} \varphi_t^T P_{t-1} \tag{201}$$

### 7.2.2 Recursive computation of $\hat{\theta}_t$

From the OLS parameter estimate (196) we have that

$$\begin{aligned} \hat{\theta}_t &= P_t \sum_{k=1}^{t} \lambda^{t-k} \varphi_k \Lambda y_k \\ &= P_t (\sum_{k=1}^{t-1} \lambda^{t-k} \varphi_k \Lambda y_k + \varphi_t \Lambda y_t) \\ &= P_t (\underbrace{\lambda \sum_{k=1}^{t-1} \lambda^{t-1-k} \varphi_k \Lambda y_k}_{P_{t-1}^{-1} \hat{\theta}_{t-1}} + \varphi_t \Lambda y_t). \end{aligned} \tag{202}$$

From the OLS parameter estimate (196) we also have that

$$\hat{\theta}_{t-1} = P_{t-1} \sum_{k=1}^{t-1} \lambda^{t-1-k} \varphi_k \Lambda y_k. \tag{203}$$

Substituting (203) into (202) gives

$$\hat{\theta}_t = P_t(\lambda P_{t-1}^{-1}\hat{\theta}_{t-1} + \varphi_t \Lambda y_t). \tag{204}$$

From (200) we have that

$$\lambda P_{t-1}^{-1} = P_t^{-1} - \varphi_t \Lambda \varphi_t^T \tag{205}$$

Substituting this into (204) gives

$$\begin{aligned} \hat{\theta}_t &= P_t((P_t^{-1} - \varphi_t \Lambda \varphi_t^T)\hat{\theta}_{t-1} + \varphi_t \Lambda y_t) \\ &= \hat{\theta}_{t-1} + P_t \varphi_t \Lambda y_t - P_t \varphi_t \Lambda \varphi_t^T \hat{\theta}_{t-1} \end{aligned} \tag{206}$$

hence,

$$\hat{\theta}_t = \hat{\theta}_{t-1} + P_t \varphi_t \Lambda (y_t - \varphi_t^T \hat{\theta}_{t-1}) \tag{207}$$

# 8 Higher order ARX modeling

It is well known that general linear dynamic systems, in state space form, or input output ARMAX formulations, may be approximated by higher order ARX systems. We will in this section discuss this and use it for subspace system identification. One advantage of this approach is that the theory is applicable for both open as well as closed loop systems. A drawback is that it requires that the input experiment is rich enough with perturbations in order to estimate the higher order ARX model, and this usually limit the approach for practical reasons. The variance of the resulting model parameters will also usually be larger than a direct prediction error approach. The discussion gives also some relationships to the subspace system identification algorithm DSR_e.

A system identification algorithm based on the identification of a higher order ARX (OLS) model may be described in the following items:

**Algorithm 8.1 (HARX algorithm)**

1. *Identify a higher order ARX model using the OLS approach. The order of the ARX model should be chosen large enough, and such that the residual is approximately white noise. One have to ensure that the input or reference signal experiment is rich enough with perturbations in order for the OLS problem to be well defined.*

2. *Form a higher order state space model from the ARX model parameters and then form the necessary number of impulse response matrices. the impulse response matrices may also be formed directly from the ARX model parameters.*

3. *Use Hankel matrix realization theory to compute a reduced order state space model of correct order.*

## 8.1 Miscellaneous examples

**Example 8.1 (Higher order ARX model)**
*Given a 1st order state space model in the form*

$$\begin{aligned} x_{k+1} &= ax_k + bu_k + ce_k, && (208)\\ y_k &= x_k + e_k. && (209) \end{aligned}$$

*This can be written as the following ARMAX model*

$$y_k + a_1 y_{k-1} = b_1 u_{k-1} + e_k + c_1 e_{k-1}, \tag{210}$$

*where $a_1 = -a$, $b_1 = b$ and $c_1 = c - a$.*

*Let us now investigate how well the parameters can be estimated by a third order ARX model.*

*Putting $k := k + 1$ in (210) and substitute for $e_k$ solved from (210) into this equation gives a second order difference equation. Repetition of this gives the following third order difference model*

$$\begin{aligned} y_{k+2} + (a_1 - c_1)y_{k+1} + (c_1^2 - c_1 a_1)y_k + c_1^2 a_1 y_{k-1} &= b_1 u_{k+1} - c_1 b_1 u_k + c_1^2 b_1 u_{k-1} \\ &+ e_{k+2} + c_1^3 e_{k-1}. \end{aligned} \tag{211}$$

*This can be written as the approximate linear regression model*

$$y_k = \overbrace{\begin{bmatrix} y_{k-1} & y_{k-2} & y_{k-3} & u_{k-1} & u_{k-2} & u_{k-3} \end{bmatrix}}^{\varphi_k^T} \overbrace{\begin{bmatrix} c_1 - a_1 \\ c_1 a_1 - c_1^2 \\ -c_1^2 \\ b_1 \\ -c_1 b_1 \\ c_1^2 b_1 \end{bmatrix}}^{\theta} + e_k + c_1^3 e_{k-3} \tag{212}$$

*The parameters in this model can be approximately estimated via an ARX model solved by the Ordinary Least Squares (OLS) method when $c_1^3 = (c-a)^3 \approx 0$. Note also that the predictor (Kalman filter) on innovations form is stable. This means that the magnitude of the eigenvalues of the predictor, $a - cd$, is less than one. Hence, $c_1^3 = (c - a)^3 \approx 0$ is a good assumption.*

**Example 8.2 (Higher order ARX model)**
*Given a 1st order state space model in the form*

$$\begin{aligned} x_{k+1} &= ax_k + bu_k + ke_k, && (213)\\ y_k &= x_k + e_k. && (214) \end{aligned}$$

*This can be written as the following 1st order ARMAX model*

$$y_k = ay_{k-1} + bu_{k-1} + e_k + (k - a)e_{k-1}, \tag{215}$$

*Note also that the noise term may be expressed as*

$$e_k = y_k - ay_{k-1} - bu_{k-1} - (k-a)e_{k-1}, \tag{216}$$

*Putting $k := k-1$ in (216) and substituting into (215) gives the 2nd order ARX model approximation*

$$y_k = ky_{k-1} - a(k-a)y_{k-2} + bu_{k-1} - b(k-a)u_{k-2} + e_k - (k-a)^2 e_{k-2}, \tag{217}$$

*when $(k-a)^2 \approx 0$. Similarly expressing $e_{k-2}$ from (216) and substituting into (217) gives the 3rd order ARX model approximation*

$$\begin{aligned} y_k &= ky_{k-1} + (-a(k-a) - (k-a)^2)y_{k-2} + a(k-a)^2 y_{k-3} \\ &+ bu_{k-1} - b(k-a)u_{k-2} + b(k-a)^2 u_{k-3} + e_k + (k-a)^3 e_{k-3}, \end{aligned} \tag{218}$$

*when $(k-a)^3 \approx 0$.*

# 9 Examples on using the Ordinary Least Squares method

**Example 9.1 (Parameters in Antoine's equation)**
*Given the Antoine equation for the relationship between temperature $T$ and vapor pressure $p$ in pure components such as e.g. saturated steam, i.e.*

$$p = e^{a - \frac{b}{T+c}}. \tag{219}$$

*This is a non-linear relation between temperature $T$ and vapor pressure $p$.*

*With some algebra we may formulate a linear regression model $y_k = \varphi_k^T \theta$*

$$\overbrace{T\ln(p)}^{y_k} = \overbrace{\begin{bmatrix} -\ln(p) & T & 1 \end{bmatrix}}^{\varphi_k^T} \overbrace{\begin{bmatrix} c \\ a \\ ac-b \end{bmatrix}}^{\theta} \tag{220}$$

*In order to show this from eq. (219) we find*

$$\ln(p) = a - \frac{b}{T+c}. \tag{221}$$

*Multiplying (221)) with $T+c$ gives*

$$(T+c)\ln(p) = a(T+c) - b, \tag{222}$$

*and from eq. (222) we find the linear regression model eq. (220).*

**Example 9.2 (Parameters in Antoine's equation)**
*Given the Antoine equation for the relationship between temperature $T$, and vapor pressure $p$, in pure components such as e.g. saturated steam, as also discussed in Example 9.1 i.e.*

$$p = e^{a - \frac{b}{T+c}}. \tag{223}$$

*This is a non-linear relation between temperature $T$ and vapor pressure $p$.*

*A linear regression model of the form $y_i = \varphi_i^T \theta$ where developed in Example 9.1, eq. (220 and we may use the OLS method to calculate the parameters $a$, $b$ and $c$ in the nonlinear eq. (223) from some observations of temperature and pressure.*

*Assume that we have given some pairs of observations $\{T_i, p_i\}\ \forall\ i = 1, \ldots, N$, of temperature $T_i$ and pressure $p_i$ in saturated steam and with $N = 10$ observations as stored in Table 1.*

Table 1: Pressure and temperature data in saturated steam

| $i$ | $p_i$ $[Bar]$ | $T_i$ $[^\circ C]$ |
|---|---|---|
| *1* | 1 | 99.1 |
| *2* | 2 | 119.6 |
| *3* | 3 | 132.9 |
| *4* | 4 | 142.9 |
| *5* | 5 | 151.1 |
| *6* | 6 | 158.1 |
| *7* | 7 | 164.2 |
| *8* | 8 | 169.6 |
| *9* | 9 | 174.5 |
| *10* | 10 | 179.0 |

*Using the results from Example 9.1 we form the linear regression matrix equation*

$$Y = XB, \tag{224}$$

*or equivalently $Y = \Phi\theta$.*

*Using eq. (220) then we may form the known data matrices $Y$ and $X$ as*

*follows*

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{10} \end{bmatrix} = \begin{bmatrix} T_1 \ln(p_1) \\ \vdots \\ T_i \ln(p_i) \\ \vdots \\ T_{10} \ln(p_{10}) \end{bmatrix} = \begin{bmatrix} 0 \\ 82.90 \\ 146.01 \\ 198.13 \\ 243.19 \\ 283.24 \\ 319.52 \\ 352.67 \\ 383.42 \\ 412.16 \end{bmatrix}, \tag{225}$$

$$X = \Phi = \begin{bmatrix} \varphi_1^T \\ \vdots \\ \varphi_i^T \\ \vdots \\ \varphi_{10}^T \end{bmatrix} = \begin{bmatrix} -\ln(p_1) & T_1 & 1 \\ \vdots & \vdots & \vdots \\ -\ln(p_i) & T_i & 1 \\ \vdots & \vdots & \vdots \\ -\ln(p_{10}) & T_{10} & 1 \end{bmatrix} = \begin{bmatrix} 0 & 99.10 & 1 \\ -0.69 & 119.60 & 1 \\ -1.10 & 132.90 & 1 \\ -1.39 & 142.92 & 1 \\ -1.61 & 151.10 & 1 \\ -1.79 & 158.08 & 1 \\ -1.95 & 164.20 & 1 \\ -2.08 & 169.60 & 1 \\ -2.20 & 174.50 & 1 \\ -2.30 & 179.00 & 1 \end{bmatrix}. \tag{226}$$

*Then we find the ordinary least squares estimate of the parameter vector $\theta$ (or equivalently the vector of regressor matrix $B$) as follows*

$$B_{\text{OLS}} = \hat{\theta} = (X^T X)^{-1} X^T Y = \begin{bmatrix} c \\ a \\ ac - b \end{bmatrix} = \begin{bmatrix} 226.37 \\ 11.68 \\ -1157.23 \end{bmatrix}. \tag{227}$$

*Finally we find the parameters in the Antoine's equation $a = 11.68$, $c = 226.37$ and $b = ac + 1157.23 = 3800.85$.*