Continuous Time Stochastic Modelling

CTSM 2.3

-

Mathematics Guide

Niels Rode Kristensen, Henrik Madsen

December 10, 2003

Technical University of Denmark

Contents

1 The mathematics behind the algorithms of CTSM **1**

1.1 Parameter estimation 1

1.1.1 Model structures 1

1.1.2 Parameter estimation methods 2

1.1.3 Filtering methods 5

1.1.4 Data issues 20

1.1.5 Optimisation issues 22

1.1.6 Performance issues 26

1.2 Other features 26

1.2.1 Various statistics 26

1.2.2 Validation data generation 28

References **31**

1

The mathematics behind the algorithms of CTSM

The following is a complete mathematical outline of the algorithms of **CTSM**.

1.1 Parameter estimation

The primary feature in **CTSM** is estimation of parameters in continuous-discrete stochastic state space models on the basis of experimental data.

1.1.1 Model structures

CTSM differentiates between three different model structures for continuous-discrete stochastic state space models as outlined in the following.

1.1.1.1 The nonlinear model

The most general of these model structures is the *nonlinear* (NL) model, which can be described by the following equations:

$$d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\theta})dt + \boldsymbol{\sigma}(\boldsymbol{u}_t, t, \boldsymbol{\theta})d\boldsymbol{\omega}_t \tag{1.1}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) + \boldsymbol{e}_k \tag{1.2}$$

where $t \in \mathbb{R}$ is time, $\boldsymbol{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ is a vector of state variables, $\boldsymbol{u}_t \in \mathcal{U} \subset \mathbb{R}^m$ is a vector of input variables, $\boldsymbol{y}_k \in \mathcal{Y} \subset \mathbb{R}^l$ is a vector of output variables, $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p$ is a vector of parameters, $\boldsymbol{f}(\cdot) \in \mathbb{R}^n$, $\boldsymbol{\sigma}(\cdot) \in \mathbb{R}^{n \times n}$ and $\boldsymbol{h}(\cdot) \in \mathbb{R}^l$ are nonlinear functions, $\{\boldsymbol{\omega}_t\}$ is an n-dimensional standard Wiener process and $\{\boldsymbol{e}_k\}$ is an l-dimensional white noise process with $\boldsymbol{e}_k \in N(\boldsymbol{0}, \boldsymbol{S}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta}))$.

1.1.1.2 The linear time-varying model

A special case of the nonlinear model is the *linear time-varying* (LTV) model, which can be described by the following equations:

$$d\boldsymbol{x}_t = (\boldsymbol{A}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\theta})\boldsymbol{x}_t + \boldsymbol{B}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\theta})\boldsymbol{u}_t)\, dt + \boldsymbol{\sigma}(\boldsymbol{u}_t, t, \boldsymbol{\theta}) d\boldsymbol{\omega}_t \tag{1.3}$$

$$\boldsymbol{y}_k = \boldsymbol{C}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k, \boldsymbol{\theta})\boldsymbol{x}_k + \boldsymbol{D}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k, \boldsymbol{\theta})\boldsymbol{u}_k + \boldsymbol{e}_k \tag{1.4}$$

where $t \in \mathbb{R}$ is time, $\boldsymbol{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ is a state vector, $\boldsymbol{u}_t \in \mathcal{U} \subset \mathbb{R}^m$ is an input vector, $\boldsymbol{y}_k \in \mathcal{Y} \subset \mathbb{R}^l$ is an output vector, $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p$ is a vector of parameters, $\boldsymbol{A}(\cdot) \in \mathbb{R}^{n\times n}$, $\boldsymbol{B}(\cdot) \in \mathbb{R}^{n\times m}$, $\boldsymbol{\sigma}(\cdot) \in \mathbb{R}^{n\times n}$, $\boldsymbol{C}(\cdot) \in \mathbb{R}^{l\times n}$ and $\boldsymbol{D}(\cdot) \in \mathbb{R}^{l\times m}$ are nonlinear functions, $\{\boldsymbol{\omega}_t\}$ is an n-dimensional standard Wiener process and $\{\boldsymbol{e}_k\}$ is an l-dimensional white noise process with $\boldsymbol{e}_k \in N(\boldsymbol{0}, \boldsymbol{S}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta}))$.

1.1.1.3 The linear time-invariant model

A special case of the linear time-varying model is the *linear time-invariant* (LTI) model, which can be described by the following equations:

$$d\boldsymbol{x}_t = (\boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{x}_t + \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{u}_t)\, dt + \boldsymbol{\sigma}(\boldsymbol{\theta}) d\boldsymbol{\omega}_t \tag{1.5}$$

$$\boldsymbol{y}_k = \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{x}_k + \boldsymbol{D}(\boldsymbol{\theta})\boldsymbol{u}_k + \boldsymbol{e}_k \tag{1.6}$$

where $t \in \mathbb{R}$ is time, $\boldsymbol{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ is a state vector, $\boldsymbol{u}_t \in \mathcal{U} \subset \mathbb{R}^m$ is an input vector, $\boldsymbol{y}_k \in \mathcal{Y} \subset \mathbb{R}^l$ is an output vector, $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p$ is a vector of parameters, $\boldsymbol{A}(\cdot) \in \mathbb{R}^{n\times n}$, $\boldsymbol{B}(\cdot) \in \mathbb{R}^{n\times m}$, $\boldsymbol{\sigma}(\cdot) \in \mathbb{R}^{n\times n}$, $\boldsymbol{C}(\cdot) \in \mathbb{R}^{l\times n}$ and $\boldsymbol{D}(\cdot) \in \mathbb{R}^{l\times m}$ are nonlinear functions, $\{\boldsymbol{\omega}_t\}$ is an n-dimensional standard Wiener process and $\{\boldsymbol{e}_k\}$ is an l-dimensional white noise process with $\boldsymbol{e}_k \in N(\boldsymbol{0}, \boldsymbol{S}(\boldsymbol{\theta}))$.

1.1.2 Parameter estimation methods

CTSM allows a number of different methods to be applied to estimate the parameters of the above model structures as outlined in the following.

1.1.2.1 Maximum likelihood estimation

Given a particular model structure, *maximum likelihood* (ML) estimation of the unknown parameters can be performed by finding the parameters $\boldsymbol{\theta}$ that maximize the likelihood function of a given sequence of measurements $\boldsymbol{y}_0$, $\boldsymbol{y}_1$, $\ldots$, $\boldsymbol{y}_k$, $\ldots$, $\boldsymbol{y}_N$. By introducing the notation:

$$\mathcal{Y}_k = [\boldsymbol{y}_k, \boldsymbol{y}_{k-1}, \ldots, \boldsymbol{y}_1, \boldsymbol{y}_0] \tag{1.7}$$

the likelihood function is the joint probability density:

$$L(\boldsymbol{\theta}; \mathcal{Y}_N) = p(\mathcal{Y}_N|\boldsymbol{\theta}) \tag{1.8}$$

or equivalently:

$$L(\boldsymbol{\theta};\mathcal{Y}_N) = \left(\prod_{k=1}^{N} p(\boldsymbol{y}_k|\mathcal{Y}_{k-1},\boldsymbol{\theta})\right) p(\boldsymbol{y}_0|\boldsymbol{\theta}) \tag{1.9}$$

where the rule $P(A \cap B) = P(A|B)P(B)$ has been applied to form a product of conditional probability densities. In order to obtain an exact evaluation of the likelihood function, the initial probability density $p(\boldsymbol{y}_0|\boldsymbol{\theta})$ must be known and all subsequent conditional densities must be determined by successively solving Kolmogorov's forward equation and applying Bayes' rule (Jazwinski, 1970), but this approach is computationally infeasible in practice. However, since the diffusion terms in the above model structures do not depend on the state variables, a simpler alternative can be used. More specifically, a method based on Kalman filtering can be applied for LTI and LTV models, and an approximate method based on extended Kalman filtering can be applied for NL models. The latter approximation can be applied, because the stochastic differential equations considered are driven by Wiener processes, and because increments of a Wiener process are Gaussian, which makes it reasonable to assume, under some regularity conditions, that the conditional densities can be well approximated by Gaussian densities. The Gaussian density is completely characterized by its mean and covariance, so by introducing the notation:

$$\hat{\boldsymbol{y}}_{k|k-1} = E\{\boldsymbol{y}_k|\mathcal{Y}_{k-1},\boldsymbol{\theta}\} \tag{1.10}$$

$$\boldsymbol{R}_{k|k-1} = V\{\boldsymbol{y}_k|\mathcal{Y}_{k-1},\boldsymbol{\theta}\} \tag{1.11}$$

and:

$$\boldsymbol{\epsilon}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1} \tag{1.12}$$

the likelihood function can be written as follows:

$$L(\boldsymbol{\theta};\mathcal{Y}_N) = \left(\prod_{k=1}^{N} \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_k^T \boldsymbol{R}_{k|k-1}^{-1}\boldsymbol{\epsilon}_k\right)}{\sqrt{\det(\boldsymbol{R}_{k|k-1})}\left(\sqrt{2\pi}\right)^l}\right) p(\boldsymbol{y}_0|\boldsymbol{\theta}) \tag{1.13}$$

where, for given parameters and initial states, $\boldsymbol{\epsilon}_k$ and $\boldsymbol{R}_{k|k-1}$ can be computed by means of a Kalman filter (LTI and LTV models) or an extended Kalman filter (NL models) as shown in Sections 1.1.3.1 and 1.1.3.2 respectively. Further conditioning on $\boldsymbol{y}_0$ and taking the negative logarithm gives:

$$\begin{aligned} -\ln\left(L(\boldsymbol{\theta};\mathcal{Y}_N|\boldsymbol{y}_0)\right) = \frac{1}{2}\sum_{k=1}^{N}\left(\ln(\det(\boldsymbol{R}_{k|k-1})) + \boldsymbol{\epsilon}_k^T \boldsymbol{R}_{k|k-1}^{-1}\boldsymbol{\epsilon}_k\right) \\ + \frac{1}{2}\left(\sum_{k=1}^{N} l\right)\ln(2\pi) \end{aligned} \tag{1.14}$$

and ML estimates of the parameters (and optionally of the initial states) can now be determined by solving the following nonlinear optimisation problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta}\left\{-\ln\left(L(\boldsymbol{\theta};\mathcal{Y}_N|\boldsymbol{y}_0)\right)\right\} \tag{1.15}$$

1.1.2.2 Maximum a posteriori estimation

If prior information about the parameters is available in the form of a prior probability density function $p(\boldsymbol{\theta})$, Bayes' rule can be applied to give an improved estimate by forming the posterior probability density function:

$$p(\boldsymbol{\theta}|\mathcal{Y}_N) = \frac{p(\mathcal{Y}_N|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{Y}_N)} \propto p(\mathcal{Y}_N|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{1.16}$$

and subsequently finding the parameters that maximize this function, i.e. by performing *maximum a posteriori* (MAP) estimation. A nice feature of this expression is the fact that it reduces to the likelihood function, when no prior information is available ($p(\boldsymbol{\theta})$ uniform), making ML estimation a special case of MAP estimation. In fact, this formulation also allows MAP estimation on a subset of the parameters ($p(\boldsymbol{\theta})$ partly uniform). By introducing the notation[1]:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}} = E\{\boldsymbol{\theta}\} \tag{1.17}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = V\{\boldsymbol{\theta}\} \tag{1.18}$$

and:

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}} = \boldsymbol{\theta} - \boldsymbol{\mu}_{\boldsymbol{\theta}} \tag{1.19}$$

and by assuming that the prior probability density of the parameters is Gaussian, the posterior probability density function can be written as follows:

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{Y}_N) \propto & \left(\prod_{k=1}^{N} \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_k^T \boldsymbol{R}_{k|k-1}^{-1} \boldsymbol{\epsilon}_k\right)}{\sqrt{\det(\boldsymbol{R}_{k|k-1})}\left(\sqrt{2\pi}\right)^l} \right) p(\boldsymbol{y}_0|\boldsymbol{\theta}) \\ & \times \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\right)}{\sqrt{\det(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})}\left(\sqrt{2\pi}\right)^p} \end{aligned} \tag{1.20}$$

Further conditioning on $\boldsymbol{y}_0$ and taking the negative logarithm gives:

$$\begin{aligned} -\ln\left(p(\boldsymbol{\theta}|\mathcal{Y}_N, \boldsymbol{y}_0)\right) \propto & \frac{1}{2}\sum_{k=1}^{N}\left(\ln(\det(\boldsymbol{R}_{k|k-1})) + \boldsymbol{\epsilon}_k^T \boldsymbol{R}_{k|k-1}^{-1}\boldsymbol{\epsilon}_k\right) \\ & + \frac{1}{2}\left(\left(\sum_{k=1}^{N} l\right) + p\right)\ln(2\pi) \\ & + \frac{1}{2}\ln(\det(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})) + \frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} \end{aligned} \tag{1.21}$$

and MAP estimates of the parameters (and optionally of the initial states) can now be determined by solving the following nonlinear optimisation problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta}\left\{-\ln\left(p(\boldsymbol{\theta}|\mathcal{Y}_N, \boldsymbol{y}_0)\right)\right\} \tag{1.22}$$

[1] In practice $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ is specified as $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \boldsymbol{\sigma}_{\boldsymbol{\theta}}\boldsymbol{R}_{\boldsymbol{\theta}}\boldsymbol{\sigma}_{\boldsymbol{\theta}}$, where $\boldsymbol{\sigma}_{\boldsymbol{\theta}}$ is a diagonal matrix of the prior standard deviations and $\boldsymbol{R}_{\boldsymbol{\theta}}$ is the corresponding prior correlation matrix.

1.1.2.3 Using multiple independent data sets

If, instead of a single sequence of measurements, multiple consecutive, but yet separate, sequences of measurements, i.e. $\mathcal{Y}_{N_1}^1$, $\mathcal{Y}_{N_2}^2$, ..., $\mathcal{Y}_{N_i}^i$, ..., $\mathcal{Y}_{N_S}^S$, are available, a similar estimation method can be applied by expanding the expression for the posterior probability density function to the general form:

$$p(\boldsymbol{\theta}|\mathbf{Y}) \propto \left(\prod_{i=1}^{S} \left(\prod_{k=1}^{N_i} \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\epsilon}_k^i)^T (\boldsymbol{R}_{k|k-1}^i)^{-1} \boldsymbol{\epsilon}_k^i\right)}{\sqrt{\det(\boldsymbol{R}_{k|k-1}^i)}\left(\sqrt{2\pi}\right)^l} \right) p(\boldsymbol{y}_0^i|\boldsymbol{\theta}) \right) \times \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\right)}{\sqrt{\det(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})}\left(\sqrt{2\pi}\right)^p} \tag{1.23}$$

where:

$$\mathbf{Y} = [\mathcal{Y}_{N_1}^1, \mathcal{Y}_{N_2}^2, \ldots, \mathcal{Y}_{N_i}^i, \ldots, \mathcal{Y}_{N_S}^S] \tag{1.24}$$

and where the individual sequences of measurements are assumed to be stochastically independent. This formulation allows MAP estimation on multiple data sets, but, as special cases, it also allows ML estimation on multiple data sets ($p(\boldsymbol{\theta})$ uniform), MAP estimation on a single data set ($S = 1$) and ML estimation on a single data set ($p(\boldsymbol{\theta})$ uniform, $S = 1$). Further conditioning on:

$$\mathbf{y_0} = [\boldsymbol{y}_0^1, \boldsymbol{y}_0^2, \ldots, \boldsymbol{y}_0^i, \ldots, \boldsymbol{y}_0^S] \tag{1.25}$$

and taking the negative logarithm gives:

$$\begin{aligned} -\ln\left(p(\boldsymbol{\theta}|\mathbf{Y}, \mathbf{y_0})\right) \propto & \frac{1}{2} \sum_{i=1}^{S} \sum_{k=1}^{N_i} \left(\ln(\det(\boldsymbol{R}_{k|k-1}^i)) + (\boldsymbol{\epsilon}_k^i)^T (\boldsymbol{R}_{k|k-1}^i)^{-1} \boldsymbol{\epsilon}_k^i \right) \\ & + \frac{1}{2} \left(\left(\sum_{i=1}^{S} \sum_{k=1}^{N_i} l \right) + p \right) \ln(2\pi) \\ & + \frac{1}{2} \ln(\det(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})) + \frac{1}{2} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} \boldsymbol{\epsilon}_{\boldsymbol{\theta}} \end{aligned} \tag{1.26}$$

and estimates of the parameters (and optionally of the initial states) can now be determined by solving the following nonlinear optimisation problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \Theta} \left\{ -\ln\left(p(\boldsymbol{\theta}|\mathbf{Y}, \mathbf{y_0})\right) \right\} \tag{1.27}$$

1.1.3 Filtering methods

CTSM computes the innovation vectors $\boldsymbol{\epsilon}_k$ (or $\boldsymbol{\epsilon}_k^i$) and their covariance matrices $\boldsymbol{R}_{k|k-1}$ (or $\boldsymbol{R}_{k|k-1}^i$) recursively by means of a Kalman filter (LTI and LTV models) or an extended Kalman filter (NL models) as outlined in the following.

1.1.3.1 Kalman filtering

For LTI and LTV models $\boldsymbol{\epsilon}_k$ (or $\boldsymbol{\epsilon}_k^i$) and $\boldsymbol{R}_{k|k-1}$ (or $\boldsymbol{R}_{k|k-1}^i$) can be computed for a given set of parameters $\boldsymbol{\theta}$ and initial states $\boldsymbol{x}_0$ by means of a continuous-discrete Kalman filter, i.e. by means of the output *prediction* equations:

$$\hat{\boldsymbol{y}}_{k|k-1} = \boldsymbol{C}\hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{D}\boldsymbol{u}_k \tag{1.28}$$

$$\boldsymbol{R}_{k|k-1} = \boldsymbol{C}\boldsymbol{P}_{k|k-1}\boldsymbol{C}^T + \boldsymbol{S} \tag{1.29}$$

the *innovation* equation:

$$\boldsymbol{\epsilon}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1} \tag{1.30}$$

the Kalman *gain* equation:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{C}^T\boldsymbol{R}_{k|k-1}^{-1} \tag{1.31}$$

the *updating* equations:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\boldsymbol{\epsilon}_k \tag{1.32}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{R}_{k|k-1}\boldsymbol{K}_k^T \tag{1.33}$$

and the state *prediction* equations:

$$\frac{d\hat{\boldsymbol{x}}_{t|k}}{dt} = \boldsymbol{A}\hat{\boldsymbol{x}}_{t|k} + \boldsymbol{B}\boldsymbol{u}_t \ , \ t \in [t_k, t_{k+1}[\tag{1.34}$$

$$\frac{d\boldsymbol{P}_{t|k}}{dt} = \boldsymbol{A}\boldsymbol{P}_{t|k} + \boldsymbol{P}_{t|k}\boldsymbol{A}^T + \boldsymbol{\sigma}\boldsymbol{\sigma}^T \ , \ t \in [t_k, t_{k+1}[\tag{1.35}$$

where the following shorthand notation applies in the LTV case:

$$\begin{gathered}\boldsymbol{A} = \boldsymbol{A}(\hat{\boldsymbol{x}}_{t|k-1}, \boldsymbol{u}_t, t, \boldsymbol{\theta}) \ , \ \boldsymbol{B} = \boldsymbol{B}(\hat{\boldsymbol{x}}_{t|k-1}, \boldsymbol{u}_t, t, \boldsymbol{\theta}) \\ \boldsymbol{C} = \boldsymbol{C}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \ , \ \boldsymbol{D} = \boldsymbol{D}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \\ \boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{u}_t, t, \boldsymbol{\theta}) \ , \ \boldsymbol{S} = \boldsymbol{S}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta})\end{gathered} \tag{1.36}$$

and the following shorthand notation applies in the LTI case:

$$\begin{gathered}\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{\theta}) \ , \ \boldsymbol{B} = \boldsymbol{B}(\boldsymbol{\theta}) \\ \boldsymbol{C} = \boldsymbol{C}(\boldsymbol{\theta}) \ , \ \boldsymbol{D} = \boldsymbol{D}(\boldsymbol{\theta}) \\ \boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\theta}) \ , \ \boldsymbol{S} = \boldsymbol{S}(\boldsymbol{\theta})\end{gathered} \tag{1.37}$$

Initial conditions for the Kalman filter are $\hat{\boldsymbol{x}}_{t|t_0} = \boldsymbol{x}_0$ and $\boldsymbol{P}_{t|t_0} = \boldsymbol{P}_0$, which may either be pre-specified or estimated along with the parameters as a part of the overall problem (see Section 1.1.3.4). In the LTI case, and in the LTV case, if $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$, $\boldsymbol{\sigma}$ and $\boldsymbol{S}$ are assumed constant between samples[2], (1.34)

[2]In practice the time interval $t \in [t_k, t_{k+1}[$ is subsampled for LTV models, and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$, $\boldsymbol{\sigma}$ and $\boldsymbol{S}$ are evaluated at each subsampling instant to provide a better approximation.

and (1.35) can be replaced by their discrete time counterparts, which can be derived from the solution to the stochastic differential equation:

$$d\boldsymbol{x}_t = (\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t)\,dt + \boldsymbol{\sigma} d\boldsymbol{\omega}_t \ , \ t \in [t_k, t_{k+1}[\tag{1.38}$$

i.e. from:

$$\boldsymbol{x}_{t_{k+1}} = e^{\boldsymbol{A}(t_{k+1}-t_k)}\boldsymbol{x}_{t_k} + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-s)}\boldsymbol{B}\boldsymbol{u}_s ds + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-s)}\boldsymbol{\sigma} d\boldsymbol{\omega}_s \tag{1.39}$$

which yields:

$$\hat{\boldsymbol{x}}_{k+1|k} = E\{\boldsymbol{x}_{t_{k+1}}|\boldsymbol{x}_{t_k}\} = e^{\boldsymbol{A}(t_{k+1}-t_k)}\hat{\boldsymbol{x}}_{k|k} + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-s)}\boldsymbol{B}\boldsymbol{u}_s ds \tag{1.40}$$

$$\begin{aligned}\boldsymbol{P}_{k+1|k} = E\{\boldsymbol{x}_{t_{k+1}}\boldsymbol{x}_{t_{k+1}}^T|\boldsymbol{x}_{t_k}\} = e^{\boldsymbol{A}(t_{k+1}-t_k)}\boldsymbol{P}_{k|k}\left(e^{\boldsymbol{A}(t_{k+1}-t_k)}\right)^T \\ + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-s)}\boldsymbol{\sigma}\boldsymbol{\sigma}^T\left(e^{\boldsymbol{A}(t_{k+1}-s)}\right)^T ds\end{aligned} \tag{1.41}$$

where the following shorthand notation applies in the LTV case:

$$\begin{aligned}\boldsymbol{A} = \boldsymbol{A}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \ , \ \boldsymbol{B} = \boldsymbol{B}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \\ \boldsymbol{C} = \boldsymbol{C}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \ , \ \boldsymbol{D} = \boldsymbol{D}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \\ \boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \ , \ \boldsymbol{S} = \boldsymbol{S}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta})\end{aligned} \tag{1.42}$$

and the following shorthand notation applies in the LTI case:

$$\begin{aligned}\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{\theta}) \ , \ \boldsymbol{B} = \boldsymbol{B}(\boldsymbol{\theta}) \\ \boldsymbol{C} = \boldsymbol{C}(\boldsymbol{\theta}) \ , \ \boldsymbol{D} = \boldsymbol{D}(\boldsymbol{\theta}) \\ \boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\theta}) \ , \ \boldsymbol{S} = \boldsymbol{S}(\boldsymbol{\theta})\end{aligned} \tag{1.43}$$

In order to be able to use (1.40) and (1.41), the integrals of both equations must be computed. For this purpose the equations are rewritten to:

$$\begin{aligned}\hat{\boldsymbol{x}}_{k+1|k} &= e^{\boldsymbol{A}(t_{k+1}-t_k)}\hat{\boldsymbol{x}}_{k|k} + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-s)}\boldsymbol{B}\boldsymbol{u}_s ds \\ &= e^{\boldsymbol{A}\tau_s}\hat{\boldsymbol{x}}_{k|k} + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-s)}\boldsymbol{B}\left(\boldsymbol{\alpha}(s-t_k) + \boldsymbol{u}_k\right) ds \\ &= \boldsymbol{\Phi}_s\hat{\boldsymbol{x}}_{k|k} + \int_0^{\tau_s} e^{\boldsymbol{A}s}\boldsymbol{B}\left(\boldsymbol{\alpha}(\tau_s - s) + \boldsymbol{u}_k\right) ds \\ &= \boldsymbol{\Phi}_s\hat{\boldsymbol{x}}_{k|k} - \int_0^{\tau_s} e^{\boldsymbol{A}s} s ds \boldsymbol{B}\boldsymbol{\alpha} + \int_0^{\tau_s} e^{\boldsymbol{A}s} ds \boldsymbol{B}\left(\boldsymbol{\alpha}\tau_s + \boldsymbol{u}_k\right)\end{aligned} \tag{1.44}$$

and:

$$\begin{aligned}\boldsymbol{P}_{k+1|k} &= e^{\boldsymbol{A}(t_{k+1}-t_k)}\boldsymbol{P}_{k|k}\left(e^{\boldsymbol{A}(t_{k+1}-t_k)}\right)^T \\ &\quad + \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(t_{k+1}-s)}\boldsymbol{\sigma}\boldsymbol{\sigma}^T\left(e^{\boldsymbol{A}(t_{k+1}-s)}\right)^T ds \\ &= e^{\boldsymbol{A}\tau_s}\boldsymbol{P}_{k|k}\left(e^{\boldsymbol{A}\tau_s}\right)^T + \int_0^{\tau_s} e^{\boldsymbol{A}s}\boldsymbol{\sigma}\boldsymbol{\sigma}^T\left(e^{\boldsymbol{A}s}\right)^T ds \\ &= \boldsymbol{\Phi}_s\boldsymbol{P}_{k|k}\boldsymbol{\Phi}_s^T + \int_0^{\tau_s} e^{\boldsymbol{A}s}\boldsymbol{\sigma}\boldsymbol{\sigma}^T\left(e^{\boldsymbol{A}s}\right)^T ds\end{aligned} \tag{1.45}$$

where $\tau_s = t_{k+1} - t_k$ and $\boldsymbol{\Phi}_s = e^{\boldsymbol{A}\tau_s}$, and where:

$$\boldsymbol{\alpha} = \frac{\boldsymbol{u}_{k+1} - \boldsymbol{u}_k}{t_{k+1} - t_k} \tag{1.46}$$

has been introduced to allow assumption of either *zero order hold* ($\boldsymbol{\alpha} = \boldsymbol{0}$) or *first order hold* ($\boldsymbol{\alpha} \neq \boldsymbol{0}$) on the inputs between sampling instants. The matrix exponential $\boldsymbol{\Phi}_s = e^{\boldsymbol{A}\tau_s}$ can be computed by means of a Padé approximation with repeated scaling and squaring (Moler and van Loan, 1978). However, both $\boldsymbol{\Phi}_s$ and the integral in (1.45) can be computed simultaneously through:

$$\exp\left(\begin{bmatrix}-\boldsymbol{A} & \boldsymbol{\sigma}\boldsymbol{\sigma}^T \\ \boldsymbol{0} & \boldsymbol{A}^T\end{bmatrix}\tau_s\right) = \begin{bmatrix}\boldsymbol{H}_1(\tau_s) & \boldsymbol{H}_2(\tau_s) \\ \boldsymbol{0} & \boldsymbol{H}_3(\tau_s)\end{bmatrix} \tag{1.47}$$

by combining submatrices of the result[3] (van Loan, 1978), i.e.:

$$\boldsymbol{\Phi}_s = \boldsymbol{H}_3^T(\tau_s) \tag{1.48}$$

and:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s}\boldsymbol{\sigma}\boldsymbol{\sigma}^T\left(e^{\boldsymbol{A}s}\right)^T ds = \boldsymbol{H}_3^T(\tau_s)\boldsymbol{H}_2(\tau_s) \tag{1.49}$$

Alternatively, this integral can be computed from the Lyapunov equation:

$$\begin{aligned}\boldsymbol{\Phi}_s\boldsymbol{\sigma}\boldsymbol{\sigma}^T\boldsymbol{\Phi}_s^T - \boldsymbol{\sigma}\boldsymbol{\sigma}^T &= \boldsymbol{A}\int_0^{\tau_s} e^{\boldsymbol{A}s}\boldsymbol{\sigma}\boldsymbol{\sigma}^T\left(e^{\boldsymbol{A}s}\right)^T ds \\ &\quad + \int_0^{\tau_s} e^{\boldsymbol{A}s}\boldsymbol{\sigma}\boldsymbol{\sigma}^T\left(e^{\boldsymbol{A}s}\right)^T ds\boldsymbol{A}^T\end{aligned} \tag{1.50}$$

but this approach has been found to be less feasible. The integrals in (1.44) are not as easy to deal with, especially if $\boldsymbol{A}$ is singular. However, this problem can be solved by introducing the singular value decomposition (SVD) of $\boldsymbol{A}$, i.e. $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, transforming the integrals and subsequently computing these.

[3] Within **CTSM** the specific implementation is based on the algorithms of Sidje (1998).

The first integral can be transformed as follows:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} s ds = \boldsymbol{U} \int_0^{\tau_s} \boldsymbol{U}^T e^{\boldsymbol{A}s} \boldsymbol{U} s ds \boldsymbol{U}^T = \boldsymbol{U} \int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} s ds \boldsymbol{U}^T \tag{1.51}$$

and, if $\boldsymbol{A}$ is singular, the matrix $\tilde{\boldsymbol{A}} = \boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{U} = \boldsymbol{U}^T\boldsymbol{A}\boldsymbol{U}$ has a special structure:

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{1.52}$$

which allows the integral to be computed as follows:

$$\begin{aligned}
\int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} s ds &= \int_0^{\tau_s} \left(\boldsymbol{I}s + \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} s^2 + \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^2 \frac{s^3}{2} + \cdots \right) ds \\
&= \int_0^{\tau_s} \left(\boldsymbol{I}s + \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} s^2 + \begin{bmatrix} \tilde{\boldsymbol{A}}_1^2 & \tilde{\boldsymbol{A}}_1\tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \frac{s^3}{2} + \cdots \right) ds \\
&= \begin{bmatrix} \int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}_1 s} s ds & \int_0^{\tau_s} \tilde{\boldsymbol{A}}_1^{-1} \left(e^{\tilde{\boldsymbol{A}}_1 s} - \boldsymbol{I} \right) s \tilde{\boldsymbol{A}}_2 ds \\ \boldsymbol{0} & \boldsymbol{I}\frac{\tau_s^2}{2} \end{bmatrix} \\
&= \begin{bmatrix} \left[\tilde{\boldsymbol{A}}_1^{-1} e^{\tilde{\boldsymbol{A}}_1 s} \left(\boldsymbol{I}s - \tilde{\boldsymbol{A}}_1^{-1} \right) \right]_0^{\tau_s} & \\ \boldsymbol{0} & \\ \tilde{\boldsymbol{A}}_1^{-1} \left[\tilde{\boldsymbol{A}}_1^{-1} e^{\tilde{\boldsymbol{A}}_1 s} \left(\boldsymbol{I}s - \tilde{\boldsymbol{A}}_1^{-1} \right) - \boldsymbol{I}\frac{s^2}{2} \right]_0^{\tau_s} \tilde{\boldsymbol{A}}_2 & \\ \boldsymbol{I}\frac{\tau_s^2}{2} & \end{bmatrix} \\
&= \begin{bmatrix} \tilde{\boldsymbol{A}}_1^{-1} \left(-\tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I} \right) + \tilde{\boldsymbol{\Phi}}_s^1 \tau_s \right) & \\ \boldsymbol{0} & \\ \tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{A}}_1^{-1} \left(-\tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I} \right) + \tilde{\boldsymbol{\Phi}}_s^1 \tau_s \right) - \boldsymbol{I}\frac{\tau_s^2}{2} \right) \tilde{\boldsymbol{A}}_2 & \\ \boldsymbol{I}\frac{\tau_s^2}{2} & \end{bmatrix}
\end{aligned} \tag{1.53}$$

where $\tilde{\boldsymbol{\Phi}}_s^1$ is the upper left part of the matrix:

$$\tilde{\boldsymbol{\Phi}}_s = \boldsymbol{U}^T \boldsymbol{\Phi}_s \boldsymbol{U} = \begin{bmatrix} \tilde{\boldsymbol{\Phi}}_s^1 & \tilde{\boldsymbol{\Phi}}_s^2 \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{1.54}$$

The second integral can be transformed as follows:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} ds = \boldsymbol{U} \int_0^{\tau_s} \boldsymbol{U}^T e^{\boldsymbol{A}s} \boldsymbol{U} ds \boldsymbol{U}^T = \boldsymbol{U} \int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} ds \boldsymbol{U}^T \tag{1.55}$$

and can subsequently be computed as follows:

$$\begin{aligned}
\int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} ds &= \int_0^{\tau_s} \left(\boldsymbol{I} + \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} s + \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^2 \frac{s^2}{2} + \cdots \right) ds \\
&= \int_0^{\tau_s} \left(\boldsymbol{I} + \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} s + \begin{bmatrix} \tilde{\boldsymbol{A}}_1^2 & \tilde{\boldsymbol{A}}_1\tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \frac{s^2}{2} + \cdots \right) ds \\
&= \begin{bmatrix} \int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}_1 s} ds & \int_0^{\tau_s} \tilde{\boldsymbol{A}}_1^{-1} \left(e^{\tilde{\boldsymbol{A}}_1 s} - \boldsymbol{I} \right) \tilde{\boldsymbol{A}}_2 ds \\ \boldsymbol{0} & \boldsymbol{I}\tau_s \end{bmatrix} \\
&= \begin{bmatrix} \left[\tilde{\boldsymbol{A}}_1^{-1} e^{\tilde{\boldsymbol{A}}_1 s} \right]_0^{\tau_s} & \tilde{\boldsymbol{A}}_1^{-1} \left[\tilde{\boldsymbol{A}}_1^{-1} e^{\tilde{\boldsymbol{A}}_1 s} - \boldsymbol{I}s \right]_0^{\tau_s} \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{I}\tau_s \end{bmatrix} \\
&= \begin{bmatrix} \tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I} \right) & \tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I} \right) - \boldsymbol{I}\tau_s \right) \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{I}\tau_s \end{bmatrix}
\end{aligned} \tag{1.56}$$

Depending on the specific singularity of $\boldsymbol{A}$ (see Section 1.1.3.3 for details on how this is determined in **CTSM**) and the particular nature of the inputs, several different cases are possible as shown in the following.

General case: Singular $\boldsymbol{A}$, first order hold on inputs

In the general case, the Kalman filter prediction can be calculated as follows:

$$\hat{\boldsymbol{x}}_{j+1} = \boldsymbol{\Phi}_s \hat{\boldsymbol{x}}_j - \boldsymbol{U} \int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} s ds \boldsymbol{U}^T \boldsymbol{B}\boldsymbol{\alpha} + \boldsymbol{U} \int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} ds \boldsymbol{U}^T \boldsymbol{B} \left(\boldsymbol{\alpha}\tau_s + \boldsymbol{u}_j \right) \tag{1.57}$$

with:

$$\int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} ds = \begin{bmatrix} \tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I} \right) & \tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I} \right) - \boldsymbol{I}\tau_s \right) \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{I}\tau_s \end{bmatrix} \tag{1.58}$$

and:

$$\begin{aligned}
\int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} s ds = \begin{bmatrix} \tilde{\boldsymbol{A}}_1^{-1} \left(-\tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I} \right) + \tilde{\boldsymbol{\Phi}}_s^1 \tau_s \right) \\ \boldsymbol{0} \end{bmatrix} \\
\begin{bmatrix} \tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{A}}_1^{-1} \left(-\tilde{\boldsymbol{A}}_1^{-1} \left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I} \right) + \tilde{\boldsymbol{\Phi}}_s^1 \tau_s \right) - \boldsymbol{I}\frac{\tau_s^2}{2} \right) \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{I}\frac{\tau_s^2}{2} \end{bmatrix}
\end{aligned} \tag{1.59}$$

Special case no. 1: Singular $\boldsymbol{A}$, zero order hold on inputs

The Kalman filter prediction for this special case can be calculated as follows:

$$\hat{\boldsymbol{x}}_{j+1} = \boldsymbol{\Phi}_s \hat{\boldsymbol{x}}_j + \boldsymbol{U} \int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} ds \boldsymbol{U}^T \boldsymbol{B} \boldsymbol{u}_j \tag{1.60}$$

with:

$$\int_0^{\tau_s} e^{\tilde{\boldsymbol{A}}s} ds = \begin{bmatrix} \tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I}\right) & \tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{\Phi}}_s^1 - \boldsymbol{I}\right) - \boldsymbol{I}\tau_s\right)\tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{I}\tau_s \end{bmatrix} \tag{1.61}$$

Special case no. 2: Nonsingular $\boldsymbol{A}$, first order hold on inputs

The Kalman filter prediction for this special case can be calculated as follows:

$$\hat{\boldsymbol{x}}_{j+1} = \boldsymbol{\Phi}_s \hat{\boldsymbol{x}}_j - \int_0^{\tau_s} e^{\boldsymbol{A}s} s ds \boldsymbol{B}\boldsymbol{\alpha} + \int_0^{\tau_s} e^{\boldsymbol{A}s} ds \boldsymbol{B}\left(\boldsymbol{\alpha}\tau_s + \boldsymbol{u}_j\right) \tag{1.62}$$

with:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} ds = \boldsymbol{A}^{-1}\left(\boldsymbol{\Phi}_s - \boldsymbol{I}\right) \tag{1.63}$$

and:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} s ds = \boldsymbol{A}^{-1}\left(-\boldsymbol{A}^{-1}\left(\boldsymbol{\Phi}_s - \boldsymbol{I}\right) + \boldsymbol{\Phi}_s\tau_s\right) \tag{1.64}$$

Special case no. 3: Nonsingular $\boldsymbol{A}$, zero order hold on inputs

The Kalman filter prediction for this special case can be calculated as follows:

$$\hat{\boldsymbol{x}}_{j+1} = \boldsymbol{\Phi}_s \hat{\boldsymbol{x}}_j + \int_0^{\tau_s} e^{\boldsymbol{A}s} ds \boldsymbol{B}\boldsymbol{u}_j \tag{1.65}$$

with:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} ds = \boldsymbol{A}^{-1}\left(\boldsymbol{\Phi}_s - \boldsymbol{I}\right) \tag{1.66}$$

Special case no. 4: Identically zero $\boldsymbol{A}$, first order hold on inputs

The Kalman filter prediction for this special case can be calculated as follows:

$$\hat{\boldsymbol{x}}_{j+1} = \hat{\boldsymbol{x}}_j - \int_0^{\tau_s} e^{\boldsymbol{A}s} s ds \boldsymbol{B}\boldsymbol{\alpha} + \int_0^{\tau_s} e^{\boldsymbol{A}s} ds \boldsymbol{B}\left(\boldsymbol{\alpha}\tau_s + \boldsymbol{u}_j\right) \tag{1.67}$$

with:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} ds = \boldsymbol{I}\tau_s \tag{1.68}$$

and:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} s ds = \boldsymbol{I}\frac{\tau_s^2}{2} \tag{1.69}$$

Special case no. 5: Identically zero $\boldsymbol{A}$, zero order hold on inputs

The Kalman filter prediction for this special case can be calculated as follows:

$$\hat{\boldsymbol{x}}_{j+1} = \hat{\boldsymbol{x}}_j + \int_0^{\tau_s} e^{\boldsymbol{A}s} ds \boldsymbol{B}\boldsymbol{u}_j \tag{1.70}$$

with:

$$\int_0^{\tau_s} e^{\boldsymbol{A}s} ds = \boldsymbol{I}\tau_s \tag{1.71}$$

1.1.3.2 Extended Kalman filtering

For NL models $\boldsymbol{\epsilon}_k$ (or $\boldsymbol{\epsilon}_k^i$) and $\boldsymbol{R}_{k|k-1}$ (or $\boldsymbol{R}_{k|k-1}^i$) can be computed for a given set of parameters $\boldsymbol{\theta}$ and initial states $\boldsymbol{x}_0$ by means of a continuous-discrete extended Kalman filter, i.e. by means of the output *prediction* equations:

$$\hat{\boldsymbol{y}}_{k|k-1} = \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \tag{1.72}$$

$$\boldsymbol{R}_{k|k-1} = \boldsymbol{C}\boldsymbol{P}_{k|k-1}\boldsymbol{C}^T + \boldsymbol{S} \tag{1.73}$$

the *innovation* equation:

$$\boldsymbol{\epsilon}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1} \tag{1.74}$$

the Kalman *gain* equation:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{C}^T\boldsymbol{R}_{k|k-1}^{-1} \tag{1.75}$$

the *updating* equations:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\boldsymbol{\epsilon}_k \tag{1.76}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{R}_{k|k-1}\boldsymbol{K}_k^T \tag{1.77}$$

and the state *prediction* equations:

$$\frac{d\hat{\boldsymbol{x}}_{t|k}}{dt} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{t|k}, \boldsymbol{u}_t, t, \boldsymbol{\theta}) \ , \ t \in [t_k, t_{k+1}[\tag{1.78}$$

$$\frac{d\boldsymbol{P}_{t|k}}{dt} = \boldsymbol{A}\boldsymbol{P}_{t|k} + \boldsymbol{P}_{t|k}\boldsymbol{A}^T + \boldsymbol{\sigma}\boldsymbol{\sigma}^T \ , \ t \in [t_k, t_{k+1}[\tag{1.79}$$

where the following shorthand notation has been applied[4]:

$$\boldsymbol{A} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}=\boldsymbol{u}_k, t=t_k, \boldsymbol{\theta}} , \ \boldsymbol{C} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_t}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}=\boldsymbol{u}_k, t=t_k, \boldsymbol{\theta}} \tag{1.80}$$

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \ , \ \boldsymbol{S} = \boldsymbol{S}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta})$$

[4]Within **CTSM** the code needed to evaluate the Jacobians is generated through analytical manipulation using a method based on the algorithms of Speelpenning (1980).

Initial conditions for the extended Kalman filter are $\hat{\boldsymbol{x}}_{t|t_0} = \boldsymbol{x}_0$ and $\boldsymbol{P}_{t|t_0} = \boldsymbol{P}_0$, which may either be pre-specified or estimated along with the parameters as a part of the overall problem (see Section 1.1.3.4). Being a linear filter, the extended Kalman filter is sensitive to nonlinear effects, and the approximate solution obtained by solving (1.78) and (1.79) may be too crude (Jazwinski, 1970). Moreover, the assumption of Gaussian conditional densities is only likely to hold for small sample times. To provide a better approximation, the time interval $[t_k, t_{k+1}[$ is therefore subsampled, i.e. $[t_k, \ldots, t_j, \ldots, t_{k+1}[$, and the equations are linearized at each subsampling instant. This also means that direct numerical solution of (1.78) and (1.79) can be avoided by applying the analytical solutions to the corresponding linearized propagation equations:

$$\frac{d\hat{\boldsymbol{x}}_{t|j}}{dt} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{j|j-1}, \boldsymbol{u}_j, t_j, \boldsymbol{\theta}) + \boldsymbol{A}(\hat{\boldsymbol{x}}_t - \hat{\boldsymbol{x}}_j) + \boldsymbol{B}(\boldsymbol{u}_t - \boldsymbol{u}_j),\ t \in [t_j, t_{j+1}[\quad (1.81)$$

$$\frac{d\boldsymbol{P}_{t|j}}{dt} = \boldsymbol{A}\boldsymbol{P}_{t|j} + \boldsymbol{P}_{t|j}\boldsymbol{A}^T + \boldsymbol{\sigma}\boldsymbol{\sigma}^T,\ t \in [t_j, t_{j+1}[\quad (1.82)$$

where the following shorthand notation has been applied[5]:

$$\boldsymbol{A} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{j|j-1}, \boldsymbol{u}=\boldsymbol{u}_j, t=t_j, \boldsymbol{\theta}},\ \boldsymbol{B} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{j|j-1}, \boldsymbol{u}=\boldsymbol{u}_j, t=t_j, \boldsymbol{\theta}} \quad (1.83)$$
$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{u}_j, t_j, \boldsymbol{\theta}),\ \boldsymbol{S} = \boldsymbol{S}(\boldsymbol{u}_j, t_j, \boldsymbol{\theta})$$

The solution to (1.82) is equivalent to the solution to (1.35), i.e.:

$$\boldsymbol{P}_{j+1|j} = \boldsymbol{\Phi}_s \boldsymbol{P}_{j|j} \boldsymbol{\Phi}_s^T + \int_0^{\tau_s} e^{\boldsymbol{A}s} \boldsymbol{\sigma}\boldsymbol{\sigma}^T \left(e^{\boldsymbol{A}s}\right)^T ds \quad (1.84)$$

where $\tau_s = t_{j+1} - t_j$ and $\boldsymbol{\Phi}_s = e^{\boldsymbol{A}\tau_s}$. The solution to (1.81) is not as easy to find, especially if $\boldsymbol{A}$ is singular. Nevertheless, by simplifying the notation, i.e.:

$$\frac{d\hat{\boldsymbol{x}}_t}{dt} = \boldsymbol{f} + \boldsymbol{A}(\hat{\boldsymbol{x}}_t - \hat{\boldsymbol{x}}_j) + \boldsymbol{B}(\boldsymbol{u}_t - \boldsymbol{u}_j)\ ,\ t \in [t_j, t_{j+1}[\quad (1.85)$$

and introducing:

$$\boldsymbol{\alpha} = \frac{\boldsymbol{u}_{j+1} - \boldsymbol{u}_j}{t_{j+1} - t_j} \quad (1.86)$$

to allow assumption of either *zero order hold* ($\boldsymbol{\alpha} = \boldsymbol{0}$) or *first order hold* ($\boldsymbol{\alpha} \neq \boldsymbol{0}$) on the inputs between sampling instants, i.e.:

$$\frac{d\hat{\boldsymbol{x}}_t}{dt} = \boldsymbol{f} + \boldsymbol{A}(\hat{\boldsymbol{x}}_t - \hat{\boldsymbol{x}}_j) + \boldsymbol{B}(\boldsymbol{\alpha}(t - t_j) + \boldsymbol{u}_j - \boldsymbol{u}_j)\ ,\ t \in [t_j, t_{j+1}[\quad (1.87)$$

[5]Within **CTSM** the code needed to evaluate the Jacobians is generated through analytical manipulation using a method based on the algorithms of Speelpenning (1980).

and by introducing the singular value decomposition (SVD) of $\boldsymbol{A}$, i.e. $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, a solvable equation can be obtained as follows:

$$\begin{aligned}
\frac{d\hat{\boldsymbol{x}}_t}{dt} &= \boldsymbol{f} + \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T(\hat{\boldsymbol{x}}_t - \hat{\boldsymbol{x}}_j) + \boldsymbol{B}\boldsymbol{\alpha}(t - t_j) \\
\boldsymbol{U}^T\frac{d\hat{\boldsymbol{x}}_t}{dt} &= \boldsymbol{U}^T\boldsymbol{f} + \boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{U}\boldsymbol{U}^T(\hat{\boldsymbol{x}}_t - \hat{\boldsymbol{x}}_j) + \boldsymbol{U}^T\boldsymbol{B}\boldsymbol{\alpha}(t - t_j) \\
\frac{d\boldsymbol{z}_t}{dt} &= \boldsymbol{U}^T\boldsymbol{f} + \boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{U}(\boldsymbol{z}_t - \boldsymbol{z}_j) + \boldsymbol{U}^T\boldsymbol{B}\boldsymbol{\alpha}(t - t_j) \\
\frac{d\boldsymbol{z}_t}{dt} &= \tilde{\boldsymbol{f}} + \tilde{\boldsymbol{A}}(\boldsymbol{z}_t - \boldsymbol{z}_j) + \tilde{\boldsymbol{B}}\boldsymbol{\alpha}(t - t_j)\ ,\ t \in [t_j, t_{j+1}[
\end{aligned} \tag{1.88}$$

where the transformation $\boldsymbol{z}_t = \boldsymbol{U}^T\hat{\boldsymbol{x}}_t$ has been introduced along with the vector $\tilde{\boldsymbol{f}} = \boldsymbol{U}^T\boldsymbol{f}$ and the matrices $\tilde{\boldsymbol{A}} = \boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{U} = \boldsymbol{U}^T\boldsymbol{A}\boldsymbol{U}$ and $\tilde{\boldsymbol{B}} = \boldsymbol{U}^T\boldsymbol{B}$. Now, if $\boldsymbol{A}$ is singular, the matrix $\tilde{\boldsymbol{A}}$ has a special structure:

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \tilde{\boldsymbol{A}}_1 & \tilde{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{1.89}$$

which makes it possible to split up the previous result in two distinct equations:

$$\begin{aligned}
\frac{d\boldsymbol{z}_t^1}{dt} &= \tilde{\boldsymbol{f}}_1 + \tilde{\boldsymbol{A}}_1(\boldsymbol{z}_t^1 - \boldsymbol{z}_j^1) + \tilde{\boldsymbol{A}}_2(\boldsymbol{z}_t^2 - \boldsymbol{z}_j^2) + \tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}(t - t_j),\ t \in [t_j, t_{j+1}[\\
\frac{d\boldsymbol{z}_t^2}{dt} &= \tilde{\boldsymbol{f}}_2 + \tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}(t - t_j),\ t \in [t_j, t_{j+1}[
\end{aligned} \tag{1.90}$$

which can then be solved one at a time for the transformed variables. Solving the equation for $\boldsymbol{z}_t^2$, with the initial condition $\boldsymbol{z}_{t=t_j}^2 = \boldsymbol{z}_j^2$, yields:

$$\boldsymbol{z}_t^2 = \boldsymbol{z}_j^2 + \tilde{\boldsymbol{f}}_2(t - t_j) + \frac{1}{2}\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}(t - t_j)^2\ ,\ t \in [t_j, t_{j+1}[\tag{1.91}$$

which can then be substituted into the equation for $\boldsymbol{z}_t^1$ to yield:

$$\begin{aligned}
\frac{d\boldsymbol{z}_t^1}{dt} = \tilde{\boldsymbol{f}}_1 &+ \tilde{\boldsymbol{A}}_1(\boldsymbol{z}_t^1 - \boldsymbol{z}_j^1) + \tilde{\boldsymbol{A}}_2\left(\tilde{\boldsymbol{f}}_2(t - t_j) + \frac{1}{2}\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}(t - t_j)^2\right) \\
&+ \tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}(t - t_j)\ ,\ t \in [t_j, t_{j+1}[
\end{aligned} \tag{1.92}$$

Introducing, for ease of notation, the constants:

$$\boldsymbol{E} = \frac{1}{2}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}\ ,\ \boldsymbol{F} = \tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2 + \tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}\ ,\ \boldsymbol{G} = \tilde{\boldsymbol{f}}_1 - \tilde{\boldsymbol{A}}_1\boldsymbol{z}_j^1 \tag{1.93}$$

and the standard form of a linear inhomogenous ordinary differential equation:

$$\frac{d\boldsymbol{z}_t^1}{dt} - \tilde{\boldsymbol{A}}_1\boldsymbol{z}_t^1 = \boldsymbol{E}(t - t_j)^2 + \boldsymbol{F}(t - t_j) + \boldsymbol{G}\ ,\ t \in [t_j, t_{j+1}[\tag{1.94}$$

gives the solution:

$$\boldsymbol{z}_t^1 = e^{\tilde{\boldsymbol{A}}_1 t}\left(\int e^{-\tilde{\boldsymbol{A}}_1 t}\left(\boldsymbol{E}(t-t_j)^2+\boldsymbol{F}(t-t_j)+\boldsymbol{G}\right)dt+\boldsymbol{c}\right),\ t\in[t_j,t_{j+1}[\tag{1.95}$$

which can be rearranged to:

$$\begin{aligned}\boldsymbol{z}_t^1 = &-\tilde{\boldsymbol{A}}_1^{-1}\left(\boldsymbol{I}(t-t_j)^2+2\tilde{\boldsymbol{A}}_1^{-1}(t-t_j)+2\tilde{\boldsymbol{A}}_1^{-2}\right)\boldsymbol{E}\\ &-\tilde{\boldsymbol{A}}_1^{-1}\left(\left(\boldsymbol{I}(t-t_j)+\tilde{\boldsymbol{A}}_1^{-1}\right)\boldsymbol{F}+\boldsymbol{G}\right)+e^{\tilde{\boldsymbol{A}}_1 t}\boldsymbol{c}\ ,\ t\in[t_j,t_{j+1}[\end{aligned} \tag{1.96}$$

Using the initial condition $\boldsymbol{z}_{t=t_j}^1=\boldsymbol{z}_j^1$ to determine the constant $\boldsymbol{c}$, i.e.:

$$\begin{aligned}\boldsymbol{z}_j^1 &= -\tilde{\boldsymbol{A}}_1^{-1}\left(2\tilde{\boldsymbol{A}}_1^{-2}\boldsymbol{E}+\tilde{\boldsymbol{A}}_1^{-1}\boldsymbol{F}+\boldsymbol{G}\right)+e^{\tilde{\boldsymbol{A}}_1 t_j}\boldsymbol{c}\\ \boldsymbol{c} &= e^{-\tilde{\boldsymbol{A}}_1 t_j}\left(\tilde{\boldsymbol{A}}_1^{-1}\left(2\tilde{\boldsymbol{A}}_1^{-2}\boldsymbol{E}+\tilde{\boldsymbol{A}}_1^{-1}\boldsymbol{F}+\boldsymbol{G}\right)+\boldsymbol{z}_j^1\right)\end{aligned} \tag{1.97}$$

the solution can be rearranged to:

$$\begin{aligned}\boldsymbol{z}_t^1 = &-\tilde{\boldsymbol{A}}_1^{-1}\left(\boldsymbol{I}(t-t_j)^2+2\tilde{\boldsymbol{A}}_1^{-1}(t-t_j)+2\tilde{\boldsymbol{A}}_1^{-2}\right)\boldsymbol{E}\\ &-\tilde{\boldsymbol{A}}_1^{-1}\left(\left(\boldsymbol{I}(t-t_j)+\tilde{\boldsymbol{A}}_1^{-1}\right)\boldsymbol{F}+\boldsymbol{G}\right)\\ &+e^{\tilde{\boldsymbol{A}}_1(t-t_j)}\left(\tilde{\boldsymbol{A}}_1^{-1}\left(2\tilde{\boldsymbol{A}}_1^{-2}\boldsymbol{E}+\tilde{\boldsymbol{A}}_1^{-1}\boldsymbol{F}+\boldsymbol{G}\right)+\boldsymbol{z}_j^1\right),\ t\in[t_j,t_{j+1}[\end{aligned} \tag{1.98}$$

which finally yields:

$$\begin{aligned}\boldsymbol{z}_{j+1}^1 &= -\tilde{\boldsymbol{A}}_1^{-1}\left(\left(\boldsymbol{I}\tau_s^2+2\tilde{\boldsymbol{A}}_1^{-1}\tau_s+2\tilde{\boldsymbol{A}}_1^{-2}\right)\boldsymbol{E}+\left(\boldsymbol{I}\tau_s+\tilde{\boldsymbol{A}}_1^{-1}\right)\boldsymbol{F}+\boldsymbol{G}\right)\\ &\quad+\tilde{\boldsymbol{\Phi}}_s^1\left(\tilde{\boldsymbol{A}}_1^{-1}\left(2\tilde{\boldsymbol{A}}_1^{-2}\boldsymbol{E}+\tilde{\boldsymbol{A}}_1^{-1}\boldsymbol{F}+\boldsymbol{G}\right)+\boldsymbol{z}_j^1\right)\\ &= -\tilde{\boldsymbol{A}}_1^{-1}\left(\left(\boldsymbol{I}\tau_s^2+2\tilde{\boldsymbol{A}}_1^{-1}\tau_s+2\tilde{\boldsymbol{A}}_1^{-2}\right)\frac{1}{2}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}\right)\\ &\quad-\tilde{\boldsymbol{A}}_1^{-1}\left(\left(\boldsymbol{I}\tau_s+\tilde{\boldsymbol{A}}_1^{-1}\right)\left(\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2+\tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}\right)+\left(\tilde{\boldsymbol{f}}_1-\tilde{\boldsymbol{A}}_1\boldsymbol{z}_j^1\right)\right)\\ &\quad+\tilde{\boldsymbol{\Phi}}_s^1\left(\tilde{\boldsymbol{A}}_1^{-1}\left(2\tilde{\boldsymbol{A}}_1^{-2}\frac{1}{2}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}+\tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2+\tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}\right)\right)\right)\\ &\quad+\tilde{\boldsymbol{\Phi}}_s^1\left(\tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{f}}_1-\tilde{\boldsymbol{A}}_1\boldsymbol{z}_j^1\right)+\boldsymbol{z}_j^1\right)\\ &= \boldsymbol{z}_j^1-\tilde{\boldsymbol{A}}_1^{-1}\left(\frac{1}{2}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}\tau_s^2+\left(\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}+\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2+\tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}\right)\tau_s\right)\\ &\quad+\left(\tilde{\boldsymbol{\Phi}}_s^1-\boldsymbol{I}\right)\tilde{\boldsymbol{A}}_1^{-2}\left(\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}+\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2+\tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}+\tilde{\boldsymbol{A}}_1\tilde{\boldsymbol{f}}_1\right)\\ &= \boldsymbol{z}_j^1-\frac{1}{2}\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}\tau_s^2-\tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}+\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2+\tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}\right)\tau_s\\ &\quad+\tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{\Phi}}_s^1-\boldsymbol{I}\right)\left(\tilde{\boldsymbol{A}}_1^{-1}\left(\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}+\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2+\tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}\right)+\tilde{\boldsymbol{f}}_1\right)\end{aligned} \tag{1.99}$$

and:

$$\boldsymbol{z}_{j+1}^2 = \boldsymbol{z}_j^2 + \tilde{\boldsymbol{f}}_2\tau_s + \frac{1}{2}\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}\tau_s^2 \tag{1.100}$$

where $\tilde{\boldsymbol{\Phi}}_s^{\,1}$ is the upper left part of the matrix:

$$\tilde{\boldsymbol{\Phi}}_s = \boldsymbol{U}^T\boldsymbol{\Phi}_s\boldsymbol{U} = \begin{bmatrix} \tilde{\boldsymbol{\Phi}}_s^{\,1} & \tilde{\boldsymbol{\Phi}}_s^{\,2} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{1.101}$$

and where the desired solution in terms of the original variables $\hat{\boldsymbol{x}}_{j+1|j}$ can be found by applying the reverse transformation $\hat{\boldsymbol{x}}_t = \boldsymbol{U}\boldsymbol{z}_t$.

Depending on the specific singularity of $\boldsymbol{A}$ (see Section 1.1.3.3 for details on how this is determined in **CTSM**) and the particular nature of the inputs, several different cases are possible as shown in the following.

General case: Singular $\boldsymbol{A}$, first order hold on inputs

In the general case, the extended Kalman filter solution is given as follows:

$$\begin{aligned} \boldsymbol{z}_{j+1|j}^1 = \boldsymbol{z}_{j|j}^1 &- \frac{1}{2}\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}\tau_s^2 \\ &- \tilde{\boldsymbol{A}}_1^{-1}\Big(\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha} + \tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2 + \tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}\Big)\tau_s \\ &+ \tilde{\boldsymbol{A}}_1^{-1}\Big(\tilde{\boldsymbol{\Phi}}_s^{\,1} - \boldsymbol{I}\Big)\Big(\tilde{\boldsymbol{A}}_1^{-1}\Big(\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha} + \tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2 + \tilde{\boldsymbol{B}}_1\boldsymbol{\alpha}\Big) + \tilde{\boldsymbol{f}}_1\Big) \end{aligned} \tag{1.102}$$

and:

$$\boldsymbol{z}_{j+1|j}^2 = \boldsymbol{z}_{j|j}^2 + \tilde{\boldsymbol{f}}_2\tau_s + \frac{1}{2}\tilde{\boldsymbol{B}}_2\boldsymbol{\alpha}\tau_s^2 \tag{1.103}$$

where the desired solution in terms of the original variables $\hat{\boldsymbol{x}}_{j+1|j}$ can be found by applying the reverse transformation $\hat{\boldsymbol{x}}_t = \boldsymbol{U}\boldsymbol{z}_t$.

Special case no. 1: Singular $\boldsymbol{A}$, zero order hold on inputs

The solution to this special case can be obtained by setting $\boldsymbol{\alpha} = 0$, which yields:

$$\boldsymbol{z}_{j+1|j}^1 = \boldsymbol{z}_{j|j}^1 - \tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2\tau_s + \tilde{\boldsymbol{A}}_1^{-1}\Big(\tilde{\boldsymbol{\Phi}}_s^{\,1} - \boldsymbol{I}\Big)\Big(\tilde{\boldsymbol{A}}_1^{-1}\tilde{\boldsymbol{A}}_2\tilde{\boldsymbol{f}}_2 + \tilde{\boldsymbol{f}}_1\Big) \tag{1.104}$$

and:

$$\boldsymbol{z}_{j+1|j}^2 = \boldsymbol{z}_{j|j}^2 + \tilde{\boldsymbol{f}}_2\tau_s \tag{1.105}$$

where the desired solution in terms of the original variables $\hat{\boldsymbol{x}}_{j+1|j}$ can be found by applying the reverse transformation $\hat{\boldsymbol{x}}_t = \boldsymbol{U}\boldsymbol{z}_t$.

Special case no. 2: Nonsingular $\boldsymbol{A}$, first order hold on inputs

The solution to this special case can be obtained by removing the SVD dependent parts, i.e. by replacing $\boldsymbol{z}_t^1$, $\tilde{\boldsymbol{A}}_1$, $\tilde{\boldsymbol{B}}_1$ and $\tilde{\boldsymbol{f}}_1$ with $\boldsymbol{x}_t$, $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{f}$ respectively, and by setting $\boldsymbol{z}_t^2$, $\tilde{\boldsymbol{A}}_2$, $\tilde{\boldsymbol{B}}_2$ and $\tilde{\boldsymbol{f}}_2$ to zero, which yields:

$$\hat{\boldsymbol{x}}_{j+1|j} = \hat{\boldsymbol{x}}_{j|j} - \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{\alpha}\tau_s + \boldsymbol{A}^{-1}\left(\boldsymbol{\Phi}_s - \boldsymbol{I}\right)\left(\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{\alpha} + \boldsymbol{f}\right) \tag{1.106}$$

Special case no. 3: Nonsingular $\boldsymbol{A}$, zero order hold on inputs

The solution to this special case can be obtained by removing the SVD dependent parts, i.e. by replacing $\boldsymbol{z}_t^1$, $\tilde{\boldsymbol{A}}_1$, $\tilde{\boldsymbol{B}}_1$ and $\tilde{\boldsymbol{f}}_1$ with $\boldsymbol{x}_t$, $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{f}$ respectively, and by setting $\boldsymbol{z}_t^2$, $\tilde{\boldsymbol{A}}_2$, $\tilde{\boldsymbol{B}}_2$ and $\tilde{\boldsymbol{f}}_2$ to zero and $\boldsymbol{\alpha} = 0$, which yields:

$$\hat{\boldsymbol{x}}_{j+1|j} = \hat{\boldsymbol{x}}_{j|j} + \boldsymbol{A}^{-1}\left(\boldsymbol{\Phi}_s - \boldsymbol{I}\right)\boldsymbol{f} \tag{1.107}$$

Special case no. 4: Identically zero $\boldsymbol{A}$, first order hold on inputs

The solution to this special case can be obtained by setting $\boldsymbol{A}$ to zero and solving the original linearized state propagation equation, which yields:

$$\hat{\boldsymbol{x}}_{j+1|j} = \hat{\boldsymbol{x}}_{j|j} + \boldsymbol{f}\tau_s + \frac{1}{2}\boldsymbol{B}\boldsymbol{\alpha}\tau_s^2 \tag{1.108}$$

Special case no. 5: Identically zero $\boldsymbol{A}$, zero order hold on inputs

The solution to this special case can be obtained by setting $\boldsymbol{A}$ to zero and $\boldsymbol{\alpha} = 0$ and solving the original linearized state propagation equation, which yields:

$$\hat{\boldsymbol{x}}_{j+1|j} = \hat{\boldsymbol{x}}_{j|j} + \boldsymbol{f}\tau_s \tag{1.109}$$

Numerical ODE solution as an alternative

The subsampling-based solution framework described above provides a better approximation to the true state propagation solution than direct numerical solution of (1.78) and (1.79), because it more accurately reflects the true time-varying nature of the matrices $\boldsymbol{A}$ and $\boldsymbol{\sigma}$ in (1.79) by allowing these to be re-evaluated at each subsampling instant. To provide an even better approximation and to handle stiff systems, which is not always possible with the subsampling-based solution framework, an option has been included in **CTSM** for applying numerical ODE solution to solve (1.78) and (1.79) simultaneously[6], which ensures intelligent re-evaluation of $\boldsymbol{A}$ and $\boldsymbol{\sigma}$ in (1.79).

[6]The specific implementation is based on the algorithms of Hindmarsh (1983), and to be able to use this method to solve (1.78) and (1.79) simultaneously, the n-vector differential equation in (1.78) has been augmented with an $n \times (n+1)/2$-vector differential equation corresponding to the symmetric $n \times n$-matrix differential equation in (1.79).

Iterated extended Kalman filtering

The sensitivity of the extended Kalman filter to nonlinear effects not only means that the approximation to the true state propagation solution provided by the solution to the state prediction equations (1.78) and (1.79) may be too crude. The presence of such effects in the output prediction equations (1.72) and (1.73) may also influence the performance of the filter. An option has therefore been included in **CTSM** for applying the *iterated extended Kalman filter* (Jazwinski, 1970), which is an iterative version of the extended Kalman filter that consists of the modified output prediction equations:

$$\hat{\boldsymbol{y}}^i_{k|k-1} = \boldsymbol{h}(\eta_i, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \tag{1.110}$$

$$\boldsymbol{R}^i_{k|k-1} = \boldsymbol{C}_i \boldsymbol{P}_{k|k-1} \boldsymbol{C}_i^T + \boldsymbol{S} \tag{1.111}$$

the modified innovation equation:

$$\boldsymbol{\epsilon}^i_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}^i_{k|k-1} \tag{1.112}$$

the modified Kalman gain equation:

$$\boldsymbol{K}^i_k = \boldsymbol{P}_{k|k-1} \boldsymbol{C}_i^T (\boldsymbol{R}^i_{k|k-1})^{-1} \tag{1.113}$$

and the modified updating equations:

$$\eta_{i+1} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{\epsilon}^i_k - \boldsymbol{C}_i(\hat{\boldsymbol{x}}_{k|k-1} - \eta_i)) \tag{1.114}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}^i_k \boldsymbol{R}^i_{k|k-1} (\boldsymbol{K}^i_k)^T \tag{1.115}$$

where:

$$\boldsymbol{C}_i = \left. \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_t} \right|_{\boldsymbol{x}=\eta_i, \boldsymbol{u}=\boldsymbol{u}_k, t=t_k, \boldsymbol{\theta}} \tag{1.116}$$

and $\eta_1 = \hat{\boldsymbol{x}}_{k|k-1}$. The above equations are iterated for $i = 1, \ldots, M$, where M is the maximum number of iterations, or until there is no significant difference between consecutive iterates, whereupon $\hat{\boldsymbol{x}}_{k|k} = \eta_M$ is assigned. This way, the influence of nonlinear effects in (1.72) and (1.73) can be reduced.

1.1.3.3 Determination of singularity

Computing the singular value decomposition (SVD) of a matrix is a computationally expensive task, which should be avoided if possible. Within **CTSM** the determination of whether or not the $\boldsymbol{A}$ matrix is singular and thus whether or not the SVD should be applied, therefore is not based on the SVD itself, but on an estimate of the reciprocal condition number, i.e.:

$$\hat{\kappa}^{-1} = \frac{1}{|\boldsymbol{A}||\boldsymbol{A}^{-1}|} \tag{1.117}$$

where $|\boldsymbol{A}|$ is the 1-norm of the $\boldsymbol{A}$ matrix and $|\boldsymbol{A}^{-1}|$ is an estimate of the 1-norm of $\boldsymbol{A}^{-1}$. This quantity can be computed much faster than the SVD, and only if its value is below a certain threshold (e.g. 1e-12), the SVD is applied.

1.1.3.4 Initial states and covariances

In order for the (extended) Kalman filter to work, the initial states $\boldsymbol{x}_0$ and their covariance matrix $\boldsymbol{P}_0$ must be specified. Within **CTSM** the initial states may either be pre-specified or estimated by the program along with the parameters, whereas the initial covariance matrix is calculated in the following way:

$$\boldsymbol{P}_0 = P_s \int_{t_0}^{t_1} e^{\boldsymbol{A}s} \boldsymbol{\sigma}\boldsymbol{\sigma}^T (e^{\boldsymbol{A}s})^T ds \tag{1.118}$$

i.e. as the integral of the Wiener process and the dynamics of the system over the first sample, which is then scaled by a pre-specified scaling factor $P_s \geq 1$.

1.1.3.5 Factorization of covariance matrices

The (extended) Kalman filter may be numerically unstable in certain situations. The problem arises when some of the covariance matrices, which are known from theory to be symmetric and positive definite, become non-positive definite because of rounding errors. Consequently, careful handling of the covariance equations is needed to stabilize the (extended) Kalman filter. Within **CTSM**, all covariance matrices are therefore replaced with their square root free Cholesky decompositions (Fletcher and Powell, 1974), i.e.:

$$\boldsymbol{P} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T \tag{1.119}$$

where $\boldsymbol{P}$ is the covariance matrix, $\boldsymbol{L}$ is a unit lower triangular matrix and $\boldsymbol{D}$ is a diagonal matrix with $d_{ii} > 0$, $\forall i$. Using factorized covariance matrices, all of the covariance equations of the (extended) Kalman filter can be handled by means of the following equation for updating a factorized matrix:

$$\tilde{\boldsymbol{P}} = \boldsymbol{P} + \boldsymbol{G}\boldsymbol{D}_g\boldsymbol{G}^T \tag{1.120}$$

where $\tilde{\boldsymbol{P}}$ is known from theory to be both symmetric and positive definite and $\boldsymbol{P}$ is given by (1.119), and where $\boldsymbol{D}_g$ is a diagonal matrix and $\boldsymbol{G}$ is a full matrix. Solving this equation amounts to finding a unit lower triangular matrix $\tilde{\boldsymbol{L}}$ and a diagonal matrix $\tilde{\boldsymbol{D}}$ with $\tilde{d}_{ii} > 0$, $\forall i$, such that:

$$\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{D}}\tilde{\boldsymbol{L}}^T \tag{1.121}$$

and for this purpose a number of different methods are available, e.g. the method described by Fletcher and Powell (1974), which is based on the modified Givens transformation, and the method described by Thornton and Bierman (1980), which is based on the modified weighted Gram-Schmidt orthogonalization. Within **CTSM** the specific implementation of the (extended) Kalman filter is based on the latter, and this implementation has been proven to have a high grade of accuracy as well as stability (Bierman, 1977).

Using factorized covariance matrices also facilitates easy computation of those parts of the objective function (1.26) that depend on determinants of covariance matrices. This is due to the following identities:

$$\det(\boldsymbol{P}) = \det(\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T) = \det(\boldsymbol{D}) = \prod_i d_{ii} \tag{1.122}$$

1.1.4 Data issues

Raw data sequences are often difficult to use for identification and parameter estimation purposes, e.g. if irregular sampling has been applied, if there are occasional outliers or if some of the observations are missing. **CTSM** also provides features to deal with these issues, and this makes the program flexible with respect to the types of data that can be used for the estimation.

1.1.4.1 Irregular sampling.

The fact that the system equation of a continuous-discrete stochastic state space model is formulated in continuous time makes it easy to deal with irregular sampling, because the corresponding state prediction equations of the (extended) Kalman filter can be solved over time intervals of varying length.

1.1.4.2 Occasional outliers

The objective function (1.26) of the general formulation (1.27) is quadratic in the innovations $\boldsymbol{\epsilon}_k^i$, and this means that the corresponding parameter estimates are heavily influenced by occasional outliers in the data sets used for the estimation. To deal with this problem, a robust estimation method is applied, where the objective function is modified by replacing the quadratic term:

$$\nu_k^i = (\boldsymbol{\epsilon}_k^i)^T (\boldsymbol{R}_{k|k-1}^i)^{-1} \boldsymbol{\epsilon}_k^i \tag{1.123}$$

with a threshold function $\varphi(\nu_k^i)$, which returns the argument for small values of ν_k^i, but is a linear function of $\boldsymbol{\epsilon}_k^i$ for large values of ν_k^i, i.e.:

$$\varphi(\nu_k^i) = \begin{cases} \nu_k^i & , \ \nu_k^i < c^2 \\ c(2\sqrt{\nu_k^i} - c) & , \ \nu_k^i \geq c^2 \end{cases} \tag{1.124}$$

where $c > 0$ is a constant. The derivative of this function with respect to $\boldsymbol{\epsilon}_k^i$ is known as *Huber's ψ-function* (Huber, 1981) and belongs to a class of functions called influence functions, because they measure the influence of $\boldsymbol{\epsilon}_k^i$ on the objective function. Several such functions are available, but Huber's ψ-function has been found to be most appropriate in terms of providing robustness against outliers without rendering optimisation of the objective function infeasible.

1.1.4.3 Missing observations.

The algorithms of the parameter estimation methods described above also make it easy to handle missing observations, i.e. to account for missing values in the output vector $\boldsymbol{y}_k^i$, for some i and some k, when calculating the terms:

$$\frac{1}{2}\sum_{i=1}^{S}\sum_{k=1}^{N_i}\left(\ln(\det(\boldsymbol{R}_{k|k-1}^i)) + (\boldsymbol{\epsilon}_k^i)^T(\boldsymbol{R}_{k|k-1}^i)^{-1}\boldsymbol{\epsilon}_k^i\right) \tag{1.125}$$

and:

$$\frac{1}{2}\left(\left(\sum_{i=1}^{S}\sum_{k=1}^{N_i} l\right) + p\right)\ln(2\pi) \tag{1.126}$$

in (1.26). To illustrate this, the case of extended Kalman filtering for NL models is considered, but similar arguments apply in the case of Kalman filtering for LTI and LTV models. The usual way to account for missing or non-informative values in the extended Kalman filter is to formally set the corresponding elements of the measurement error covariance matrix $\boldsymbol{S}$ in (1.73) to infinity, which in turn gives zeroes in the corresponding elements of the inverted output covariance matrix $(\boldsymbol{R}_{k|k-1})^{-1}$ and the Kalman gain matrix $\boldsymbol{K}_k$, meaning that no updating will take place in (1.76) and (1.77) corresponding to the missing values. This approach cannot be used when calculating (1.125) and (1.126), however, because a solution is needed which modifies both $\boldsymbol{\epsilon}_k^i$, $\boldsymbol{R}_{k|k-1}^i$ and l to reflect that the effective dimension of $\boldsymbol{y}_k^i$ is reduced. This is accomplished by replacing (1.2) with the alternative measurement equation:

$$\overline{\boldsymbol{y}}_k = \boldsymbol{E}\left(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) + \boldsymbol{e}_k\right) \tag{1.127}$$

where $\boldsymbol{E}$ is an appropriate permutation matrix, which can be constructed from a unit matrix by eliminating the rows that correspond to the missing values in $\boldsymbol{y}_k$. If, for example, $\boldsymbol{y}_k$ has three elements, and the one in the middle is missing, the appropriate permutation matrix is given as follows:

$$\boldsymbol{E} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.128}$$

Equivalently, the equations of the extended Kalman filter are replaced with the following alternative output prediction equations:

$$\hat{\overline{\boldsymbol{y}}}_{k|k-1} = \boldsymbol{E}\boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \tag{1.129}$$

$$\overline{\boldsymbol{R}}_{k|k-1} = \boldsymbol{E}\boldsymbol{C}\boldsymbol{P}_{k|k-1}\boldsymbol{C}^T\boldsymbol{E}^T + \boldsymbol{E}\boldsymbol{S}\boldsymbol{E}^T \tag{1.130}$$

the alternative innovation equation:

$$\overline{\boldsymbol{\epsilon}}_k = \overline{\boldsymbol{y}}_k - \hat{\overline{\boldsymbol{y}}}_{k|k-1} \tag{1.131}$$

the alternative Kalman gain equation:

$$\overline{\boldsymbol{K}}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{C}^T\boldsymbol{E}^T\overline{\boldsymbol{R}}_{k|k-1}^{-1} \tag{1.132}$$

and the alternative updating equations:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \overline{\boldsymbol{K}}_k\overline{\boldsymbol{\epsilon}}_k \tag{1.133}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \overline{\boldsymbol{K}}_k\overline{\boldsymbol{R}}_{k|k-1}\overline{\boldsymbol{K}}_k^T \tag{1.134}$$

The state prediction equations remain the same, and the above replacements in turn provide the necessary modifications of (1.125) to:

$$\frac{1}{2}\sum_{i=1}^{S}\sum_{k=1}^{N_i}\left(\ln(\det(\overline{\boldsymbol{R}}_{k|k-1}^i)) + (\overline{\boldsymbol{\epsilon}}_k^i)^T(\overline{\boldsymbol{R}}_{k|k-1}^i)^{-1}\overline{\boldsymbol{\epsilon}}_k^i\right) \tag{1.135}$$

whereas modifying (1.126) amounts to a simple reduction of l for the particular values of i and k with the number of missing values in $\boldsymbol{y}_k^i$.

1.1.5 Optimisation issues

CTSM uses a *quasi-Newton* method based on the BFGS updating formula and a soft line search algorithm to solve the nonlinear optimisation problem (1.27). This method is similar to the one described by Dennis and Schnabel (1983), except for the fact that the gradient of the objective function is approximated by a set of finite difference derivatives. In analogy with ordinary Newton-Raphson methods for optimisation, quasi-Newton methods seek a minimum of a nonlinear objective function $\mathcal{F}(\boldsymbol{\theta})$: $\mathbb{R}^p \to \mathbb{R}$, i.e.:

$$\min_{\boldsymbol{\theta}} \mathcal{F}(\boldsymbol{\theta}) \tag{1.136}$$

where a minimum of $\mathcal{F}(\boldsymbol{\theta})$ is found when the gradient $\boldsymbol{g}(\boldsymbol{\theta}) = \frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ satisfies:

$$\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{0} \tag{1.137}$$

Both types of methods are based on the Taylor expansion of $\boldsymbol{g}(\boldsymbol{\theta})$ to first order:

$$\boldsymbol{g}(\boldsymbol{\theta}^i + \boldsymbol{\delta}) = \boldsymbol{g}(\boldsymbol{\theta}^i) + \frac{\partial \boldsymbol{g}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^i}\,\boldsymbol{\delta} + o(\boldsymbol{\delta}) \tag{1.138}$$

which by setting $\boldsymbol{g}(\boldsymbol{\theta}^i + \boldsymbol{\delta}) = \boldsymbol{0}$ and neglecting $o(\boldsymbol{\delta})$ can be rewritten as follows:

$$\boldsymbol{\delta}^i = -\boldsymbol{H}_i^{-1}\boldsymbol{g}(\boldsymbol{\theta}^i) \tag{1.139}$$

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + \boldsymbol{\delta}^i \tag{1.140}$$

i.e. as an iterative algorithm, and this algorithm can be shown to converge to a (possibly local) minimum. The Hessian $\boldsymbol{H}_i$ is defined as follows:

$$\boldsymbol{H}_i = \frac{\partial \boldsymbol{g}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^i} \tag{1.141}$$

but unfortunately neither the Hessian nor the gradient can be computed explicitly for the optimisation problem (1.27). As mentioned above, the gradient is therefore approximated by a set of finite difference derivatives, and a secant approximation based on the BFGS updating formula is applied for the Hessian. It is the use of a secant approximation to the Hessian that distinguishes quasi-Newton methods from ordinary Newton-Raphson methods.

1.1.5.1 Finite difference derivative approximations

Since the gradient $\boldsymbol{g}(\boldsymbol{\theta}^i)$ cannot be computed explicitly, it is approximated by a set of finite difference derivatives. Initially, i.e. as long as $||\boldsymbol{g}(\boldsymbol{\theta})||$ does not become too small during the iterations of the optimisation algorithm, *forward difference* approximations are used, i.e.:

$$g_j(\boldsymbol{\theta}^i) \approx \frac{\mathcal{F}(\boldsymbol{\theta}^i + \delta_j \boldsymbol{e}_j) - \mathcal{F}(\boldsymbol{\theta}^i)}{\delta_j} \ , \ j = 1, \ldots, p \tag{1.142}$$

where $g_j(\boldsymbol{\theta}^i)$ is the j'th component of $\boldsymbol{g}(\boldsymbol{\theta}^i)$ and $\boldsymbol{e}_j$ is the j'th basis vector. The error of this type of approximation is $o(\delta_j)$. Subsequently, i.e. when $||\boldsymbol{g}(\boldsymbol{\theta})||$ becomes small near a minimum of the objective function, *central difference* approximations are used instead, i.e.:

$$g_j(\boldsymbol{\theta}^i) \approx \frac{\mathcal{F}(\boldsymbol{\theta}^i + \delta_j \boldsymbol{e}_j) - \mathcal{F}(\boldsymbol{\theta}^i - \delta_j \boldsymbol{e}_j)}{2\delta_j} \ , \ j = 1, \ldots, p \tag{1.143}$$

because the error of this type of approximation is only $o(\delta_j^2)$. Unfortunately, central difference approximations require twice as much computation (twice the number of objective function evalutions) as forward difference approximations, so to save computation time forward difference approximations are used initially. The switch from forward differences to central differences is effectuated for $i > 2p$ if the line search algorithm fails to find a better value of $\boldsymbol{\theta}$.

The optimal choice of step length for forward difference approximations is:

$$\delta_j = \eta^{\frac{1}{2}} \theta_j \tag{1.144}$$

whereas for central difference approximations it is:

$$\delta_j = \eta^{\frac{1}{3}} \theta_j \tag{1.145}$$

where η is the relative error of calculating $\mathcal{F}(\boldsymbol{\theta})$ (Dennis and Schnabel, 1983).

1.1.5.2 The BFGS updating formula

Since the Hessian $\boldsymbol{H}_i$ cannot be computed explicitly, a secant approximation is applied. The most effective secant approximation $\boldsymbol{B}_i$ is obtained with the so-called BFGS updating formula (Dennis and Schnabel, 1983), i.e.:

$$\boldsymbol{B}_{i+1} = \boldsymbol{B}_i + \frac{\boldsymbol{y}_i \boldsymbol{y}_i^T}{\boldsymbol{y}_i^T \boldsymbol{s}_i} - \frac{\boldsymbol{B}_i \boldsymbol{s}_i \boldsymbol{s}_i^T \boldsymbol{B}_i}{\boldsymbol{s}_i^T \boldsymbol{B}_i \boldsymbol{s}_i} \tag{1.146}$$

where $\boldsymbol{y}_i = \boldsymbol{g}(\boldsymbol{\theta}_{i+1}) - \boldsymbol{g}(\boldsymbol{\theta}_i)$ and $\boldsymbol{s}_i = \boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i$. Necessary and sufficient conditions for $\boldsymbol{B}_{i+1}$ to be positive definite is that $\boldsymbol{B}_i$ is positive definite and that:

$$\boldsymbol{y}_i^T \boldsymbol{s}_i > 0 \tag{1.147}$$

This last demand is automatically met by the line search algorithm. Furthermore, since the Hessian is symmetric and positive definite, it can also be written in terms of its square root free Cholesky factors, i.e.:

$$\boldsymbol{B}_i = \boldsymbol{L}_i \boldsymbol{D}_i \boldsymbol{L}_i^T \tag{1.148}$$

where $\boldsymbol{L}_i$ is a unit lower triangular matrix and $\boldsymbol{D}_i$ is a diagonal matrix with $d_{jj}^i > 0$, $\forall j$, so, instead of solving (1.146) directly, $\boldsymbol{B}_{i+1}$ can be found by updating the Cholesky factorization of $\boldsymbol{B}_i$ as shown in Section 1.1.3.5.

1.1.5.3 The soft line search algorithm

With $\boldsymbol{\delta}^i$ being the secant direction from (1.139) (using $\boldsymbol{H}_i = \boldsymbol{B}_i$ obtained from (1.146)), the idea of the soft line search algorithm is to replace (1.140) with:

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + \lambda_i \boldsymbol{\delta}^i \tag{1.149}$$

and choose a value of $\lambda_i > 0$ that ensures that the next iterate decreases $\mathcal{F}(\boldsymbol{\theta})$ and that (1.147) is satisfied. Often $\lambda_i = 1$ will satisfy these demands and (1.149) reduces to (1.140). The soft line search algorithm is globally convergent if each step satisfies two simple conditions. The first condition is that the decrease in $\mathcal{F}(\boldsymbol{\theta})$ is sufficient compared to the length of the step $\boldsymbol{s}_i = \lambda_i \boldsymbol{\delta}^i$, i.e.:

$$\mathcal{F}(\boldsymbol{\theta}^{i+1}) < \mathcal{F}(\boldsymbol{\theta}^i) + \alpha \boldsymbol{g}(\boldsymbol{\theta}^i)^T \boldsymbol{s}_i \tag{1.150}$$

where $\alpha \in]0, 1[$. The second condition is that the step is not too short, i.e.:

$$\boldsymbol{g}(\boldsymbol{\theta}^{i+1})^T \boldsymbol{s}_i \geq \beta \boldsymbol{g}(\boldsymbol{\theta}^i)^T \boldsymbol{s}_i \tag{1.151}$$

where $\beta \in]\alpha, 1[$. This last expression and $\boldsymbol{g}(\boldsymbol{\theta}^i)^T \boldsymbol{s}_i < 0$ imply that:

$$\boldsymbol{y}_i^T \boldsymbol{s}_i = \left(\boldsymbol{g}(\boldsymbol{\theta}^{i+1}) - \boldsymbol{g}(\boldsymbol{\theta}^i)\right)^T \boldsymbol{s}_i \geq (\beta - 1) \boldsymbol{g}(\boldsymbol{\theta}^i)^T \boldsymbol{s}_i > 0 \tag{1.152}$$

which guarantees that (1.147) is satisfied. The method for finding a value of λ_i that satisfies both (1.150) and (1.151) starts out by trying $\lambda_i = \lambda_p = 1$. If this trial value is not admissible because it fails to satisfy (1.150), a decreased value is found by cubic interpolation using $\mathcal{F}(\boldsymbol{\theta}^i)$, $\boldsymbol{g}(\boldsymbol{\theta}^i)$, $\mathcal{F}(\boldsymbol{\theta}^i + \lambda_p \boldsymbol{\delta}^i)$ and $\boldsymbol{g}(\boldsymbol{\theta}^i + \lambda_p \boldsymbol{\delta}^i)$. If the trial value satisfies (1.150) but not (1.151), an increased value is found by extrapolation. After one or more repetitions, an admissible λ_i is found, because it can be proved that there exists an interval $\lambda_i \in [\lambda_1, \lambda_2]$ where (1.150) and (1.151) are both satisfied (Dennis and Schnabel, 1983).

1.1.5.4 Constraints on parameters

In order to ensure stability in the calculation of the objective function in (1.26), simple constraints on the parameters are introduced, i.e.:

$$\theta_j^{\min} < \theta_j < \theta_j^{\max} \ , \ j = 1, \ldots, p \tag{1.153}$$

These constraints are satisfied by solving the optimisation problem with respect to a transformation of the original parameters, i.e.:

$$\tilde{\theta}_j = \ln\left(\frac{\theta_j - \theta_j^{\min}}{\theta_j^{\max} - \theta_j}\right) \ , \ j = 1, \ldots, p \tag{1.154}$$

A problem arises with this type of transformation when θ_j is very close to one of the limits, because the finite difference derivative with respect to θ_j may be close to zero, but this problem is solved by adding an appropriate penalty function to (1.26) to give the following modified objective function:

$$\mathcal{F}(\boldsymbol{\theta}) = -\ln\left(p(\boldsymbol{\theta}|\mathbf{Y}, \mathbf{y_0})\right) + P(\lambda, \boldsymbol{\theta}, \boldsymbol{\theta}^{\min}, \boldsymbol{\theta}^{\max}) \tag{1.155}$$

which is then used instead. The penalty function is given as follows:

$$P(\lambda, \boldsymbol{\theta}, \boldsymbol{\theta}^{\min}, \boldsymbol{\theta}^{\max}) = \lambda \left(\sum_{j=1}^{p} \frac{|\theta_j^{\min}|}{\theta_j - \theta_j^{\min}} + \sum_{j=1}^{p} \frac{|\theta_j^{\max}|}{\theta_j^{\max} - \theta_j}\right) \tag{1.156}$$

for $|\theta_j^{\min}| > 0$ and $|\theta_j^{\max}| > 0$, $j = 1, \ldots, p$. For proper choices of the Lagrange multiplier λ and the limiting values $\theta_j^{\min}$ and $\theta_j^{\max}$ the penalty function has no influence on the estimation when θ_j is well within the limits but will force the finite difference derivative to increase when θ_j is close to one of the limits.

Along with the parameter estimates **CTSM** computes normalized (by multiplication with the estimates) derivatives of $\mathcal{F}(\boldsymbol{\theta})$ and $P(\lambda, \boldsymbol{\theta}, \boldsymbol{\theta}^{\min}, \boldsymbol{\theta}^{\max})$ with respect to the parameters to provide information about the solution. The derivatives of $\mathcal{F}(\boldsymbol{\theta})$ should of course be close to zero, and the absolute values of the derivatives of $P(\lambda, \boldsymbol{\theta}, \boldsymbol{\theta}^{\min}, \boldsymbol{\theta}^{\max})$ should not be large compared to the corresponding absolute values of the derivatives of $\mathcal{F}(\boldsymbol{\theta})$, because this indicates that the corresponding parameters are close to one of their limits.

1.1.6 Performance issues

Solving optimisation problems of the general type in (1.27) is a computationally intensive task. The binary code within **CTSM** has therefore been optimized for maximum performance on all supported platforms, i.e. Linux, Solaris and Windows. On Solaris systems **CTSM** also supports shared memory parallel computing using the OpenMP Application Program Interface (API).

More specifically, the finite difference derivative approximations used to approximate the gradient of the objective function can be computed in parallel, and Figure 1.1 shows the performance benefits of this approach in terms of reduced execution time and demonstrates the resulting scalability of the program for the bioreactor example used in the User's Guide. In this example there are 11 unknown parameters, and in theory using 11 CPU's should therefore be most optimal. Nevertheless, using 12 CPU's seems to be slightly better, but this may be due to the inherent uncertainty of the determination of execution time. The apparently non-existing effect of adding CPU's in the interval 6-10 is due to an uneven distribution of the workload, since in this case at least one CPU performs two finite difference computations, while the others wait.

1.2 Other features

Secondary features of **CTSM** include computation of various statistics and facilitation of residual analysis through validation data generation.

1.2.1 Various statistics

Within **CTSM** an estimate of the uncertainty of the parameter estimates is obtained by using the fact that by the central limit theorem the estimator in (1.27) is asymptotically Gaussian with mean $\boldsymbol{\theta}$ and covariance:

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}} = \boldsymbol{H}^{-1} \tag{1.157}$$

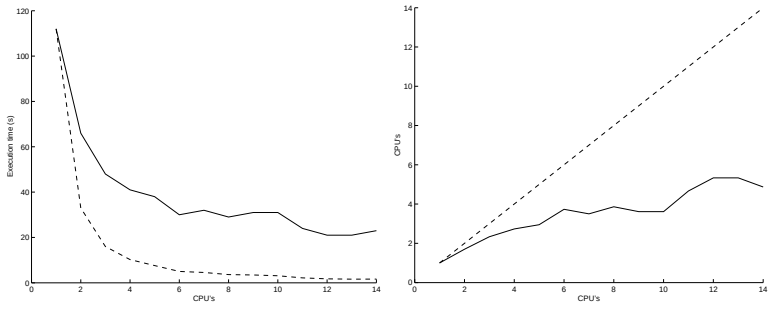
where the matrix $\boldsymbol{H}$ is given by:

$$\{h_{ij}\} = -E\left\{\frac{\partial^2}{\partial\theta_i\partial\theta_j}\ln\left(p(\boldsymbol{\theta}|\mathbf{Y},\mathbf{y_0})\right)\right\}\ ,\ i,j=1,\ldots,p \tag{1.158}$$

and where an approximation to $\boldsymbol{H}$ can be obtained from:

$$\{h_{ij}\} \approx -\left(\frac{\partial^2}{\partial\theta_i\partial\theta_j}\ln\left(p(\boldsymbol{\theta}|\mathbf{Y},\mathbf{y_0})\right)\right)\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}\ ,\ i,j=1,\ldots,p \tag{1.159}$$

which is the Hessian evaluated at the minimum of the objective function, i.e. $\boldsymbol{H}_i|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$. As an overall measure of the uncertainty of the parameter estimates,



(a) Performance. (b) Scalability.

Figure 1.1. Performance (execution time vs. no. of CPU's) and scalability (no. of CPU's vs. no. of CPU's) of **CTSM** when using shared memory parallel computing. Solid lines: **CTSM** values; dashed lines: Theoretical values (linear scalability).

the negative logarithm of the determinant of the Hessian is computed, i.e.:

$$-\ln\left(\det\left(\boldsymbol{H}_i|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}\right)\right) \tag{1.160}$$

The lower the value of this statistic, the lower the overall uncertainty of the parameter estimates. A measure of the uncertainty of the individual parameter estimates is obtained by decomposing the covariance matrix as follows:

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}} = \boldsymbol{\sigma}_{\hat{\boldsymbol{\theta}}}\boldsymbol{R}\boldsymbol{\sigma}_{\hat{\boldsymbol{\theta}}} \tag{1.161}$$

into $\boldsymbol{\sigma}_{\hat{\boldsymbol{\theta}}}$, which is a diagonal matrix of the standard deviations of the parameter estimates, and $\boldsymbol{R}$, which is the corresponding correlation matrix.

The asymptotic Gaussianity of the estimator in (1.27) also allows marginal t-tests to be performed to test the hypothesis:

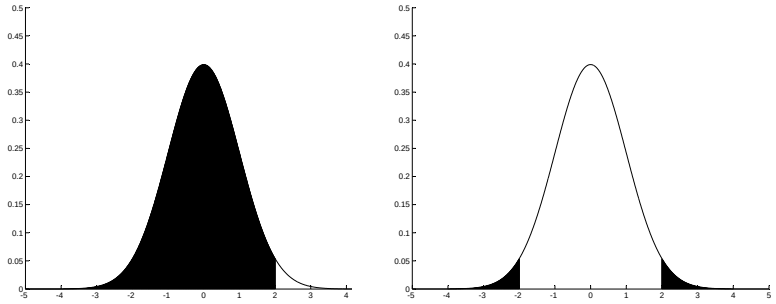
$$H_0\text{:}\quad \theta_j = 0 \tag{1.162}$$

against the corresponding alternative:

$$H_1\text{:}\quad \theta_j \neq 0 \tag{1.163}$$

i.e. to test whether a given parameter θ_j is marginally insignificant or not. The test quantity is the value of the parameter estimate divided by the standard deviation of the estimate, and under H_0 this quantity is asymptotically t-distributed with a number of degrees of freedom DF that equals the total number of observations minus the number of estimated parameters, i.e.:

$$z^t(\hat{\theta}_j) = \frac{\hat{\theta}_j}{\sigma_{\hat{\theta}_j}} \quad \in \quad t(\mathrm{DF}) = t\left(\left(\sum_{i=1}^{S}\sum_{k=1}^{N_i} l\right) - p\right) \tag{1.164}$$



(a) $P(t < |z^t(\hat{\theta}_j)|)$. (b) $P(t<-|z^t(\hat{\theta}_j)| \wedge t>|z^t(\hat{\theta}_j)|)$.

Figure 1.2. Illustration of computation of $P(t<-|z^t(\hat{\theta}_j)| \wedge t>|z^t(\hat{\theta}_j)|)$ via (1.167).

where, if there are missing observations in $\boldsymbol{y}_k^i$ for some i and some k, the particular value of l is reduced with the number of missing values in $\boldsymbol{y}_k^i$. The critical region for a test on significance level α is given as follows:

$$z^t(\hat{\theta}_j) < t(\mathrm{DF})_{\frac{\alpha}{2}} \ \vee \ z^t(\hat{\theta}_j) > t(\mathrm{DF})_{1-\frac{\alpha}{2}} \tag{1.165}$$

and to facilitate these tests, **CTSM** computes $z^t(\hat{\theta}_j)$ as well as the probabilities:

$$P\left(t<-|z^t(\hat{\theta}_j)| \wedge t>|z^t(\hat{\theta}_j)|\right) \tag{1.166}$$

for $j = 1, \ldots, p$. Figure 1.2 shows how these probabilities should be interpreted and illustrates their computation via the following relation:

$$P\left(t<-|z^t(\hat{\theta}_j)| \wedge t>|z^t(\hat{\theta}_j)|\right) = 2\left(1 - P(t < |z^t(\hat{\theta}_j)|)\right) \tag{1.167}$$

with $P(t < |z^t(\hat{\theta}_j)|)$ obtained by approximating the cumulative probability density of the t-distribution $t(\mathrm{DF})$ with the cumulative probability density of the standard Gaussian distribution $N(0,1)$ using the test quantity transformation:

$$z^N(\hat{\theta}_j) = z^t(\hat{\theta}_j)\frac{1 - \frac{1}{4\mathrm{DF}}}{\sqrt{1 + \frac{(z^t(\hat{\theta}_j))^2}{2\mathrm{DF}}}} \quad \in \quad N(0,1) \tag{1.168}$$

The cumulative probability density of the standard Gaussian distribution is computed by approximation using a series expansion of the error function.

1.2.2 Validation data generation

To facilitate e.g. residual analysis, **CTSM** can also be used to generate validation data, i.e. state and output estimates corresponding to a given input data set, using either pure simulation, prediction, filtering or smoothing.

1.2.2.1 Pure simulation data generation

The state and output estimates that can be generated by means of pure simulation are $\hat{\boldsymbol{x}}_{k|0}$ and $\hat{\boldsymbol{y}}_{k|0}$, $k = 0, \ldots, N$, along with their standard deviations $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|0}) = \sqrt{\mathrm{diag}(\boldsymbol{P}_{k|0})}$ and $\mathrm{SD}(\hat{\boldsymbol{y}}_{k|0}) = \sqrt{\mathrm{diag}(\boldsymbol{R}_{k|0})}$, $k = 0, \ldots, N$. The estimates are generated by the (extended) Kalman filter without updating.

1.2.2.2 Prediction data generation

The state and output estimates that can be generated by prediction are $\hat{\boldsymbol{x}}_{k|k-j}$, $j \geq 1$, and $\hat{\boldsymbol{y}}_{k|k-j}$, $j \geq 1$, $k = 0, \ldots, N$, along with their standard deviations $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|k-j}) = \sqrt{\mathrm{diag}(\boldsymbol{P}_{k|k-j})}$ and $\mathrm{SD}(\hat{\boldsymbol{y}}_{k|k-j}) = \sqrt{\mathrm{diag}(\boldsymbol{R}_{k|k-j})}$, $k = 0, \ldots, N$. The estimates are generated by the (extended) Kalman filter with updating.

1.2.2.3 Filtering data generation

The state estimates that can be generated by filtering are $\hat{\boldsymbol{x}}_{k|k}$, $k = 0, \ldots, N$, along with their standard deviations $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|k}) = \sqrt{\mathrm{diag}(\boldsymbol{P}_{k|k})}$, $k = 0, \ldots, N$. The estimates are generated by the (extended) Kalman filter with updating.

1.2.2.4 Smoothing data generation

The state estimates that can be generated by smoothing are $\hat{\boldsymbol{x}}_{k|N}$, $k = 0, \ldots, N$, along with their standard deviations $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|N}) = \sqrt{\mathrm{diag}(\boldsymbol{P}_{k|N})}$, $k = 0, \ldots, N$. The estimates are generated by means of a nonlinear smoothing algorithm based on the extended Kalman filter (for a formal derivation of the algorithm, see Gelb (1974)). The starting point is the following set of formulas:

$$\hat{\boldsymbol{x}}_{k|N} = \boldsymbol{P}_{k|N} \left(\boldsymbol{P}_{k|k-1}^{-1} \hat{\boldsymbol{x}}_{k|k-1} + \overline{\boldsymbol{P}}_{k|k}^{-1} \hat{\overline{\boldsymbol{x}}}_{k|k} \right) \tag{1.169}$$

$$\boldsymbol{P}_{k|N} = \left(\boldsymbol{P}_{k|k-1}^{-1} + \overline{\boldsymbol{P}}_{k|k}^{-1} \right)^{-1} \tag{1.170}$$

which states that the smoothed estimate can be computed by combining a forward filter estimate based only on past information with a backward filter estimate based only on present and "future" information. The forward filter estimates $\hat{\boldsymbol{x}}_{k|k-1}$, $k = 1, \ldots, N$, and their covariance matrices $\boldsymbol{P}_{k|k-1}$, $k = 1, \ldots, N$, can be computed by means of the EKF formulation given above, which is straightforward. The backward filter estimates $\hat{\overline{\boldsymbol{x}}}_{k|k}$, $k = 1, \ldots, N$, and their covariance matrices $\overline{\boldsymbol{P}}_{k|k}$, $k = 1, \ldots, N$, on the other hand, must be computed using a different set of formulas. In this set of formulas, a transformation of the time variable is used, i.e. $\tau = t_N - t$, which gives the SDE model, on which the backward filter is based, the following system equation:

$$d\boldsymbol{x}_{t_N - \tau} = -\boldsymbol{f}(\boldsymbol{x}_{t_N - \tau}, \boldsymbol{u}_{t_n - \tau}, t_N - \tau, \boldsymbol{\theta}) d\tau - \boldsymbol{\sigma}(\boldsymbol{u}_{t_N - \tau}, t_N - \tau, \boldsymbol{\theta}) d\boldsymbol{\omega}_\tau \tag{1.171}$$

where $\tau \in [0, t_N]$. The measurement equation remains unchanged. For ease of implementation a coordinate transformation is also introduced, i.e. $\boldsymbol{s}_t = \overline{\boldsymbol{P}}_t^{-1}\hat{\boldsymbol{x}}_t$, and the basic set of formulas in (1.169)-(1.170) is rewritten as follows:

$$\hat{\boldsymbol{x}}_{k|N} = \boldsymbol{P}_{k|N}\left(\boldsymbol{P}_{k|k-1}^{-1}\hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{s}_{k|k}\right) \tag{1.172}$$

$$\boldsymbol{P}_{k|N} = \left(\boldsymbol{P}_{k|k-1}^{-1} + \overline{\boldsymbol{P}}_{k|k}^{-1}\right)^{-1} \tag{1.173}$$

The backward filter consists of the *updating* equations:

$$\boldsymbol{s}_{k|k} = \boldsymbol{s}_{k|k+1} + \boldsymbol{C}_k^T\boldsymbol{S}_k^{-1}\left(\boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) + \boldsymbol{C}_k\hat{\boldsymbol{x}}_{k|k-1}\right) \tag{1.174}$$

$$\overline{\boldsymbol{P}}_{k|k}^{-1} = \overline{\boldsymbol{P}}_{k|k+1}^{-1} + \boldsymbol{C}_k^T\boldsymbol{S}_k^{-1}\boldsymbol{C}_k \tag{1.175}$$

and the *prediction* equations:

$$\frac{d\boldsymbol{s}_{t_N-\tau|k}}{d\tau} = \boldsymbol{A}_\tau^T\boldsymbol{s}_{t_N-\tau|k} - \overline{\boldsymbol{P}}_{t_N-\tau|k}^{-1}\boldsymbol{\sigma}_\tau\boldsymbol{\sigma}_\tau^T\boldsymbol{s}_{t_N-\tau|k} \tag{1.176}$$

$$- \overline{\boldsymbol{P}}_{t_N-\tau|k}^{-1}\left(\boldsymbol{f}(\hat{\boldsymbol{x}}_{t_N-\tau|k}, \boldsymbol{u}_{t_N-\tau}, t_N-\tau, \boldsymbol{\theta}) - \boldsymbol{A}_\tau\hat{\boldsymbol{x}}_{t_N-\tau|k}\right) \tag{1.177}$$

$$\frac{d\overline{\boldsymbol{P}}_{t_N-\tau|k}^{-1}}{d\tau} = \overline{\boldsymbol{P}}_{t_N-\tau|k}^{-1}\boldsymbol{A}_\tau + \boldsymbol{A}_\tau^T\overline{\boldsymbol{P}}_{t_N-\tau|k}^{-1} - \overline{\boldsymbol{P}}_{t_N-\tau|k}^{-1}\boldsymbol{\sigma}_\tau\boldsymbol{\sigma}_\tau^T\overline{\boldsymbol{P}}_{t_N-\tau|k}^{-1} \tag{1.178}$$

which are solved, e.g. by means of an ODE solver, for $\tau \in [\tau_k, \tau_{k+1}]$. In all of the above equations the following simplified notation has been applied:

$$\begin{aligned} \boldsymbol{A}_\tau = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}\Big|_{\hat{\boldsymbol{x}}_{t_N-\tau|k}, \boldsymbol{u}_{t_N-\tau}, t_N-\tau, \boldsymbol{\theta}} \;,\; \boldsymbol{C}_k = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_t}\Big|_{\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}} \\ \boldsymbol{\sigma}_\tau = \boldsymbol{\sigma}(\boldsymbol{u}_{t_N-\tau}, t_N-\tau, \boldsymbol{\theta}) \;,\; \boldsymbol{S}_k = \boldsymbol{S}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta}) \end{aligned} \tag{1.179}$$

Initial conditions for the backward filter are $\boldsymbol{s}_{N|N+1} = \boldsymbol{0}$ and $\overline{\boldsymbol{P}}_{N|N+1}^{-1} = \boldsymbol{0}$, which can be derived from an alternative formulation of (1.172)-(1.173):

$$\hat{\boldsymbol{x}}_{k|N} = \boldsymbol{P}_{k|N}\left(\boldsymbol{P}_{k|k}^{-1}\hat{\boldsymbol{x}}_{k|k} + \boldsymbol{s}_{k|k+1}\right) \tag{1.180}$$

$$\boldsymbol{P}_{k|N} = \left(\boldsymbol{P}_{k|k}^{-1} + \overline{\boldsymbol{P}}_{k|k+1}^{-1}\right)^{-1} \tag{1.181}$$

by realizing that the smoothed estimate must coincide with the forward filter estimate for $k = N$. The smoothing feature is only available for NL models.

References

Bierman, G. J. (1977). *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York, USA.

Dennis, J. E. and Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, USA.

Fletcher, R. and Powell, J. D. (1974). On the Modification of LDL^T Factorizations. *Math. Comp.*, **28**, 1067–1087.

Gelb, A. (1974). *Applied Optimal Estimation*. The MIT Press, Cambridge, USA.

Hindmarsh, A. C. (1983). ODEPACK, A Systematized Collection of ODE Solvers. In R. S. Stepleman, editor, *Scientific Computing (IMACS Transactions on Scientific Computation, Vol. 1)*, pages 55–64. North-Holland, Amsterdam.

Huber, P. J. (1981). *Robust Statistics*. Wiley, New York, USA.

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory*. Academic Press, New York, USA.

Moler, C. and van Loan, C. F. (1978). Nineteen Dubious Ways to Compute the Exponential of a Matrix. *SIAM Review*, **20**(4), 801–836.

Sidje, R. B. (1998). Expokit: A Software Package for Computing Matrix Exponentials. *ACM Transactions on Mathematical Software*, **24**(1), 130–156.

Speelpenning, B. (1980). Compiling Fast Partial Derivatives of Functions Given by Algorithms. Technical Report UILU-ENG 80 1702, University of Illinois-Urbana, Urbana, USA.

Thornton, C. L. and Bierman, G. J. (1980). UDU^T Covariance Factorization for Kalman Filtering. In C. T. Leondes, editor, *Control and Dynamic Systems*. Academic Press, New York, USA.

van Loan, C. F. (1978). Computing Integrals Involving the Matrix Exponential. *IEEE Transactions on Automatic Control*, **23**(3), 395–404.