Continuous Time Stochastic Modelling

CTSM 2.3

-

User's Guide

Niels Rode Kristensen, Henrik Madsen

December 10, 2003

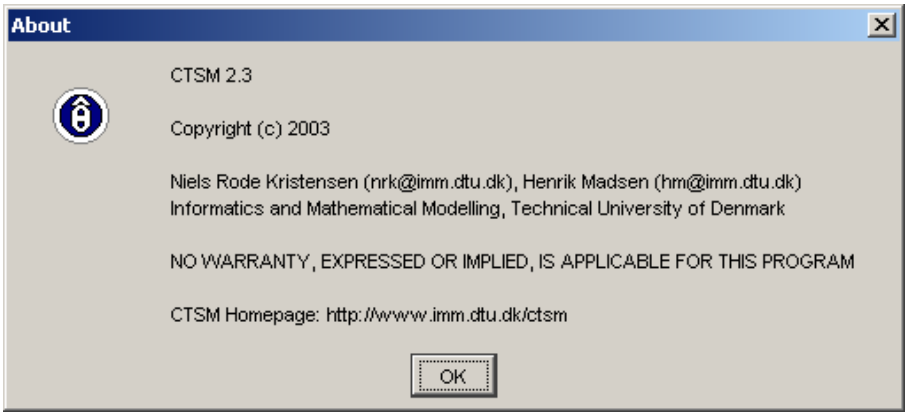
Technical University of Denmark

Contents

1

Introduction

CTSM is a computer program for performing **C**ontinuous **T**ime **S**tochastic **M**odelling. The program has been developed at Informatics and Mathematical Modelling (IMM) at the Technical University of Denmark (DTU).



Continuous time stochastic modelling means semi-physical modelling of dynamic systems based on stochastic differential equations. Stochastic differential equations contain a diffusion term to account for random effects, but are otherwise structurally similar to ordinary differential equations. Therefore conventional modelling principles can be applied to set up the model structure.

With the model structure given, the program provides methods for estimating any unknown parameters of the model from data, including the parameters of the diffusion term. These methods are able to handle both linear and nonlinear models, and the program also provides flexibility with respect to the data sets that can be used for estimation. The parameter estimation methods implemented in the program are a *maximum likelihood* (ML) method and a *maximum a posteriori* (MAP) method. Both methods allow several independent data sets to be used. Once the parameters have been estimated, various statistical methods can be applied to investigate the quality of the model, and features that facilitate application of such methods are also included in the program.

CTSM and the original program on which it is based, CTLSM (Madsen and Melgaard, 1991; Melgaard and Madsen, 1993), has been successfully applied for modelling a variety of systems, including building heat dynamics (Madsen and Holst, 1995), environmental systems (Jacobsen and Madsen, 1996), fed-batch processes within chemical and biotechnological industry (Kristensen, 2002) and pharmacokinetic and pharmacodynamic systems (Tornøe, 2002).

This manual is meant to give a short introduction to the ideas behind and the use of **CTSM**. Chapter 2 gives a brief mathematical description of the features in the program and provides instructions for installing it, Chapter 3 contains a step-by-step guide to using the program, Chapter 4 provides some basic tips and tricks, and in chapter 5 some hints for advanced troubleshooting are given.

Appendix A and B contain tutorial examples illustrating the procedure for parameter estimation for a linear and a nonlinear model respectively.

2

Getting started

This chapter gives a brief mathematical description of the features in **CTSM** and provides instructions for downloading and installing the program.

A complete description of the mathematics behind the algorithms of the program can be found in the Mathematics Guide.

2.1 Features of CTSM

CTSM provides features for performing *maximum likelihood* (ML) and *maximum a posteriori* (MAP) estimation of the unknown parameters of continuous-discrete stochastic state space models. Continuous-discrete stochastic state space models are models that consist of a set of stochastic differential equations describing the dynamics of a system in continuous time and a set of algebraic equations describing how measurements are obtained at discrete time instants. Several independent data sets can be used for the estimation, and the program also provides features for dealing with varying sample times, occasional outliers and missing observations. In addition, the program provides features for performing statistical tests and for facilitating residual analysis.

2.1.1 Model structures

Within **CTSM** the class of continuous-discrete stochastic state space models is divided into three distinct subclasses, which are handled separately.

The class of nonlinear (NL) models

The class of NL models is the class of models described by the equations:

$$d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\theta})dt + \boldsymbol{\sigma}(\boldsymbol{u}_t, t, \boldsymbol{\theta})d\boldsymbol{\omega}_t \tag{2.1}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k, \boldsymbol{\theta}) + \boldsymbol{e}_k \tag{2.2}$$

where $t \in \mathbb{R}$ is time, $\boldsymbol{x}_t \in \mathbb{R}^n$ is a vector of state variables, $\boldsymbol{u}_t \in \mathbb{R}^m$ is a vector of input variables, $\boldsymbol{y}_k \in \mathbb{R}^l$ is a vector of output variables, $\boldsymbol{\theta} \in \mathbb{R}^p$ 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 $\boldsymbol{e}_k \in N(\boldsymbol{0}, \boldsymbol{S}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta}))$.

The class of linear time-varying (LTV) models

The class of LTV models is the class of models described by the equations:

$$d\boldsymbol{x}_t = (\boldsymbol{A}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\theta})\boldsymbol{x}_t + \boldsymbol{B}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\theta})\boldsymbol{u}_t)\, dt + \boldsymbol{\sigma}(\boldsymbol{u}_t, t, \boldsymbol{\theta}) d\boldsymbol{\omega}_t \tag{2.3}$$

$$\boldsymbol{y}_k = \boldsymbol{C}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k, \boldsymbol{\theta})\boldsymbol{x}_k + \boldsymbol{D}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k, \boldsymbol{\theta})\boldsymbol{u}_k + \boldsymbol{e}_k \tag{2.4}$$

where $t \in \mathbb{R}$ is time, $\boldsymbol{x}_t \in \mathbb{R}^n$ is a state vector, $\boldsymbol{u}_t \in \mathbb{R}^m$ is an input vector, $\boldsymbol{y}_k \in \mathbb{R}^l$ is an output vector, $\boldsymbol{\theta} \in \mathbb{R}^p$ is a parameter vector, $\boldsymbol{A}(\cdot) \in \mathbb{R}^{n \times n}$, $\boldsymbol{B}(\cdot) \in \mathbb{R}^{n \times m}$, $\boldsymbol{\sigma}(\cdot) \in \mathbb{R}^{n \times n}$, $\boldsymbol{C}(\cdot) \in \mathbb{R}^{l \times n}$ 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 $\boldsymbol{e}_k \in N(\boldsymbol{0}, \boldsymbol{S}(\boldsymbol{u}_k, t_k, \boldsymbol{\theta}))$.

The class of linear time invariant (LTI) models

The class of LTI models is the class of models described by the equations:

$$d\boldsymbol{x}_t = (\boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{x}_t + \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{u}_t)\, dt + \boldsymbol{\sigma}(\boldsymbol{\theta}) d\boldsymbol{\omega}_t \tag{2.5}$$

$$\boldsymbol{y}_k = \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{x}_k + \boldsymbol{D}(\boldsymbol{\theta})\boldsymbol{u}_k + \boldsymbol{e}_k \tag{2.6}$$

where $t \in \mathbb{R}$ is time, $\boldsymbol{x}_t \in \mathbb{R}^n$ is a state vector, $\boldsymbol{u}_t \in \mathbb{R}^m$ is an input vector, $\boldsymbol{y}_k \in \mathbb{R}^l$ is an output vector, $\boldsymbol{\theta} \in \mathbb{R}^p$ is a parameter vector, $\boldsymbol{A}(\cdot) \in \mathbb{R}^{n \times n}$, $\boldsymbol{B}(\cdot) \in \mathbb{R}^{n \times m}$, $\boldsymbol{\sigma}(\cdot) \in \mathbb{R}^{n \times n}$, $\boldsymbol{C}(\cdot) \in \mathbb{R}^{l \times n}$ 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 $\boldsymbol{e}_k \in N(\boldsymbol{0}, \boldsymbol{S}(\boldsymbol{\theta}))$.

2.1.2 Parameter estimation methods

CTSM allows a number of different parameter estimation setups to be used.

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 sequence of measurements $\boldsymbol{y}_0$, $\boldsymbol{y}_1$, …, $\boldsymbol{y}_k$, …, $\boldsymbol{y}_N$ of the output variables. By introducing the notation:

$$\mathcal{Y}_k = [\boldsymbol{y}_k, \boldsymbol{y}_{k-1}, \ldots, \boldsymbol{y}_1, \boldsymbol{y}_0] \tag{2.7}$$

the likelihood function is the joint probability density:

$$L(\boldsymbol{\theta}; \mathcal{Y}_N) = p(\mathcal{Y}_N | \boldsymbol{\theta}) \tag{2.8}$$

or equivalently:

$$L(\boldsymbol{\theta}; \mathcal{Y}_N) = \left(\prod_{k=1}^{N} p(\boldsymbol{y}_k | \mathcal{Y}_{k-1}, \boldsymbol{\theta}) \right) p(\boldsymbol{y}_0 | \boldsymbol{\theta}) \tag{2.9}$$

where the rule $P(A \cap B) = P(A|B)P(B)$ has been applied to form a product of conditional probability densities. By introducing the notation:

$$\hat{\boldsymbol{y}}_{k|k-1} = E\{\boldsymbol{y}_k | \mathcal{Y}_{k-1}, \boldsymbol{\theta}\} \tag{2.10}$$

$$\boldsymbol{R}_{k|k-1} = V\{\boldsymbol{y}_k | \mathcal{Y}_{k-1}, \boldsymbol{\theta}\} \tag{2.11}$$

and:

$$\boldsymbol{\epsilon}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1} \tag{2.12}$$

and by assuming that the conditional probability densities are Gaussian, the likelihood function can be written as follows:

$$L(\boldsymbol{\theta}; \mathcal{Y}_N) = \left(\prod_{k=1}^{N} \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_k^T \boldsymbol{R}_{k|k-1}^{-1} \boldsymbol{\epsilon}_k\right)}{\sqrt{\det(\boldsymbol{R}_{k|k-1})}\left(\sqrt{2\pi}\right)^l} \right) p(\boldsymbol{y}_0 | \boldsymbol{\theta}) \tag{2.13}$$

where $\boldsymbol{\epsilon}_k$ and $\boldsymbol{R}_{k|k-1}$ can be computed by means of a Kalman filter (LTI models) or an iterated extended Kalman filter (LTV and NL models). Further conditioning on $\boldsymbol{y}_0$ and taking the negative logarithm gives:

$$\begin{aligned} -\ln\left(L(\boldsymbol{\theta}; \mathcal{Y}_N | \boldsymbol{y}_0)\right) = \frac{1}{2} \sum_{k=1}^{N} \left(\ln(\det(\boldsymbol{R}_{k|k-1})) + \boldsymbol{\epsilon}_k^T \boldsymbol{R}_{k|k-1}^{-1} \boldsymbol{\epsilon}_k \right) \\ + \frac{1}{2} \left(\sum_{k=1}^{N} l \right) \ln(2\pi) \end{aligned} \tag{2.14}$$

and ML estimates of the parameters (and the initial states if these are unknown) can now be determined by solving the nonlinear optimisation problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \Theta} \left\{ -\ln\left(L(\boldsymbol{\theta}; \mathcal{Y}_N | \boldsymbol{y}_0)\right) \right\} \tag{2.15}$$

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:

$$p(\boldsymbol{\theta} | \mathcal{Y}_N) = \frac{p(\mathcal{Y}_N | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{Y}_N)} \propto p(\mathcal{Y}_N | \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{2.16}$$

and finding the parameters that maximize this function, i.e. by performing *maximum a posteriori* (MAP) estimation. By introducing the notation:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}} = E\{\boldsymbol{\theta}\} \tag{2.17}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = V\{\boldsymbol{\theta}\} \tag{2.18}$$

and:

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}} = \boldsymbol{\theta} - \boldsymbol{\mu}_{\boldsymbol{\theta}} \tag{2.19}$$

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

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{Y}_N) \propto & \left(\prod_{k=1}^{N} \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_k^T \boldsymbol{R}_{k|k-1}^{-1} \boldsymbol{\epsilon}_k\right)}{\sqrt{\det(\boldsymbol{R}_{k|k-1})}\left(\sqrt{2\pi}\right)^l} \right) p(\boldsymbol{y}_0|\boldsymbol{\theta}) \\ & \times \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\right)}{\sqrt{\det(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})}\left(\sqrt{2\pi}\right)^p} \end{aligned} \tag{2.20}$$

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

$$\begin{aligned} -\ln\left(p(\boldsymbol{\theta}|\mathcal{Y}_N, \boldsymbol{y}_0)\right) \propto & \frac{1}{2}\sum_{k=1}^{N}\left(\ln(\det(\boldsymbol{R}_{k|k-1})) + \boldsymbol{\epsilon}_k^T \boldsymbol{R}_{k|k-1}^{-1}\boldsymbol{\epsilon}_k\right) \\ & + \frac{1}{2}\left(\left(\sum_{k=1}^{N} l\right) + p\right)\ln(2\pi) \\ & + \frac{1}{2}\ln(\det(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})) + \frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} \end{aligned} \tag{2.21}$$

and MAP estimates of the parameters (and the initial states if these are unknown) can now be determined by solving the nonlinear optimisation problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta}\left\{-\ln\left(p(\boldsymbol{\theta}|\mathcal{Y}_N, \boldsymbol{y}_0)\right)\right\} \tag{2.22}$$

Using several independent data sets

Instead of a single sequence of measurements, several separate sequences, i.e. $\mathcal{Y}_{N_1}^1$, $\mathcal{Y}_{N_2}^2$, …, $\mathcal{Y}_{N_i}^i$, …, $\mathcal{Y}_{N_S}^S$, possibly of varying length, may be available. In this case a similar estimation method can be applied by expanding the expression for the posterior probability density function to a more general form:

$$\begin{aligned} p(\boldsymbol{\theta}|\mathbf{Y}) \propto & \prod_{i=1}^{S}\left(\prod_{k=1}^{N_i} \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\epsilon}_k^i)^T (\boldsymbol{R}_{k|k-1}^i)^{-1}\boldsymbol{\epsilon}_k^i\right)}{\sqrt{\det(\boldsymbol{R}_{k|k-1}^i)}\left(\sqrt{2\pi}\right)^l} \right) p(\boldsymbol{y}_0^i|\boldsymbol{\theta}) \\ & \times \frac{\exp\left(-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^T \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\right)}{\sqrt{\det(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})}\left(\sqrt{2\pi}\right)^p} \end{aligned} \tag{2.23}$$

where:

$$\mathbf{Y} = [\mathcal{Y}_{N_1}^1, \mathcal{Y}_{N_2}^2, \ldots, \mathcal{Y}_{N_i}^i, \ldots, \mathcal{Y}_{N_S}^S] \tag{2.24}$$

and where the individual sequences of measurements are assumed to be stochastically independent. Further conditioning on:

$$\mathbf{y_0} = [\boldsymbol{y}_0^1, \boldsymbol{y}_0^2, \ldots, \boldsymbol{y}_0^i, \ldots, \boldsymbol{y}_0^S] \tag{2.25}$$

and taking the negative logarithm gives:

$$\begin{aligned} -\ln\left(p(\boldsymbol{\theta}|\mathbf{Y}, \mathbf{y_0})\right) \propto & \frac{1}{2}\sum_{i=1}^{S}\sum_{k=1}^{N_i}\left(\ln(\det(\boldsymbol{R}_{k|k-1}^i)) + (\boldsymbol{\epsilon}_k^i)^T(\boldsymbol{R}_{k|k-1}^i)^{-1}\boldsymbol{\epsilon}_k^i\right) \\ & + \frac{1}{2}\left(\left(\sum_{i=1}^{S}\sum_{k=1}^{N_i} l\right) + p\right)\ln(2\pi) \\ & + \frac{1}{2}\ln(\det(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})) + \frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^T\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}\boldsymbol{\epsilon}_{\boldsymbol{\theta}} \end{aligned} \tag{2.26}$$

and estimates of the parameters (and the initial states if these are unknown) can now be determined by solving the nonlinear optimisation problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta}\left\{-\ln\left(p(\boldsymbol{\theta}|\mathbf{Y}, \mathbf{y_0})\right)\right\} \tag{2.27}$$

2.1.3 Data issues

Within **CTSM** features have also been implemented for dealing with varying sample times, occasional outliers and missing observations.

2.1.4 Statistical tests

In addition to the parameter estimates themselves, **CTSM** computes the standard deviations of the estimates, the corresponding t-test scores and associated probabilities for testing the significance of the parameters, as well as the correlation matrix of the estimates. Altogether these quantities allow a number of statistical tests to be performed to investigate the quality of the model.

2.1.5 Residual analysis

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

2.2 Installing CTSM

CTSM is available for Linux, Solaris and Windows platforms. This section provides instructions for downloading and installing the program on all three platforms. More information can be found on the program homepage:

`http://www.imm.dtu.dk/ctsm`

2.2.1 Downloading CTSM

CTSM can be downloaded or installed directly from the download section of the program homepage. To be able to install directly from the program homepage, your web browser must support Java applets.

2.2.2 Installing CTSM on Linux platforms

CTSM for Linux has only been tested on Red Hat Linux 7.3 but is expected to work on other Linux implementations as well. To install on Linux:

1. Use the installer applet on the program homepage or download the installer to a temporary directory, open a shell, `cd` to the temporary directory and type: `sh ./install.bin`.
2. Follow the on-screen instructions provided by the installer to automatically install the program on your system.
3. Update your system's LD_LIBRARY_PATH environment variable as instructed by the installer when installation is complete.

Updating the LD_LIBRARY_PATH environment variable

To permanently add `newDir` for `bash`, `ksh` and `keysh` shells:

1. Open the `.profile` file in your home directory.
2. Add the line: `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:newDir`.
3. Add the line: `export LD_LIBRARY_PATH`.
4. Save, log out and log in again.

To permanently add `newDir` for `csh` and `tcsh` shells:

1. Open the `.login` file in your home directory.
2. Add the line: `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:newDir`.
3. Save, log out and log in again.

2.2.3 Installing CTSM on Solaris platforms

CTSM for Solaris has been optimized specifically for SUN servers equipped with ULTRASPARC-III processors using the Forte Developer 7 compiler suite. It will not run on systems with pre-ULTRASPARC-III processors and other versions of the compiler suite (use the workaround described on the program homepage for Solaris systems where Forte Developer 7 is installed but is not the default compiler suite). To find out if your system fulfills these requirements, contact your system administrator. To install on Solaris:

1. Use the installer applet on the program homepage or download the installer to a temporary directory, open a shell, `cd` to the temporary directory and type: `sh ./install.bin`.
2. Follow the on-screen instructions provided by the installer to automatically install the program on your system.
3. Update your system's LD_LIBRARY_PATH environment variable as instructed by the installer when installation is complete.

Updating the LD_LIBRARY_PATH environment variable

To permanently add `newDir` for `bash`, `ksh` and `keysh` shells:

1. Open the `.profile` file in your home directory.
2. Add the line: `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:newDir`.
3. Add the line: `export LD_LIBRARY_PATH`.
4. Save, log out and log in again.

To permanently add `newDir` for `csh` and `tcsh` shells:

1. Open the `.login` file in your home directory.
2. Add the line: `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:newDir`.
3. Save, log out and log in again.

2.2.4 Installing CTSM on Windows platforms

CTSM for Windows has only been tested on Windows 2000 and XP, but is expected to work on other Windows versions as well. To install on Windows:

1. Use the installer applet on the program homepage or download the installer to a temporary directory and double-click `install.exe`.

2. Follow the on-screen instructions provided by the installer to automatically install the program on your system.
3. Update your system's PATH environment variable as instructed by the installer when installation is complete.
4. Make sure your system's TEMP environment variable is DOS-compliant (no blanks and no long path names).

Updating the PATH environment variable

To permanently add `newDir` on Windows 95, 98 and ME:

1. Open `C:\Autoexec.bat`.
2. Add the line: `PATH=%PATH%;newDir`.
3. Save and reboot.

To permanently add `newDir` on Windows NT, 2000 and XP:

1. Right-click My Computer.
2. Choose Properties - Advanced - Environment Variables.
3. Edit the PATH by adding: `newDir`.
4. Save and reboot.

Checking the TEMP environment variable

To check the TEMP variable on Windows 95, 98 and ME:

1. Open `C:\Autoexec.bat`.
2. If there is a `TEMP` line, make sure it is DOS-compliant.
3. Save and reboot, if you made any changes.

To check the TEMP variable on Windows NT, 2000 and XP:

1. Right-click My Computer.
2. Choose Properties - Advanced - Environment Variables.
3. Make sure the `TEMP` variable is DOS-compliant.
4. Save and reboot, if you made any changes.

3

Using CTSM

The various features of **CTSM** can all be controlled via the program's graphical user interface (GUI). The following is a step-by-step guide, which shows how to do this, with screen shots taken from the tutorial example in Appendix B.

3.1 Starting up CTSM

Depending on the platform, there are different ways of starting up **CTSM**.

3.1.1 Starting up CTSM on Linux platforms

If **CTSM** has been installed correctly and if the system's LD_LIBRARY_PATH environment variable has been updated appropriately (see Section 2.2.2), the program can be started from a shell by typing `Ctsm` in the directory where the program is installed or in the directory of the installed link.

3.1.2 Starting up CTSM on Solaris platforms

If **CTSM** has been installed correctly and if the system's LD_LIBRARY_PATH environment variable has been updated appropriately (see Section 2.2.3), the program can be started from a shell by typing `Ctsm` in the directory where the program is installed or in the directory of the installed link.

3.1.3 Starting up CTSM on Windows platforms

If **CTSM** has been installed correctly and if the system's PATH variable has been updated and the TEMP variable is DOS-compliant (see Section 2.2.4), the program can be started by double-clicking the installed shortcut.

3.2 Setting up a model

When **CTSM** has been started up, the GUI (Windows Look and Feel) appears as in Figure 3.1. The only enabled actions are New model and Open model (see Section 3.7 for details about opening an existing model).

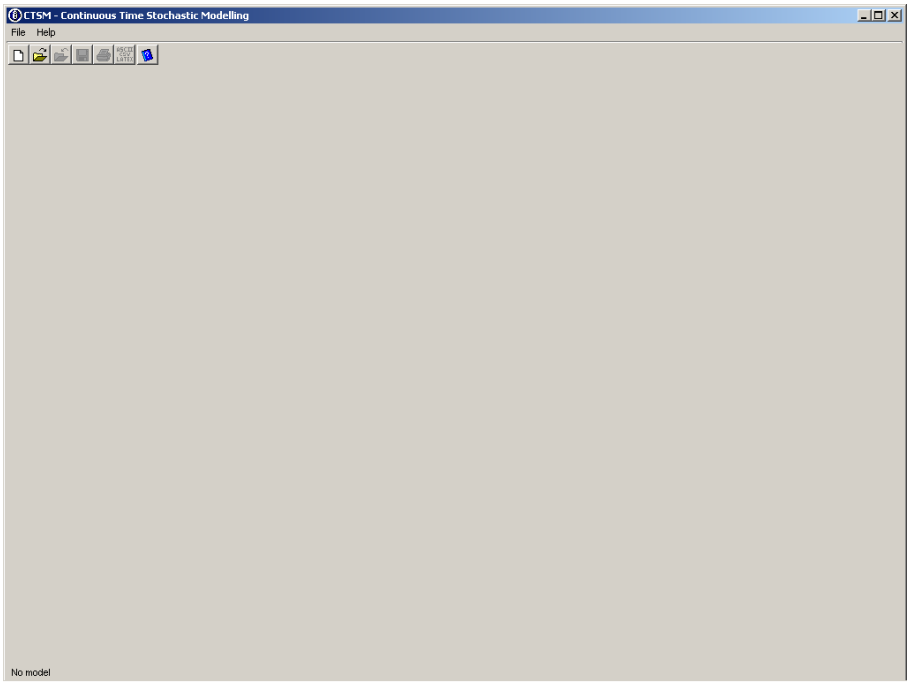


Figure 3.1. GUI when **CTSM** has been started.

3.2.1 Creating a new model

To create a new model, select the New model action. The New Model Specification dialog now appears (see Figure 3.2), and you can specify the type and dimensions of the new model, you want to create. **CTSM** distinguishes between three different model types: LTI, LTV and NL, and the dimensions that must be specified are the number of inputs, the number of outputs, the number of states and the number of algebraic equations. At present, **CTSM** can only handle explicit algebraic equations, i.e. algebraic equations that can be substituted directly into the other equations of the model.

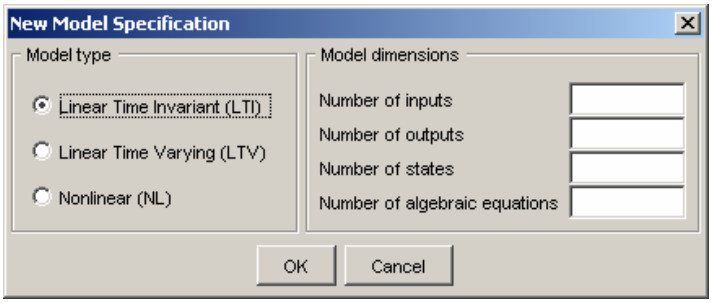


Figure 3.2. The empty New Model Specification dialog.

> Models must have at least 1 output and 1 state. Models without inputs and algebraic equations are allowed. The upper limit on the number of inputs, outputs, states and algebraic equations is 50.

Figure 3.3 shows the New Model Specification dialog with values appropriately assigned. To proceed from here, press OK.

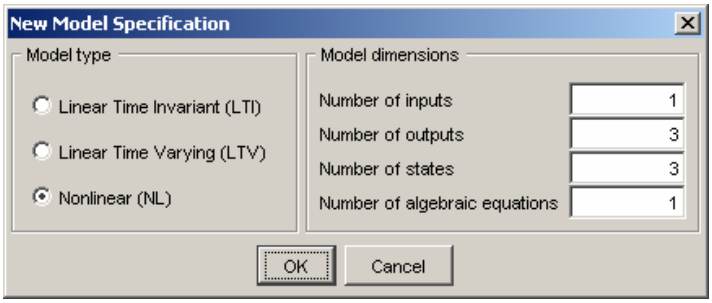


Figure 3.3. The New Model Specification dialog with values assigned.

3.2.2 Typing in the model equations

When a new model has been created, the GUI appears as in Figure 3.4 (with the Model tab and the Close model, Save model and Analyze model actions enabled), and the model equations can now be typed in.

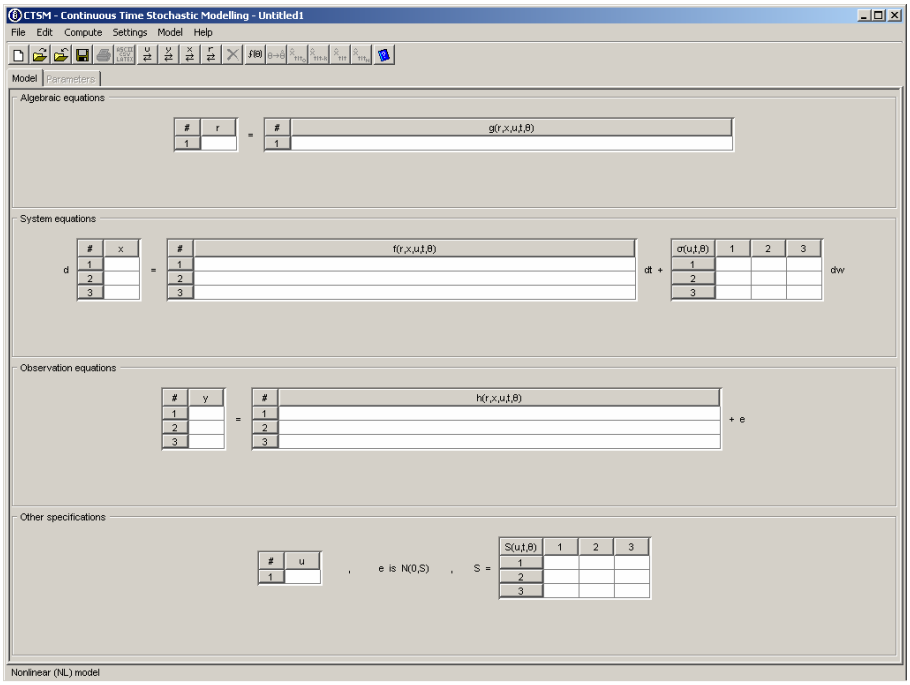


Figure 3.4. GUI when a new model has been created.

> Typing in the model equations means typing in the symbolic names of the inputs ($\boldsymbol{u}$), outputs ($\boldsymbol{y}$) and states ($\boldsymbol{x}$) and the symbolic expressions for the elements of vectors $\boldsymbol{f}$ and $\boldsymbol{h}$ (NL models), matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ (LTI and LTV models) and matrices $\boldsymbol{\sigma}$ and $\boldsymbol{S}$. If the model has algebraic equations, their symbolic names and the symbolic expressions for the right hand sides must be typed in too.

The rules for typing in model equations are determined by **CTSM**'s interpreter:

- Characters accepted by the interpreter are letters **A-Z** and **a-z**, integers **0-9**, operators +, **-**, *****, /, ^, parentheses **(** and **)** and decimal separator **.**.

- The interpreter is case sensitive with respect to the symbolic names of inputs, outputs, states, algebraic equations and parameters, but not with respect to common mathematical functions. This means that the names '**k1**' and '**K1**' are different, whereas the names '**exp()**' and '**EXP()**' are the same. The single character '**t**' is treated as the time variable, whereas the single character '**T**' is treated as any other single character.

- The number formats accepted by the interpreter are the following: Scientific (i.e. **1.2E+1**), standard (i.e. **12.0**) and integer (i.e. **12**).

- Each factor, i.e. each collection of latin letters and integers separated by operators or parentheses, which is not a number or a common mathematical function, is checked to see if it corresponds to the symbolic name of any of the inputs, outputs, states or algebraic equations or to the time variable. If not, the factor is regarded as a parameter.

- The common mathematical functions recognized by the interpreter are the following: **abs()**, **sign()**, **sqrt()**, **exp()**, **log()**, **sin()**, **cos()**, **tan()**, **arcsin()**, **arctan()**, **sinh()** and **cosh()**.

3.2.3 Analyzing the model equations

An example of how the GUI appears when the model equations have been typed in is shown in Figure 3.5. To proceed from here by letting **CTSM**'s interpreter analyze the model equations, select the Analyze model action.

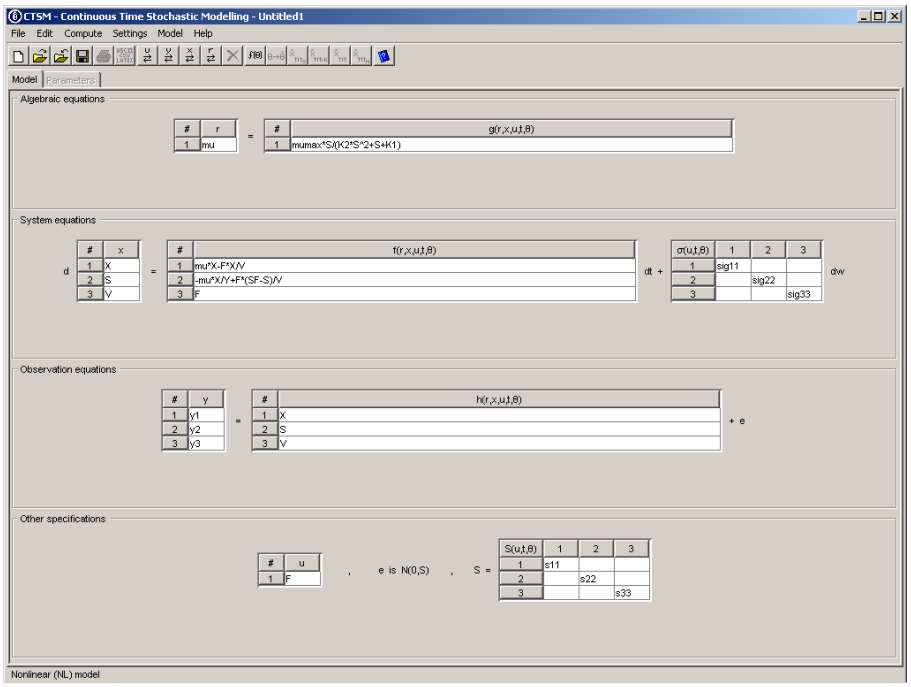


Figure 3.5. GUI when the model equations have been typed in.

When analyzing the model equations the interpreter first checks for compliance with the rules for typing in model equations (see Section 3.2.2), and then analyzes each symbolic expression in turn to determine the symbolic names of the parameters of the model, which are then displayed along with the symbolic names of the initial states under the Parameters tab as shown in Figure 3.6.

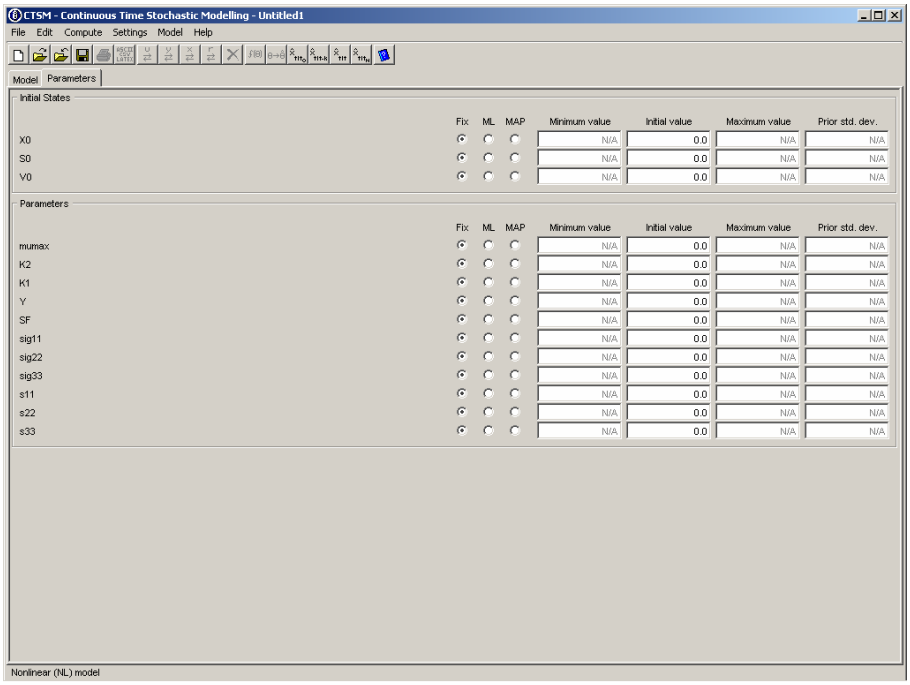


Figure 3.6. GUI when the model has been analyzed.

Once the model equations have been analyzed the Analyze model action remains disabled until a change is made to the model equations.

3.3 Estimating parameters

When the initial states and parameters are first displayed under the Parameters tab (see Figure 3.6), they all have a default Initial value of 0 and radio buttons set to Fix. To perform an estimation, you have to specify an estimation method.

3.3.1 Specifying estimation method

The rules for specifying estimation method are the following:

- For fixed parameters, i.e. with radio buttons set to Fix, only the Initial value must be specified. The program will use this value as a fixed value.

- For parameters to be estimated with the *maximum likelihood* method (see Section 2.1), i.e. with radio buttons set to ML, the Minimum value and Maximum value must also be specified. In this case the program will use the Initial value as an initial guess.

- For parameters to be estimated with the *maximum a posteriori* method (see Section 2.1), i.e. with radio buttons set to MAP, the Prior std. dev. and the corresponding elements of the Prior Correlation Matrix, which appears below, must also be specified. In this case the program will regard the Initial value as the prior mean and use it as an initial guess.

An example of how the GUI appears when an estimation method has been specified for all initial states and parameters is shown in Figure 3.7.

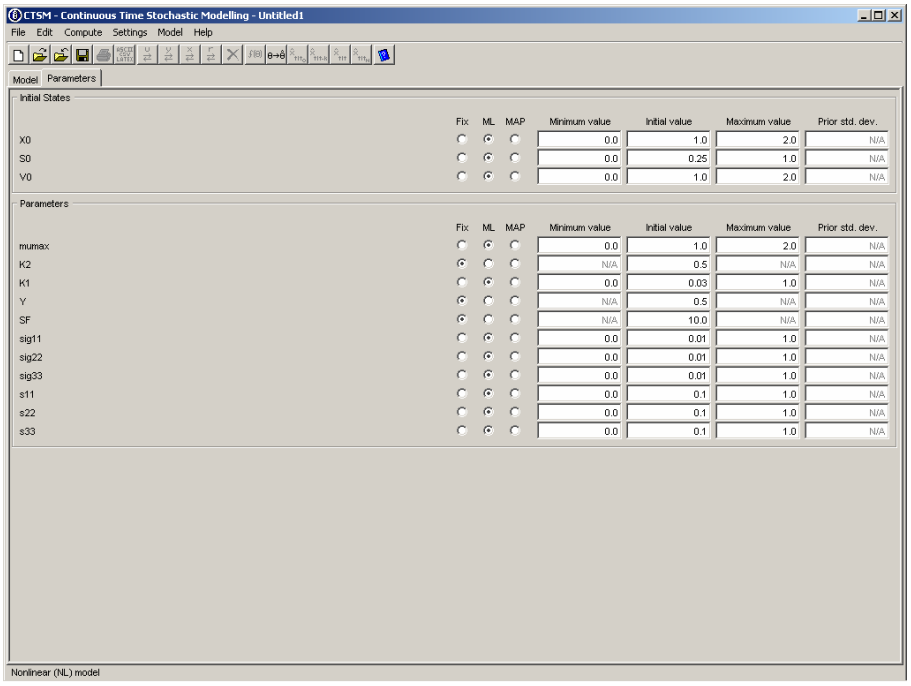


Figure 3.7. GUI when estimation methods have been specified.

3.3.2 Specifying estimation data

To start an estimation, select the Estimate parameters action. The Specification of Data for Estimation dialog now appears (see Figure 3.8), and you can select the data file(s), you want to use for the estimation.

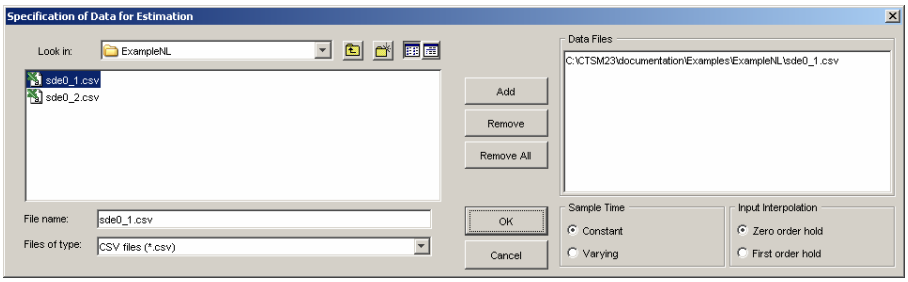


Figure 3.8. The Specification of Data for Estimation dialog.

> **CTSM** expects data in semi-colon delimited CSV files (`*.csv`) with $1 + m + l$ columns, i.e. as many columns as there are inputs and outputs plus a column for time instants. Within each row the column entries must be separated by semi-colons '**;**' and appear in the order: time, inputs and outputs. For models with more than one input and/or more than one output the inputs and outputs must appear in the order they were typed in. Missing observations must be indicated with the number $1E300$ (or larger). Note that when selecting more than one data file, the program will use the same initial states when processing all the selected files.

Apart from specifying data file(s), you must also specify to the program whether to use Constant or Varying sample time and whether to use a Zero order hold or a First order hold on the inputs between sample instants.

> If Constant sample time is specified, the program will determine the sample time from the first two time instants in the data file(s). If Varying sample time is specified, sample times will be determined successively from the time instants.

When all necessary information has been provided, press OK to proceed.

3.3.3 Monitoring computation progress

An example of how the GUI appears when the computation has commenced is shown in Figure 3.9. You can monitor the progress of the computation via the Trace of optimisation panel, which shows the number of iterations, the number of objective function evaluations, the gradient approximation method and the value of the objective function, and via the Trace of parameter estimates panel, which shows the values of the individual parameter estimates.

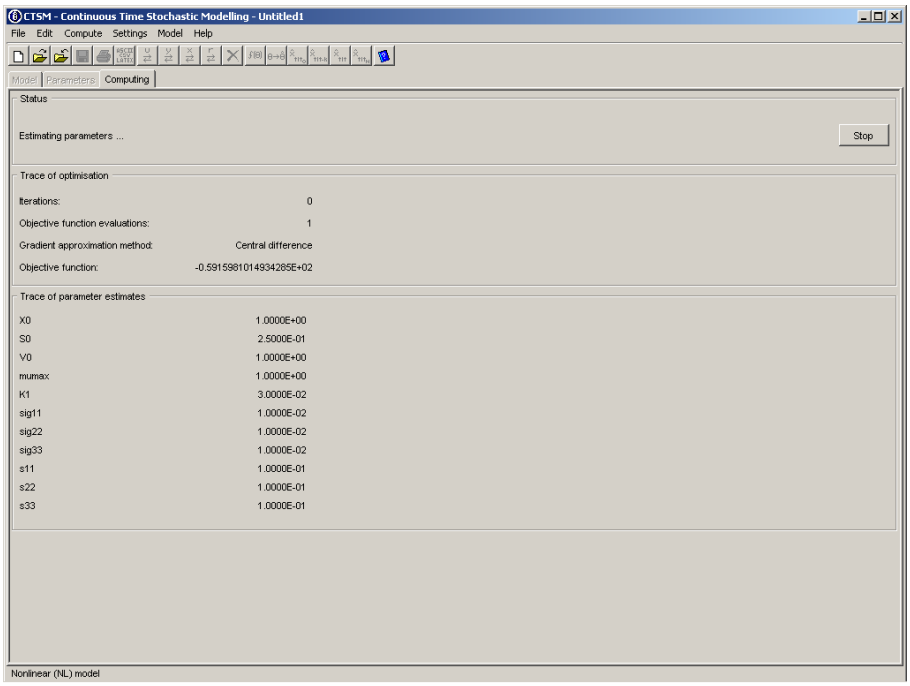


Figure 3.9. GUI when the computation has commenced.

> The computation can be stopped at any time by pressing the Stop button in the Status panel. The Stop button terminates execution, so it cannot be used to pause the computation.

3.3.4 Interpreting estimation results

Once the computation is complete, and the initial states and parameters have been estimated (see Section 5.1 for an explanation of common error messages if the computation stops prematurely), the GUI appears as in Figure 3.10.

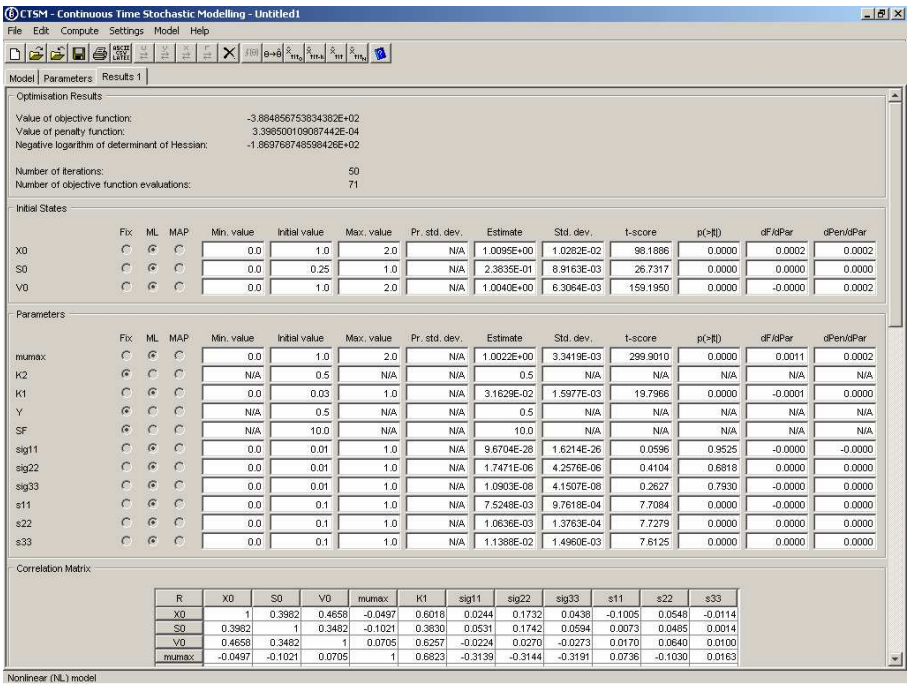


Figure 3.10. GUI when the initial states and parameters have been estimated.

For each result set (here Results 1) a number of panels are displayed: An Optimisation Results panel with information about the course of the optimisation, three, or four if MAP estimation has been used, panels with the results of the estimation, a Data panel with information about the data file(s) used for the estimation, a Model panel with the model equations, and a Settings panel.

The Optimisation Results panel displays the value of the objective function, and the value of the prior probability density function if MAP estimation has been used, along with the value of the penalty function and the negative logarithm of the determinant of the Hessian (see the Mathematics Guide for details), the number of iterations and the number of objective function evaluations.

> If the value of the penalty function is significant compared to the value of the objective function, a parameter may be close to one of its limits, and you should consider to loosen this limit.

The panels with the results of the estimation display information about all the initial states and parameters, the Correlation Matrix of the parameter estimates, and the Prior Correlation Matrix if MAP estimation has been used.

For each initial state and parameter the information provided under the Parameters tab is redisplayed along with the Estimate, the Std. dev. and the corresponding t-score (see Section 2.1). The p(>|t|) value is the fraction of probability of the corresponding t-distribution outside the limits set by the t-score.

> Loosely speaking, the p(>|t|) value is the probability that the particular initial state or parameter is insignificant, i.e. equal to 0.

The dF/dPar and dPen/dPar values are derivatives of the objective function and the penalty function with respect to the particular initial state or parameter.

> If the dF/dPar value is not close to zero, the solution found may not be the true optimum, and you should consider changing some settings for the optimisation and repeating the computation. If the dPen/dPar value is significant compared to the dF/dPar value, the particular initial state or parameter may be close to one of its limits, and you should consider to loosen this limit.

Below the initial states and parameters the Correlation Matrix of the parameter estimates (see Section 2.1) is displayed, and below that the Prior Correlation Matrix is redisplayed if MAP estimation has been used.

> If the Correlation Matrix has off-diagonal values close to 1 or -1, it is an indication that the model is overparameterized, and you should consider eliminating some of the parameters.

3.3.5 Printing estimation results

The results of an estimation, i.e. the contents of a result set, can be printed on paper by selecting the Print selected result set action.

> **CTSM** prints the contents of the Optimisation Results panel, the panels with the results of the estimation, the Data panel and the Settings panel. The Model panel is not printed. The Correlation Matrix and the Prior Correlation Matrix are only printed if they are small enough to fit (in terms of width) on a single piece of paper.

3.3.6 Exporting estimation results

Some of the contents of a result set can be exported to a file by selecting the Export selected result set action. This will make the Specification of Export of

Results dialog appear (see Figure 3.11), and you can specify which parts of the exportable results you want to export and to what output file.

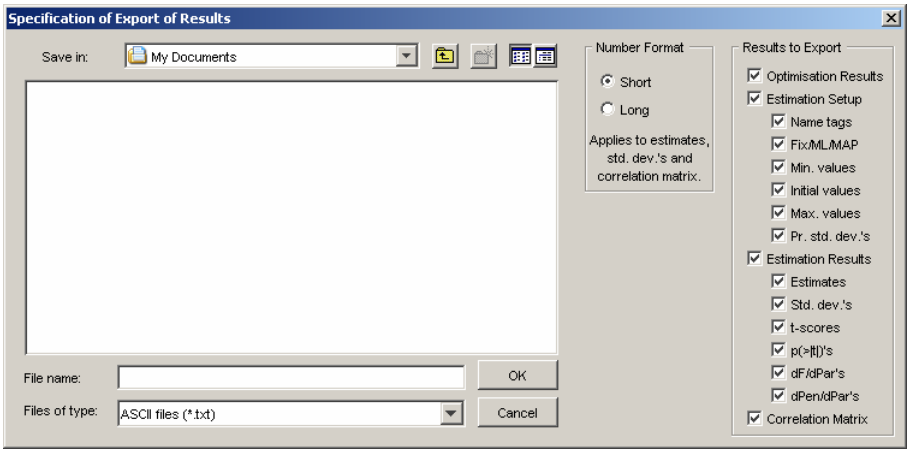


Figure 3.11. The Specification of Export of Results dialog.

> **CTSM** exports results in three different file formats: Tab delimited ASCII files (`*.txt`), semi-colon delimited CSV files (`*.csv`) and appropriately formatted LaTeX files (`*.tex`).

3.3.7 Deleting estimation results

The results of an estimation, i.e. the contents of a result set, can be permanently deleted by selecting the Remove selected result set action.

3.4 Generating validation data

Having estimated the initial states and parameters, you have the possibility of generating validation data for performing residual analysis.

> **CTSM** saves validation data in semi-colon delimited CSV files (`*.csv`). Data files of this type can easily be imported into MATLAB, S-Plus and standard spreadsheet programs.

3.4.1 Generating pure simulation data

Pure simulation data, i.e. state and output estimates based only on the inputs in a given data file (which must, however, also include outputs) can be generated for a given set of initial states and parameters by selecting the Generate pure simulation data action, which will make the Specification of Data for Validation dialog appear (see Figure 3.12), so you can select the data file, you want to use (see Section 3.3.2 for details about specifying a data file). The state and output estimates generated are $\hat{\boldsymbol{x}}_{k|0}$ and $\hat{\boldsymbol{y}}_{k|0}$, along with their standard deviations $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|0}) = \sqrt{\mathrm{diag}(\boldsymbol{P}_{k|0})}$ and $\mathrm{SD}(\hat{\boldsymbol{y}}_{k|0}) = \sqrt{\mathrm{diag}(\boldsymbol{R}_{k|0})}$, corresponding to each time instant t_k in the data file (see Section 2.1). Apart from specifying the data file, you must also specify whether to use Constant or Varying sample time and whether to use a Zero order hold or a First order hold on the inputs between sample instants (see Section 3.3.2). When all necessary information has been provided, press OK to proceed. Once the pure simulation data set has been generated, you must specify an output file for saving the data.

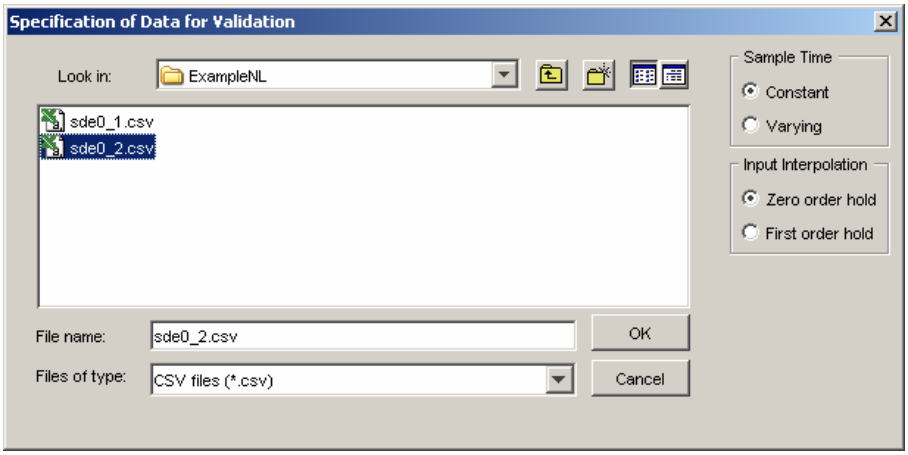


Figure 3.12. The Specification of Data for Validation dialog.

CTSM saves pure simulation data in semi-colon delimited CSV files (`*.csv`) with $1 + 2n + 2l$ columns. Within each row the column entries will be separated by semi-colons '**;**' and appear in the order: t_k, $\hat{\boldsymbol{x}}_{k|0}$, $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|0})$, $\hat{\boldsymbol{y}}_{k|0}$ and $\mathrm{SD}(\hat{\boldsymbol{y}}_{k|0})$. For models with more than one state and/or more than one output the state and output estimates will appear in the order they were typed in.

3.4.2 Generating prediction data

Prediction data, i.e. state and output estimates based on the inputs and lagged values of the outputs in a given data file can be generated for a given set of initial states and parameters by selecting the Generate prediction data action, which will make the Specification of Data for Validation dialog appear (see Figure 3.12), so you can select the data file, you want to use (see Section 3.3.2 for details about specifying a data file). The state and output estimates generated are $\hat{\boldsymbol{x}}_{k|k-j}$, $j \geq 1$, and $\hat{\boldsymbol{y}}_{k|k-j}$, $j \geq 1$, along with their standard deviations $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|k-j}) = \sqrt{\mathrm{diag}(\boldsymbol{P}_{k|k-j})}$ and $\mathrm{SD}(\hat{\boldsymbol{y}}_{k|k-j}) = \sqrt{\mathrm{diag}(\boldsymbol{R}_{k|k-j})}$, corresponding to each time instant t_k in the data file (see Section 2.1). Apart from specifying the data file, you must also specify j, whether to use Constant or Varying sample time and whether to use a Zero order hold or a First order hold on the inputs between sample instants (see Section 3.3.2). When all necessary information has been provided, press OK to proceed. Once the prediction data set has been generated, you must specify an output file for saving the data.

> **CTSM** saves prediction data in semi-colon delimited CSV files (`*.csv`) with $1 + 2n + 2l$ columns. Within each row the column entries will be separated by semi-colons '**;**' and appear in the order: t_k, $\hat{\boldsymbol{x}}_{k|k-j}$, $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|k-j})$, $\hat{\boldsymbol{y}}_{k|k-j}$ and $\mathrm{SD}(\hat{\boldsymbol{y}}_{k|k-j})$. For models with more than one state and/or more than one output the state and output estimates will appear in the order they were typed in.

3.4.3 Generating filtering data

Filtering data, i.e. state estimates based on the inputs and non-lagged values of the outputs in a given data file can be generated for a given set of initial states and parameters by selecting the Generate filtering data action, which will make the Specification of Data for Validation dialog appear (see Figure 3.12), so you can select the data file, you want to use (see Section 3.3.2 for details about specifying a data file). The state estimates generated are $\hat{\boldsymbol{x}}_{k|k}$, along with their standard deviations $\mathrm{SD}(\hat{\boldsymbol{x}}_{k|k}) = \sqrt{\mathrm{diag}(\boldsymbol{P}_{k|k})}$, corresponding to each time instant t_k in the data file (see Section 2.1). Apart from specifying the data file, you must also specify to the program whether to use Constant or Varying sample time and whether to use a Zero order hold or a First order hold on the inputs between sample instants (see Section 3.3.2). When all necessary information has been provided, press OK to proceed. Once the filtering data set has been generated, you must specify an output file for saving the data.

CTSM saves filtering data in semi-colon delimited CSV files (`*.csv`) with $1 + 2n$ columns. Within each row the column entries will be separated by semi-colons '**;**' and appear in the order: t_k, $\hat{\boldsymbol{x}}_{k|k}$ and $\text{SD}(\hat{\boldsymbol{x}}_{k|k})$. For models with more than one state the estimates will appear in the order they were typed in.

3.4.4 Generating smoothing data

For nonlinear models, smoothing data, i.e. state estimates based on all of the inputs and all of the outputs in a given data file can be generated for a given set of initial states and parameters by selecting the Generate smoothing data action, which will make the Specification of Data for Validation dialog appear (see Figure 3.12), so you can select the data file, you want to use (see Section 3.3.2 for details about specifying a data file). The state estimates generated are $\hat{\boldsymbol{x}}_{k|N}$, along with their standard deviations $\text{SD}(\hat{\boldsymbol{x}}_{k|N}) = \sqrt{\text{diag}(\boldsymbol{P}_{k|N})}$, corresponding to each time instant t_k in the data file (see Section 2.1). Apart from specifying the data file, you must also specify to the program whether to use Constant or Varying sample time and whether to use a Zero order hold or a First order hold on the inputs between sample instants (see Section 3.3.2). When all necessary information has been provided, press OK to proceed. Once the smoothing data set has been generated, you must specify an output file for saving the data.

CTSM saves smoothing data in semi-colon delimited CSV files (`*.csv`) with $1 + 2n$ columns. Within each row the column entries will be separated by semi-colons '**;**' and appear in the order: t_k, $\hat{\boldsymbol{x}}_{k|N}$ and $\text{SD}(\hat{\boldsymbol{x}}_{k|N})$. For models with more than one state the estimates will appear in the order they were typed in.

3.5 Changing the settings for a model

Using the Settings menu, it is possible to control a number of the computational features within **CTSM**, which may be useful to obtain better results.

3.5.1 Changing filter settings

Selecting the Filter ... menu item brings up the Filter Settings dialog (see Figure 3.13), which holds the controls for the (iterated extended) Kalman filter part of **CTSM** (see the Mathematics Guide for details).

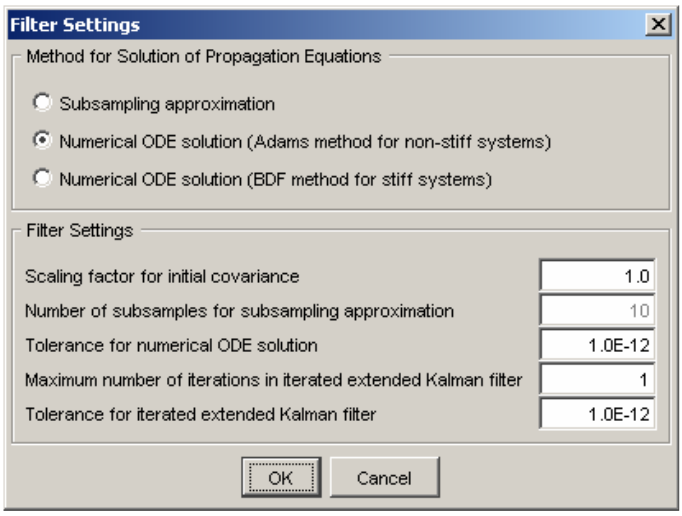


Figure 3.13. The Filter Settings dialog.

LTI models

For LTI models, only the Scaling factor for initial covariance (default: 1.0) is displayed. It is used in the calculation of the covariance of the initial states.

> The larger the scaling factor, the more uncertain the initial states.

LTV models

For LTV models, the Number of subsamples in iterated extended Kalman filter (default: 10) is displayed in addition to the above scaling factor. This is the number of subsamples used in the subsampling approximation to the true solution of the propagation equations in the iterated extended Kalman filter.

> The more subsamples, the more accurate the approximation.

NL models

For NL models, all settings are displayed. The top panel of the dialog provides a possibility for choosing which method to apply for solving the propagation equations of the iterated extended Kalman filter: 1) Subsampling approximation, 2) numerical ODE solution with the Adams method for non-stiff systems (default) or 3) numerical ODE solution with the BDF method for stiff systems.

In the lower panel the Tolerance for numerical ODE solution (default: 1.0E-12) is the tolerance used by the ODE solvers in the iterated extended Kalman filter.

> The lower the tolerance, the more accurate the ODE solution.

The Maximum number of iterations in iterated extended Kalman filter (default: 1) is the maximum number of iterations performed to reduce the effects of measurement equation nonlinearities in the iterated extended Kalman filter, and the Tolerance for iterated extended Kalman filter (default: 1.0E-12) is the corresponding tolerance. The measurement equation is iterated until the tolerance is met or until the maximum number of iterations is reached.

> The more iterations and the lower the tolerance, the better.

3.5.2 Changing optimisation settings

Selecting the Optimisation ... menu item brings up the Optimisation Settings dialog (see Figure 3.14), which holds the basic controls for the optimisation part of **CTSM** (see the Mathematics Guide for details).

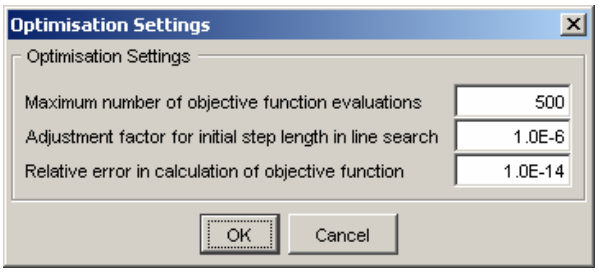


Figure 3.14. The Optimisation Settings dialog.

The Maximum number of objective function evaluations (default: 500) is simply a limit on the number of objective function evaluations, the Adjustment factor for initial step length in line search (default: 1.0E-6) is a value used to adjust the initial step length in the line search, and the Relative error in calculation of objective function (default: 1.0E-14) is a value used to determine a step size for the finite difference gradient approximations within the optimisation algorithm.

> There is usually no need to adjust any of these values.

3.5.3 Changing advanced settings

Selecting the Advanced ... menu item brings up the Advanced Settings dialog (see Figure 3.15), which holds the advanced controls for the optimisation part of **CTSM** and controls for some computational settings (see the Mathematics Guide for details). The Cut-off value for Huber's psi-function (default: 3.0) is the constant c in Huber's ψ-function. The Padé approximation order (default: 6) is the order of the Padé approximation used to compute matrix exponentials. The Tolerance for singular value decomposition (default: 1.0E-12) is a value used to determine if the $\boldsymbol{A}$ matrix is singular or not. The Lagrange multiplier in penalty function (default: 1.0E-4) is the λ value used in the penalty function. The Minimum absolute value used for normalizing in penalty function is a value used to ensure numerical stability in calculation of the penalty function.

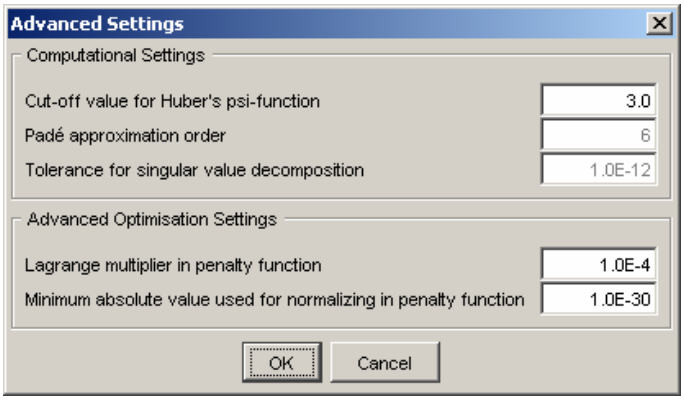


Figure 3.15. The Advanced Settings dialog.

> There is usually no need to adjust any of these values.

3.6 Modifying the structure of a model

Using the Add/remove inputs, Add/remove outputs, Add/remove states and Add/remove algebraic equations actions, the structure of a model can be modified. As an example, Figure 3.16 shows the Add/remove inputs dialog that appears when the Add/remove inputs action is selected. Using this dialog, existing input variables can be eliminated and new input variables can be added.

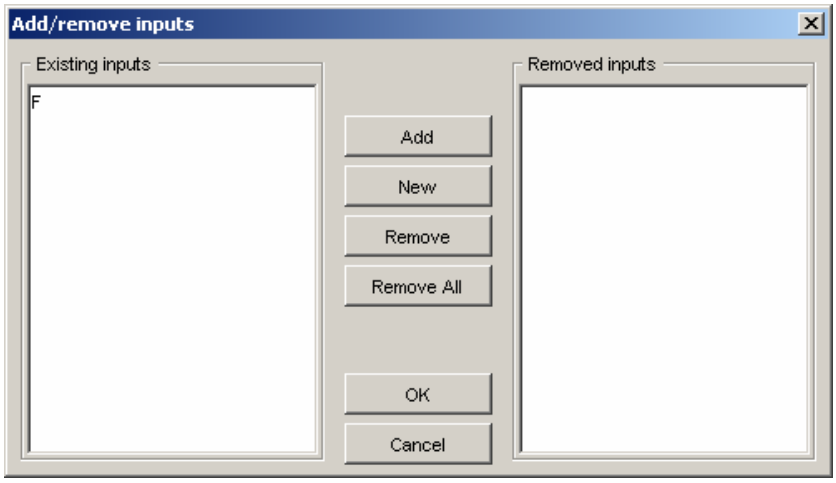


Figure 3.16. The Add/remove inputs dialog.

3.7 Saving and opening a model

To save a model, you simply select the Save model action, which will allow you to specify an output file for saving the current model.

> **CTSM** saves models in binary files of type `*.ctsm`. When a model is saved, the current state of the model equations, the current specifications under the Parameters tab, all result sets and the current settings are saved. Model files are platform independent, so the same model file can be used on both Linux, Solaris and Windows.

To open a model, you simply select the Open model action, which will allow you to select the file with the model you want to open.

> **IMPORTANT:** Models created in **CTSM 2.0**, in **CTSM 2.1** and in **CTSM 2.2** are unfortunately incompatible with **CTSM 2.3**. Trying to open such models will result in an error.

4

Tips and tricks

This chapter provides some tips for making **CTSM** run more smoothly and some tricks and workarounds for handling complicated modelling tasks.

4.1 Tips for making the estimation run smoothly

To avoid numerical problems and to ensure an appropriate level of accuracy when performing estimations in **CTSM**, the following tips may be useful.

4.1.1 Scaling of variables

Numerical problems may be avoided by using appropriately scaled values of the inputs, outputs and states of the model. Therefore, if possible, these variables should have physical units that confine their numerical values to the interval $[-1, 1]$. If this is not possible, scaled or normalized variables should be used.

4.1.2 Selecting appropriate parameter values

To ensure an appropriate level of accuracy of the estimation results (especially to obtain accurate standard deviations and an accurate correlation matrix), the Number of iterations should not be too low. What too low means depends on the particular model, but as a rule of thumb the Number of iterations should be above 20. The Number of iterations can be influenced by appropriatelty selecting the Minimum values, Initial values and Maximum values of the initial states and parameters to be estimated, e.g. in accordance with these tips:

1. Use a non-zero Initial value if possible.
2. If you know the "true" value, use a different Initial value.
3. Do not set the Minimum value and Maximum value too close to each other.
4. Do not set the Minimum value and Maximum value too far from each other.

Another tip that may be useful is that it is sometimes a good idea to let the program estimate all of the initial states and parameters instead of fixing some of them. In particular, it is usually more robust to estimate the initial states.

4.1.3 Data issues

CTSM has built-in features for automatically handling varying sample times, occasional outliers and missing observations in the data sets used for estimation.

With respect to the latter, the program recognizes the number $1E300$ (or larger) as a missing observation. When a missing observation is reached the (iterated extended) Kalman filter predicts its value based on previous observations and the calculation of the objective function is modified accordingly. If, however, there are missing input values in the data sets it is more difficult. In this case it is necessary to fill in the missing input values before running **CTSM**.

4.2 Various tricks and workarounds

CTSM has certain limitations with respect to introducing discontinuous functionalities in a model, because Dirac's δ-function, Heaviside's unit function and other threshold-based model components cannot be implemented directly, but these limitations may be circumvented with one of the following workarounds.

4.2.1 A general trick for modelling thresholds

The workaround for modelling thresholds consists of replacing a sharp threshold with an equivalent smooth threshold, i.e. by replacing the threshold model:

$$f(\boldsymbol{z}) = \begin{cases} f_1(\boldsymbol{z}) & , g(\boldsymbol{z}) \leq a \\ f_2(\boldsymbol{z}) & , g(\boldsymbol{z}) > a \end{cases} \tag{4.1}$$

with the corresponding smooth threshold model:

$$f(\boldsymbol{z}) = \frac{f_1(\boldsymbol{z})}{1 + \exp(-\gamma(a - g(\boldsymbol{z})))} + \frac{f_2(\boldsymbol{z})}{1 + \exp(\gamma(a - g(\boldsymbol{z})))} \tag{4.2}$$

where the smoothness is introduced by means of two sigmoidal functions. The γ parameter is used to tune to sharpness of the threshold. The larger γ, the sharper threshold. In the above formulation $\boldsymbol{z}$ is a vector variable, signifying that, in general, thresholds depending on the states $\boldsymbol{x}$, the inputs $\boldsymbol{u}$ and the parameters $\boldsymbol{\theta}$ as well as the time variable t can be modelled this way. A special case of (4.1) depending only on the time variable t is Heaviside's unit function:

$$f(t) = \begin{cases} 0 & , t \leq 0 \\ 1 & , t > 0 \end{cases} \tag{4.3}$$

which can be modelled by replacing it with:

$$f(t) = \frac{1}{1 + \exp(-\gamma t)} \tag{4.4}$$

which is a special case of (4.2). In a similar fashion, simple thresholds depending on other variables can be modelled. Replacing sharp thresholds with smooth approximations may affect the accuracy of the results, but the only other danger in doing this is the effect of a very large value of γ on numerical stability.

4.2.2 Tricks for modelling pulses and steps in inputs

A workaround that can be used for modelling pulses and steps in one or more of the inputs $\boldsymbol{u}$ is to modify the corresponding columns in the data files.

Modelling a single pulse in a specific input

- Start by setting all values in the column corresponding to the specific input in the data file equal to 0.
- If there are not already rows in the data file that correspond to the time of the beginning and the end of the pulse, add new lines that do.
- Fill in the value of the size of the pulse in the beginning row and in all rows between the beginning and end rows (not in the end row itself).
- If you added new lines, fill in the values for the other inputs in these rows as well (you may have to use interpolation), and remember to mark all the corresponding outputs as missing ($1E300$).
- Finally, remember to use Zero order hold interpolation between inputs when performing the estimation to correctly obtain the desired pulse.

As an example, the following regularly sampled data file for a two-input, two-output system has a pulse of size 1 between time 2.5 and 3.5 in the first input:

```
0;     0;   1;      0.1;     0.1
1;     0;   0.5;    0.2;     0.1
2;     0;   0;      0.3;     0.1
2.5;   1;   0.25;   1E300;   1E300
3;     1;   0.5;    0.1;     0.2
3.5;   0;   0.75;   1E300;   1E300
4;     0;   1;      0.2;     0.3
5;     0;   0.5;    0.3;     0.3
6;     0;   0;      0.4;     0.3
```

Modelling a single step in a specific input

- Start by setting all values previous to the step in the column corresponding to the specific input in the data file equal to 0.
- If there is not already a row in the data file that corresponds to the time of the step, add a new line that does.
- Fill in the value of the size of the step in this and all subsequent rows.
- If you added a new line, fill in the values for the other inputs in this row as well (you may have to use interpolation), and remember to mark all the corresponding outputs as missing ($1E300$).
- Finally, remember to use Zero order hold interpolation between inputs when performing the estimation to correctly obtain the desired step.

As an example, the following regularly sampled data file for a two-input, two-output system has a step of size 1 at time 2.5 in the first input:

```
0;     0;   1;      0.1;     0.1
1;     0;   0.5;    0.2;     0.1
2;     0;   0;      0.3;     0.1
2.5;   1;   0.25;   1E300;   1E300
3;     1;   0.5;    0.2;     0.3
4;     1;   1;      0.3;     0.3
5;     1;   0.5;    0.4;     0.3
6;     1;   0;      0.5;     0.3
```

5

Troubleshooting

This chapter focuses on what to do if problems occur while running **CTSM**.

5.1 Common error messages

This section explains the error messages that may occur while running **CTSM** and seeks to provide information about how to avoid the corresponding errors.

5.1.1 GUI input errors

All input to **CTSM** provided via the dialogs for New Model Specification, Specification of Data for Estimation, Specification of Export of Results, Specification of Data for Validation, Filter Settings, Optimisation Settings and Advanced Settings is checked immediately upon being typed in, and all errors are reported via error messages intended to be selfexplanatory. The latter also applies to errors in the typed in model equations revealed when the model is analyzed.

5.1.2 Computational errors

Within **CTSM**'s computational code, i.e. within the code used for estimation or generation of validation data, a number of errors may occur due to misspecification or numerical problems, some of which are more common than others.

Common errors

The following computational errors are the most common:

> Unable to calculate matrix exponential!

To solve this problem (within the iterated extended Kalman filter), try increasing the Number of subsamples in iterated extended Kalman filter (see Section 3.5.1).

> Unable to perform numerical ODE solution!

To solve this problem (within the iterated extended Kalman filter), try decreasing the Tolerance for numerical ODE solution (see Section 3.5.1).

> The maximum number of objective function evaluations has been exceeded!

> The maximum objective function value (1e300) has been exceeded!

> The measurement noise covariance matrix is not positive definite!

> The state covariance matrix is not positive definite!

To solve these problems (with optimisation), try different Initial value's and different limits on some of the initial states and parameters (see Section 3.3.1).

Uncommon errors

The following computational errors are much less common:

> The amount of data available is insufficient to perform the estimation!

To solve this problem, provide a data file with more observations.

> The prior covariance matrix is not positive definite!

To solve this specification problem, make sure the Prior Correlation Matrix and the Prior std. dev.'s have been correctly specified (see Section 3.3.1).

> Unable to determine reciprocal condition number!

> Unable to compute singular value decomposition!

> Unable to solve system of linear equations!

To solve these problems (with linear algebra), try different Initial value's and maybe different limits on some of the initial states and parameters (see Section 3.3.1).

5.1.3 Exceptions

In addition to the above errors, a number of exceptions may be triggered if something unexpected happens within the program's code. If an exception occurs, **CTSM** displays an error message explaining in which part of the code the exception occured. If you experience this, you are kindly asked to file a bug report, which describes, if possible, what made the exception occur.

5.2 Frequently asked questions

Other reports on problems with **CTSM**, and how they may be solved, can be found in the FAQ section of the program homepage:

`http://www.imm.dtu.dk/ctsm`

Appendices

A

Example: LTI model of the heat dynamics of a wall

This appendix provides an example of using **CTSM** for estimating parameters in an LTI model of the heat dynamics of a wall. The model file generated in this example and the two data files used can be found in a subdirectory of the `documentation` directory (a subdirectory of the main directory `CTSM23`). The files are also available for download on the program homepage:

`http://www.imm.dtu.dk/ctsm`

A.1 Model equations

Imagine that we want to estimate the parameters in a simple model of the heat dynamics of a wall. The system equation of the model is:

$$\begin{aligned} d\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = (& \begin{bmatrix} -\frac{1}{G_1}\left(\frac{1}{H_1}+\frac{1}{H_2}\right) & \frac{1}{G_1 H_2} \\ \frac{1}{G_2 H_2} & -\frac{1}{G_2}\left(\frac{1}{H_2}+\frac{1}{H_3}\right) \end{bmatrix} \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} \\ & + \begin{bmatrix} \frac{1}{G_1 H_1} & 0 \\ 0 & \frac{1}{G_2 H_3} \end{bmatrix} \begin{pmatrix} T_e \\ T_i \end{pmatrix}) \, dt + \begin{bmatrix} \sigma_{11} & 0 \\ 0 & \sigma_{22} \end{bmatrix} d\boldsymbol{\omega}_t \end{aligned} \tag{A.1}$$

where the state variables T_1 and T_2 are the outer wall temperature and the inner wall temperature, the input variables T_e and T_i are the outdoor temperature and the indoor temperature, and G_1, G_2, H_1, H_2 and H_3 are parameters of the thermal network describing the wall. The measurement equation is:

$$(q_i)_k = \begin{bmatrix} 0 & -\frac{1}{H_3} \end{bmatrix} \begin{pmatrix} T_1 \\ T_2 \end{pmatrix}_k + \begin{bmatrix} 0 & \frac{1}{H_3} \end{bmatrix} \begin{pmatrix} T_e \\ T_i \end{pmatrix}_k + e_k \tag{A.2}$$

where the output variable (q_i) is the heat flux and $e_k \in N(0, S)$. The true parameter values used for generating the data we will use are given in Table A.1.

G_1	G_2	H_1	H_2	H_3	σ_{11}	σ_{22}	S
100	50	1	2	0.5	0	0	0.01

Table A.1. True values of the parameters of the wall heat dynamics model.

A.2 Setting up the model

We begin our work by starting up **CTSM** (see Section 3.1).

A.2.1 Creating a new model

We then select the New model action, provide the appropriate information (i.e. that the model is linear time invariant and has 2 inputs, 1 output and 2 states) in the New Model Specification dialog and press OK (see Figure A.1).

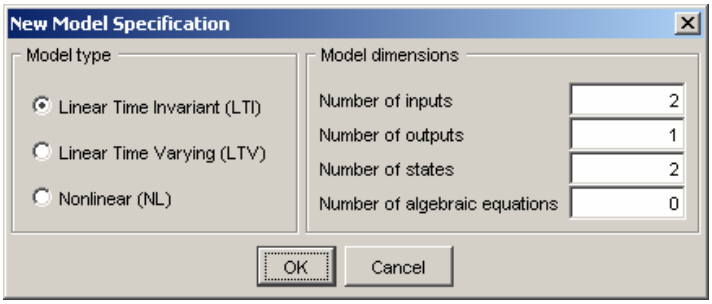


Figure A.1. The New Model Specification dialog with values assigned.

A.2.2 Typing in the model equations

The GUI now appears as in Figure A.2 and we are ready to type in the model equations. After doing this, the GUI appears as in Figure A.3.

A.2.3 Analyzing the model equations

We are now ready to let **CTSM**'s interpreter analyze the model equations, so we select the Analyze model action. When analyzing the model equations the interpreter first checks for compliance with the rules for typing in model equations (see Section 3.2.2) and then determines the symbolic names of the parameters of the model, which are then displayed along with the symbolic names of the initial states under the Parameters tab as in Figure A.4.

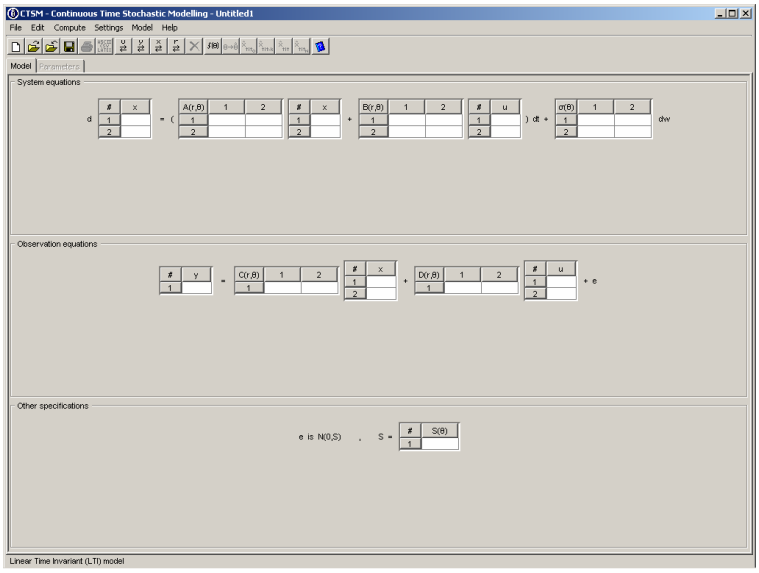


Figure A.2. GUI when the new model has been created.

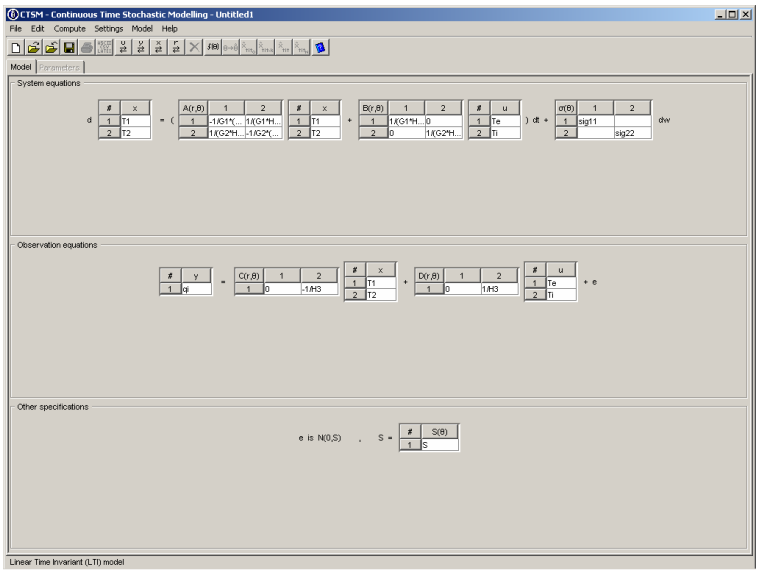


Figure A.3. GUI when the model equations have been typed in.

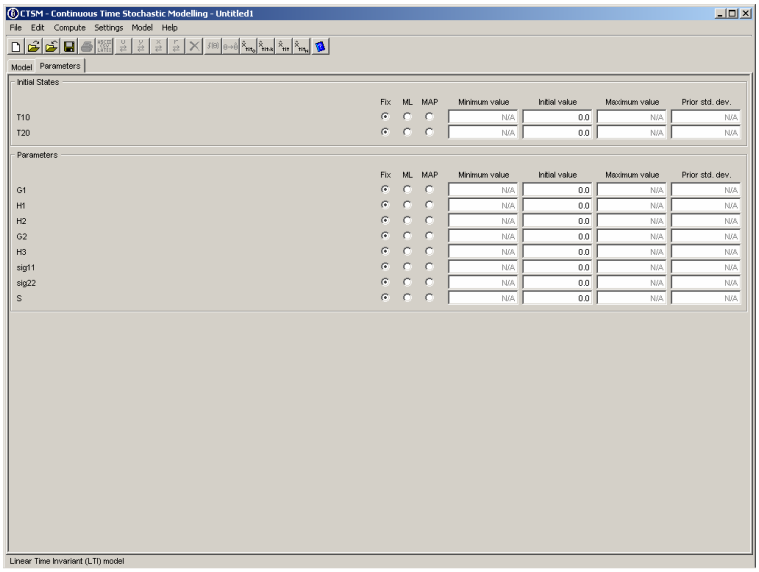


Figure A.4. GUI when the model has been analyzed.

As expected, **CTSM** finds 2 initial states and 8 parameters in the model.

A.3 Estimating parameters

We are now ready to set up an estimation problem.

A.3.1 Specifying estimation method

To set up an estimation problem, we have to specify an estimation method for each of the initial states and parameters (see Section 3.3.1). After we have finished doing this, the GUI appears as in Figure A.5.

A.3.2 Specifying estimation data

We are now ready to start the estimation, so we select the Estimate parameters action and the dialog for Specification of Data for Estimation appears (see Figure A.6), and we can select the data file(s), we want to use for the estimation. Apart from specifying data file(s), we must provide some additional information for the program as shown in Section 3.3.2. We want to use the data file `ltidata.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that and press OK, and the computation starts.

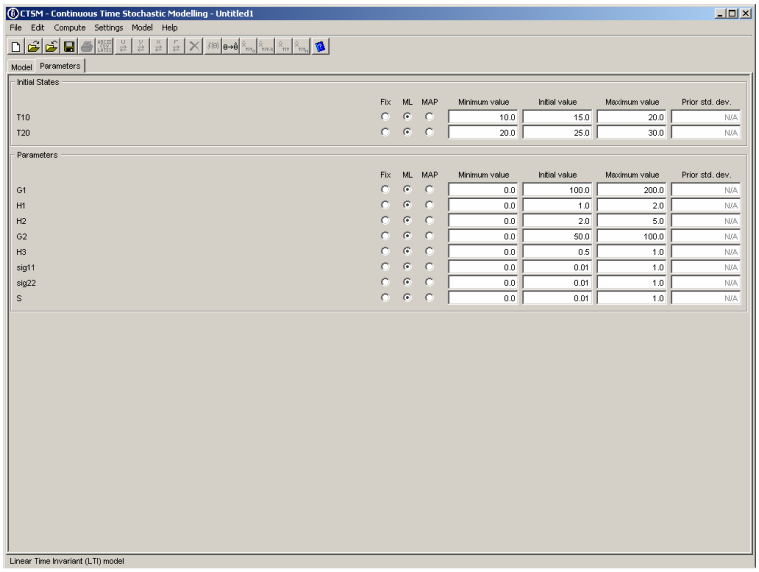


Figure A.5. GUI when estimation methods have been specified.

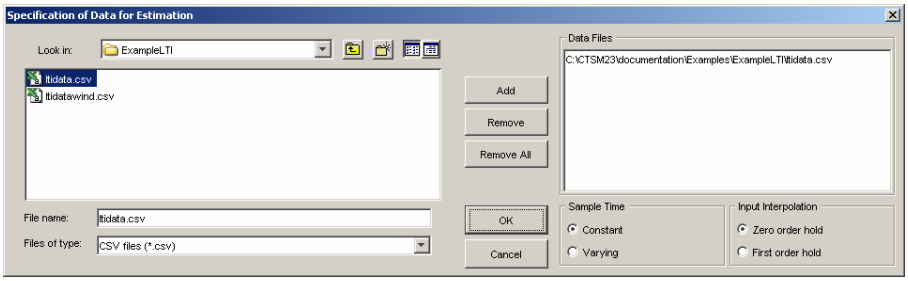


Figure A.6. The Specification of Data for Estimation dialog.

A.3.3 Monitoring computation progress

When the computation has commenced, the GUI appears as in Figure A.7, and we can monitor the progress of the computation as shown in Section 3.3.3.

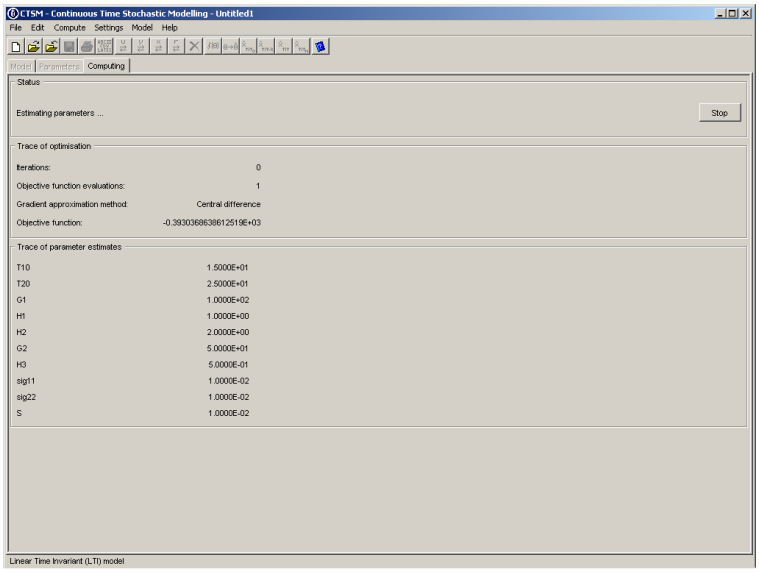


Figure A.7. GUI when the computation has commenced.

A.3.4 Interpreting estimation results

Once the computation is complete, and the initial states and parameters have been succesfully estimated, the GUI appears as in Figure A.8 and we are ready to take a look at the results (see Section 3.3.4 for details about interpreting results). Inspecting the results of the estimation we have performed, there does not seem to be any particular need for changing optimisation settings or loosening any limits on the initial states or parameters to try to obtain more accurate results. Furthermore, comparing the Estimates with the true values listed in Table A.1, we see that they are in very good agreement.

A.4 Generating validation data

Having estimated the initial states and parameters, we have the possibility of generating validation data for performing residual analysis.

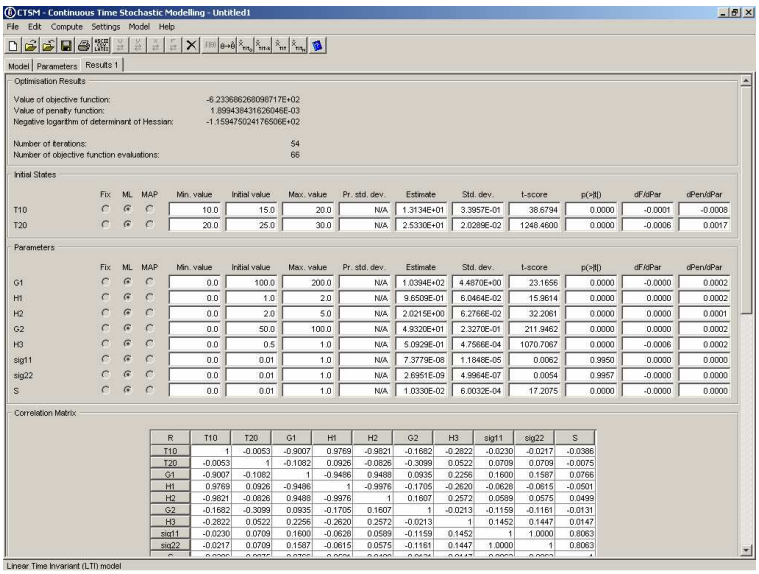


Figure A.8. GUI when the initial states and parameters have been estimated.

A.4.1 Generating pure simulation data

To demonstrate how to generate pure simulation data, we select the Generate pure simulation data action and the dialog for Specification of Data for Validation appears (see Figure A.9), and we can select the data file, we want to use (see Section 3.3.2 for details about specifying a data file). We want to use the data file `ltidatawind.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that, press OK, and computation starts.

A.4.2 Generating prediction data

To demonstrate how to generate prediction data, we select the Generate prediction data action and the dialog for Specification of Data for Validation appears (see Figure A.9), and we can select the data file, we want to use (see Section 3.3.2 for details about specifying a data file). We want to use the data file `ltidatawind.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that, press OK, and computation starts.

A.4.3 Generating filtering data

To demonstrate how to generate filtering data, we select the Generate filtering data action and the dialog for Specification of Data for Validation appears

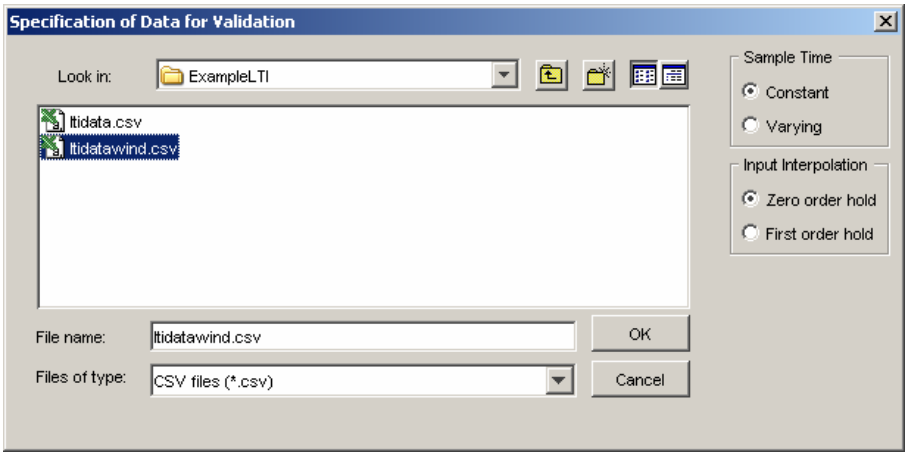


Figure A.9. The Specification of Data for Validation dialog.

(see Figure A.9), and we can select the data file, we want to use (see Section 3.3.2 for details about specifying a data file). We want to use the data file `ltidatawind.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that, press OK, and computation starts.

A.5 Saving the model

We now want to finish working with the model, but before we do that, we would like to save it, so we can use it later on. We do that by selecting the Save model action, which allows us to specify an output file for saving the model.

We save the model in the file `lti.ctsm`, and if we want to work with it again, we can re-open it by selecting the Open model action.

B

Example: NL model of a fed-batch bioreactor

This appendix provides an example of using **CTSM** for estimating parameters in an NL model of a fed-batch bioreactor. The model file generated in this example and the two data files used can be found in a subdirectory of the `documentation` directory (a subdirectory of the main directory `CTSM23`). The files are also available for download on the program homepage:

`http://www.imm.dtu.dk/ctsm`

B.1 Model equations

Imagine that we want to estimate the parameters in a simple model of a fed-batch bioreactor. The system equation of the model is:

$$d\begin{pmatrix} X \\ S \\ V \end{pmatrix} = \begin{pmatrix} \mu(S)X - \frac{FX}{V} \\ -\frac{\mu(S)X}{Y} + \frac{F(S_F - S)}{V} \\ F \end{pmatrix} dt + \begin{bmatrix} \sigma_{11} & 0 & 0 \\ 0 & \sigma_{22} & 0 \\ 0 & 0 & \sigma_{33} \end{bmatrix} d\boldsymbol{\omega}_t \tag{B.1}$$

where the states X, S and V are the biomass concentration, the substrate concentration and the volume, the input F is the feed flow rate, Y is a yield coefficient and S_F is the feed concentration. The growth rate $\mu(S)$ is:

$$\mu(S) = \mu_{\max} \frac{S}{K_2 S^2 + S + K_1} \tag{B.2}$$

where $\mu_{\max}$, K_1 and K_2 are kinetic parameters. The measurement equation is:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}_k = \begin{pmatrix} X \\ S \\ V \end{pmatrix}_k + \boldsymbol{e}_k \ , \quad \boldsymbol{e}_k \in N(\boldsymbol{0}, \boldsymbol{S}) \ , \quad \boldsymbol{S} = \begin{bmatrix} S_{11} & 0 & 0 \\ 0 & S_{22} & 0 \\ 0 & 0 & S_{33} \end{bmatrix} \tag{B.3}$$

where y_1, y_2 and y_3 are outputs. The true parameter values used for generating the data we will be using for the estimation are given in Table B.1.

$\mu_{\max}$	K_1	K_2	Y	S_F	σ_{11}	σ_{22}	σ_{33}	S_{11}	S_{22}	S_{33}
1	0.03	0.5	0.5	10	0	0	0	0.01	0.001	0.01

Table B.1. True values of the parameters of the fed-batch bioreactor model.

B.2 Setting up the model

We begin our work by starting up **CTSM** (see Section 3.1).

B.2.1 Creating a new model

We then select the New model action, provide the appropriate information (i.e. that the model is nonlinear and has 1 input, 3 outputs and 3 states) in the New Model Specification dialog and press OK (see Figure B.1).

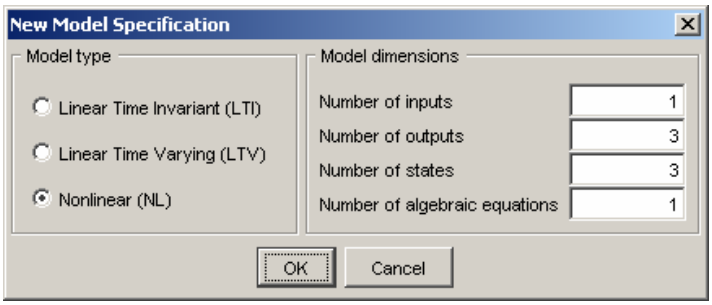


Figure B.1. The New Model Specification dialog with values assigned.

Note that we have also specified that we want to use 1 algebraic equation to avoid having to type in the expression for the growth rate $\mu(S)$ more than once.

B.2.2 Typing in the model equations

The GUI now appears as in Figure B.2 and we are ready to type in the model equations. After doing this, the GUI appears as in Figure B.3. Note that we have typed in the expression for the growth rate $\mu(S)$ on the right hand side of the algebraic equation and introduced the abbreviation mu on the left hand side. This abbreviation we have then used in place of $\mu(S)$ in the system equations of the model to reduce the amount of typing we have to do.

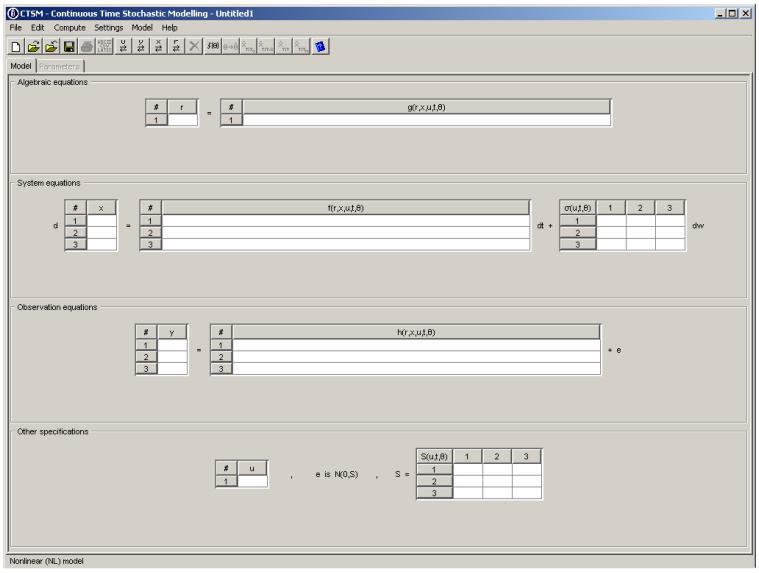


Figure B.2. GUI when the new model has been created.

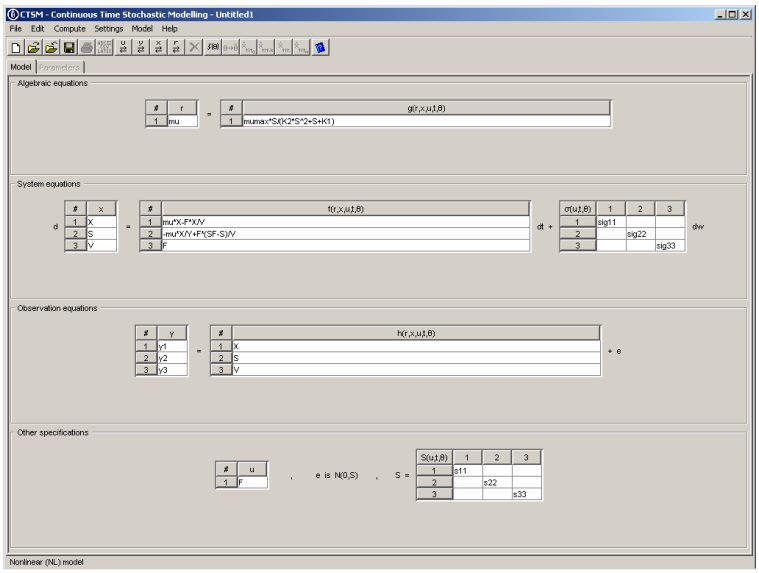


Figure B.3. GUI when the model equations have been typed in.

B.2.3 Analyzing the model equations

We are now ready to let **CTSM**'s interpreter analyze the model equations, so we select the Analyze model action. When analyzing the model equations the interpreter first checks for compliance with the rules for typing in model equations (see Section 3.2.2) and then determines the symbolic names of the parameters of the model, which are then displayed along with the symbolic names of the initial states under the Parameters tab as in Figure B.4.

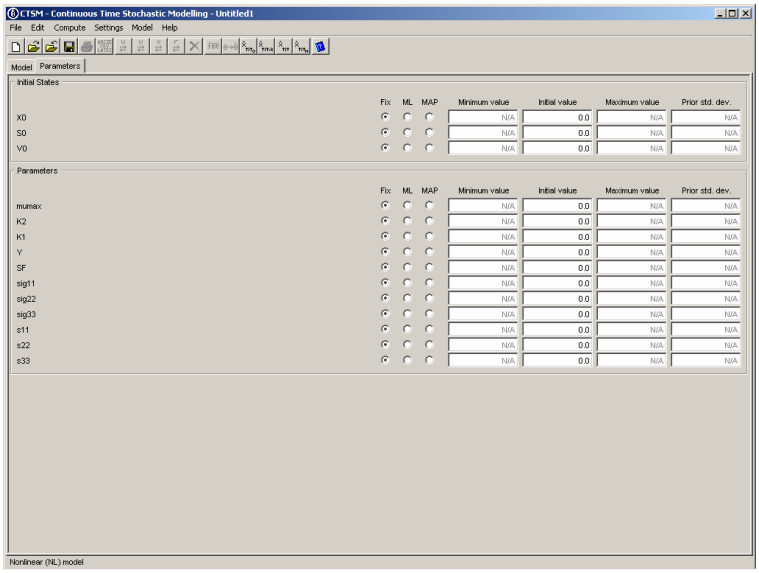


Figure B.4. GUI when the model has been analyzed.

As expected, **CTSM** finds 3 initial states and 11 parameters in the model.

B.3 Estimating parameters

We are now ready to set up an estimation problem.

B.3.1 Specifying estimation method

To set up an estimation problem, we have to specify an estimation method for each of the initial states and parameters (see Section 3.3.1). After we have finished doing this, the GUI appears as in Figure B.5.

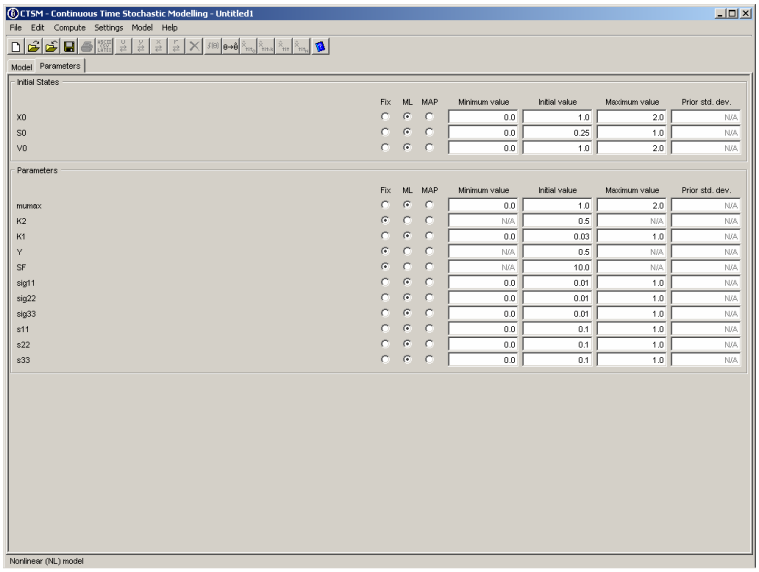


Figure B.5. GUI when estimation methods have been specified.

B.3.2 Specifying estimation data

We are now ready to start the estimation, so we select the Estimate parameters action and the dialog for Specification of Data for Estimation appears (see Figure B.6), and we can select the data file(s), we want to use for the estimation.

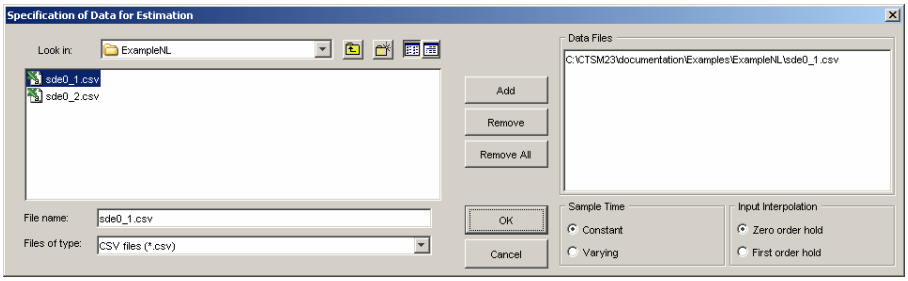


Figure B.6. The Specification of Data for Estimation dialog.

Apart from specifying data file(s), we must provide some additional information for the program as shown in Section 3.3.2. We only want to use the data file `sde0_1.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that and press OK, and the computation starts.

B.3.3 Monitoring computation progress

When the computation has commenced, the GUI appears as in Figure B.7, and we can monitor the progress of the computation as shown in Section 3.3.3.

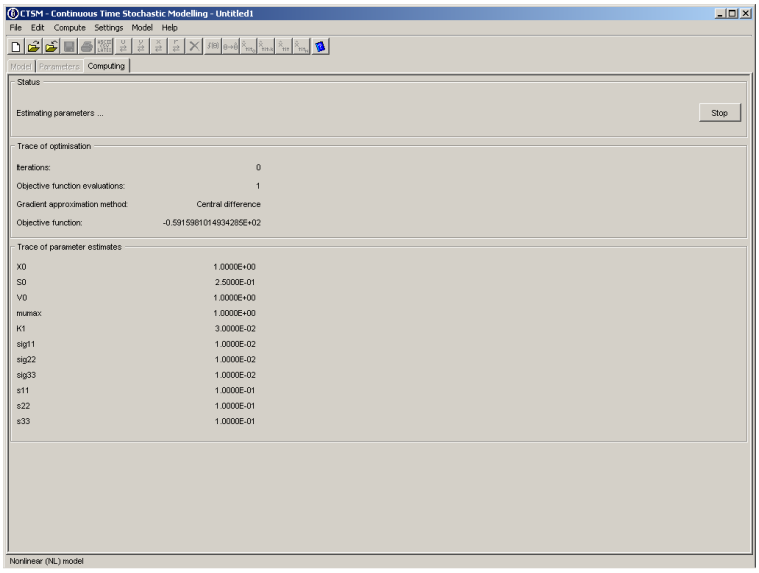


Figure B.7. GUI when the computation has commenced.

B.3.4 Interpreting estimation results

Once the computation is complete, and the initial states and parameters have been succesfully estimated, the GUI appears as in Figure B.8 and we are ready to take a look at the results (see Section 3.3.4 for details about interpreting results). Inspecting the results of the estimation we have performed, there does not seem to be any particular need for changing optimisation settings or loosening any limits on the initial states or parameters to try to obtain more accurate results. Furthermore, comparing the Estimates with the true values listed in Table B.1, we see that they are in very good agreement.

B.4 Generating validation data

Having estimated the initial states and parameters, we have the possibility of generating validation data for performing residual analysis.

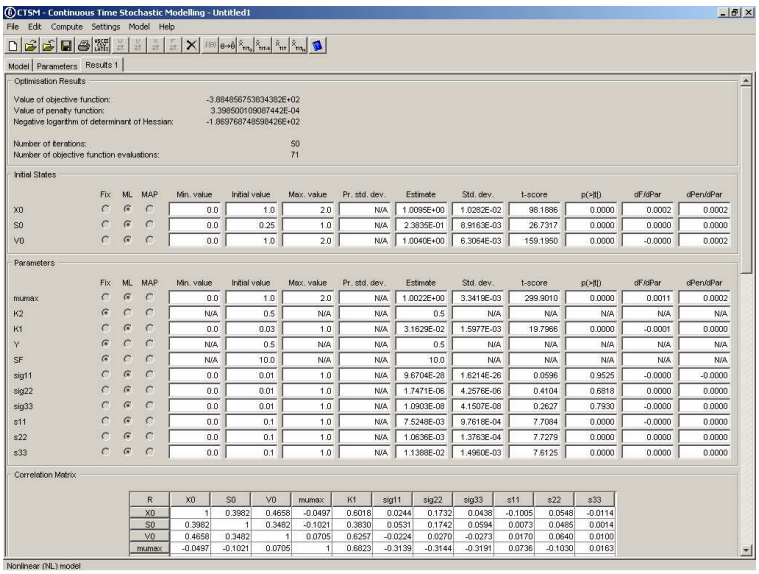


Figure B.8. GUI when the initial states and parameters have been estimated.

B.4.1 Generating pure simulation data

To demonstrate how to generate pure simulation data, we select the Generate pure simulation data action and the dialog for Specification of Data for Validation appears (see Figure B.9), and we can select the data file, we want to use (see Section 3.3.2 for details about specifying a data file). We want to use the data file `sde0_2.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that and press OK, and computation starts.

B.4.2 Generating prediction data

To demonstrate how to generate prediction data, we select the Generate prediction data action and the dialog for Specification of Data for Validation appears (see Figure B.9), and we can select the data file, we want to use (see Section 3.3.2 for details about specifying a data file). We want to use the data file `sde0_2.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that and press OK, and computation starts.

B.4.3 Generating filtering data

To demonstrate how to generate filtering data, we select the Generate filtering data action and the dialog for Specification of Data for Validation appears (see

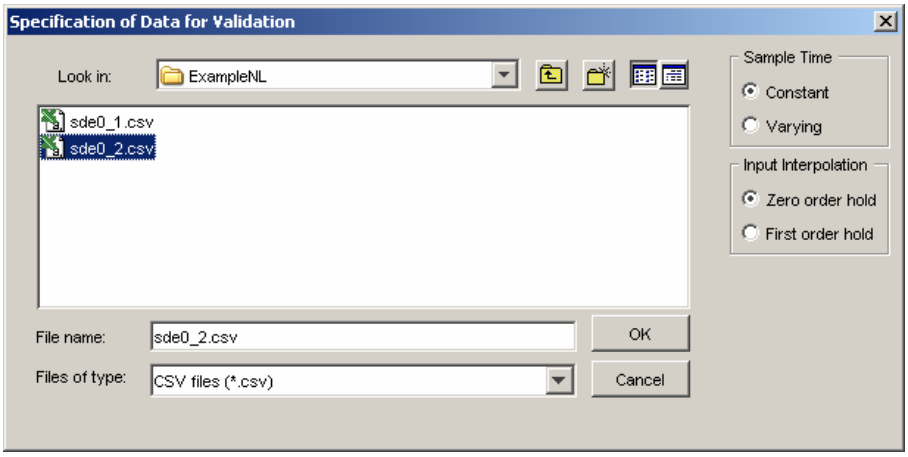


Figure B.9. The Specification of Data for Validation dialog.

Figure B.9), and we can select the data file, we want to use (see Section 3.3.2 for details about specifying a data file). We want to use the data file `sde0_2.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that and press OK, and computation starts.

B.4.4 Generating smoothing data

To demonstrate how to generate smoothing data, we select the Generate smoothing data action and the dialog for Specification of Data for Validation appears (see Figure B.9), and we can select the data file, we want to use (see Section 3.3.2 for details about specifying a data file). We want to use the data file `sde0_2.csv` with Constant sample time and Zero order hold interpolation between inputs, so we specify that and press OK, and computation starts.

B.5 Saving the model

We now want to finish working with the model, but before we do that, we would like to save it, so we can use it later on. We do that by selecting the Save model action, which allows us to specify an output file for saving the model.

We save the model in the file `nl.ctsm`, and if we want to work with it again, we can re-open it by selecting the Open model action.

References

Jacobsen, J. L. and Madsen, H. (1996). Grey Box Modelling of Oxygen Levels in a Small Stream. *Environmetrics*, **7**(1), 109–121.

Kristensen, N. R. (2002). *Fed-Batch Process Modelling for State Estimation and Optimal Control - A Stochastic Grey-Box Modelling Framework*. Ph.D. thesis, Technical University of Denmark, Lyngby, Denmark.

Madsen, H. and Holst, J. (1995). Estimation of Continuous-Time Models for the Heat Dynamics of a Building. *Energy and Buildings*, **22**, 67–79.

Madsen, H. and Melgaard, H. (1991). The Mathematical and Numerical Methods Used in CTLSM. Technical Report 7, IMM, Technical University of Denmark, Lyngby, Denmark.

Melgaard, H. and Madsen, H. (1993). CTLSM - A Program for Parameter Estimation in Stochastic Differential Equations. Technical Report 1, IMM, Technical University of Denmark, Lyngby, Denmark.

Tornøe, C. W. (2002). *Grey-Box PK/PD Modelling of Insulin*. Master's thesis, Technical University of Denmark, Lyngby, Denmark.