Implementation of Grey-Box Identification in JModelica.org

Elias Palmkvist

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
ISRN LUTFD2/TFRT--5941--SE
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Media-Tryck.
Lund 2014

Abstract

Grey-box identification is a tool to identify and improve nonlinear system models by estimating parameters. The estimation is done by optimizing a cost function using measurement data. The robustness of the estimations can then be analyzed with statistics. JModelica.org is a platform for modeling and optimization of dynamical models. In order to do grey-box identification one need models and be able to optimize. JModelica.org supports modeling and optimization so it has a huge potential to support grey-box identification. So far there is no complete solution for grey-box identification in JModelica.org. This work is focusing on how to implement grey-box identification in JModelica.org in order to estimate parameters for nonlinear models. The theory of grey-box identification has been investigated as well as the possibilities with JModelica.org. Finally, an interactive method to estimate model parameters and a method to calculate the confidence intervals for the estimates have been implemented. The implementation has been tested for nonlinear models and works as expected.

Acknowledgements

First of all, I would like thank my advisor Toivo Henningsson for the support. His insights and feedback have been extremely valuable during the work and he has always helped me as fast as I got stuck. This work would not have been completed without his support.

I would also like to thank Alf Isaksson from ABB for taking his time to share his experiences and knowledge in grey-box identification, Rolf Johansson at the department of Automatic Control at LTH for the support and feedback and Fredrik Magnusson at the department of Automatic Control at LTH for the help to understand how the optimization works.

Finally, I would like to thank all people at Modelon AB, especially the other master thesis students, for the time as well as Toivo and Johan Åkesson for giving me the opportunity to do my master thesis project within this topic.

Contents

List of Figures

List of Tables

1

Introduction

Models of dynamic systems are used in the design phase of product development as well as for control design. The model design is a very important procedure to get as accurate a model as possible. When modeling dynamic physical systems there are three main parts in the model: the equations, the constants and the parameters. The equations can be seen as the skeleton of the model and are derived from physical equations. The equations usually contain a set of constants and parameters. Constants are well known without any uncertainties and could be for example the physical constant of the electric charge of an electron e or the mathematical constant π and will not cause any errors to the model. The parameters are instead quantities with some uncertainty, for example the area A of a body or the mass concentration ρ in a liquid. It can be hard to know the correct parameter values during the design of the model so there is a need to estimate them by using some measurement data.

In grey-box identification, the uncertainties of nonlinear models are estimated. For model design and verification, grey-box identification is used to identify the uncertain parameters. The noise model can be included in the optimization. Examples of two methods that are used for parameter estimation of nonlinear systems are Maximum Likelihood estimation and Output Error estimation. In [Bohlin, 2006] a MATLAB toolbox for grey-box identification is presented and in [Sørlie, 1996] a grey-box package was developed.

Modelica is a modeling language, mainly used for modeling dynamical system. In order to improve Modelica models grey-box identification can be used. JModelica.org is a platform for simulation and optimization of Modelica models. With the support for simulation and optimization in JModelica.org there is a huge potential to support grey-box identification but so far there is no complete solution for grey-box implementation i JModelica.org.

1.1 This Thesis

The objective of this thesis is to see how far it is possible to implement grey-box identification in JModelica.org. The focus of the work is to identify uncertain model

parameters from measurement data. The work is limited to systems with white Gaussian measurement noise without any stochastic system disturbances. The implemented methods are limited to Maximum Likelihood estimation and Output Error estimation. The more concrete goals are to be able to present a step by step method that can be used for parameter estimation but also for statistical analysis of the result.

1.2 Outline

The report starts with an introduction to grey-box identification in chapter 2 followed by a presentation of JModelica.org in chapter 3. The implementation of grey-box identification in JModelica.org is described in chapter 4 and results are presented in chapter 5. Finally, discussion and conclusion are presented in chapter 6.

1.3 Notation

$N(\mu, r)$ denotes a normal distribution with expected value μ and covariance r. $\chi^2_p(x)$ denotes the survival function (1-cumulative distribution function) of x for a χ^2 distribution with p degrees of freedom. When presenting results, relative error is calculated as

$$\frac{\text{estimated value - true value}}{\text{true value}}$$

and relative value as

$$\frac{\text{estimated value}}{\text{true value}}.$$

2

Grey-Box Identification

There are two main methods for modeling a system. The model can be derived from already known physical equations or by statistical analysis from measurement data. Grey-box identification combines the two methods. Grey-box identification is the common name for the identification process for system with partly unknown dynamics, usually in terms of unknown parameter values. For parameter identification, the identification is made off-line over a whole batch of measurement data and is usually solved by minimizing a cost function.

The target of grey-box identification is to make a system model as accurate as possible compared to the real system. When identifying nonlinear system it is extremely helpful to use some dynamics of the system. Compared to black-box identification, where nothing about the system is known, grey-box identification use some information about the system, such as the main dynamics, to identify the system. In order to get some information about how the system behaves, one also needs some measurement data. To improve the model, by making it more accurate, one can estimate the parameters in the model but also extend the system if the system model does not include all variables or equations. In this work the physical equations are assumed to be known but with some uncertainties in the parameters. Two types of estimation methods are used in this thesis, Output Error estimation (OE) and Maximum Likelihood estimation (ML), where measurement data is used to estimate the unknown parameters.

An example of how to rewrite a system in order to identify some parameters is shown in [Li and Ding, 2013]. In [Gunnar et al., 2006] a three step method is used to estimate the parameters of a linear actuator. The two first steps estimate initial values for OE estimation in the last step. For the parameters of a feed-water heater model OE is used to initially estimate the parameters and to update the estimates in [Barszcz and Czop, 2011]. Another method using OE as the cost function but instead solving the estimations by using signal basis for parameter estimations and an iterative scheme is presented in [Maruta and Toshuharu, 2013]. A parameter estimation method with low computational time but still reliable is the two-stage paradigm presented in [Garatti and Bittanti, 2013].

We will consider models on the form

$$\begin{cases} \dot{x}(t) = f(x(t), u(t), \theta) \\ y(t) = h(x(t), u(t), \theta) + v(t) \\ x(0) = x_0 \end{cases} \tag{2.1}$$

where $y(t)$ is the measurable system output, $v(t)$ is the measurement noise with covariance matrix R, $x(t)$ is the system states, $u(t)$ is the system inputs and θ is the set of unknown parameters to be estimated. The discrete measurements used for the identification are given by

$$h = \frac{t_n}{n-1} \tag{2.2}$$

$$T = [t_0, t_1, \cdots, t_n] \tag{2.3}$$

$$t_k = t_0 + h \cdot k \tag{2.4}$$

$$t_0 = 0 \tag{2.5}$$

$$y_M(t_k) = y(t_k) + v(t_k) \tag{2.6}$$

$$v(t_k) = \begin{bmatrix} v_1(k) & v_2(k) & \cdots & v_m(k) \end{bmatrix}^T \tag{2.7}$$

$$R = \begin{bmatrix} r_1 & 0 & \cdots & 0 \\ 0 & r_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_m \end{bmatrix} \tag{2.8}$$

$$v_i(k) \in N(0, r_i) \tag{2.9}$$

where h is the sample time, n is the sample size, t_n is the final time, $y_M(t_k)$ is the measured output, $v(t_k)$ is the measurement noise vector, R is the measurement noise covariance matrix and m is the number of output signals. Equation (2.6) is rearranged to

$$v(t_k) = y_M(t_k) - y(t_k) \tag{2.10}$$

and is used in the optimization.

This chapter describes the two estimation methods used in this work as well as some theory used to analyze the results. It is assumed that the measurement noise is white Gaussian and that there are no stochastic system disturbances.

2.1 Output Error (OE) Estimation

OE estimation is a simple estimation method that just minimizes the error between the predicted system output from the model and the measurements. Since most system outputs are measured with noise, the initial states x_0 are unknown and have to

be estimated as well as the parameters, which are the main target. Parameter estimation for the system in Eq. (2.1) is done by minimizing the cost function [Isaksson et al., 2010]

$$\min_{x_0,\theta,x} V_{OE} = \sum_{k=1}^{N} v(t_k)^T v(t_k)$$

subject to Eq. (2.1) and Eq. (2.10).

The advantage with OE estimation is mainly that it is easy to implement and that the noise distribution does not effect the setup. A drawback is that all outputs are treated equally even if some measurements may be better than others.

2.2 Maximum Likelihood (ML) Estimation

The idea with ML estimation is to formulate the problem as "which set of parameters makes the given measurements most likely?". For a set of parameters θ and measurements $y = [y_1, y_2, \cdots, y_n]$, the estimated parameters are given by the $\hat{\theta}$ that maximizes the following likelihood function:

$$L(\theta, Y) = \prod_{k=1}^{N} f_v(y_k; \theta)$$

subject to Eq. (2.1) and Eq. (2.10). The probability density function, $f_v(v)$, for v in Eq. (2.7) is the Gaussian density

$$f_v(v) = \frac{1}{\sqrt{(2\pi)^k |R|}} \exp\left(-\frac{1}{2} v^T R^{-1} v \right) \tag{2.11}$$

where R is given from Eq. (2.8).

If the parameters θ in the system in Eq. (2.1) are supposed to be estimated from N measurements on the outputs, $y \in \Re^{n\times 1}$, the ML optimization problem will be

$$\max_{\theta} L = \prod_{k=1}^{N} f_v\big(v(t_k)\big) \tag{2.12}$$

subject to Eq. (2.1) and Eq. (2.10).

Equation (2.12) is simplified with Eq. (2.11) into

$$(\hat{\theta},\hat{x}) = \arg\max_{\hat{\theta},\hat{x}} L = \prod_{k=1}^{N} f_v\big(v(t_k)\big) = \prod_{k=1}^{N} \frac{1}{\sqrt{(2\pi)^k |R|}} \exp\left(-\frac{1}{2} v(t_k)^T R^{-1} v(t_k) \right) =$$

$$\arg\max_{\hat{\theta},\hat{x}} \log L = c + \sum_{k=1}^{N} -\frac{1}{2} v(t_k)^T R^{-1} v(t_k) - \frac{1}{2} \log |R| =$$

$$\arg\min_{\hat{\theta},\hat{x}} = \sum_{k=1}^{N} v(t_k)^T R^{-1} v(t_k) + \log |R|$$

where $\hat{\theta}$ are the estimated parameters and $\hat{x}$ the optimal state trajectories.

As one can see, ML estimation is similar to OE estimation, except for R. The advantage with ML estimation is that it gives an opportunity to estimate the noise. In ML estimation the noise is estimated together with the parameters and the estimated noise covariances can be seen as a weight matrix designed during the estimation. Since the initial states usually are unknown due to measurement noise they also have to be estimated.

Thus the ML cost, for a continuous system with N samples, no process noise and Gaussian measurement noise, will look like (see also [Isaksson, 2013])

$$\min_{x_0,\theta,R,x} V_{ML} = \sum_{k=1}^{N} v(t_k)^T R^{-1} v(t_k) + \log |R| \tag{2.13}$$

subject to Eq. (2.1) and Eq. (2.10). R is the measurement noise covariance matrix and the system input-signals $u(t)$ are assumed to be known.

2.3 Large Sample Theory

In the previous sections, two methods to estimate parameters have been presented. Both of these methods will give estimated parameters from given measurement data. However, the result does not say anything about how accurate they are or how much one can trust the result. For models that are not including all equations or variables to fully represent the system, it might be impossible to identify the parameters correctly. Since the models in this work are assumed to include all equations and variables, this will not be a hazard.

The estimation will face some trouble even if the model is designed in a correct way. Intuitively, by increasing the number of samples, the performance of the estimation should increase. In [Mendel, 1995] it is stated that ML estimators are asymptotically Gaussian with no bias, which means that the estimation is unbiased for an infinite sample size. Unfortunately there are no infinite sample sizes in reality so the sample size chosen in the estimation has to depend on its purpose. If a small bias can be acceptable for the purpose of the estimation, then it is possible to use

a finite sample size and still get sufficiently good estimates. A simple example of when the estimation is biased for finite samples and unbiased for infinite samples is shown in Ex. 1.

EXAMPLE 1—NOISE ESTIMATION [MENDEL, 1995]
Assume a SISO system with the following dynamics:

$$\begin{gathered} y_k = a + v_k \\ \forall k \in [1,2,\cdots,N] \\ v_k \in N(0,r) \end{gathered}$$

where y_k are the measurements. The target is to estimate the covariance r and the parameter a with ML estimation. The cost function to minimize will be

$$\min_r V_{ML} = \sum_{k=1}^{N} \frac{v_k^2}{r} + \log(r) = \sum_{k=1}^{N} \frac{(y_k - a)^2}{r} + \log(r)$$

and is solved by assuming that the only point where the derivatives of the cost function is zero is equal to the local minimum.

$$\begin{aligned} \frac{\partial V_{ML}}{\partial a} = 0 &\iff \sum_{k=1}^{N} -\frac{2(y_k - a)}{r} = 0 \implies \hat{a} = \frac{1}{N}\sum_{k=1}^{N} y_k \\ \frac{\partial V_{ML}}{\partial r} = 0 &\iff \sum_{k=1}^{N} -\frac{(y_k - a)^2}{r^2} + \frac{1}{r} = 0 \implies \hat{r} = \frac{1}{N}\sum_{k=1}^{N} (y_k - \hat{a})^2 \end{aligned}$$

The expected value of the estimated covariance $\hat{r}$ is

$$\begin{aligned} E\{\hat{r}\} &= \frac{1}{N}\sum_{k=1}^{N} E\left\{(y_k - \hat{a})^2\right\} = \sum_{k=1}^{N} E\left\{(v_k - \frac{1}{N}\sum_{i=1}^{N} v_i)^2\right\} \\ &= \frac{1}{N}\sum_{k=1}^{N}\left(E\left\{v_k^2 - \frac{2}{N}\sum_{i=1}^{N} v_k v_i + \frac{1}{N^2}(\sum_{i=1}^{N} v_i)^2\right\}\right) \\ &= \frac{1}{N}\sum_{k=1}^{N}\left(r - \frac{2}{N}r + \frac{1}{N^2}Nr\right) \\ &= r - \frac{1}{N}r = \frac{N-1}{N}r \end{aligned}$$

As one can see the estimate $\hat{r}$ is biased for a finite sample size but asymptotically unbiased for an infinite sample size. □

The bias in the estimation with finite sample size is not the biggest problem. A worse scenario is if the estimation converges into an unwanted local minimum. In [Bohlin, 1991] it is stated that estimation with finite sample size might have many local minima but in this thesis no complete analysis for this is carried out.

2.4 Falsification and Over-parametrization

For system identification the falsification procedure investigates whether the estimated model is close enough to the real system. Since the work in this thesis only treats parameter estimation, the falsification procedure will analyze how many and which parameters to release in the optimization in order to find the best model with as low complexity as possible. Another failure to investigate is if the solver has converged to an unwanted minimum. By analyzing how reasonable the given measurements would be given the estimated parameters, the probability that the estimated model is too complex can be evaluated.

When identifying parameters it might be impossible to estimate all parameters from a given set of data. When the system is represented with more parameters than necessary it is called over-parametrization and might result in bad estimates.

The objective of the estimations in this work is to improve already existing models by statistical optimization and analysis from measurement data from the real system. In [Bohlin, 1991] there are several different falsification techniques presented for different kinds of purposes. Many of the tests are more theoretical and hard to implement. The most practical falsification technique for the purpose of this thesis is the Likelihood-Ratio test presented in the next subsection.

2.4.1 The Likelihood-Ratio test

The Likelihood-Ratio (LR) test [Bohlin, 1991] offers an implementable and practical method to analyze the optimization result from ML estimation. The theory holds for infinite samples but for practical reasons large sample sizes are assumed to give the same results as for infinite sample size. Therefore the sample size N is assumed to be arbitrarily large. For estimation with finite sample size, some parameters will affect the system differently. From a given set of measurement data, one wants to find the optimal set of parameters to estimate in order to find the best model with least complexity. If the number of degrees of freedom increases, it is more likely that the problem is over-parametrized and all parameters cannot be estimated from the given data but also that the optimization converges to an unwanted minimum. On the other hand, more free parameters might increase the performance and give a better model than for a lesser number of parameters. The LR test is a tool to find how many and which parameters to estimate in order to get the statistically best model.

Let θ be the set of parameters to estimate and let $F(\theta)$ be the minimum in Eq. (2.13):

$$F(\theta) = \min_{x_0,\theta,R,x} V_{ML} = \sum_{k=1}^{N} \hat{v}(t_k)^T \hat{R}^{-1} \hat{v}(t_k) + \log|\hat{R}| \tag{2.14}$$

Consider two sets of parameters θ_0 and θ_1, resulting in two new models of the system. The sets of parameters are chosen so that $\dim(\theta_1) > \dim(\theta_0)$ and θ_1 contain

all parameters in θ_0. If these two models represent the system equally well then $F(\theta_0) - F(\theta_1) \in \chi^2_p$, where $p = \dim(\theta_1) - \dim(\theta_0)$.

With this theory it is possible to perform a hypothesis test with the null hypothesis, H_0, and the alternative hypothesis, H_1:

$$\begin{cases} H_0 : \text{"}\theta_0 \text{ gives equally good model as } \theta_1\text{"} \\ H_1 : \text{"}\theta_1 \text{ gives a better model than } \theta_0\text{"} \end{cases}$$

The risk, α is given from, α = ("H_1 is preferred if H_0 is true"). The mathematical expression for α is then:

$$\alpha = \chi^2_p\big(F(\theta_0) - F(\theta_1)\big) \tag{2.15}$$

$$p = \dim(\theta_1) - \dim(\theta_0) \tag{2.16}$$

where $\chi^2_p(x)$ is the survival function of the chi-squared distribution with p degrees of freedom.

3

JModelica.org

JModelica.org is an open source platform, developed by Modelon AB in collaboration with the Department of Automatic Control Lund University, for powerful system simulations and optimizations. Modelica is a modeling language while Optimica is an extension of Modelica for optimization, both are supported in JModelica.org. To be able to do grey-box identification one needs a tool for model design, an optimization tool and a tool for analyzing and handling of the data. The three components used for this purpose are Modelica for model design and simulations, Optimica for optimization and Python, which is also supported in JModelica.org, for execution of the previous tools and for handling the data. A user guide for JModelica.org can be found at [Modelon AB, 2013].

3.1 Modelica

Modelica is an open source language developed by the Modelica Association [Modelica Association, 2014] for modeling physical systems. Modelica is object oriented and equation based and is today, for example, used for modeling in the automotive and power industry. An example of a Modelica code representation of a state space model can be seen in Ex. 2.

EXAMPLE 2
This is a simple example of how to represent a system on state space form in Modelica.

The following Modelica code:

```
model simple
        parameter Real a = -1.0;
        parameter Real b = 2.0;
        parameter Real c = -1.5;
        parameter Real d = -0.5;

        parameter Real x10 = 0.0;
        parameter Real x20 = 0.0;
```

```
        Real x1(fixed = true, start = x10)
        Real x2(fixed = true, start = x20)

        Real y1;
        Real y2;

        input Real u1;
        input Real u2;

        equation
                der(x1) = a*x1 + b*x2 + u1;
                der(x2) = c*x1 + d*x2 + u2;
                y1 = x1;
                y2 = x2;
end simple;
```

represents the following system:

$$\begin{cases} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \\ \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \end{cases}$$

with the control signals $u_i(t)$, the output signals $y_i(t)$ and $a = -1.0, b = 2.00, c = -1.5$ and $d = -0.5$. □

In JModelica.org the Modelica code used for simulations are compiled into a FMU (Functional Mock-up Unit) model that can be loaded into Python.

3.2 Optimica

Optimica is an extension of Modelica used for optimization purposes. As for Modelica code, Optimica code is compiled into a model. CasADi is a tool for automatic differentiation. The current default model for optimization in JModelica.org is CasADi and therefore are the Optimica models used in this thesis are compiled into CasADi models. Unfortunately, there is currently no support for minimizing sums in Optimica with CasADi. How to handle this is explained in Sec. 3.4. A simple implementation of OE estimation is shown in Ex. 3.

EXAMPLE 3
This example illustrates OE estimation of the system in Ex. 2 where the unknown parameters $\theta = [a, b, c, d]$ are to be estimated from measurements of 40 s. It is as-

sumed that the Modelica model in Ex. 2 is accessible from the following Optimica model.

```
optimization OE(objectiveIntegrand=(v1^2)*(r1)^(-1) +
                                   (v2^2)*(r2)^(-1) ),
                                   startTime=0.0, finalTime = 40.0)

        simple simp(x10 = x10, x20 = x20, a = a, b = b, c = c, d = d);

        //noise covariance parameters
        parameter Real r1(free = false) = 1;
        parameter Real r2(free = false) = 1;

        //initial states
        parameter Real x10(free = true,  initialGuess = 0.0) ;
        parameter Real x20(free = true,  initialGuess = 0.0) ;

        // parameters to estimate
        parameter Real a(free = true, initialGuess = 1.0,
                min = -5, max = 5);
        parameter Real b(free = true, initialGuess = 1.0,
                min = -5, max = 5);
        parameter Real c(free = true, initialGuess = 1.0,
                min = -5, max = 5);
        parameter Real d(free = true, initialGuess = 1.0,
                min = -5, max = 5);

        // control signals
        input Real u1;
        input Real u2;

        // measurement signals
        input Real y1;
        input Real y2;

        Real v1;
        Real v2;

equation
        v1 = (y1-simp.y1);
        v2 = (y2-simp.y2);
        u1 = simp.u1;
        u2 = simp.u2;
end OE;
```

Note that both the measurements and the control signals are declared as inputs since they are not predefined, but changes from estimation to estimation. All pa-

rameters to estimate are declared with `free = true`. Since there is no support for minimizing sums, the cost function is declared as an integral to minimize `objectiveIntegrand`. □

3.3 Python

Python [Python Software Foundation, 20140328] is a programing language that is used in JModelica.org to execute Modelica and Optimica code. A Python script is used to compile and load the Modelica models into FMUs as well as the Optimica models into CasADi models. Also all data handling and analysis are made through Python.

3.4 Limitations

Optimica is developed for continuous time optimization, while grey-box identification is done in discrete time. The cost function is evaluated at every collocation point at each element [Magnusson, 2012]. The default discretization method for optimization in JModelica.org is Radau collocation [Modelon AB, 2013]. When the collocation points per element are set to 1, the evaluations in the integrand are made only at the endpoint of each element. When the number of elements is set to the number of measurements-1 the elements will have the same length as the sampling time, and the endpoints are matched into the measurement points and basically no interpolation is done. By the modifications described above, the cost function, implemented as

$$\min_{x_0,\theta,R,x} V_{ML} = \int_{t_1}^{t_N} v(t_k)^T R^{-1} v(t_k) + \log|R|$$

can be treated as the following cost function:

$$\min_{x_0,\theta,R,x} V_{ML} = h \sum_{k=2}^{N} v(t_k)^T R^{-1} v(t_k) + \log|R|$$

where h is the sampling time for the given measurements. Note that since we only evaluate the endpoint of each collocation element we will lose the first measurement, at time = 0.0, in the cost function.

The numerical solver will lose some of its accuracy with the necessary conditions. When setting the number of collocation points to 1, the discretization method will be implicit Euler discretization:

$$\begin{cases} \dot{x}(t) = f(x,u,t) \\ x_{k+1} = x_k + h\dot{x}_{k+1} \\ x_0 = x(0) \end{cases}$$

This will give a systematic error in the optimization. The reason that this has to be done is because CasADi optimization with JModelica.org is currently not supporting cost functions as sums, but the optimization options are changed to make the cost function to act like a sum. If the system is not evaluated over the measurement points, the interpolation will be heavily affected by the measurement noises. This result will be worse than for the case with a numerical solver with systematic errors. There will not be any systematic errors for systems with constant derivative between the samples.

4

Implementation

The theory presented in chapter 2 has to be implemented in order to be able to use grey-box identification. This chapter describes how the grey-box identification has been implemented in JModelica.org. Section 4.1 describes how to simulate Modelica models in order to analyze the estimation for simulated systems. The optimization is presented in Sec. 4.2 followed by implementation of estimation analysis in Sec. 4.3.

4.1 Simulation

All estimation done in this work has been done based on artificial measurement data. The measurement data is generated simply by adding white Gaussian noise to the signals given from simulation of Modelica models. The advantage of estimating parameters from artificial measurement data is that the whole system is known and the robustness of the implementation is easier to analyze. As described in Sec. 3.4, the optimization algorithm has some limitation in the discretization of the system. Two types of discretization during the simulations has been done in this work, one with implicit Euler in order to use the same discretization method as the optimization, and one for the default settings used for FMUs. All measurement noise was generated from the `random` module in the NumPy library [The Scipy Community, 2013].

It was not possible to simulate FMUs with implicit Euler with one collocation point. To be able to generate measurement data with implicit Euler, another approach was used by generating measurement data from Optimica with the same settings as for ML estimation with implicit Euler. The optimization was then made by optimizing the outputs from the Modelica model, but with no free parameters. Since Optimica simulates the model in order to find the optimal solution and no parameters are free, the final solution will be the system simulated with the given parameters. The setup in form of equations for generating N samples can be seen as

$$\min_{\theta,x} V_{sim} = \sum_{k=1}^{N} y(t_k)^T y(t_k)$$

where the $y(t)$ is the system output, $x(t)$ is the states of the systems, $u(t)$ is the system inputs and θ is the free parameters. By setting $\theta_0 = [\,]$, no parameters are estimated and the system will be simulated and all parameters can be defined in the Optimica code.

The implementation of simulation with the default settings in FMU is straight forward and can be seen in chapter 3.

4.2 Optimization

Two types of identification have been implemented in this work. OE estimation and ML estimation. The limitations explained in Sec. 3.4 makes the approach a little bit different depending on which type of estimation is done.

As long as no noise intensities are supposed to be estimated, it seems that the sum in the cost function can be replaced with an integral without degrading the performance of the estimation. Simply

$$\min_{x_0,\theta,x} V_{ML} = \sum_{k=1}^{N} v(t_k)^T R^{-1} v(t_k) + \log|R|$$

is replaced with

$$\min_{x_0,\theta,x} V_{ML} = \int_{t_1}^{t_N} v(t_k)^T R^{-1} v(t_k) + \log|R| \tag{4.1}$$

For ML estimation with free noise parameters the cost function is written in Optimica as Eq. (4.1) but the collocation points are changed as explained in Sec. 3.4 to get a sum instead of an integral. The number of element is always set to be the sample size-1 to make the element endpoints match each sample point.

4.3 Analysis

4.3.1 Interactive Identification

This iterative procedure is inspired by the MoCaVa MATLAB toolbox [Bohlin, 2006] and uses the theory from the LR-test in order to iteratively find the best model with least complexity. Let θ be the set of parameters to be estimated and $M(\theta)$ the model found by ML estimation with the free parameters in θ.

The null model is the current optimization result given from the free parameters in θ_0 and is used as a reference in order to see if the model can be improved or not.

From Eq. (2.14), $F(\theta)$ is called the cost. For two different kinds of models with different complexity the risk is calculated from Eq. (2.15).

In this section the step by step procedure of how to find the best model with lowest complexity is explained. Initial guesses on parameter values including initial states and measurement noise covariance are needed.

One wants to investigate whether it is worth to have more free parameters than in the null model and which parameters to release. Let $[\theta_1, \theta_2, \theta_3, \cdots, \theta_n]$ be the conditional parameter sets that include all parameters in the null model but also some more parameters. The risk α is calculated as

$$\alpha = \chi_p^2(F(\theta_i) - F(\theta_0))$$
$$p = \dim(\theta_i) - \dim(\theta_0)$$

where the risk is the probability to choose the new model, given from θ_i, if the old model is equally good as the new model. This works for a single conditional model. If there are n conditional models the risk is

$$\bar{\alpha}_i = 1 - (1 - \alpha_i)^n$$

where $\bar{\alpha}_i$ is the multiple conditional risk and accounts for the fact that the risk of false positives is increasing when we have many hypothesis to choose from.

If the risk is low enough one can extend the null model by releasing the parameters one compared with. With many conditional models, the one with the greatest cost reduction, $F(\theta_0) - F(\theta_i)$ is chosen if more than one model gives a low enough risk.

The first parameters to release are always the initial states and the noise covariances, usually this can be done with a single conditional model including all these parameters. If it is possible to release these parameters one can start to try to release other parameters. For this case there might be multiple conditional models but compared to the null model just one extra parameter is released for each conditional model. This is repeated until no improvement is reached. If one parameter is chosen to be released it will always be released for more complex models. If one suspects constant disturbances in the system, constant disturbances should be included in the model and should be released just after the initial states and the noise covariances.

4.3.2 Confidence Intervals

The LR test helps to reach the best model given the measurement data. However, one might want to get some kind of information about the accuracy of the result. In this work one big assumption is made in order to get a straight-forward method to analyze the estimation. It is almost impossible to get an estimated model equal to the true model, but if the error is small enough, the statistics will still give some useful

and interesting information about the system. Bootstrapping [Zoubir and Iskander, 2007] is used to regenerate the sample data and use this to estimate the estimation covariance. This approach only works for ML estimation, since measurement noise covariance is needed.

ML estimation estimates the initial states, the measurement noise intensity and the system parameters, the control signals are given, so one has all parameters to regenerate measurement data by simulation. The idea is to simulate the system with the estimated parameters and add white Gaussian noise with the estimated covariance and then estimate the parameters from the simulated data. This is repeated for a large number of times to get the distribution of the estimated parameters. Because the real value for the simulated system is known one can investigate bias in the estimation but also the covariance of the estimation.

If parameter estimates are uncorrelated to each other one can estimate the covariance from the single distribution of each parameter. In this work, all parameters are treated separately, just because it is easier to visualize.

5

Results

In this chapter, the results from the implementation of grey-box identification in chapter 4 is presented. Analysis of the identification is presented in Sec. 5.1 and Sec. 5.2 presents some examples of full estimations.

5.1 Identification Analysis

This section shows some tests made to analyze the performance of the estimation. In Sec. 5.1.1 an example of pure noise estimation is presented. Subsection 5.1.2 illustrates the difference given for different kind of discretization methods used to create the measurement data. Subsection 5.1.3 shows the advantage of using ML estimation compared to OE. To show the performance of ML estimation, the parameters of a smaller nonlinear system are estimated and presented in Sec. 5.1.4. Finally, the performance of ML implementation for white Gaussian measurement noise for a two-point noise distribution is presented in Sec. 5.1.5.

5.1.1 Noise Estimation

ML estimation of covariance for a measurement with only noise was done as a first simple estimation step. The advantage with this estimation is that it is simple to calculate the result by hand and therefore one can compare the result with the theory.

This estimation was done by making a Modelica model with a constant output set to 0. Two ML estimates of the noise covariance were made, one with known expected value of the noise and one with unknown expected value of the noise. From Ex. 1, the ML estimates for estimating the covariance r and the expected value μ are

$$\begin{cases} \hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} y_i \\ \\ \hat{r} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{\mu})^2 \end{cases}$$

where $\hat{r}$ is the estimated covariance and $\hat{\mu}$ is the estimated expected value.

If the expected value is known, the noise covariance ML estimate is

$$\hat{r} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2.$$

1000 samples was used to estimate white Gaussian noise with $\mu = 0$ and $r = 1$. The result can be seen is Table 5.1.

Table 5.1 The ML estimated values for white Gaussian noise sampled with 1000 samples.

Parameter	$\hat{r}$ (μ known)	$\hat{r}$ (μ unknown)	$\hat{\mu}$ (μ unknown)
Theoretical value	1.1201	1.0161	0.0054
Estimated value	1.1201	1.0161	0.0054
Estimated value - Theoretical value	$-2.95 \cdot 10^{-10}$	$-2.9149 \cdot 10^{-8}$	$-2.6887 \cdot 10^{-13}$

Since the differences between the estimated value and the value expected to get from ML estimation is negligible, the conclusion is that it is possible to estimate white Gaussian noise with the ML estimation with a negligible error.

5.1.2 Discretization

This example will show the effect of different discretization in the optimization and the simulation. A system was simulated with two different kinds of discretization, implicit Euler and the default discretization in JModelica.org. The state-space representation of the system is

$$\begin{cases} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -a_1 & a_2 \\ -b_1 & -b_2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \\ \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} x_1(t) \cdot x_2(t) \\ x_2(t)/\big(1 + x_1(t)\big) \end{bmatrix} \\ x_1(0) = x_{10} \\ x_2(0) = x_{20} \end{cases} \tag{5.1}$$

where $u_i(t)$ are the control signals, $y_i(t)$ is the system outputs and the parameters $[a_1, a_2, b_1, b_2, x_{10}, x_{20}] = [1.0, 1.0, 1.0, 3.0, 0.5, 0.9]$. The simulation with the two different discretizations can be seen in Fig. 5.1 and the differences between the simulated signals can be seen in Fig. 5.2.

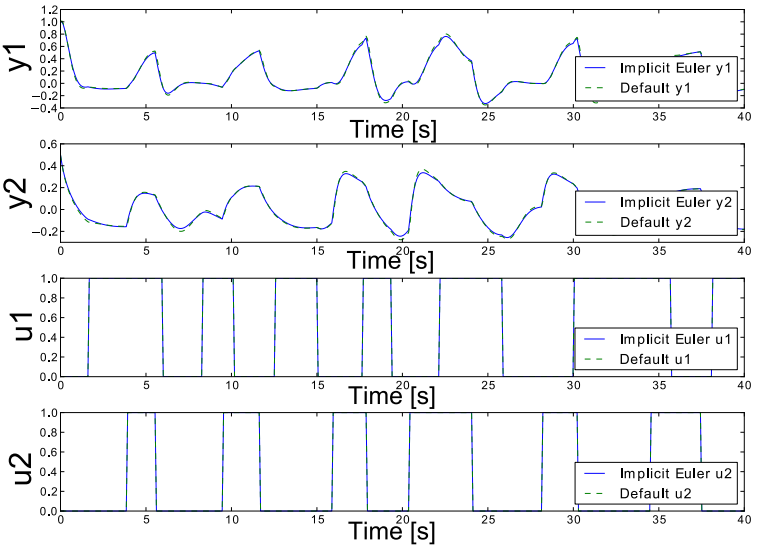

Figure 5.1 System in Eq. (5.1) simulated with the default discretization in JModelica.org and with implicit Euler.

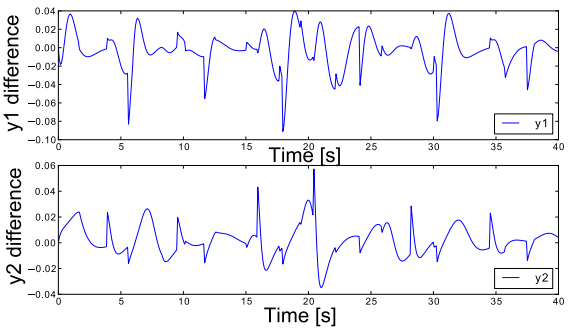

Figure 5.2 The differentiates (Implicit Euler - Default) between the simulation outputs from Fig. 5.1.

The two different sets of outputs given from the simulation were used, without noise, to estimate the parameters in the system with OE estimation. The result can be seen in Table 5.2.

One can see that the simulation with implicit Euler gives an unbiased estimation

Table 5.2 Relative error of the OE estimated parameters for the system in Eq. (5.1) with the signals in Fig. 5.1.

Parameters	a_1	a_2	b_1	b_2	x_{10}	x_{20}
Imp. Euler Sim.	0.00%	0.00%	0.00%	0.00%	0.00	0.00%
Default Sim.	-0.98%	-1.02%	2.16%	-7.72%	12.6%	-3.62%

while the default simulation gives a clear bias. Intuitively, one could expect that the default discretization should give the best estimation since it is a better discretization than implicit Euler. The result is due to the different discretization between the optimization and the simulation. With implicit Euler, the optimization can reproduce the data exactly using the correct parameters. With the default discretization, it will try to adapt them to account for the differences in discretization.

5.1.3 ML and OE comparison

In this subsection, it is investigated how the noise covariance affects the results given from ML and OE estimation. The system in Eq. (5.1) was simulated with implicit Euler over 40 s and sampled with 500 samples with white Gaussian measurement noise.

300 different estimates were made with a new noise realization for each and with noise covariances with different magnitude, $r_1 = 0.1$ and $r_2 = 0.001$.

As one can see in Fig. 5.3, the estimate covariance is much smaller for ML estimation than for OE estimation. The measurement noise covariances are chosen with different magnitude to show the advantage with ML estimation compared to OE estimation.

In Fig. 5.4, it is demonstrated how the ratio of measurement noise intensities affects the estimation covariance. One can see that ML estimation gives a lower covariance than OE for measurement covariances with different magnitude but not for noise covariances with the same magnitude. This is because the noise intensity estimates can be seen as measurement signal weights in the optimization. OE is always unweighted which gives the best performance for noise covariances with same magnitude while ML has to estimate the covariance which gives a higher degree of freedom in the optimization. This is more significant for a low sample size, because then it is harder to estimate the correct noise covariance with ML estimation.

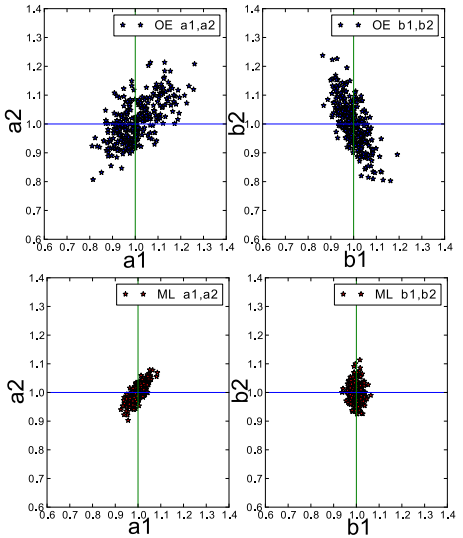

Figure 5.3 The estimation distribution for ML and OE for 300 noise realizations of the system in Eq. (5.1) simulated with implicit Euler with white Gaussian measurement noises with covariance $r_1 = 0.1$ and $r_2 = 0.001$. All plots are showing the relative value and are presented with the same scale.

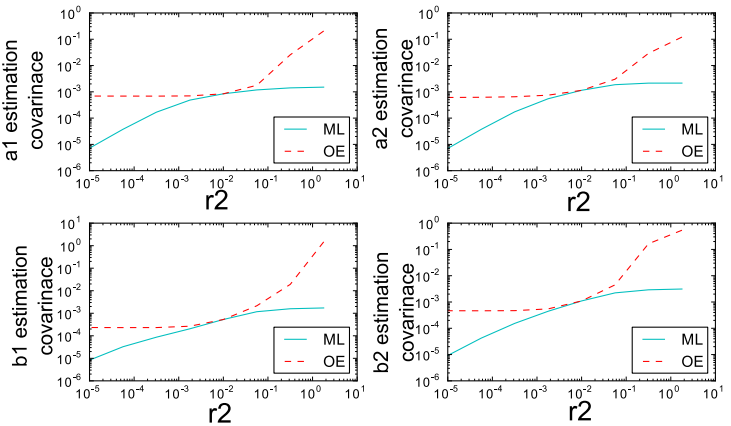

Figure 5.4 The estimation covariance as a function of r_2 for the system in Eq. (5.1). 150 different noise realizations was scaled with the noise covariances $r_1 = 0.01$ and r_2 given from the x-axis.

5.1.4 ML estimation for a nonlinear system

This example shows the performance of the implemented ML estimation for a small nonlinear system. The state-space representation of the system is

$$\begin{cases} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} -a_1 \cdot x_1(t) \cdot x_3(t) + a_2 \cdot x_2(t) + x_3(t) \\ -b_1 \cdot x_1(t) - b_2 \cdot x_2(t) + x_4(t) \\ -c_1 \cdot x_3(t) + c_2 \cdot x_4(t) \\ -d_1 \cdot x_3(t) - d_2 \cdot x_3(t) \cdot x_4(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \\ \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \\ y_4(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} \\ x_1(0) = x_{10}, x_2(0) = x_{20}, x_3(0) = x_{30}, x_4(0) = x_{40} \end{cases} \tag{5.2}$$

where $x(t)$ is the states, $y(t)$ is the outputs, $u(t)$ is the control signals and $\theta = [a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2, x_{10}, x_{20}, x_{30}, x_{40}]$ are parameters.

The system was simulated with implicit Euler for 40 s and sampled with 300 samples. 300 different estimates were made with a new noise realization for each loop and with noise covariances with different magnitude, $r_1 = 0.001, r_2 = 0.0001, r_3 = 0.005, r_4 = 0.005$, the inputs $u_i(t)$

$$u_1(t) = \begin{cases} 0 & \text{for } t < 5.0 \\ 1 & \text{for } t \geq 5.0 \end{cases}$$
$$u_2(t) = \begin{cases} 0 & \text{for } t < 25.0 \\ 1 & \text{for } t \geq 25.0 \end{cases}$$

and the parameters

$$[a_1 = 1.0, a_2 = 3.0, b_1 = 0.5, b_2 = 0.9, c_1 = 0.8, c_2 = 1.0, d_1 = 0.7, \\ d_2 = 0.3, x_{10} = 1.0, x_{20} = 1.0, x_{30} = 1.0, x_{40} = 1.0].$$

The estimate distribution can be seen together with a 95 % confidence interval in Fig. 5.5 and the confidence values can be seen in Table 5.3. From the estimation distribution it looks like the bias is negligible and the implementation manages to estimate the parameters as expected. The intersection between the confidence intervals are showing the mean values for the estimations. Since the plots show the relative value, an estimate is unbiased if the mean value is 1. A small bias can be expected, especially for the noise covariances, since the sample size is finite. For all parameter estimates, the bias is negligible and one can confirm that the ML estimation works for this system.

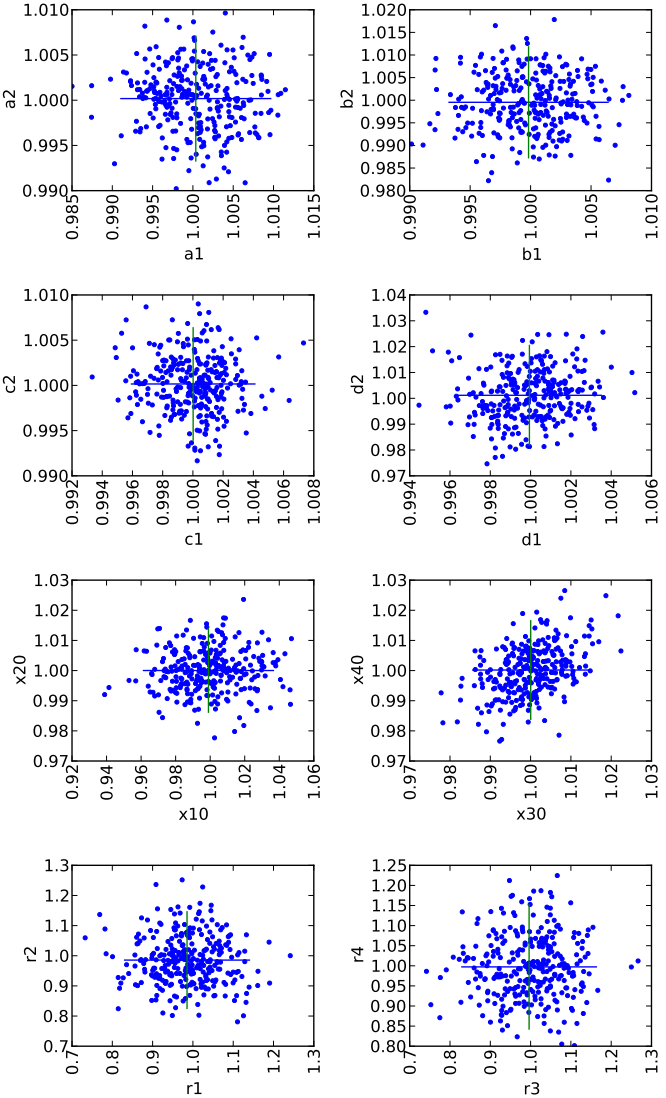

Figure 5.5 Relative value for the estimated parameters for 300 different noise realizations for the system in Eq. (5.2) . The system was discretized and simulated with implicit Euler. $r_1 = 0.001, r_2 = 0.0001, r_3 = 0.005, r_4 = 0.005$. The lines show the 95% confidence interval for each parameter estimate.

Table 5.3 Confidence interval of the relative value given from the estimations from Fig. 5.5.

Parameter	a_1	a_2	b_1	b_2
Mean value	1.0004	1.0002	0.9998	0.9995
Interval min	0.9911	0.9933	0.9933	0.9873
Interval max	1.0097	1.0071	1.0064	1.0118
Parameter	c_1	c_2	d_1	d_2
Mean value	1.0000	1.0002	0.9999	1.0012
Interval min	0.9959	0.9939	0.9964	0.9819
Interval max	1.0041	1.0064	1.0035	1.0205
Parameter	x_{10}	x_{20}	x_{30}	x_{40}
Mean value	1.0000	1.0000	1.0000	1.0002
Interval min	0.9614	0.9862	0.9856	0.9838
Interval max	1.0366	1.0139	1.0144	1.0165
Parameter	r_1	r_2	r_3	r_4
Mean value	0.9854	0.9856	0.9963	0.9973
Interval min	0.8308	0.8253	0.8294	0.8427
Interval max	1.1403	1.1458	1.1631	1.1518

5.1.5 Parameter estimation for two-point noise distribution

The system in Eq. (5.1) was simulated over 40 s, with a two-point noise distribution with zeros mean values and the covariances $r_1 = 0.01$ and $r_2 = 0.001$. Note that the covariances are the square root of the absolute noise values. The parameters were estimated with ML estimation with the same cost function as for white Gaussian measurement noise. The measurement signals, the optimal trajectory found from the optimization, and the residuals can be seen in Fig. 5.6 and a histogram from the residuals can be seen in Fig. 5.7. One can see that the residuals differ from a white Gaussian noise realization. In the histograms, it looks like the residuals can be seen as two separate white Gaussian distributions with mean values equal to the points in the two point distribution.

A plot over the estimation distribution for 500 different noise realizations can be seen in Fig. 5.8. One can see that even if the noise is not white Gaussian it looks like the estimates are normal distributed. The noise covariance estimates are biased what is expected for a ML estimation but they are not Gaussian. Just by looking at the parameter estimation distribution it is impossible do discern the kind of measurement noise.

The same system was sampled with 50 samples instead, to investigate how a low sample size would effect the optimal trajectory. A histogram plot of the residuals can be seen in Fig. 5.9. One can see from the histogram plot that the residuals differ from two white Gaussian distributions. Some residual are close to zero which

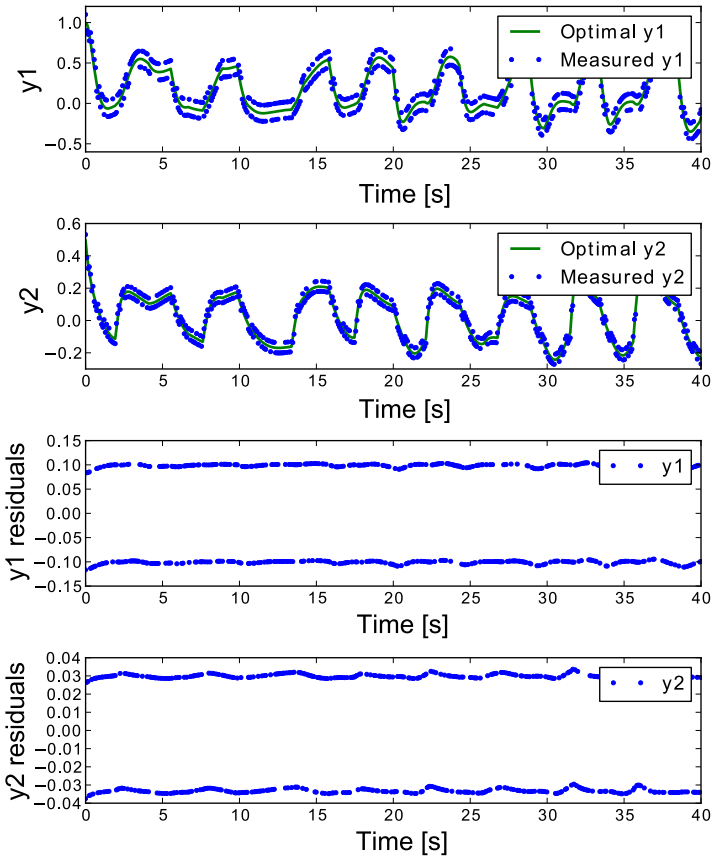

Figure 5.6 The system in Eq. (5.1) with two-point noise distribution with the covariances $r_1 = 0.01$ and $r_2 = 0.001$ estimated with ML estimation. The residuals = optimal trajectory - measured signal.

indicates that the solver does not manage to find a good optimal trajectory.

The conclusions given from this test is that an analysis over the residuals is necessary in order to identify the measurement noise distribution. ML estimation that is formulated for white Gaussian measurement noise but in fact is measuring two-point distributed noise are resulting in the same estimation distribution. A sample size big enough for its purpose is also necessary in order to identify the system.

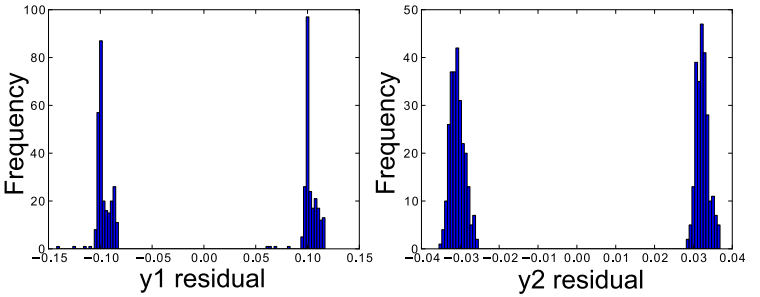

Figure 5.7 Histogram over the residuals for the system in Eq. (5.1) when sampled with 500 samples with the two-point noise covariances $r_1 = 0.01$ and $r_2 = 0.001$.

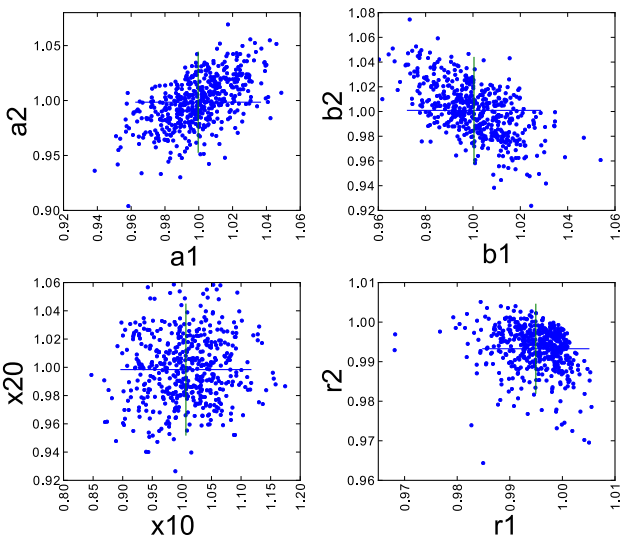

Figure 5.8 The estimation distribution for the parameters in system in Eq. (5.1) for 500 different two-point noise distribution with the covariances $r_1 = 0.01$ and $r_2 = 0.001$. The resuslt are presented in relative error. The lines are showing the 95 % confidence interval.

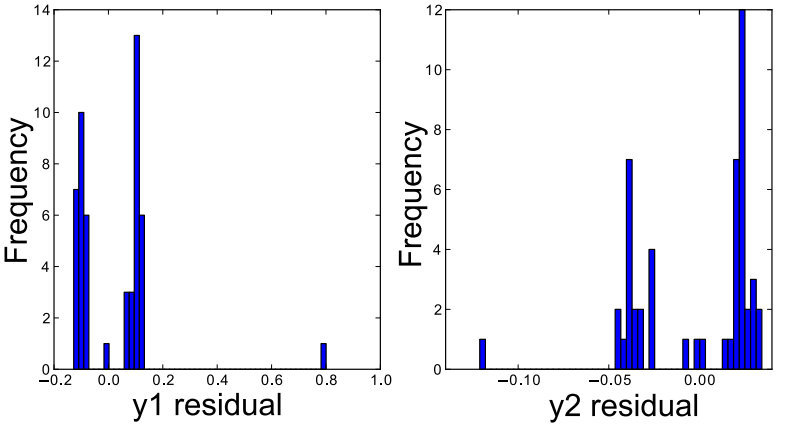

Figure 5.9 Histogram over the residuals for the system in Eq. (5.1) when sampled with 50 samples with the two-point noise covariances $r_1 = 0.01$ and $r_2 = 0.001$.

5.2 Full estimations

This section will show three full estimations including the analyses of the results. The examples are chosen to show why or why not it might be possible to estimate all parameters. The first example shows an example of a system where not all parameters are possible to estimate. The second example shows a full estimation where all parameters are estimated but also analysis of the estimates. The last example will show how the methods work to fit a less complex model with data from a more complex model.

5.2.1 RC-circuit Example

This example demonstrates how the interactive identification will falsify a too complex model in order to represent the system. Assume that R and the C are supposed to be estimated for the circuit in Fig. 5.10. The state-space representation of the system is

$$\begin{cases} \dot{x}(t) = \dfrac{-1}{R \cdot C} \cdot x(t) + \dfrac{1}{R \cdot C} u(t) \\ \\ y(t) = x(t) + v(t) \\ x(0) = x_0 \end{cases} \tag{5.3}$$

$$y(t) = V_{out}(t), u(t) = V_{in}, v(t) \in N(0, r). \tag{5.4}$$

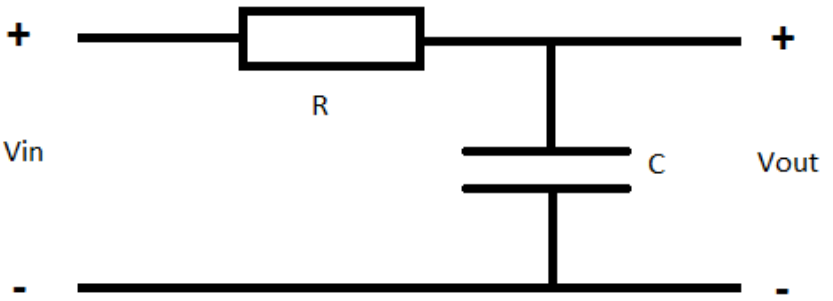

Figure 5.10 RC-circuit with measured signals V_{in} and V_{out}

The system was simulated with implicit Euler and 300 samples. The parameters where set to, $x_0(0) = 0\,V$, $C = 0.11\,mF$ and $R = 0.9\,k\Omega$, and the measurement noise on V_{out} was white Gaussian with covariance $r = 0.001$.

Table 5.4 Initial guess of parameters for the RC-circuit in Eq. (5.3).

.

Parameter	$R\ [k\Omega]$	$C\ [mF]$	$x_0\ [V]$	r
Initial Guess	15	0.10	0.0	0.1

The initial model parameters, to improve, can be seen in Table 5.4. The measured signal and the signal given from the initial model can be seen in Fig. 5.11. The cost $F(\theta)$ given from the initial model, $\theta_0 = 0$, was $F(\theta_0) = -663.4388$.

The first parameters to release in the optimization was the initial state x_0 and the noise covariance r. This gives the null model $\theta_0 = 0$ and the conditional model $\theta_1 = [x_0, r]$.The initial guesses are given from Table 5.4. The optimal trajectory found from the estimation can be seen in Fig. 5.12 and the statistics in Table 5.5.

Table 5.5 Cost statistics for $\theta_0 = 0$ and the extra parameter (Released parameter) for the system in Eq. (5.3).

Released parameter	$[x_0, r]$
Cost	-1217.9394
Cost Reduction	554.5006
Risk	0.00

Since the risk that the null model is equally good as the conditional model, is low, the procedure can go forward.

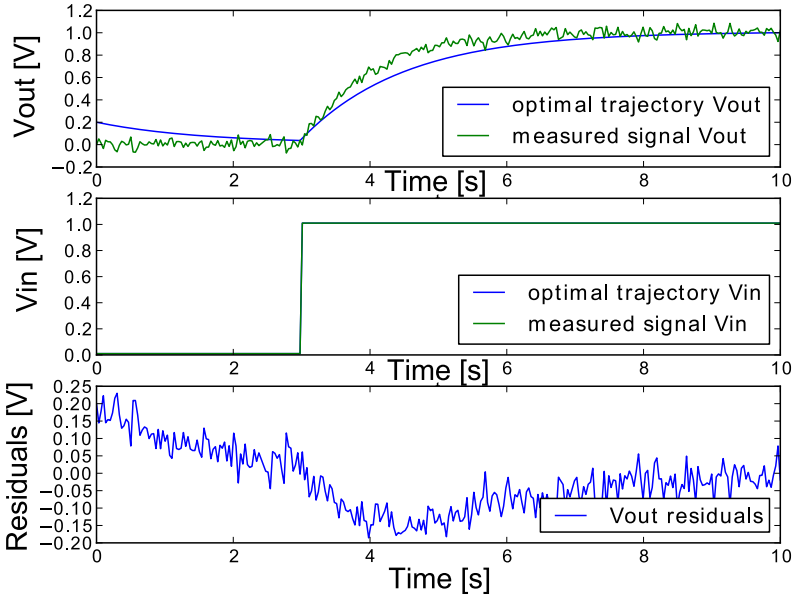

Figure 5.11 The measured output and the initial model (Table 5.4) for the system in Eq. (5.3). The residuals = optimal trajectory - measured signal.

Next, R and C are released so the conditional models are given by $\theta_1 = [x_0, r, R]$ and $\theta_2 = [x_0, r, C]$ and the null model is given by $\theta_0 = [x_0, r]$. The result can be seen in Table 5.6.

Table 5.6 Cost statistics for $\theta_0 = [x_0, r]$ and the extra parameter (Released parameter) for the system in Eq. (5.3).

Released parameter	R	C
Cost	-1739.5368	-1739.5368
Cost Reduction	521.5974	521.5974
Risk	0.00	0.00

It shows that the result is identical for the case were R or C is added to the estimation, this suggesting that the optimal trajectories for both cases are identical. The estimated results can be seen in Fig. 5.13.

Since the improvements compared to the first estimation is significantly better with no risk both parameters are released to see if it is possible to improve the esti-

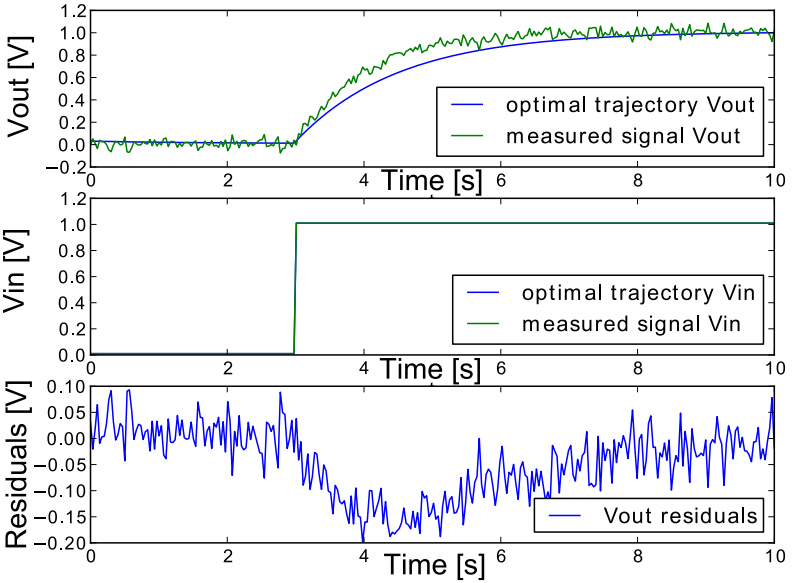

Figure 5.12 The system in Eq. (5.3) was estimated with fixed $C = 0.1mF$ and fixed $R = 1.5k\Omega$. The residuals = optimal trajectory - measured signal.

mates even more. Because the cost reduction is the same for R free and C free one can use any of these models as the reference model. Let $F(\theta_0) = -1739.5369, \theta_0 = [x_0, r, C]$ and $\theta_1 = [x_0, r, C, R]$. The result can be seen in Table 5.7.

Table 5.7 Cost statistics for $\theta_0 = [x_0, r, C]$ and the extra parameter (Released parameter) for the system in Eq. (5.3).

Released parameter	R
Cost	-1739.5368
Cost Reduction	0.00
Risk	1.00

Unfortunately, the risk of releasing all parameters is 100 % and it is impossible to get a better model than the previous given from R or C released separately. The real values used in the simulation as compared to the ones given from the estimations can be seen in Table 5.8 and it is shown that none of the estimates converge to the right values.

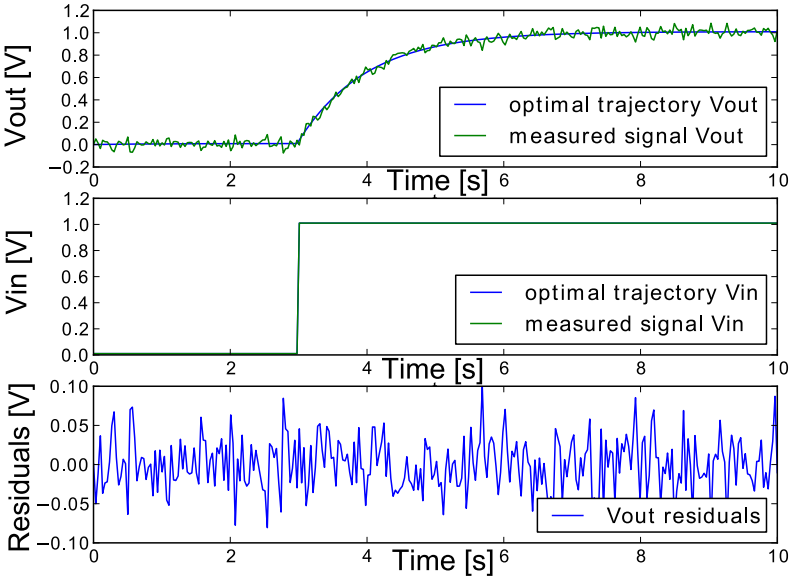

Figure 5.13 The system in Eq. (5.3) was estimated with free C and fixed $R = 1.5k\Omega$. The residuals = optimal trajectory - measured signal.

Table 5.8 Real and estimated parameters for the RC-circuit in Fig. 5.10 with measurement noise covariance 0.001.

Parameter	$R\ [k\Omega]$	$C\ [mF]$	$x_0\ [V]$
Real Value	9.0	0.11	0.0
R free	9.9	0.10	0.00
C free	15	0.066	0.00
R&C free	11	0.094	0.00

This is a known case of over-parametrization were the output does not depend individually on R or C. Instead it depends on the time constant $\tau = R \cdot C$. Note that for all estimates in Table 5.8 the product of R and C is 1.0. For a model designer that is just interested in the $V_{in} - V_{out}$ relationship, the relevant parameter is τ. A more relevant model would have replaced $R \cdot C$ with τ. It looks like the residuals are white Gaussian, which usually means that there is no more possibilities to improve the system even if not all parameters are released into the optimization.

5.2.2 Drum Boiler Example

This system and approach is taken from [Bohlin, 2006] with some modifications. It shows how to approach grey-box identification in order to estimate the unknown parameters with the lowest necessary complexity. All parameters are estimated by ML estimation and all costs $F(\theta)$ are given from Eq. (2.14).

The state-space model of the system is

$$\begin{cases} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} \frac{K_1}{T_D} \cdot (A_1 \cdot f_c(t) - A_2 \cdot u_c(t) \cdot x_1(t) \\ \frac{K_2}{T_R} \cdot (A_2 \cdot u_c(t) \cdot x_1(t) - A_3 \cdot x_2(t) \end{bmatrix} \\ \begin{bmatrix} E \\ P \end{bmatrix} = \begin{bmatrix} K_3 \cdot \big(A_4 \cdot A_2 \cdot u_c(t) \cdot x_1(t) + (1 - A_4) \cdot A_3 \cdot x_2(t)\big) + w_E \\ x_1(t) + w_P \end{bmatrix} \\ x_1(0) = x_{10} \\ x_2(0) = x_{20} \end{cases} \tag{5.5}$$

with the states $x(t)$, known constants in Table 5.9, unknown parameters in Table 5.10, constant disturbances w, the inputs $u_c(t)$ (control valve position) and $f_c(t)$ (fuel flow [kg/s]) and the outputs $E(t)$ (power [MW]) and $P(t)$ (drum pressure [N/m^2]). The constant disturbances, w_P, w_E are included into the unknown parameters since they might be estimated as well.

Table 5.9 Constants for system in Eq. (5.5)

Constant	K_1	A_1	A_2	K_2	A_3	K_3
Value	0.3	14	3.3	0.064	15	0.32

The system was sampled with 500 samples over 1000 s and simulated with implicit Euler in order to eliminate the error from differences in discretization between the simulation and the optimization. The simulation can be seen in Fig. 5.14.

This example will show how to estimate parameters step by step while trying to avoid a too complex model. Assume that the parameters are unknown but with initial guesses and the model one wants to improve is given from the parameters in Table 5.10. As a first step the initial model is run to compare with the measurements as well as to get a cost. The result can be seen in Fig. 5.15. The cost $F(\theta)$ was 3196935.

The next step is to investigate whether it is possible to improve the model by releasing the initial states $[x_{10}, x_{20}]$ and the measurement noise covariances $[r_E, r_P]$ (and see if the improvements given from this estimation is better in order to increase the complexity). The results can be seen in Table 5.11. Figure 5.16 shows the opti-

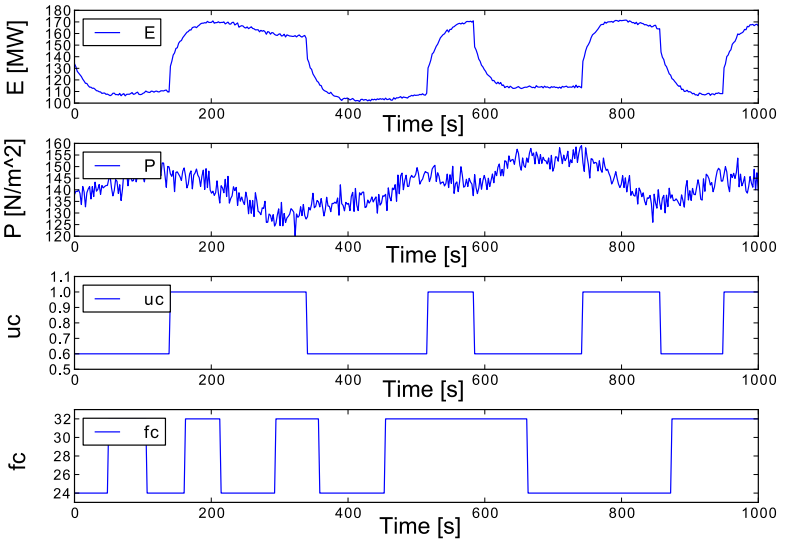

Figure 5.14 The system in Eq. (5.5) simulated with unknown parameters and white Gaussian measurements noise.

Table 5.10 Initial parameters for system in Eq. (5.5)

Parameter	$T_D\ [s]$	$T_R\ [s]$	A_4	w_E	w_P	x_{10}	x_{20}	r_E	r_P
Initial Guess	250.0	30.0	0.50	0.0	0.0	0.0	0.0	1.0	1.0

mal trajectories found by the estimation. The risk is low, so more parameters can be investigated and released.

Table 5.11 Cost statistics for the null model, $\theta_0 = 0$, for the system in Eq. (5.5)

Released parameters	$[x_{10}, x_{20}, r_E, r_P]$
Cost	5658.666
Cost Reduction	3191276
Risk	0.00

Since it looks like there is some kind of disturbances, the constant disturbances are released in order to investigate if the new model is better than the old one. So the null model θ_0 and the conditional model θ_1 are

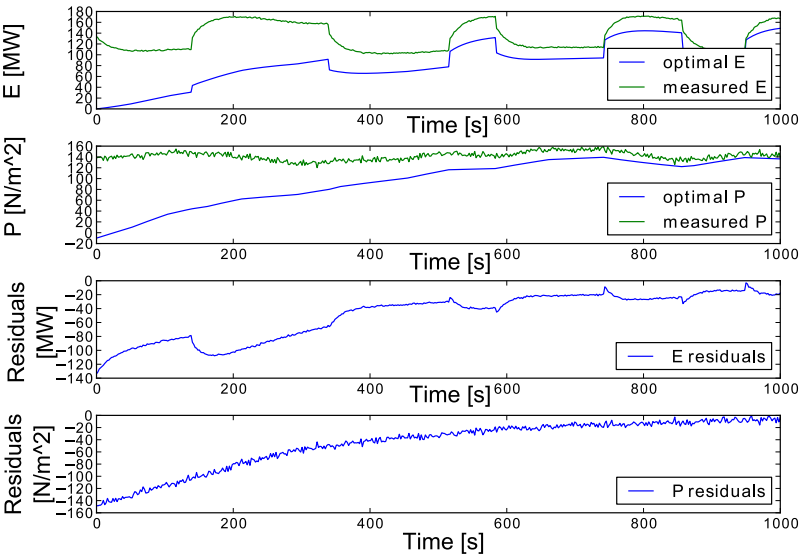

Figure 5.15 The system in Eq. (5.5) estimated with $\theta = 0$. The residuals = optimal trajectory - measured signal .

$\theta_0 = [x_{10}, x_{x20}, r_E, r_P]$
$\theta_1 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E]$.
The results can be seen in Table 5.12 and Fig. 5.17.

Table 5.12 Cost statistics for the null model, $\theta_0 = [x_{10}, x_{20}, r_E, r_P]$, for the system in Eq. (5.5)

Released parameters	$[w_E, w_P]$
Cost	3369.623
Cost Reduction	2289.044
Risk	0.0

Since the risk to release the disturbances is zero, more free parameters can be investigated in order to improve the model. The estimated parameters from the last test will be the new initial guesses shown in Table 5.13.

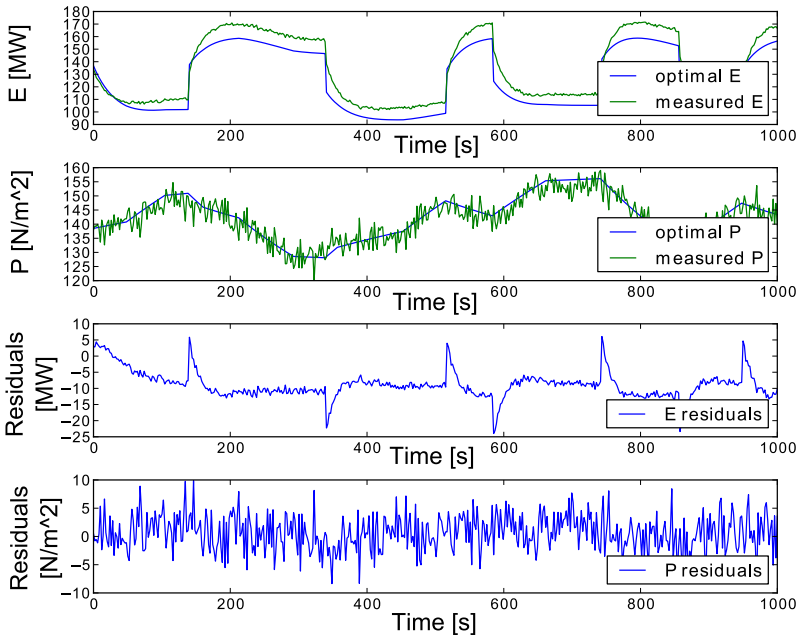

Figure 5.16 The system in Eq. (5.5) estimated with $\theta = [x_{10}, x_{20}, r_E, r_P]$. The residuals = optimal trajectory - measured signal .

Table 5.13 Initial guess after estimated system in Eq. (5.5) with $\theta = [x_{10}, x_{20}, r_E, r_P, w_P, w_E]$.

Parameter	T_D [s]	T_R [s]	A_4	w_E	w_P
Initial Guess	250.0	30.0	0.50	9.57	-10.7
Parameter	x_{10}	x_{20}	r_E	r_P	
Initial Guess	148	29.2	11.6	9.98	

Next, we try to release T_D, T_R and A_4 separately. In other words

$$\begin{aligned}\theta_0 &= [x_{10}, x_{20}, r_E, r_P, w_P, w_E] \\ \theta_1 &= [x_{10}, x_{20}, r_E, r_P, w_P, w_E, T_D] \\ \theta_2 &= [x_{10}, x_{20}, r_E, r_P, w_P, w_E, T_R] \\ \theta_3 &= [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4].\end{aligned}$$

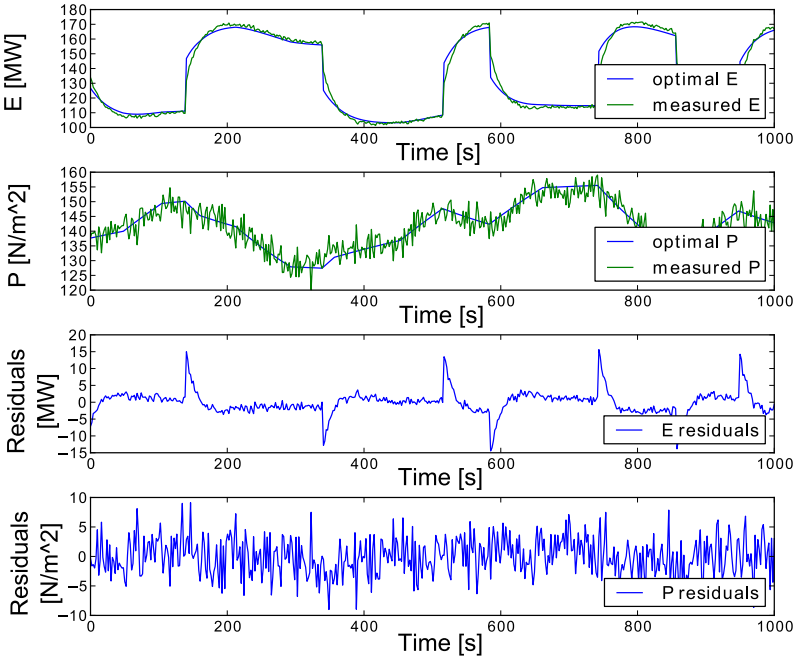

Figure 5.17 The system in Eq. (5.5) estimated with $\theta = [x_{10}, x_{20}, r_E, r_P, w_P, w_E]$. The residuals = optimal trajectory - measured signal .

The results can be seen in Table 5.14.

Table 5.14 Cost statistics for the null model, $\theta_0 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E]$ for system in Eq. (5.5).

Released parameter	T_D	T_R	A_4
Cost	3281.737	3361.316	3121.397
Cost Reduction	87.886	8.307	248.226
Risk	0.000	0.012	0.000

Since the risk is zero for more than one parameter, A_4 is chosen, since it gives the greatest cost reduction. The optimal test results with A_4 released can be seen in Fig. 5.18. One can see that the residuals for E do not look like the white Gaussian noise, so further improvements can probably be achieved.

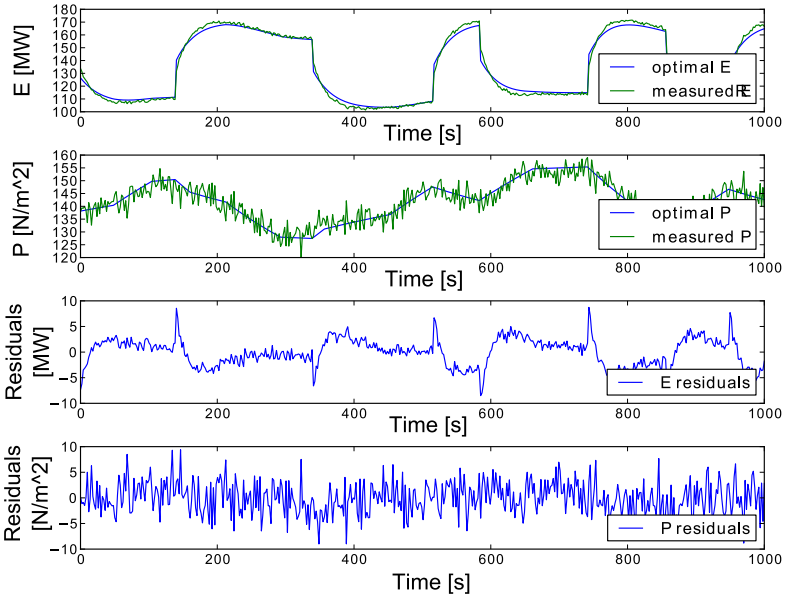

Figure 5.18 The system in Eq. (5.5) estimated with $\theta = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4]$. The residuals = optimal trajectory - measured signal .

For this step, the null model is given from the estimation of
$\theta_0 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4]$
and the more complex models are given from
$\theta_1 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4, T_D]$ and
$\theta_2 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4, T_R]$.
The initial guesses are given from θ_0 and can be seen in Table 5.15.

Table 5.15 Initial guess after estimated system in Eq. (5.5) with $\theta = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4]$.

Parameter	T_D [s]	T_R[s]	A_4	w_E	w_P
Initial Guess	250.0	30.0	0.393	9.59	-10.8
Parameter	x_{10}	x_{20}	r_E	r_P	
Initial Guess	149	27.4	7.03	10.0	

The results of the new models can be seen in Table 5.16.
Since the risk is zero for T_D, the model given from this estimation can be said to

Table 5.16 Cost statistics for the null model, $\theta_0 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4]$, for system in Eq. (5.5).

Released parameter	T_D	T_R
Cost	3093.534	4714.292
Cost Reduction	27.863	-1532.865
Risk	0.000	1.00

be better than the null model. The test results for this estimation can be seen in Fig. 5.19. Note that the cost reduction for releasing T_R is negative, this is probably due to that the optimization converged to an unwanted minimum.

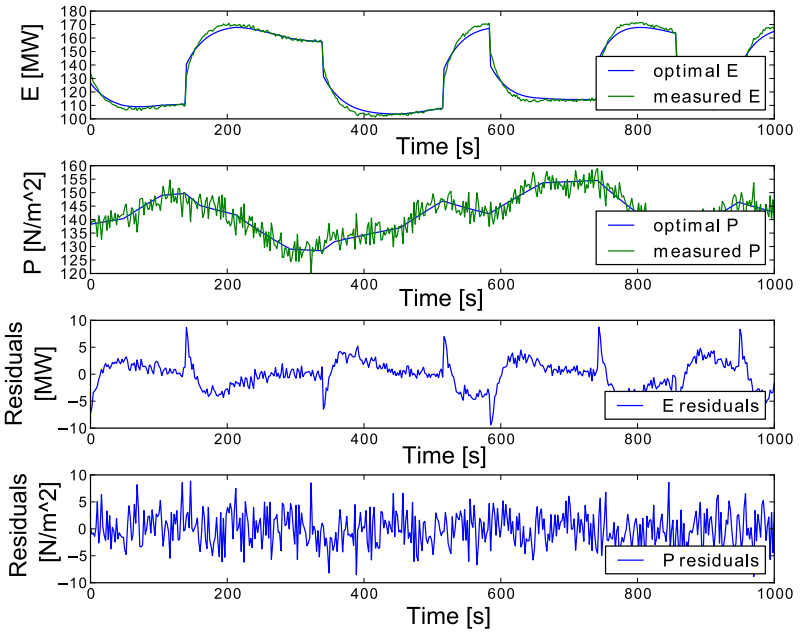

Figure 5.19 The system in Eq. (5.5) estimated with $\theta = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4, T_D]$. The residuals = optimal trajectory - measured signal .

There are still some errors greater than one can expect from white Gaussian noise, as can be seen in the residuals. Therefore the procedure goes forward and the new null model is given from
$\theta_0 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4, T_D]$

and the conditional model is given from
$\theta_1 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4, T_D, T_R]$.

Table 5.17 Initial guess after estimated system in Eq. (5.5) with $\theta = [x_{10}, x_{20}, r_E,$- $r_P, w_P, w_E, A_4, T_D]$.

Parameter	T_D $[s]$	T_R $[s]$	A_4	w_E	w_P
Initial Guess	272	30.0	0.405	9.68	-10.6
Parameter	x_{10}	x_{20}	r_E	r_P	
Initial Guess	149	27.5	7.04	9.46	

Table 5.18 Cost statistics for the null model, $\theta_0 = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4, T_D]$, for system in Eq. (5.5).

Released parameter	T_R
Cost	1765.775
Cost Reduction	1327.759
Risk	0.000

With the initial guesses in Table 5.17, the system was estimated with all parameters free and the cost statistics can be seen in Table 5.18. There is no risk to release T_R and the best model is found by estimating all parameters. The optimal trajectory can be seen in Fig. 5.20 and one can see the significant improvement compared to the initial model in Fig. 5.15. The real parameters used in the simulation to generate the measurement data compared to the estimated parameters can be seen in Table 5.19. One can see that all parameters except for r_P are estimated with an error less than 2% compared to the real value.

Table 5.19 Real and estimated parameters for system in Eq. (5.5)

Parameter	T_D $[s]$	T_R $[s]$	A_4	w_E	w_P
Real Value	285	19.0	0.260	10.0	-10.0
Estimated Value	283	19.1	0.260	10.0	-10.2
Parameter	x_{10}	x_{20}	r_E	r_P	
Real Value	148	28.0	0.50	10.0	
Estimated Value	148	28.0	0.50	9.41	

To investigate the robustness of the estimation the final estimated parameters were used to calculate the confidence interval with bootstrapping for each parameter, as described in Sec. 4.3.2. The 95% confidence interval, calculated from 500

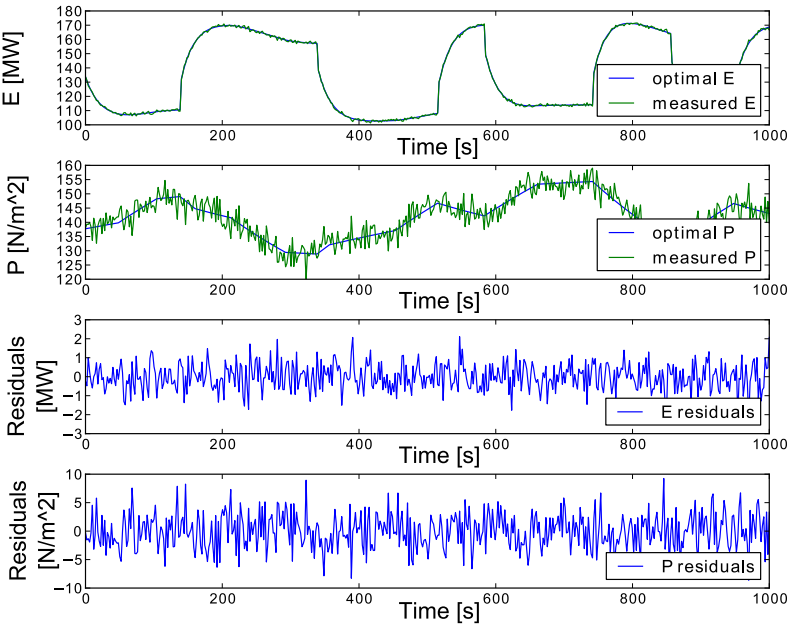

Figure 5.20 The system in Eq. (5.5) estimated with $\theta = [x_{10}, x_{20}, r_E, r_P, w_P, w_E, A_4, T_D, T_R]$. The residuals = optimal trajectory - measured signal .

simulations of the model given from the estimated values, is shown in Table 5.19 and the estimated relative error is shown in Table 5.21.

Table 5.20 Estimated confidence intervals for the final estimated parameters for system in Eq. (5.5)

Parameters	T_D [s]	T_R [s]	A_4	w_E	w_P
Bias	0.0%	-0.1%	-0.3%	-0.1%	-0.1%
Interval min	-0.9%	-4.8%	-9.8%	-3.0%	-1.1%
Interval max	1.0%	4.6%	9.1%	2.8%	0.9%
Parameters	x_{10}	x_{20}	r_E	r_P	
Bias	0.0%	0.2%	-0.5%	-1.2%	
Interval min	-0.2%	-2.5%	-12.4%	-13.1%	
Interval max	0.2%	3.0%	11.3%	10.8%	

Table 5.21 Estimated relative error for the final estimated parameters for system in Eq. (5.5)

Parameters	T_D $[s]$	T_R $[s]$	A_4	w_E	w_P
Relative Error	-0.7%	-0.5%	0.0%	0.0 %	2%
Parameters	x_{10}	x_{20}	r_E	r_P	
Relative Error	0.0%	0.0%	0.0%	-5.9%	

One can see that the real values are in the confidence interval except for the estimation of w_p. Since the confidence intervals are not a guarantee for the real value this could happen and it is not unlikely to happen for 1 of 9 parameters with 95% confidence intervals.

5.2.3 Heat Exchanger

The interactive method was applied to a simplified heat-exchanger model in order to fit to a more complex model of the system. This was done as an experimental example to see if it is possible to use the method for other purposes than just identifying the system from real measurement data. The complex model was simulated and since the simulation was noise free, OE was used as identification method. For this identification the measurements came from the complex model.

In Table 5.22 the step by step values for each parameter to estimate is shown. Step 1 is the initial model to improve and step 7 is the final model that came out from the interactive method.

One can see that C_3 could not be estimated and that some parameter values changed drastically compared to the initial model. A drawback with this identification was that the initial state parameters $[p_0, h_{out0}, T_{wall0}]$ were dependent to the other unknown parameters. The initial model compared to the measurements can be seen in Fig. 5.21 and the final improved model can be seen in Fig. 5.22. If the target is to have a model that just follows the two outputs given to the optimization, the model has been improved. It is obvious from the plot that the initial states were hard to estimate since there is a clear offset in the two output signals. There was also a long sequence that was constant which does not contain any information about the system and therefore is decreasing the performance of the estimation. This sequence was kept in the optimization because it facilitated the estimation procedure to estimate over one time sequence.

Table 5.22 The estimated parameter values step by step for the Heat Exchanger. A grey cell corresponds to a free parameter.

Parameter	C_1	C_2	C_3	α_{wall}	p_0	h_{out0}	T_{wall0}
step 1	0.0680	0.800	0.400	3900	$1.70 \cdot 10^7$	$3.39 \cdot 10^6$	811
step 2	0.821	0.800	0.400	3900	$1.70 \cdot 10^7$	$3.39 \cdot 10^6$	811
step 3	119	0.800	0.400	527	$1.70 \cdot 10^7$	$3.39 \cdot 10^6$	811
step 4	119	0.800	0.400	527	$1.70 \cdot 10^7$	$1.00 \cdot 10^6$	811
step 5	874	0.00	0.400	826	$1.70 \cdot 10^7$	$1.00 \cdot 10^6$	811
step 6	806	0.00	0.400	867	$1.70 \cdot 10^7$	$1.00 \cdot 10^6$	800
step 7	806	0.00	0.400	869	$1.71 \cdot 10^7$	$1.00 \cdot 10^6$	800

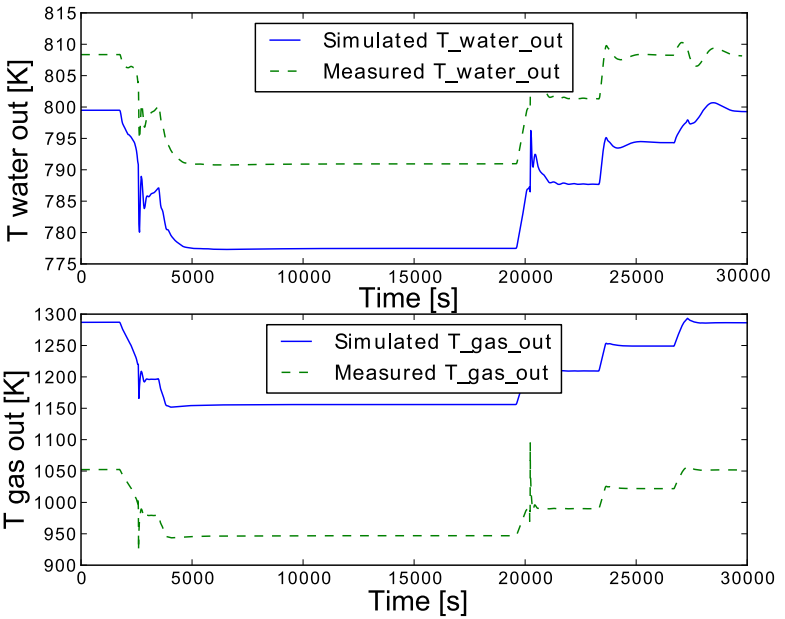

Figure 5.21 The simulated model from step 1 in Table 5.22 compared to the measured signal.

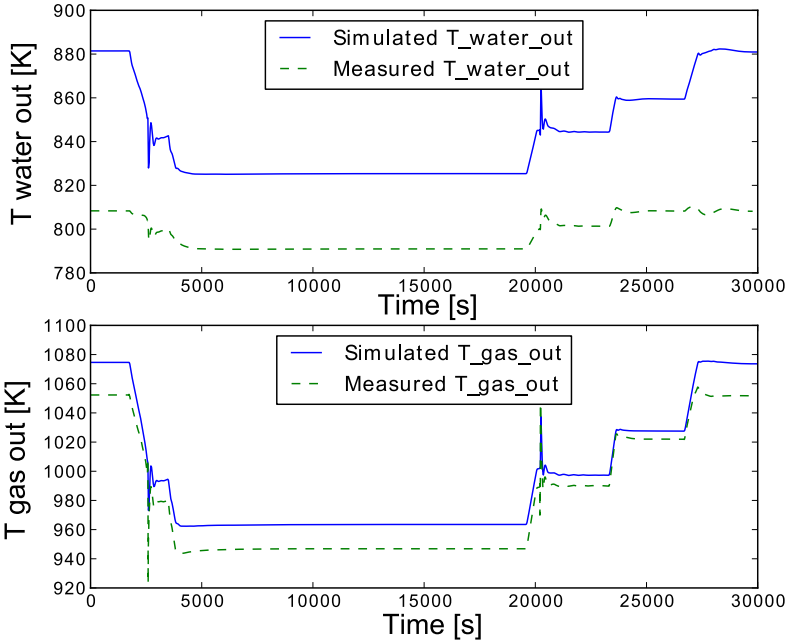

Figure 5.22 The simulated model from step 7 in Table 5.22 compared to the measured signal.

6

Discussion and Conclusion

In this work the possibilities of implementing grey-box identification in JModelica.org have been investigated. Focus was on Maximum Likelihood (ML) estimation for processes with white Gaussian measurement noise, in order to identify improved parameter values for Modelica models from measurement data.

The behavior of the software and how to use it was investigated followed with a method for identification and analyze the estimates. The conclusions given from this work are focusing on challenges and performance of the implementation.

6.1 Discussion

Because no solution was found to change the discretization in the optimization, the discretization in the simulation was instead changed to implicit Euler to reduce the discretization error. To be able to estimate parameters for real systems without bias, the discretization in the optimization has to be improved to reduce the discretization error.

The identification works for simulations with implicit Euler. Estimation bias due to discretization seems to come from differences in discretization between the measurement data generation and the optimization. It is very likely that the implementation will work for real systems, if it is possible to minimize a sum in Optimica with a better discretization than implicit Euler.

A possible way to get around the discretization problem might be to estimate the measurement noise by itself and then use the estimated noise covariances as fixed weights in the optimization. Then it is possible to improve the discretization by choosing another discretization method with more than one collocation point.

The confidence interval has been calculated for each parameter independently. The advantage with that implementation is that it is easy to implement but also easy to analyze. A less conservative confidence interval for the whole system could be achieved by calculating a confidence region that treats all parameters together, but then the result is more complex to analyze since it is very likely that the parameter estimates are correlated.

The sample size larger than 500-1000, depending on the system, was affecting the solver. If the sample size has been given too large it has been impossible for the solver to converge. So depending on the model and the number of free parameters, the sample size has been upper limited. At the same time, the theoretical performance increases with a large sample size. The sample sizes used in this work was been chosen with that in mind.

When the two estimation methods, OE and ML estimation, were used to estimate parameters for different noise intensities, the differences for equal noise intensities was shown to be small. If the measurement noise covariance is known, the noise parameters in the optimization can be fixed to the corresponding noise to reduce the degrees of freedom. However, the advantage was shown to be small and so fixing the noise parameters might not make a big difference.

When the discretization effects the estimation, another method that could be useful was presented in [Garatti and Bittanti, 2013]. This method is often more robust for systems with bad initial guesses that might converge to an unwanted minimum. It also disregards the distribution of the noise which is an advantage for unknown noise distributions.

6.2 Conclusions

The implementation can estimate noise intensity parameters for pure noise signals with negligible errors compared to the theoretical values. For ML estimation implemented for nonlinear systems, the parameter estimation distribution looked like what is expected from a working ML estimation.

In order to generate sums to minimize, the optimization had to be discretized using the implicit Euler method. The same system was simulated, with implicit Euler and with the default method in JModelica.org, in order to generate measurement data. It was demonstrated that the simulation with implicit Euler resulted in better estimations. This is probably due to the identification trying to account differences in discretization by changing parameters.

Maximum Likelihood (ML) and Output Error (OE) estimation has been implemented and was compared for different ratios of measurement noise covariances. For same noise intensities, the OE estimation gave a little bit lower estimation error. For noise covariance with different ratios, ML estimation gave a significantly lower estimation covariance.

ML estimation for white Gaussian measurement noise was tested for a system with two-point noise distribution. The optimization manage to find an optimal trajectory close to the real system when using a large sample size. The parameter estimate distribution given from measurements with two-point noise distribution were similar to what one can expect from white Gaussian measurements. On the other hand, the residuals differ from the residuals given from white Gaussian noise and reveal the two-point distribution.

An interactive method for how to approach grey-box identification was presented. The implemented interactive method was used for a simple RC circuit to see how over-parametrization is handled. It was shown that the interactive method falsified the option of releasing too many parameters, which indicated that it works fine for over-parametrization.

Bootstrapping was used to get confidence intervals for the estimated parameters for a drum boiler model. For this model, the dynamics of the model were all known and all parameters were free in the final estimation. For eight of nine parameters, the 95 % confidence interval that was calculated contained the real parameter value. This indicates that the bootstrapping method works fine for the purpose of analyzing the robustness of the estimation.

As a final case in this work, OE estimation was used to fit a simplified model to data from a more complex model. The interactive method was used and the model was improved with respect to minimizing the cost function. However, there was still a clear difference between the optimal trajectory and the signal given from the more complex model. A drawback with this model was that the initial state values were dependent of the unknown parameters so it was hard to estimate the correct initial states. Therefore, no real conclusion about the robustness of the iterative method for this purpose can be drawn.

6.2.1 Future Work

In this work all measurement data was generated from simulations. The target is however to improve models for real systems. A next step would be to try to apply grey-box identification with the implementation in this work for real systems but with similar complexity as the models used in this work.

The main focus with this work was to investigate how to do the optimization for identification. The other side of the coin, how the model design affects the optimization needs further investigations.

The work has been limited to system with only measurement noise, identifying nonlinear systems with process disturbances using ML estimation has not been investigated. More work has to be done in order to implement a method for systems with process disturbances.

Since this work has been limited to system with white Gaussian measurement noise, the implementation is limited. To be able to identify parameters for system with other noise distributions more analysis has to be done. For example, ML estimation might not be asymptotically unbiased for other distributions.

The implementation of the iterative identification was focused on the approach and need to be more user friendly in order to be easy to use.

A

Optimica code

A.1 Drum Boiler

```
model DrumBoiler
        //parameters
        parameter Real T_D = 250.0; //drum time constant [s]
        parameter Real TR = 30.0; // Reheater time constant [s]
        parameter Real A4 = 0.5; // Drum yield
        parameter Real w_E = 0.0; //Power disturbance
        parameter Real w_P = 0.0; //Preassure disturbace
        parameter Real x10 = 0.0; //initial state
        parameter Real x20 = 0.0; //initial state

        //constants
        parameter Real K1(free = false) = 0.3;
        parameter Real A1(free = false) = 14;
        parameter Real A2(free = false) = 3.3;
        parameter Real K2(free = false) = 0.064;
        parameter Real A3(free = false) = 15;
        parameter Real K3(free = false) = 0.32;

        input Real uc; //control valve position
        input Real fc; // fuel flow [kg/s]

        Real x1(fixed = true, start = x10);
        Real x2(fixed = true, start = x20);

        Real E; //Power[M/W]
        Real P;  //Drum pressure[n/m^2]

        equation
                der(x1) = K1*(A1*fc-A2*uc*x1)/T_D;
                der(x2) = K2*(A2*uc*x1-A3*x2)/TR;
                E = K3*(A4*A2*uc*x1 + (1-A4)*A3*x2) + w_E;
```

```
                P = x1 + w_P;


end DrumBoiler;

optimization ML(objectiveIntegrand= (v1^2)*(r1)^(-1) +
        (v2^2)*(r2)^(-1) + log(r1*r2),
        startTime=0.0, finalTime = 1000.0)

        //noise covariances
        parameter Real r1(free = true,initialGuess = 1.0,
                 min = 0.0, max = 100.0)=1.0;
        parameter Real r2(free = true,initialGuess = 1.0,
                 min = 0.0, max = 100.0)=1.0;

        DrumBoiler DB(T_D = T_D,TR = TR, A4 = A4, w_E = w_E,
                w_P = w_P, x10 = x10, x20 = x20);


        parameter Real T_D(free = true, initialGuess = 250.0,
                 nominal = 284.87, min = 0,
                 max = 500) = 250.0; //drum time constant [s]

        parameter Real TR(free = true, initialGuess = 30.0,
                nominal = 19.0, min = 0,
                max = 100) = 30.0; // Reheater time constant [s]

        parameter Real A4(free = true, initialGuess = 0.393,
                 nominal = 0.26, min = 0.0,
                  max = 1.0) = 0.5; // Drum yield

        parameter Real w_E(free = true,
                initialGuess = 9.59) = 0.0; //Power disturbance

        parameter Real w_P(free = true,
                initialGuess = -10.8) = 0.0; //Preassure disturbace

        parameter Real x10(free = true,
                initialGuess = 0.0,
                nominal = 148) = 0.0; //initial state

        parameter Real x20(free = true,
                initialGuess = 0.0,
                nominal = 28) = 0.0; //initial state

        //controlsignals
```

```
        input Real uc;
        input Real fc;

        input Real E;
        input Real P;

        Real v1;
        Real v2;

        equation
                v1 = (DB.E-E);
                v2 = (DB.P-P);
                uc = DB.uc;
                fc = DB.fc;
end ML;
```

Bibliography

Barszcz, T. and P. Czop (2011). "Estimation of feedwater parameters based on a grey-box approach". *Int. J. Appl. Math. Comput. Sci.* **21** (4), pp. 703–715.

Bohlin, T. (1991). *Interactive Sytem Identification: Prospects and Pitfalls*. Springer-Verlag, Heidelberger Platz 3, Berlin, Germany.

Bohlin, T. (2006). *Practical Grey-Box Process Identification - Theory and Application*. Springer-Verlag, Heidelberger Platz 3, Berlin, Germany.

Garatti, S. and S. Bittanti (2013). "A new paradigm for parameter estimation in system modeling". *Int. J. Adapt. Control Signal Process* **27**, pp. 667–687.

Gunnar, J., E. Wernholt, G. Hovland, and T. Brogårgh (2006). "Nonlinear grey-box identification of linear actuators containing hysteresis". *IEEE International Conference on Robotics and Automation, Orlando-Florida, May 2006*.

Isaksson, A. J. (2013). "Some aspects of industrial system identification". *10th IFAC International Symposium on Dynamics and Control - Dycops, Mumbai, India, 18-20 Dec 2013*.

Isaksson, A. J., D. Törnqvist, J. Sjöberg, and L. Ljung (2010). "Grey-box identification based on horizon estimation and nonlinear optimization". *Proc. of the Symposium on Stochastic Systems (SSS'09), Kobe, Japan, Nov 2009*.

Li, J. and R. Ding (2013). "Parameter estimation methods for nonlinear systems". *Applied Mathematics and Computation* **219**, pp. 4278–4287.

Magnusson, F. (2012). *Collocation methods in JModelica.org*. Master's Thesis ISRN LUTFD2/TFRT--5892--SE. Department of Automatic Control, Lund University, Sweden.

Maruta, I. and S. Toshuharu (2013). "Projection-based identification algorithm for grey-box continuous-time models". *System and Control Letters* **62**, pp. 1090–1097.

Mendel, J. M. (1995). *Lessons in Estimation Theory for Signal Processing, Communications and Control*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Modelica Association (2014). *Modelica and the Modelica association*. URL: `https://www.modelica.org/`.

Modelon AB (2013). *Jmodelica.org user guide*. URL: `http://www.jmodelica.org/api-docs/usersguide/1.11.0/`.

Python Software Foundation (20140328). *Python homepage*. URL: `https://www.python.org/`.

Sørlie, J. A. (1996). *On Grey-Box Model Definition and Symbolic Derivation of Extended Kalman Filters*. PhD Thesis TRITA--REG--9601. Department of Signals, Sensors and Systems, Royal Institute of Technology, Sweden.

The Scipy Community (2013). *Numpy reference*. URL: `http://docs.python.org/2/library/`.

Zoubir, A. M. and D. R. Iskander (2007). "Bootstrap methods and applications". *Signal Processing Magazine, IEEE* **24** (4), pp. 10–19.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER´S THESIS

Date of issue
April 2014

Document Number
ISRN LUTFD2/TFRT--5941--SE

Author(s)
Elias Palmkvist

Supervisor
Toivo Henningsson, Modelon AB
Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle

Implementation of Grey-Box Identification in JModelica.org

Abstract

Grey-box identification is a tool to identify and improve nonlinear system models by estimating parameters. The estimation is done by optimizing a cost function using measurement data. The robustness of the estimations can then be analyzed with statistics. JModelica.org is a platform for modeling and optimization of dynamical models. In order to do grey-box identification one need models and be able to optimize. JModelica.org supports modeling and optimization so it has a huge potential to support grey-box identification. So far there is no complete solution for grey-box identification in JModelica.org. This work is focusing on how to implement greybox identification in JModelica.org in order to estimate parameters for nonlinear models. The theory of grey-box identification has been investigated as well as the possibilities with JModelica.org. Finally, an interactive method to estimate model parameters and a method to calculate the confidence intervals for the estimates have been implemented. The implementation has been tested for nonlinear models and works as expected.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-64

Recipient's notes

Security classification

http://www.control.lth.se/publications/