Sjöberg J., Fyhr F., Grönstedt, T.:
Estimating parameters in physical models using MathModelica
2nd International Modelica Conference, Proceedings, pp. 325-330

Paper presented at the 2nd International Modelica Conference, March 18-19, 2002,
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Oberpfaffenhofen, Germany.

All papers of this workshop can be downloaded from
http://www.Modelica.org/Conference2002/papers.shtml

Estimating parameters in physical models using MathModelica*

Jonas Sjöberg, Fredrik Fyhr and Tomas Grönstedt
Department of Machine- and Vehicle Systems
Chalmers University of Technology
412 96 Gothenburg, Sweden
Phone +46-31-772 1855, Fax: +46-31-772 3690
Email: jonas.sjoberg@me.chalmers.se

Abstract

This paper describes a program which extends *MathModelica* so that model parameters can be estimated using measured data. Given initial values of the parameters, the parameter estimates are iteratively changed so that the sum of squared errors of the difference between the model output and the data is minimized. In each iteration an extended differential equation has to be simulated. The developed program imports the Modelica model into *Mathematica* and derives a symbolic expression for this extended differential equation. The extended model i converted to Modelica format and MathModelica is used to simulate it efficiently.

1 Introduction

A mathematical model of a plant can be based on well-known physical laws. These physical laws often contains parameters which numerical values might not be exactly known. There might be, for example, spring constants, masses, resistances and other basic parameters. The program described here has been developed to estimate such parameters using measured signals from the system.

Estimating models of dynamic systems is called system identification. It is a well established engineering research field. Introductory books are, for examples, [5, 3], and more advanced ones [4, 6]. These books, and available software tools, deal with either linear models or discrete time models. For many real world problems there is a need of nonlinear continuous-time models.

There are many reasons to estimate parameters in models built on physical principles. Some examples follows.

- Some parameters are maybe only approximately known.
- The change of the parameters in the estimation can be used as a way to validate the original model.
- The system might need to be re-tuned after age and wear.
- An online version of the program could be used for monitoring and failure detection of plants in continuous use.

The current program builds on *MathModelica*. The rational for this is that when the user has obtained a model for simulation, no extra effort is needed to estimate its parameters. This is in opposite to most existing identification tools of today where you have to transform the model into their special format. It is also a question of flexibility, thanks to the great generality of Modelica you can specify almost any type of model. This can either by done by using the Modelica syntax in a *Mathematica* notebook or by using the graphical user interface, Figure 1, where you build a model by combining sub models from different libraries.

The following example illustrates the idea of the program.

Example: Consider the electric circuit in Figure 2. It is easy to build this model with the model editor. The only non-standard part is the resistor which is nonlinear and described by

$$u_R = R_1 i + R_2 i^5$$

where u_R is the voltage, i the current, and R_1 and R_2 are parameters. There are also parameters describing the inductance, L and the capacitor, C.

*Financial support from Volvo Aero Corporation AB is gratefully acknowledged.

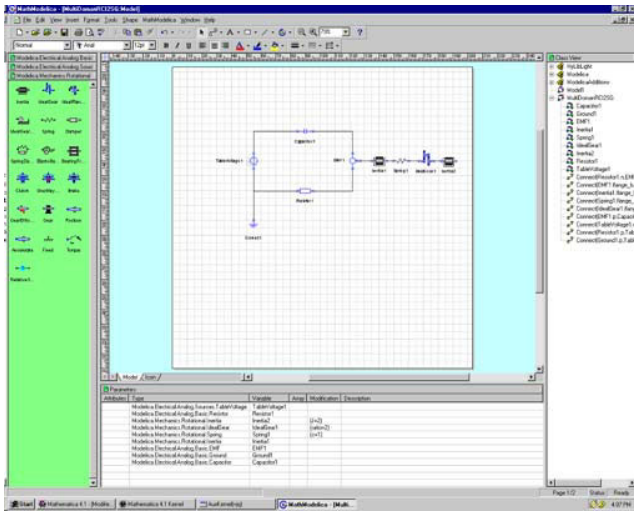

Figure 1: MathModelica model building editor.

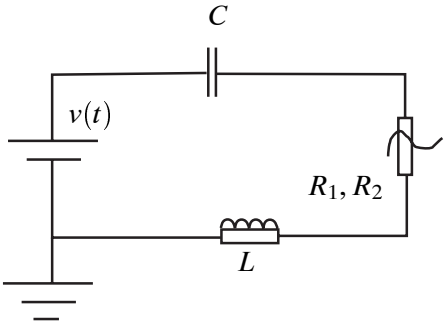

Figure 2: A nonlinear circuit with unknown parameters C, L, and R_1 and R_2. Voltage over the resistor is described by $R_1 i + R_2 i^5$

Data was obtained by simulating the model with "true" parameter values and with a step voltage of 10 Volt at $t = 0$. The obtained current is displayed in Figure 3. The sampled data values used in the parameter estimation are indicated with dots. The model was initialized

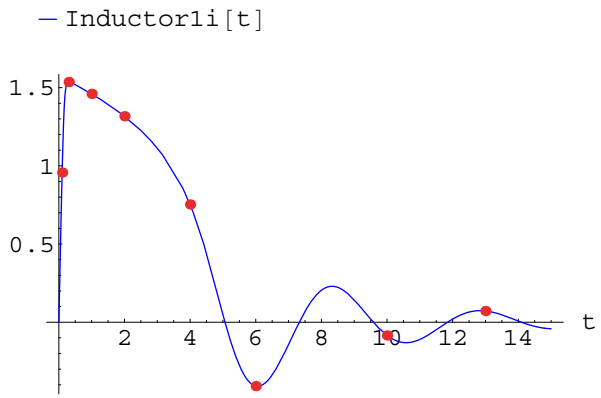

Figure 3: Current through the inductor of the circuit in Figure (2). The marked values are data samples used to estimate the parameters.

with parameter values different from the ones used to obtain the data, as indicated in Table 1. From the initialization the parameters where iteratively improved using the developed program. In Figure 4 the simulated current is shown after each iteration. As seen in the figure, the simulated values coincide with the data after some 6-7 iterations.

Parameter	True value	Initial estimates
R_1	0.5	2
R_2	1	2
C	0.5	2
L	1	2

Table 1: True parameters used to obtain the data and initial parameter values used in the optimization.

□

The rest of the paper is organized in the following way. Section 2 gives the mathematical description of the considered system identification problem. How this theory is solved by the program is described in Section 3. Another example is given in Section 4 which is followed by a discussion on possibilities and problems with the current approach in Section 5. The paper is then concluded in Section 6.

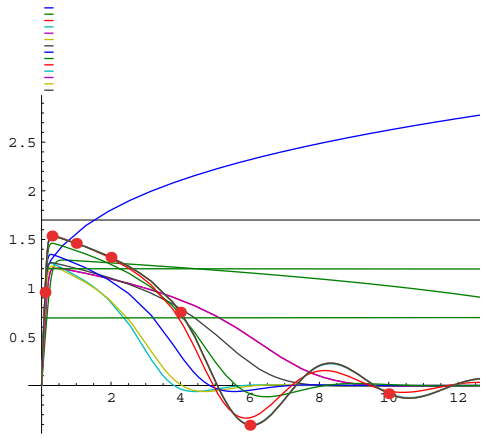

Figure 4: Simulated current after consecutive parameter estimate updates.

2 The equations describing the calculations

Assume that we are interested of a specific system and consider a model of it, described by a differential equation, DAE or ODE

$$f(\dot{x}(t),x(t),u(t),\theta) = 0 \quad (1)$$
$$\hat{y}(t,\theta) = h(x(t),\theta) \quad (2)$$

where $x(t)$ are the states of the model, $u(t)$ is the input signal, $y(t)$ is the output signal. The differential equation is then specified by the functions f and h which also depend on the parameters which are stored in a parameter vector θ.
Assume further that a data set of N samples has been collected from the system, $\{y(t),u(t)\}_{t=1}^{N}$. The goal is then to tune θ so that the simulated output $\hat{y}(t)$ resembles $y(t)$ when the model is simulated with the input $\{u(t)\}_{t=1}^{N}$. This is obtained by introducing a criterion of fit. It can be almost any arbitrary differential function, but to keep things easy we choose the mean squared error

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}(y(t)-\hat{y}(t,\theta))^2 \quad (3)$$

Then the estimate is defined as

$$\hat{\theta} = \arg\min_{\theta} V_N(\theta) \quad (4)$$

It is generally not possible to find a closed form expression for $\hat{\theta}$. Instead, starting with an initial parameter guess $\hat{\theta}^{(0)}$ the estimate is iteratively computed by a gradient based algorithm

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} - \mu R^{-1}\left.\frac{dV_N(\theta)}{d\theta}\right|_{\theta=\hat{\theta}^{(i)}} \quad (5)$$

where R is a positive definite matrix approximating the Hessian, and μ is a step length to assure descent steps. Different standard minimization algorithms, for example Gauss-Newton, Levenberg-Marquardt and steepest-descent, are covered by (5) and they differ on the choice of R. See, eg, [1, 2].
A key part in the iterative minimization (5) is the computation of the derivative of the criterion. It becomes

$$\frac{dV_N(\theta)}{d\theta} = -\frac{2}{N}\sum_{t=1}^{N}(y(t)-\hat{y}(t,\theta))\frac{d\hat{y}(t,\theta)}{d\theta} \quad (6)$$

This leads us to the derivative of the model output

$$\frac{d\hat{y}(t,\theta)}{d\theta} = \frac{\partial h(x(t),\theta)}{\partial\theta} + \frac{\partial h(x(t),\theta)}{\partial x(t)}\frac{dx(t)}{d\theta} \quad (7)$$

which cannot be obtained without the signal

$$\tilde{x}(t) = \frac{dx(t)}{d\theta} \quad (8)$$

To obtain this signal we have to take the derivative of the original state space equation in (1). This gives us

$$\frac{\partial f(\dot{x}(t),x(t),u(t),\theta)}{\partial\dot{x}(t)}\frac{d\dot{x}(t)}{d\theta} + \frac{\partial f(\dot{x}(t),x(t),u(t),\theta)}{\partial x(t)}\frac{dx(t)}{d\theta} + \frac{\partial f(\dot{x}(t),x(t),u(t),\theta)}{\partial\theta} = \frac{\partial f(\dot{x}(t),x(t),u(t),\theta)}{\partial\dot{x}(t)}\dot{\tilde{x}}(t) + \frac{\partial f(\dot{x}(t),x(t),u(t),\theta)}{\partial x(t)}\tilde{x}(t) + \frac{\partial f(\dot{x}(t),x(t),u(t),\theta)}{\partial\theta} = 0 \quad (9)$$

which is a new differential equation. Since it contains $x(t)$ it is coupled with the original differential equation (1) describing the model.
By introducing

$$z(t) = \begin{bmatrix} x(t) \\ \tilde{x}(t) \end{bmatrix} \quad (10)$$

the two coupled differential equations (1) and (9) can be described as

$$F(\dot{z}(t),z(t),u(t),\theta) = 0 \quad (11)$$

where the definition of F follows from (1) and (9).
Hence, to perform the iterative minimization (5) the differential equation (11) has to be simulated in each iteration using the current value of θ.

3 The program

The different parts of the *Mathematica* program can now be described in more detail:

1. Given a Modelica model, describing (1), with initial parameter values and data from the true system.
2. A subset of the parameters are selected for estimation.
3. The extended differential equation (11) is symbolically computed from the original model (1) and transformed into Modelica standard.
4. The extended differential equation (11) is simulated with the current parameter values.
5. The selected parameters are updated (5).
6. Until convergence, go to 4.

The main part of the program is the derivation of the extended model. The other steps consists of interface issues or well-known algorithms which have to be included into the program.

4 Example

A first example was given already in the introduction. Here follows a second one where we have a different type of nonlinear resistor. The system is described in Figure 5. The input to the system is the voltage at the voltage source and the output is the current. The

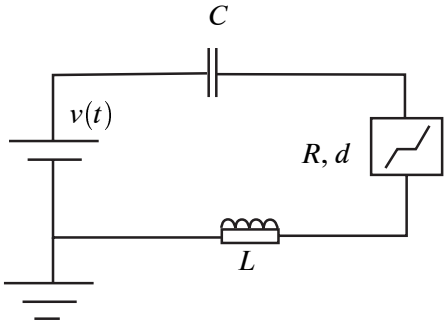

Figure 5: Nonlinear circuit with a parameterized dead zone.

resistor is described by an unknown resistance, R, and a dead zone, d, see Figure 6.
Estimation data was obtained by selecting a set of "true" parameters, given in Table 2 and simulating the model with a step input of 5 Volt. Figure 7 depicts the

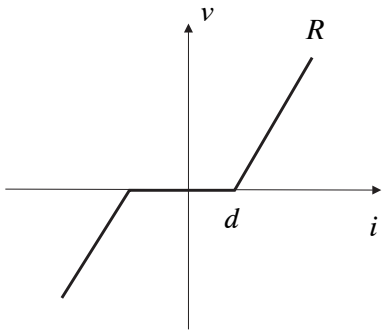

Figure 6: Description of the dead zone parameterization.

voltage over the resistor and one can clearly see the cut-off due to the dead zone.

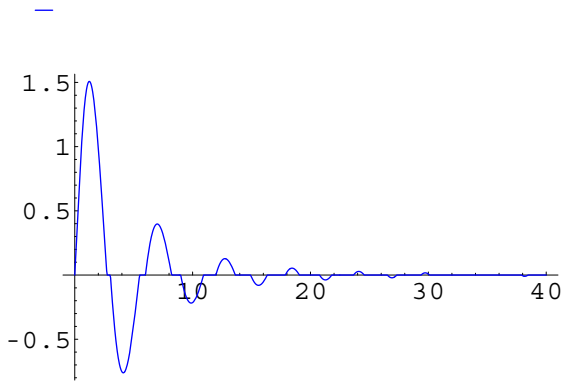

Figure 7: Simulation of the true system, the voltage over the resistor versus time.

The model was initialized with parameter values as indicated in Table 2. The simulation of the initial model together with the estimation data are depicted in Figure 8.

Parameter	True value	Initial value	Final value
R	0.5	0.7	0.503
d	0.4	0.35	0.402
C	0.8	1.1	0.805
L	1	1.1	0.993

Table 2: Parameter values for the circuit with a dead zone in the resistance.

The result of the tuning is illustrated in Figure 9 where the simulated current is depicted after each iteration together with the estimation data. Table 2 gives the final parameter values. From the figure it is clear that

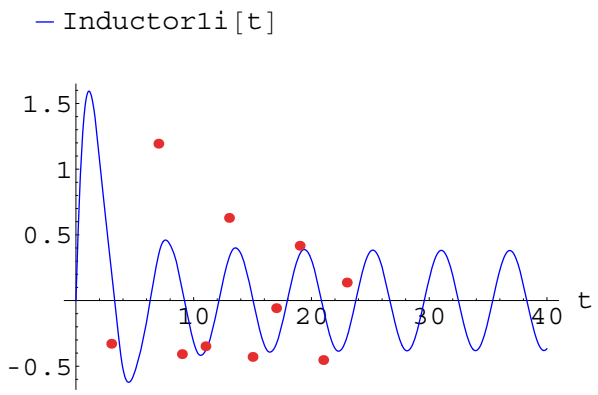

Figure 8: Simulation of the model with the initial parameters together with estimation data.

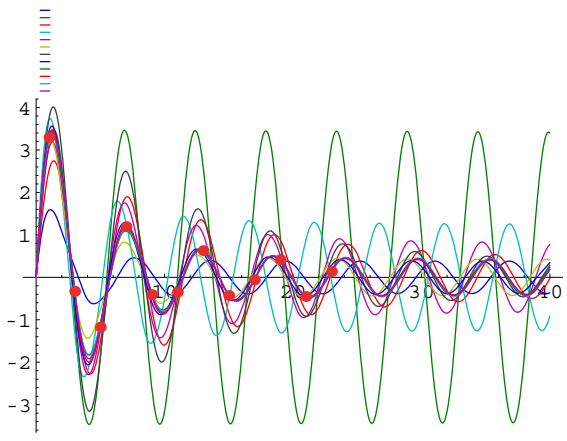

Figure 9: Simulated current after consecutive parameter estimate updates.

the parameters converge within some 10 iterations.

5 Discussion

Although the program is far from ready the following functionality is supported, or can easily be supported by making smaller changes.

- Data can be sampled at irregular sampling instances. Different signals can be sampled individually.
- Systems described by DAE can be handled in the same (automatic) way as ODE systems. The example in the introduction was actually a DAE example.
- Discontinuous (but piecewise smooth) differential equations can be handled, at least a formal result can be obtained.
- Multiple input multiple output systems are handled.
- Criterion of fit can be changed.

There are also potential problems.

- The gradient search can only guarantee convergence to a local minimum. Hence, a good initial parameter guess is necessary to obtain convergence to the global minimum.
- The order of the extended differential equation is often high, it becomes the number of parameters times the number of states in the original differential equation. This gives a high computational burden which might limit the applicability of the program.
- Stability problems may occur. Depending on the parameter values the differential equations might be stable or unstable. In the general case, where the model is nonlinear, it is not possible to monitor stability.

6 Conclusions

A *Mathematica* program has been developed which extends MathModelica so that parameters can be estimated using measured data. The program builds on the following principles.

- An existing modeling tool, the MathModelica graphical user interface, is used to describe the model.
- Mathematica is used to create the to the model specific extended differential equation which needs to be simulated in the estimation process.
- Existing, efficient numerical differential equation solver is used to simulate the extended differential equation.

So far only preliminary studies have been carried out. More experience is needed and the program has to be developed further before it becomes as easy to use, as it is supposed to be.

References

[1] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[2] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987.

[3] R. Johansson. *System Modeling & Identification*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1993.

[4] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1999.

[5] L. Ljung and T. Glad. *Modeling of Dynamic Systems*. Prentice Hall, Englewood Cliffs, N.J., 1994.

[6] T. Söderström and P. Stoica. *System Identification*. Prentice-Hall International, Hemel Hempstead, Hertfordshire, 1989.