TOOLS FOR SEMI-PHYSICAL MODELING

P. LINDSKOG and L. LJUNG

Department of Electrical Engineering, Linköping University, S-581 83 Linköping, SWEDEN,
Email: lindskog@isy.liu.se, ljung@isy.liu.se

Abstract. By semi-physical modeling we mean such an application of system identification, where physical insight into the application is used to come up with suitable nonlinear transformations of the raw measurements, so as to allow for a good model structure. Semi-physical modeling is less "ambitious" than physical modeling, in that no complete physical structure is sought, just suitable inputs and outputs that can be subjected to more or less standard model structures, such as linear regressions. In this contribution we discuss various tools that can support the process of semi-physical modeling. We will deal both with analytical tools such as differential algebra and more informal ones such as the programming environment.

Key Words. Grey box modeling; least-squares estimation; nonlinear transformations, semi-physical modeling; software tools; system identification

1. INTRODUCTION

System identification is applied to a wide variety of processes. Loosely speaking the idea is to fit a model structure to observed data, so as to come up with a model which as well as possible reproduces the past behavior of the system.

The crux in this process is really to find a suitable model structure for the process and for the intended application of the model. Several approaches to the design of such a model structure have been described and discussed earlier in the literature:

1. *Black boxes:* By black boxes we mean a family of (usually linear) models, whose parameters do not have physical significance, but where the objective is to find a good linear system that fits the observed data. One could, in case of linear models, think of black box structures as direct parameterizations of the frequency functions.

2. *Physically parameterized models:* These are the results of more or less laborious modeling, where all the physical insight about the behavior of the process is condensed into a model, usually in state space form, which contains both known and unknown parameters. The unknown parameters describe the model structure, and they typically have a physical significance of their own, such as unknown physical constants *etc.*

Between these two "extremes" on the scale of design of a model structure there is a zone where considerable and important physical insight is used in the identification process, but not to the extent that a formal physically parameterized model is constructed. In this contribution we call this "middle zone" *semi-physical modeling.* This process of identification is of course by no means new; however we shall explore some of its key features and discuss what formal and informal tools will help the user in this process.

2. SEMI-PHYSICAL MODELING

By semi-physical modeling we will mean the process to take physical insight about the behavior of the system into account, to use that insight to find adequate nonlinear transformations of the raw measurements so that the new variables – the new inputs and outputs – stand a better chance to describe the true system when they are subjected to standard model structures (typically linear in the new variables).

Let us illustrate the point in a toy example.

Suppose that we want to build a model for how the voltage applied to an electric heater affects the temperature in a room. Physical modeling entails writing down all equations relating to *the power of the heater, heat transfer, heat convection and so on. This involves several complicated equations, expressions, unknown heat transfer coefficients and so on. A simple black box approach would instead be to use, say, an ARX model (a linear difference equation) with the applied voltage as the input and the room temperature as the output. But that is too simple! A moment's reflection reveals that it is the heater power rather than the voltage that causes the temperature to change. Thus try to use a linear difference equation with the squared voltage as input and the room temperature as output.*

Clearly, semi-physical modeling is in frequent use in practice. It is however also true that many failures of the identification process are to be blamed on not applying this principle. No doubt, several of these failures are due to the lack of relevant and general software tools that support a systematic and interactive modeling procedure. The objective herein is therefore to discuss and point at tools that can be useful for this kind of modeling.

3. AN ILLUSTRATIVE EXAMPLE

The oil crises in the early seventies highly motivated the search for alternative and more environmental friendly energy sources. It became, *e.g.*, popular to utilize solar energy for the heating of houses. Data from such a heating system, earlier investigated by Ljung (1987), will in this section be used to emphasize and highlight steps involved in a typical semi-physical modeling session.

Consider the solar-heated house in Fig. 1.

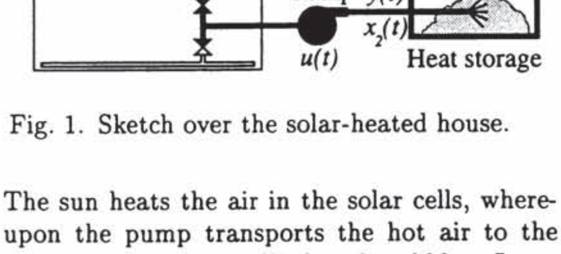


Fig. 1. Sketch over the solar-heated house.

The sun heats the air in the solar cells, whereupon the pump transports the hot air to the heat storage – a box filled with pebbles. Later during the night the energy flow is reversed and the house is heated. The modeling aim is to describe the storage inlet temperature and in-

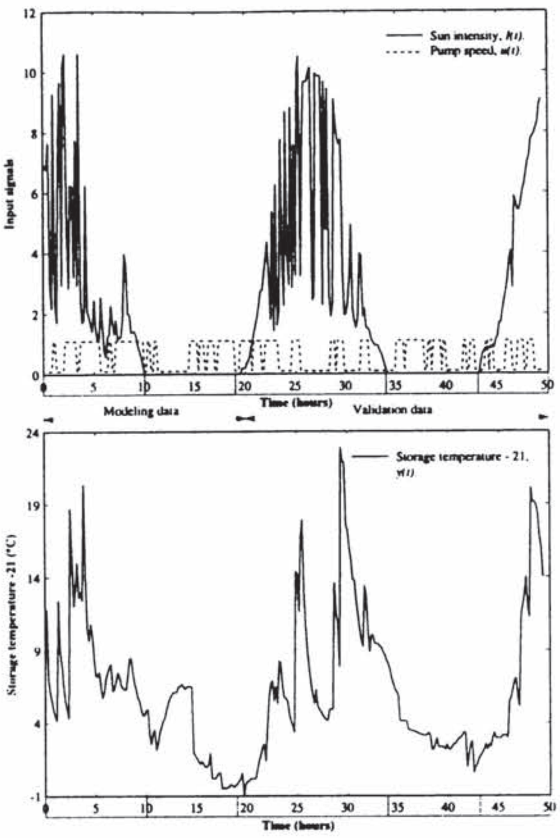


Fig. 2. Sun intensity $I(t)$, pump speed $u(t)$ (input signals) and inlet storage temperature $y(t)$ (output signal). Grey time slots indicate darkness during the night.

vestigate how it is affected by the pump speed and the sun intensity. At our disposal there are three measured signals:

$I(t)$ the sun radiation at time instance t (a non-controllable input),

$u(t)$ the pump speed (a controllable input), or actually a binary variable indicating if the pump is on or off, and

$y(t)$ the storage inlet temperature (the output).

Measures of these signals were taken every tenth minute over a period of forty-eight hours, *i.e.*, 296 samples. The first 120 samples (20 hours) were exclusively used for model building, whereas the remaining data were reserved for validation tests. Next, the outdoor mean temperature (21° C) was removed from the storage inlet temperature, and in order to avoid numerical problems (division by zero) a constant level of 0.1 was added to the pump speed. The resulting signals are given in Fig. 2.

As a preliminary (guided by the "try simple things first" principle) it is of interest to see how

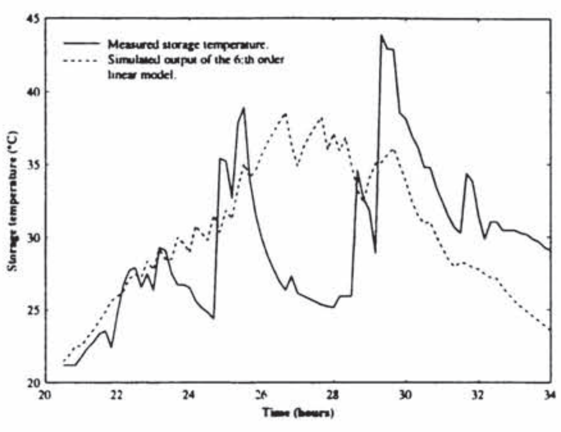


Fig. 3. Measured storage temperature compared to the simulated output of the sixth order linear model.

an ordinary linear model structure, such as

$$\begin{aligned} y(t) &= -a_1 y(t-1) - a_2 y(t-2) \\ &\quad + b_1 u(t-1) + b_2 u(t-2) \\ &\quad + c_1 I(t-1) + c_2 I(t-2) \end{aligned} \tag{1}$$

would perform on this data. Focusing on the second day-time period, it is clear from the simulation detailed in Fig. 3 that such a model has great difficulties in explaining the solar house dynamics. As a matter of fact, staying within this linear model class and varying the model order does not help – the same kind of discrepancy between measured and simulated output is still there.

If the "try simple things first" principle is unacceptable a good advice is to proceed to the slightly more complicated "try simple physical things next". In our case this means that simple physical insight into the heating process should be taken into account. After a moment's reflection one realizes that a linear model is not very realistic since the sun intensity and the pump speed hardly are additive. The solar panel, and indirectly the sun intensity, can scarcely affect the storage temperature when the pump is off. Instead we should expect a multiplicative relationship between the available input signals.

To investigate this suspicion we can see what happens if the system is modeled according to the law of conservation of energy. As a first step let $x_1(t)$ denote the mean solar panel temperature and let $x_2(t)$ equal $y(t)$. In discrete time the supply of energy to the solar panel from one time to the other (proportional to $x_1(t+1) - x_1(t)$) is equal to the energy injected into the system minus the energy lost to the surroundings. The way the energy is supplied or lost might in reality be governed by very complicated relationships, but in semi-physical modeling one should strive to simplify them. Therefore, assume that the sun directly is responsible for the entire supply of energy ($\eta_1 I(t)$), and that the losses to the surroundings are divided into two parts: losses to the environment ($\eta_2 x_1(t)$) and losses to the storage when the pump is operating ($\eta_3 x_1(t) u(t)$), *i.e.*,

$$\begin{aligned} x_1(t+1) - x_1(t) &= \eta_1 I(t) - \eta_2 x_1(t) \\ &\quad - \eta_3 x_1(t) u(t). \end{aligned} \tag{2}$$

Similarly, the supply of energy to the storage over a certain time period (proportional to $x_2(t+1) - x_2(t)$) equals the surplus energy from the solar panel ($\eta_3 x_1(t) u(t)$) minus the losses through the storage walls ($\eta_4 x_2(t)$), *i.e.*,

$$x_2(t+1) - x_2(t) = \eta_3 x_1(t) u(t) - \eta_4 x_2(t). \tag{3}$$

If only the temperature in the panel was measured Equation (2) would have been redundant and η_3 and η_4 could have been estimated directly from the latter equation. Since this is not the case $x_1(t)$ must be eliminated, which here can be done in the following way. First, determine $x_1(t)$ and its shifted cousin $x_1(t+1)$ from (3). Next, substitute the resulting two expressions into (2). Solving for $y(t) = x_2(t)$ finally gives the input-output description

$$\begin{aligned} y(t) &= (1-\eta_4) y(t-1) \\ &\quad + \eta_1 \eta_3 u(t-1) I(t-2) \\ &\quad - \eta_3 u(t-1) y(t-1) \\ &\quad + \eta_3 (1-\eta_4) u(t-1) y(t-2) \\ &\quad + (1-\eta_2) \frac{u(t-1) y(t-1)}{u(t-2)} \\ &\quad - (1-\eta_2)(1-\eta_4) \frac{u(t-1) y(t-2)}{u(t-2)}. \end{aligned} \tag{4}$$

The original physical parameters enter this equation in a complicated nonlinear fashion. However, the model can be reparameterized as

$$\begin{aligned} &\theta_1 = (1-\eta_4), && \varphi_1(t) = y(t-1), \\ &\theta_2 = \eta_1 \eta_3, && \varphi_2(t) = u(t-1) I(t-2), \\ &\theta_3 = -\eta_3, && \varphi_3(t) = u(t-1) y(t-1), \\ &\theta_4 = \eta_3 (1-\eta_4), && \varphi_4(t) = u(t-1) y(t-2), \\ &\theta_5 = (1-\eta_2), && \varphi_5(t) = \frac{u(t-1) y(t-1)}{u(t-2)}, \\ &\theta_6 = -(1-\eta_2)(1-\eta_4) && \\ & && \varphi_6(t) = \frac{u(t-1) y(t-2)}{u(t-2)}, \end{aligned}$$

which gives a system expressed as a linear regression:

$$y(t) = \sum_{i=1}^{6} \theta_i \varphi_i(t) = \theta^T \varphi(t). \tag{5}$$

Although the linearized parameters are known it might here be impossible to substitute back

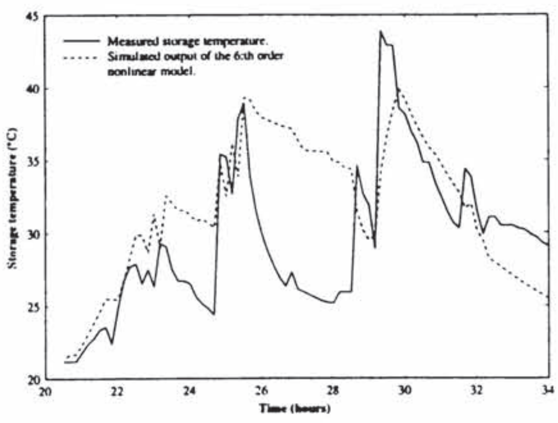


Fig. 4. Measured storage temperature compared to the simulated output of the sixth order nonlinear model.

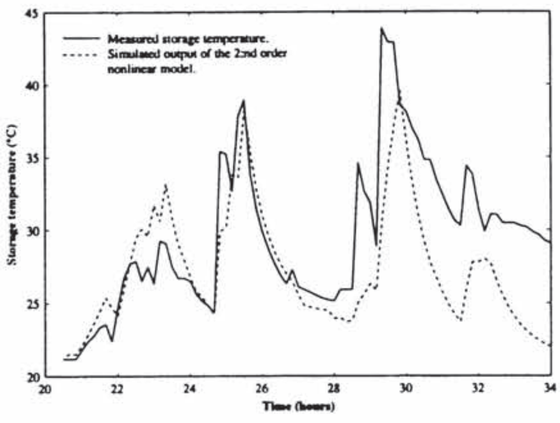


Fig. 5. Measured storage temperature compared to the simulated output of the second order nonlinear model.

and uniquely determine the original parameters η. Since the original equations already are approximations this is often a price worth paying.

This is as far as symbolic calculations can be of any help in the assignment of delivering a suitable model structure. It is now the task of an estimator to come up with reasonable parameter values. As Structure (5) is a linear regression, its parameters can be estimated efficiently by the least-squares algorithm. As can be seen from the simulation detailed in Fig. 4 the sixth order nonlinear model performs better than the previously discussed linear one. The mean temperature deviation (4.1 compared to 4.6 for the linear model) is, however, still too high.

It should at this point be asked if all of the suggested regressors are equally important in explaining the output. Or posed another way, can some of the regressors be removed without a decrease in the prediction ability of the model? By only picking the first two regressors from the proposed model structure, *i.e.*,

$$y(t) = \theta_1 y(t-1) + \theta_2 u(t-1) I(t-2), \quad (6)$$

the answer is according to Fig. 5 obvious – some of the regressors seem to be redundant. In fact, they are not only redundant, their inclusion results in a worse model, which mainly is due to so-called "overfit" to the modeling data.

To lastly summarize the example the mean temperature of the second order nonlinear model is 3.3 and as we expected the significant input signal is a mixture between $I(t)$ and $u(t)$.

4. REQUIRED TOOLS

From the solar house example it should be clear that semi-physical modeling requires both *symbolic* and *numerical* calculations. Let us in more detail and from a user's perspective dwell on these demands.

The first and most challenging step is to obtain a useful model structure, *i.e.*, dynamic relationships between the input signals $\boldsymbol{u}(t)$ and the measured outputs signals $\boldsymbol{y}(t)$. Based on physical considerations the model builder is often able to write down some few equations, which are believed to pick up the essential features of the plant in question. Apart from the measured signals, the given equations typically contain nonmeasurable variables $\boldsymbol{x}(t)$, known physical constants and a set of unknown parameters η. From these equations the problem of finding a suitable model structure can be stated as follows.

Given a set of user defined equations generate, without changing the solution space, a new set of equations which do not depend on $\boldsymbol{x}(t)$.

In terms of algorithms this requirement suggests a general symbolic equation solver which eliminates the unknown signals from the original set of equations. In a broad context these equations may contain arbitrary dynamics and nonlinearities, combinations of discrete- and continuous-time variables, time delays and so on. The given problem is for such a general framework considered to be extremely hard.

However, if all the given expressions are *differential-algebraic*, *i.e.*, all relations between $\boldsymbol{u}(t)$, $\boldsymbol{y}(t)$, $\boldsymbol{x}(t)$, η and their time derivatives (of arbitrary order) are purely algebraic, the situation is more manageable. In practice, this means that there exist algorithms, which are able to solve the stated problem. One such algorithm, computational similar to ordinary Gauss elimination, was outlined by Ritt (1950) in the 1930's (see also Glad (1992) for software details).

Ritt's algorithm only works for time-continuous systems, and it is not immediate to extend it to the time-discrete case. But as data is collected using computers this situation must also be supported. One way is to stay within the differential-algebraic framework (and only allow differential and pure algebraic equations, DAE's for short), apply Ritt's algorithm and finally transform the result into the time-discrete domain by approximating the derivatives with suitable discrete expressions. Another possibility would be to rely on so-called *difference algebra*, although very few algorithmic tools exist in this area today.

A second tool for solving polynomial differential equations originate from *commutative algebra* (see Cox *et al.* (1992) for a basic introduction and overview). Algorithmically, the unknown variables are eliminated by computing a particular so-called *Gröbner basis* for the system of equations according to a scheme invented by Buchberger (1985) in the 1970's.

Although these symbolic algorithms are different in nature, one cannot say that one approach is always superior to the others. The differential-algebraic framework seems to be better if the original equations are unstructured, since the equations without modifications can be sent to the solver. Gröbner basis calculations on the other hand typically require some modifications of the equations. For systems given in state-space form these modifications are especially simple, which means that both approaches very well can be used. However, one major advantage with commutative algebra is that difference equations (at least those condensed into a state-space form) can be handled. This was the approach pursued in solar house example.

At this point the practiced model designer would most likely object and claim that many model descriptions do not fit into the polynomial framework. Exponentials are, *e.g.*, commonly encountered in chemical and biological modeling, while other structures often include trigonometric functions. Thus, the model builder must at least be given the freedom to use simple nonlinearities such as sines and square-roots in his or her equations. It turns out that this can be done in many cases, and hence we need a pre-processing algorithm of the following kind.

Given a set of equations, transform this set into a form convenient for the available polynomial algorithms, without throwing away possible solutions.

The mentioned algorithms are in general very demanding in terms of computational time and space complexity. It is not unusual that systems compounded of some three or four equations and variables require several hours of CPU time and some 10 Mb of memory. Beforehand it would therefore be worth a lot, even though we have to resort to heuristic arguments, to be able to determine some sort of upper complexity bound for the set of equations. If the complexity is just too high the advice is either to reduce the number of equations, or alternatively, to simplify some of the nonlinear relationships. To achieve this a second pre-processing tool is needed:

Before sending the set of equations to a solver decide if the problem is computational feasible.

The result from the equation solver is finally rearranged so as to fit into a linear regression framework. Hence, the last symbolic tool required is a post-processing algorithm:

Given an input-output description transform, if necessary, it to a time-discrete counterpart, solve for $y(t)$ and collect the suggested combinations of old inputs and outputs in a regression vector $\varphi(t)$. Also, introduce new parameters θ so that $y(t) = \theta^T \varphi(t)$ holds.

The reason for this kind of transformation is, of course, that the new, and linear, parameters can be efficiently (with respect to computational burden, numerical accuracy *etc.*) estimated from data by using the *least-squares algorithm*. It should again be emphasized that any physical significance of the original parameters normally are lost in this operation.

The number of regressors delivered from the symbolic package might in practice be very high. For a number of reasons, including model understanding and computational accuracy, a good low order model is preferred before a higher order one. For large systems an "all possible subset" search for the few best regressors is as good as impossible due to the combinatorial explosion. Fortunately, there exist quite a few standard statistical procedures for sorting out the most relevant regressors. Two such algorithms both resembling a statistical F-test, the *stepwise regression* and the *backward elimination* procedures (see, *e.g.*, Draper and Smith (1981)), were interactively used to select the regressors of the solar house model. The last need is thus a combined estimation and model reduction algorithm:

From measured data and a number of suggested regressors estimate a "good" model of low order.

It is true that the result from the solver can have physical significance, but in the reparameterization and the model reduction steps we have disregarded this fact. Hence, the term *semi-physical* modeling.

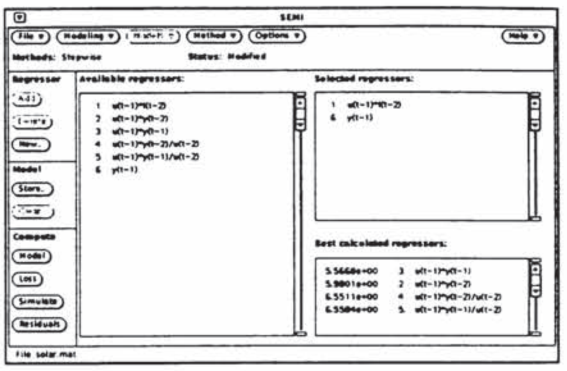
5. SOFTWARE PACKAGE

For numerical calculations there exist several interactive, efficient and wide spread software packages, like $Matrix_X$ and *Matlab*. The computer standard of today has also made symbolic computations feasible, and general and commercial products such as *Maple* and *Mathematica* have emerged during the last few years. Although these packages stand alone are powerful, we believe that in semi-physical modeling there is a need for some kind of supervisor, which is responsible for simplifying and administering the computations. The process of sorting out a few regressors is, *e.g.*, highly interactive and subject to the designer's thoughts and ideas about what a good model really is.

To improve the situation we suggest that a semi-physical modeling tool should consist of a graphical oriented user interface for entering equations and variables, handling data and last but not least for aiding in selecting the regressors. Furthermore, means for validation (simulation, residual tests *etc.*) must also be included. As several models typically are investigated in parallel an opportunity to store the most promising ones in a model data base is desirable.

To meet these demands we have developed the SEMI software tool (whose interface is shown in Fig. 6). In SEMI the symbolic and numerical services are provided by using *Maple* and *Matlab* as computational engines, which are running as separate processes. These processes are invoked via a graphical user interface (coded in *C*). Besides administering and presenting the calculations, this program is also responsible for keeping track of earlier investigated models. For symbolic equation solving, either Ritt's algorithm or Gröbner basis procedures can be employed. The only symbolic requirement of Sec. 4 not yet available is the one concerning the model structure complexity. The numerical services provided by SEMI include a least-squares minimization algorithm, means for model reduction and choice of regressors (the stepwise regression and backward elimination schemes) and some tools for model validation.

A session with SEMI typically shows the following steps. First, the variables of the system are specified, whereupon the equations are entered as *Maple* expressions. Next, the unknown variables are eliminated (if possible) by the chosen equation solver, which returns a number of possible regressors (displayed in Fig. 6 under "Available regressors:"). After having specified the data set to use, the final step is to interactively use the model reduction schemes and the validation tools mentioned earlier to (hopefully)



```
Methods: Stepwise                    Status: Modified

Available regressors:
1  u(t-1)*I(t-2)
2  u(t-1)*y(t-2)
3  u(t-1)*y(t-1)
4  u(t-1)*y(t-2)/u(t-2)
5  u(t-1)*y(t-1)/u(t-2)
6  y(t-1)

Selected regressors:
1  u(t-1)*I(t-2)
6  y(t-1)

Best calculated regressors:
5.5668e+00   3  u(t-1)*y(t-1)
5.9801e+00   2  u(t-1)*y(t-2)
6.5511e+00   4  u(t-1)*y(t-2)/u(t-2)
6.5584e+00   5  u(t-1)*y(t-1)/u(t-2)

File solar.mat
```

Fig. 6. The SEMI interface. The system under investigation is the solar house.

find a suitable model. Using SEMI this procedure was the one followed in the solar house example.

6. CONCLUSION

In this contribution we have discussed the tools of the user in the process of semi-physical modeling. We believe that this process is, or at least should be, a useful and commonly used way to come up with good model structures for identification. To support semi-physical modeling in practice we have developed the SEMI software.

7. REFERENCES

Buchberger, B. (1985). Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. In Bose, N., editor, *Multidimensional Systems Theory*, pages 184–232. Dordrecht Reidel.

Cox, D., Little, J., and O'Shea, D. (1992). *Ideals, Varietes, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate texts in mathematics. Springer.

Draper, N. and Smith, H. (1981). *Applied Regression Analysis.* John Wiley & Sons, New York, N.Y. USA, 2nd edition.

Glad, S. (1992). Implementing Ritt's Algorithm of Differential Algebra. Technical Report LiTH-ISY-I-1338, Department of Electrical Engineering, Linköping University, Linköping, Sweden.

Ljung, L. (1987). *System Identification: Theory for the User.* Prentice Hall, Englewood Cliffs, N.J. USA.

Ritt, J. (1950). *Differential Algebra.* American Mathematical Society, Providence, R. I. USA.