TOOLS FOR SEMIPHYSICAL MODELLING

P. LINDSKOG AND L. LJUNG

Department of Electrical Engineering, Linköping University, S–581 83 Linköping, Sweden

SUMMARY

Semiphysical modelling is often interpreted as an application of system identification where physical insight into the application is used to come up with suitable non-linear transformations of the raw measurements so as to allow for a good model structure. This modelling procedure is less 'ambitious' than those used for traditional physical modelling 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 paper we discuss a semiphysical modelling procedure and various tools supporting it. These include constructive algorithms originating from commutative and differential algebra as well as more informal tools such as the programming environment.

KEY WORDS: grey box modelling and identification; non-linear input-output transformations; linear regression; least squares estimation

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 reproduces as well as possible the past behaviour of the system.[1] In mathematical terms the problem is often considered as finding a function $g(t, \boldsymbol{\theta}, \boldsymbol{\varphi}(t))$ which maps the known (at time t) quantities $\boldsymbol{\varphi}(t)$ to the output $y(t)$ using a finite number of model parameters $\boldsymbol{\theta}$. Hence the output is given by

$$y(t) = g(t, \boldsymbol{\theta}, \boldsymbol{\varphi}(t)) \tag{1}$$

Once the structure of $g(t, \boldsymbol{\theta}, \boldsymbol{\varphi}(t))$ has been determined, it 'only' remains to estimate the parameters $\boldsymbol{\theta}$ and to decide whether the model is good enough or not.

The major crux in this process is really to find a suitable model structure, i.e. a suitable $g(\cdot)$, 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 model that fits the observed data. For linear models one can think of black box structures as direct parametrizations of the frequency functions in terms of different function expansions.
- *Physically parameterized models.* These are the results of more or less laborious modelling where all the physical insight about the behaviour 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 typically they have a physical significance of their own, such as unknown physical constants, etc.

Between these two 'extremes' on the scale of the 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 parametrized model is constructed. This is the 'middle zone' in focus in this contribution and we will refer to it as *semiphysical modelling*.[2]

The idea of utilizing simple physical insight in identification is by no means new and indeed is in frequent use in practice. However, the fact remains that many failures of the identification process can be blamed on not applying this principle.[2] These failures are no doubt mainly due to the lack of relevant and general software tools that support a systematic and interactive modelling procedure. It is the purpose of this paper to address this need by discussing and highlighting formal as well as more informal software tools that will help the user in this process.

The paper is organized as follows. The basic ideas behind our view of semiphysical modelling is discussed in the following section. In Section 3 we investigate data from a solar-heated house in order to identify general tools that are useful for semiphysical modelling. The types of formal algorithmic requirements identified in this investigation are discussed in Section 4, while in Section 5 we consider more software-oriented issues. In the latter section we also tie together the tools from Section 4 by describing a software tool, SEMI, which supports semiphysical modelling in practice.

2. SEMIPHYSICAL MODELLING

By semiphysical modelling we mean the process of taking physical insight about the behaviour of the system into account and using that insight to find adequate non-linear transformations of the raw measurements so that the new variables—the new inputs and outputs—stand a better chance of describing the true system when they are subjected to standard model structures (typically linear in the new variables). Let us illustrate the point via a toy example.

Example

Suppose that we want to build a model of how the voltage $u(t)$ applied to an electric heater affects the temperature $y(t)$ in a room. Physical modelling 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, etc. and is normally the result of a quite time-consuming procedure which often returns a complicated model that is hard to understand and analyse. On the other hand, a simple black box approach would be to use, say, an ARX model (a linear difference equation) with the applied voltage as input and the room temperature as output. However, 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, i.e. try something like

$$y(t) = g(t, \boldsymbol{\theta}, \boldsymbol{\varphi}(t)) = \theta_1 y(t-1) + \theta_2 y(t-2) + \theta_3 u^2(t-1) + \theta_4 u^2(t-2) = \boldsymbol{\theta}^T \boldsymbol{\varphi}(t) \quad (2)$$

with

$$\boldsymbol{\theta}^T = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix} \quad (3)$$

$$\boldsymbol{\varphi}(t) = \begin{bmatrix} y(t-1) & y(t-2) & u^2(t-1) & u^2(t-2) \end{bmatrix}^T \quad (4)$$

One major motivation for employing a structure such as (2) is that the unknown parameters $\boldsymbol{\theta}$ can be efficiently (with respect to computational burden, numerical accuracy, etc.) estimated using the *least squares algorithm*. This is an important observation, since in practice $g(\cdot)$ is searched for in a family of structures which usually hosts a large number of possibilities. Owing to the many advantages with least squares estimation, we are here searching for model structures that are linear in the parameters:

$$y(t) = g(t, \boldsymbol{\theta}, \boldsymbol{\varphi}(t)) = \sum_{i=1}^{n} \theta_i \varphi_i(t) \quad (5)$$

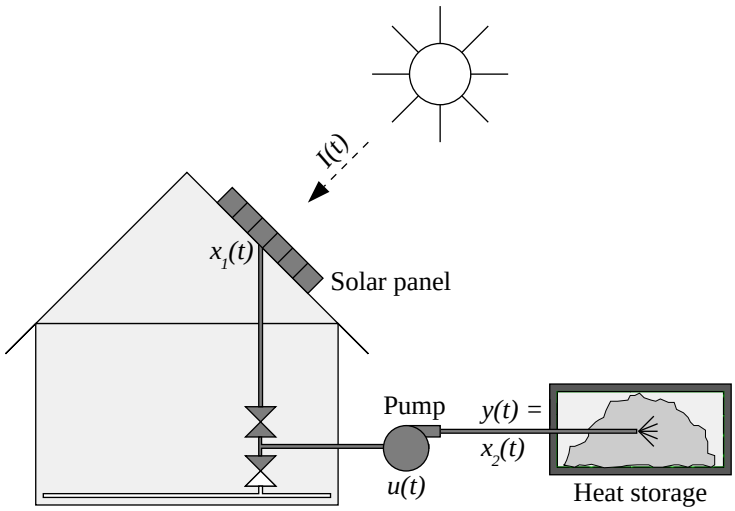

Figure 1. Sketch over the solar-heated house

Observe that the regressors $\varphi_i(t)$ can contain any combination of known variables, linear as well as non-linear. While these regressors can be the result of physical modelling and commonsense reasoning, the parameters may, as we will see in the following section, not have a direct physical interpretation. Hence the term *semi*physical modelling.

The obvious question now is how to arrive generally at a linear regression structure based on physical insight. Let us, by going through a simple but non-trivial application, outline a procedure for achieving this.

3. AN ILLUSTRATIVE EXAMPLE

The oil crises in the early 1970s greatly motivated the search for alternative and more environmentally friendly energy sources. It became popular, for example, to utilize solar energy for the heating of houses. Data from such a heating system investigated earlier by Ljung[1] will in this section be reused to illustrate a typical semiphysical modelling procedure and its usefulness.

Consider the solar-heated house in Figure 1. 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 modelling aim is to describe the storage inlet temperature and investigate how it is affected by the pump speed and the sun intensity. At our disposal are three measured signals:

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

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

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

To begin with, measures of these signals were taken every 10th minute over a period of 48 h, i.e. 296 samples were collected altogether. The first 120 samples (20 h) were exclusively used for model building, while the remaining data were reserved for validation tests. The outdoor mean temperature (21 °C) was then subtracted 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 shown in Figure 2.

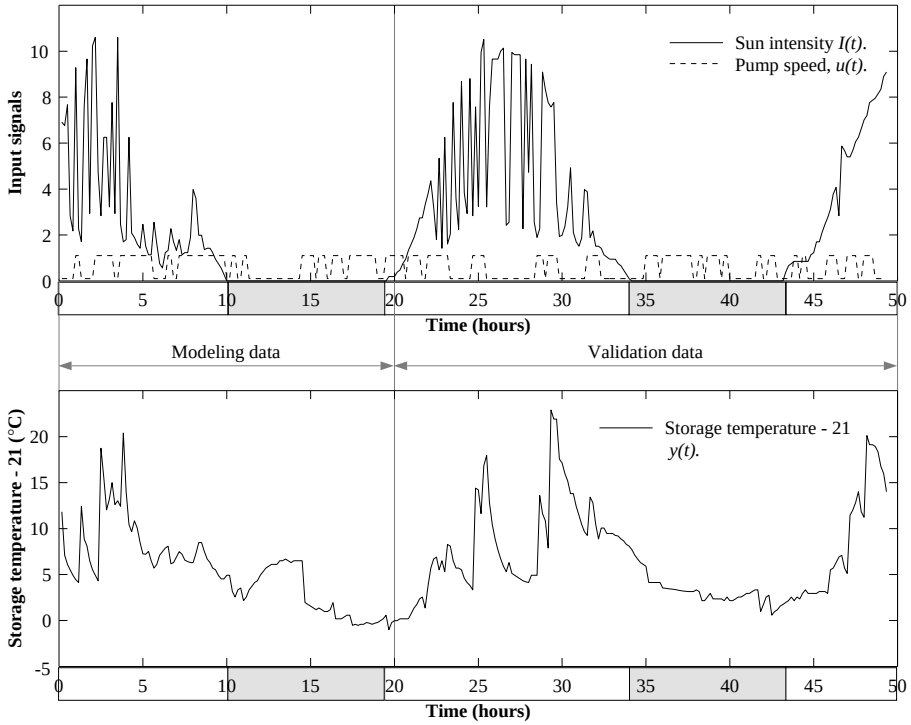

Figure 2. Sun intensity $I(t)$, pump speed $u(t)$ (input signals) and inlet storage temperature $y(t)$ (output signal). Grey time slots indicate darkness during the night

As a preliminary (guided by the 'try simple things first' principle) it is of interest to see how an ordinary linear model structure such as

$$y(t) = -\theta_1 y(t-1) - \theta_2 y(t-2) + \theta_3 u(t-1) + \theta_4 u(t-2) + \theta_5 I(t-1) + \theta_6 I(t-2) \quad (6)$$

will perform on these data. Focusing on the second daytime period, it is clear from the simulation detailed in Figure 3 that the least-squared-fitted model has severe difficulties in explaining the heating dynamics of the system. 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 outputs is still there.

If the "try simple things first" path is unacceptable, a reasonable first advice is to proceed to the slightly more complicated 'try simple physical things next'. In our case this means that 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 are hardly additive. The solar panel, and indirectly the sun intensity, can scarcely affect the storage temperature when the pump is off! We should instead expect a multiplicative relationship between the available input signals.

To investigate this suspicion, we can see what happens if the system is modelled 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 heating of the air in the solar panel (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 this kind of modelling one should strive to simplify them. Therefore assume that the sun is directly 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 ($\eta_3 x_1(t)u(t)$),

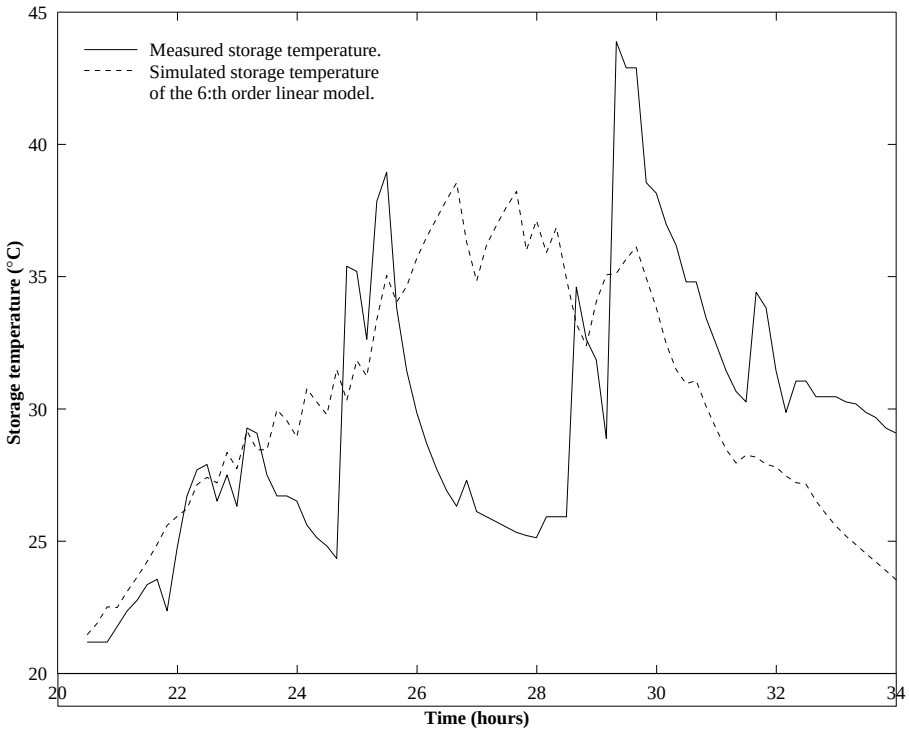

Figure 3. Measured storage temperature compared with the simulated output of the sixth-order linear model (structure (6))

i.e.

$$x_1(t+1) - x_1(t) = \eta_1 I(t) - \eta_2 x_1(t) - \eta_3 x_1(t)u(t) \tag{7}$$

In the same way, 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.

$$x_2(t+1) - x_2(t) = \eta_3 x_1(t)u(t) - \eta_4 x_2(t) \tag{8}$$

If only the temperature in the panel was measured, equation (7) 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 can be done in the following way. First determine $x_1(t)$ and its shifted relative $x_1(t+1)$ from equation (8). Next substitute the resulting two expressions into equation (7). Utilizing the fact that $y(t) = x_2(t)$, performing a time shift and solving for $y(t)$ finally gives the input-output description

$$\begin{aligned} y(t) = {} & (1-\eta_4)y(t-1) + \eta_1\eta_3 u(t-1)I(t-2) - \eta_3 u(t-1)y(t-1) + \eta_3(1-\eta_4)u(t-1)y(t-2) \\ & + (1-\eta_2)\frac{u(t-1)y(t-1)}{u(t-2)} - (1-\eta_2)(1-\eta_4)\frac{u(t-1)y(t-2)}{u(t-2)} \end{aligned} \tag{9}$$

Estimating these parameters would require some kind of iterative search scheme. From a computational point of view this is unfortunate, especially for complex systems with many parameters. To overcome this difficulty, it is appealing to carry out the reparametrization

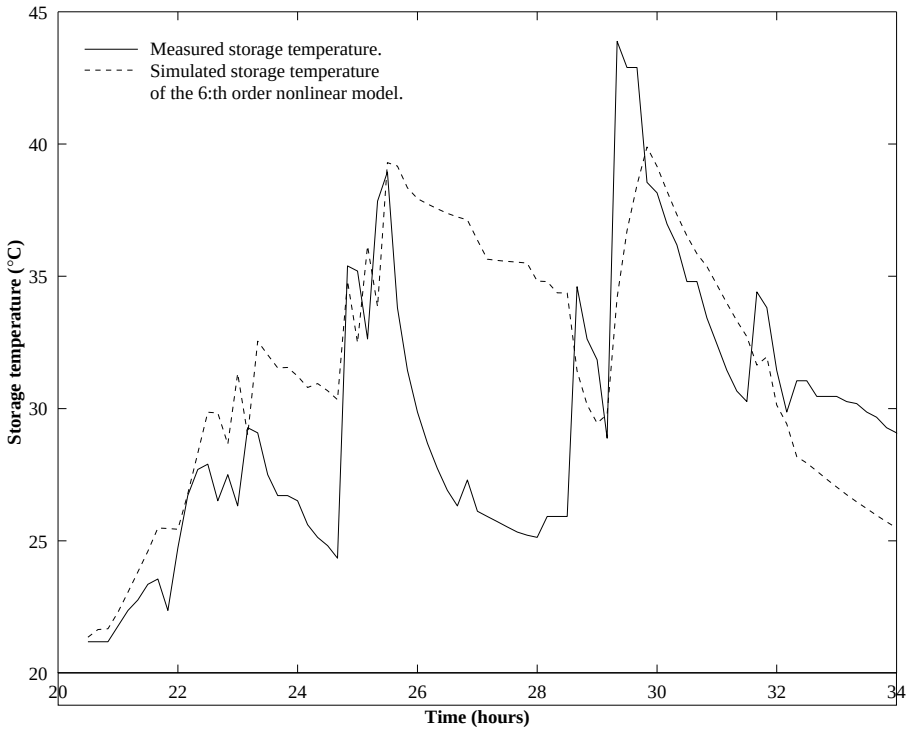

Figure 4. Measured storage temperature compared with the simulated output of the sixth-order non-linear model (structure (10))

$$
\begin{aligned}
\theta_1 &= g_1(\boldsymbol{\eta}) = (1-\eta_4) & \varphi_1(t) &= y(t-1)\\
\theta_2 &= g_2(\boldsymbol{\eta}) = \eta_1\eta_3 & \varphi_2(t) &= u(t-1)I(t-2)\\
\theta_3 &= g_3(\boldsymbol{\eta}) = -\eta_3 & \varphi_3(t) &= u(t-1)y(t-1)\\
\theta_4 &= g_4(\boldsymbol{\eta}) = \eta_3(1-\eta_4) & \varphi_4(t) &= u(t-1)y(t-2)\\
\theta_5 &= g_5(\boldsymbol{\eta}) = (1-\eta_2) & \varphi_5(t) &= u(t-1)y(t-1)/u(t-2)\\
\theta_6 &= g_6(\boldsymbol{\eta}) = -(1-\eta_2)(1-\eta_4) & \varphi_6(t) &= u(t-1)y(t-2)/u(t-2)
\end{aligned}
$$

which gives a system expressed as a true linear regression

$$y(t) = \sum_{i=1}^{6} g_i(\boldsymbol{\eta})\varphi_i(t) = \sum_{i=1}^{6} \theta_i\varphi_i(t) = \boldsymbol{\theta}^T\boldsymbol{\varphi}(t) \tag{10}$$

where we have a linear relationship between the new parameters $\boldsymbol{\theta}$ and the constructed regression vector $\boldsymbol{\varphi}(t)$. Notice that although the linearized parameters are known, it might be impossible here to substitute back and uniquely determine the original parameters $\boldsymbol{\eta}$. However, 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 assigning a suitable model structure. It is now the task of an estimator to come up with reasonable parameter values. Since Structure (10) is a linear regression, its parameters can be estimated using the least squares algorithm. As can be seen from the simulation detailed in Figure 4, the sixth-order non-linear model performs somewhat better than the previously discussed linear one. The mean temperature deviation (4.0 compared with 4.2 for the linear model) is, however, still too high.

It should be asked at this point whether all the 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

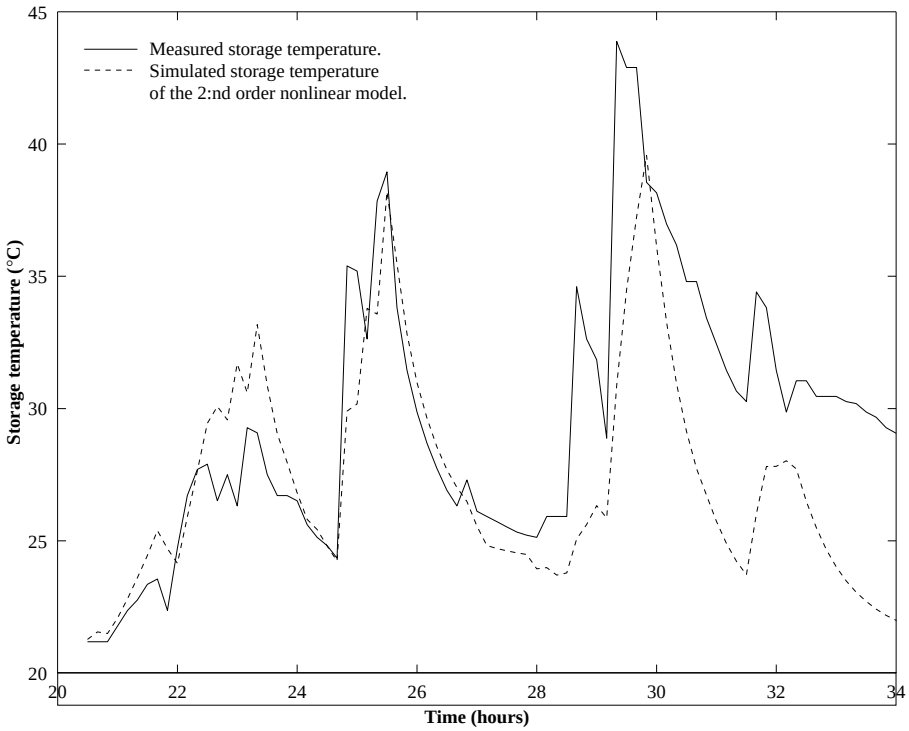

Figure 5. Measured storage temperature compared with the simulated output of the second-order non-linear model (structure (11))

the proposed model structure, i.e.

$$y(t) = \theta_1 y(t-1) + \theta_2 u(t-1) I(t-2) \tag{11}$$

the answer is according to Figure 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 is mainly due to so-called 'overfit' to the modelling data. The selection of only a few of the physically motivated regressors is another fact that justifies the term *semi*physical modelling.

To finally summarize the example, the mean temperature deviation of the second-order non-linear model is 3.1 and, as expected, the significant input signal is a mixture between $I(t)$ and $u(t)$. Notice also that the difference in time shift between the sun intensity and the pump speed has a nice physical interpretation: it takes one time unit to transport the energy from the solar panel to the pump and another time unit to transport it to the storage.

4. REQUIRED MODELLING TOOLS

From the previous section it should be clear that semiphysical modelling requires both *symbolic* and *numeric* computations. Let us dwell on these demands in more detail from a user's perspective. This section is more or less directed towards the class of tools desired without paying that much attention to algorithmic details. For those issues we just give the appropriate references.

4.1. Symbolic computations

The first and clearly most challenging step is to obtain a useful model structure, i.e. dynamic relationships between the input signals $\boldsymbol{u}(t)$ and the measured output signal $y(t)$. Based on

physical considerations, the model builder is often able to write down some fundamental equations which, at least approximatively, are believed to describe the system in question. Apart from the measured signals the given equations typically contain non-measurable variables $\boldsymbol{x}(t)$, known physical constants and a set of unknown parameters $\boldsymbol{\eta}$. From these equations the problem of finding a suitable model structure can be stated as follows.

Requirement 1

Given a set of equations, generate, without changing the solution space, a new set of equations having at least one equation which does 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 non-linearities, combinations of time-discrete and time-continuous variables, time delays and so forth. The given problem for such a general framework is considered to be extremely hard.

However, if all the given expressions are *differential algebraic equations* (DAEs), i.e. all relations between $\boldsymbol{u}(t)$, $y(t)$, $\boldsymbol{x}(t)$, $\boldsymbol{\eta}$ 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, computationally similar to Gauss elimination, was outlined by Ritt[3] in the 1950s.

In short, *Ritt's algorithm* is as follows. Suppose that we are given a set of *differential polynomials* $\boldsymbol{\Phi}$ in $\boldsymbol{u}(t)$, $y(t)$, $\boldsymbol{x}(t)$ and $\boldsymbol{\eta}$. The usual state space representation

$$\boldsymbol{\Phi} = \{\dot{x}_1(t) - f_1(\boldsymbol{x}(t), u(t), \boldsymbol{\eta}), \ldots, \dot{x}_n(t) - f_n(\boldsymbol{x}(t), u(t), \boldsymbol{\eta}), y(t) - h(\boldsymbol{x}(t), u(t), \boldsymbol{\eta})\} \quad (12)$$

with all $f_i(\boldsymbol{x}(t), u(t), \boldsymbol{\eta})$ and $h(\boldsymbol{x}(t), u(t), \boldsymbol{\eta})$ being polynomials is an example of such a set. In general, though, $\boldsymbol{\Phi}$ can contain any kind of differential polynomials, including pure algebraic ones (a polynomial). Suppose further that we add new differential polynomials which are formed from elements in $\boldsymbol{\Phi}$ via multiplication by arbitrary differential polynomials, addition and differentiation. The infinite set of all polynomials that can be constructed in this way is termed the *differential ideal* generated by $\boldsymbol{\Phi}$ and is denoted by $[\boldsymbol{\Phi}]$. The important point here is that the solutions to $\boldsymbol{\Phi} = \boldsymbol{0}$ are solutions to every equation that can be created from $[\boldsymbol{\Phi}]$. Among all expressions in this set we are looking for a particular representative, a basis, having as 'simple' a structure as possible. In differential algebra the common way of expressing 'simplicity' is to *rank* the polynomials according to how desired they are. There exist several possible rankings, but so-called Ritt ranking[3] is sufficient for our purposes:

$$u(t) < \dot{u}(t) \ldots < y(t) < \dot{y}(t) < \ldots < x_1(t) < \dot{x}_1(t) < \ldots < x_n(t) < \dot{x}_n(t) < \ldots \quad (13)$$

Here '$<$' should be interpreted as 'is ranked lower than', or equally, 'is more desirable than'. As an example, a differential polynomial containing an unknown variable or a derivative of it is considered to be less desirable than another one in the inputs and outputs only.

Using this rule and the allowed operations (multiplication, addition and differentiation) when extending $\boldsymbol{\Phi}$, a 'simpler' characterization of $[\boldsymbol{\Phi}]$ can be obtained successively until finally the 'simplest' conceivable characterization of $[\boldsymbol{\Phi}]$ is returned. Formally this is Ritt's algorithm; see References 3 and 4 for further details. The characterization obtained is often labeled the *characteristic set* of $[\boldsymbol{\Phi}]$ and shows some nice features. Besides the result on solutions stated above, it will have a triangular structure with at least one differential polynomial in the measured variables only—the searched-for expression.

Ritt's algorithm only works for time-continuous systems and it is not immediate to extend it to the time-discrete case. However, as data are collected using computers, this situation must also be supported. One way is to stay within the differential algebraic framework, apply Ritt's algorithm and finally transform the result into the time-discrete domain by approximating

the derivatives with suitable discrete-time expressions. Another possibility would be to rely on so-called *difference algebra*, although to our knowledge very few constructive tools exist in this area today.

A second tool for solving polynomial differential equations originate from *commutative algebra*; see Reference 5 for a basic introduction and overview. In many aspects it provides the same kind of constructive machinery for non-linear set of equations as linear algebra offers to sets of linear equations. The parallels to differential algebra are also evident: the concept of differential ideal corresponds to *ideal*, the concept of ranking corresponds to *ordering* and the corresponding term for a characteristic set is known as a *Gröbner basis*. Using an ordering similar to the ranking (13), the unknown variables are eliminated according to a scheme invented by Buchberger[4] in the 1970s. The result is a Gröbner basis with at least one relationship in measured variables only.

Of course, there are fundamental differences between differential and commutative algebra too: the objects manipulated (using addition and multiplication only) are polynomials (and not differential polynomials), which means that the original elimination problem must be rearranged to a purely algebraic one. As shown by Forsman,[5] this can readily be done for models given in state space form, thus meaning that both differential and commutative algebra can be used for such time-continuous structures. However, one important advantage with *Buchberger's algorithm* is that difference equations (at least those condensed into a state space form) can be handled similarly to the way time-continuous state space models are handled.

In summary, one cannot say that one approach is always superior to the others. Differential algebra and Ritt's algorithm seems to be prefered if the original polynomials are unstructured, since these without modification can be sent to the solver. Gröbner basis calculations on the other hand can very well be used for models entered in state space form.

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, for example, are commonly encountered in chemical and biological modelling, while other structures often include trigonometric functions. Thus the model builder must at least be given the freedom to use simple non-linearities such as sinusoids and square-roots in his or her equations. Then, as we intend to stick to the polynomial framework, the following need arises naturally.

Requirement 2

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

Most functions encountered in traditional modelling belong to the class of *elementary functions*, which besides polynomials comprises the trigonometric, inverse trigonometric, exponential and logarithmic functions as well as all other functions that can be constructed from these by adding, subtracting, multiplying, dividing or forming compositions. According to Rubel and Singer,[6] one nice feature of the elementary functions is that they satisfy a differential polynomial. Algorithmically (see Reference 9 for the full story) the idea is simply to replace all non-polynomial expressions by such a polynomial. If, for example, $\sin(x_1(t))$ is encountered in the set of expressions, a new variable, say $z_1(t)$, is introduced so that $z_1(t) = \sin(x_1(t))$. This relation can be written as a differential polynomial

$$\dot{z}_1^2(t) - \dot{x}_1^2(t)(1 - z_1^2(t)) \tag{14}$$

which is added to the model structure. In terms of differential algebra, all the introduced variables $\boldsymbol{z}(t)$ can be treated as unknowns, which along with $\boldsymbol{x}(t)$ are then eliminated using Ritt's algorithm. The price paid for introducing new polynomials in this manner is that extra (false) solutions are also added. This is fortunately not so serious as it first might look, since, in the estimation step, data are employed to pick out a reasonable solution. The important point here is just that the description is flexible enough to comprise the solutions of the original structure.

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 Mbyte 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 to simplify some of the non-linear relationships. To achieve this, a second pre-processing tool is needed.

Requirement 3

Before sending the set of equations to the equation solver, decide whether the problem is computationally feasible.

The computational complexity (in time and space) is doubtless the main limiting factor for this kind of modelling. For general problems the used complexity figures[7] grow exponentially. As a consequence, using the best available Gröbner basis software, Lazard[7] stresses that problems in six to eight variables can be managed but that there is no hope to solve general problems with 10 or more unknowns. Despite the pessimistic complexity figures it should be remembered that these measures reflect the worst possible case. The 'average' modelling situation seems to be much more manageable, mainly because structures derived from physics tend to have an inherently sparse structure. However, how to accurately judge the computational complexity associated with a specific problem is still an open research topic.

In the case where the equation solver returns an input-output relation, the next step is to reparametrize it so that it fits into a linear regression framework. The last symbolic tool required is thus a post-processing tool.

Requirement 4

Given an input-output description, transform it if necessary to a time-discrete counterpart. Also, rewrite the result as a linear-in-the-parameters predictor or, when this is impossible, return a linear-in-the-parameters input-output structure.

Calculating or approximating the derivatives can in principle be done in two different ways. Either we regard the derivatives as separate signals that are created from known quantities using appropriate filters or we use suitable approximations. In view of the definition of derivatives the simplest and most natural approach is to replace the derivatives with the usual difference quotient, i.e. to use the Euler approximation[8] (T is the sampling period):

$$\frac{\mathrm{d}^{\,n}x(t)}{\mathrm{d}\,t^n} \approx \left(\frac{1}{T}\right)^n \sum_{i=0}^{n} \frac{n!}{(n-i)!i!}(-1)^{n-i}x(t+i), \qquad n = 1, 2, \ldots \tag{15}$$

Following the latter approach and then shifting the signals so that the highest time index of any signal is t, the resulting expression will, provided causality, contain an output variable $y(t)$. Knowing this, we can try to solve for $y(t)$ and at the same time separate the parameters and the regressors into two parts: $\boldsymbol{g}(\boldsymbol{\eta})$ and $\boldsymbol{\varphi}(t)$. The main point here is that $\boldsymbol{\varphi}(t)$ contains combinations of variables whose values are known at time t and thus $y(t) = \boldsymbol{g}(\boldsymbol{\eta})^T\boldsymbol{\varphi}(t) = \boldsymbol{\theta}^T\boldsymbol{\varphi}(t)$ is a linear-in-the-parameters predictor structure.

However, this procedure is not always possible. A first problem is that variables may show up in the denominator of a regressor, meaning that there is a risk that such a regressor will become numerically undefined. Another difficulty is that it might be impossible to separate the parameters from the regressors, while a third problem is that $y(t)$ can occur implicitly so that it is impossible to solve for it, at least uniquely. An alternative here is to avoid the predictor formulation and instead work directly with the input-output expression. After

reparametrization the key point is that the result is a linear regression, but know the structure is $\boldsymbol{\theta}^T \boldsymbol{\varphi}(t) = 0$, where, unlike in the predictor case, $\boldsymbol{\varphi}(t)$ also contains unknown outputs. Thus, instead of minimizing a prediction error, here we have to content ourselves with minimizing an equation error (or, actually, avoiding the degenerate solution $\boldsymbol{\theta} = \mathbf{0}$, a normalized equation error).

4.2. Numeric computations

The purpose of the symbolic computations is to transform the actual problem to one that is important and well-known in the identification community, namely to a linear regression structure where least squares estimation can be applied. One standard reference in this area is Draper and Smith,[9] in which many of the methods mentioned below are discussed and evaluated.

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 to be preferred to a higher-order one. For large systems an 'all possible subset' search for the few best regressors is as good as impossible owing to the combinatorial explosion. Altogether these observations call for a combined estimation and regressor selection procedure.

Requirement 5

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

Having N input-output data, $\boldsymbol{\theta}$ can be estimated using the usual *least squares algorithm*[1]

$$\hat{\boldsymbol{\theta}}_N = \left[\frac{1}{N} \sum_{t=1}^{N} \boldsymbol{\varphi}(t) \boldsymbol{\varphi}^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^{N} \boldsymbol{\varphi}(t) y(t) \tag{16}$$

provided that the inverse exists. From a numerical point of view it can be awkward to compute this inverse, especially when there are many regressors. To reduce this problem one can use so-called *ridge regression*.[9] For equation error structures a common situation is that some of the regressors are also disturbed. In such a case the so-called *total linear least squares* procedure can be applied.[10]

The common idea behind all regressor selection procedures is to explore the most promising regressors in one way or another. The methods suggested in the literature can be divided into two main categories.

- The first category consists of methods where statistical hypothesis testing is employed in a stepwise fashion in order to screen out a subset of regressors. The so-called *forward selection*, *backward elimination* and *stepwise regression* procedures all belong to this category. See References 9 and 12 for further details.
- The other category consists of methods that tries to analyse the correlation structure of the regression matrix. The idea is to include all available regressors but to weight their influence on the output. *Principal component regression* (PCR)[9] and *partial least squares* (PLS)[11] are two such procedures.

5. SOFTWARE ISSUES

A computer-based tool for semiphysical modelling requires powerful software pieces of various kinds. On one hand the aim of this section is to identify these pieces and on the other hand to present and motivate the solution we have adopted.

The following issues should be considered and supported in one way or another.

1. There should be means for entering and representing the set of equations assumed to describe the underlying system. Some sort of modelling interface is thus required.
2. Computer algebra computations are needed in the model structure selection phase.
3. Numerical and graphical routines for pre-treatment of data (detrending, removal of outliers, filtering, etc.) should be offered.
4. There ought to be facilities for interactively selecting the most 'important' regressors as well as schemes for parameter estimation (here different variations of the least squares algorithm).
5. Different tools for model validation (prediction, simulation, residual tests and so on) must be included.
6. It should be possible to work with several models in parallel. Furthermore, the route of modelling actions leading to one specific model should be traceable. These needs call for some sort of model database responsible for keeping track of the most promising models and aiding in answering what has been done in the modelling session up to now.
7. Optionally, and on top of the other tools, there is a need for a more advanced help system which is able to assist the user throughout the modelling session. Such help systems can come in many shapes, but a knowledge-based system (KBS) is perhaps the most sophisticated and interesting alternative.

One pronounced trend in developing scientific software is to embody the theory into a high-level program package accessed via a graphical user interface (GUI); see, e.g. the identification tools outlined by Ljung[12] and Van Overschee *et al.*[13] With this and the specification list above in mind it is not hard to imagine the considerable undertaking it would be to implement a semiphysical modelling tool from scratch in a conventional programming language such as C.

This is fortunately not necessary, as numerous packages supporting various parts of the listed items are available on the market. Desiring an open and easy-to-extend tool, we suggest to interfacing with such packages. However, in order to achieve a transparent usage and a common software environment, there is a need for a supervisor to administer the computations. This task may very well be entrusted to the graphical user interface, which in addition can be designed to release the user from the burdensome task of entering frequently used commands textually. Moreover, the GUI can be made responsible, at least partly, for keeping track of earlier investigated models. A particularly intriguing way of interfacing is to use the required packages as stand-alone computational units, running as separate processes, thereby providing the supervisor with different services. Besides being 'open', a further advantage with this approach is that it encourages distributed computations, i.e. parallel processing, and perhaps this is really what is needed to solve more complex modelling problems.

The interfacing philosophy was adopted in SEMI, a hybrid prototype semiphysical modelling tool structured as shown in Figure 6. The present version includes the packages marked grey in the figure, meaning that appropriate tools for items 1, 6 and 7 have not yet been included (though the open software structure supports such extensions). The graphical user interface is coded in C, taking advantage of special packages for interactive graphics. Numerical and other graphical capabilities (items 3–5) are provided through *Matlab* via a number of m-files, while *Maple* is currently used as the main symbolic engine (item 2).

The interface window of SEMI is shown in Figure 7. Most computations are initiated via pull-down menus or via buttons, selection lists, etc. Using the solar-heated house as an example, a typical modelling session with SEMI shows the following steps.

1. Specify variables used to model the house:

```
INPUTS :=   [u(t), I(t)]:
OUTPUT :=   [y(t)]:
UNKNOWNS := [x[1](t), x[2](t)]:
```

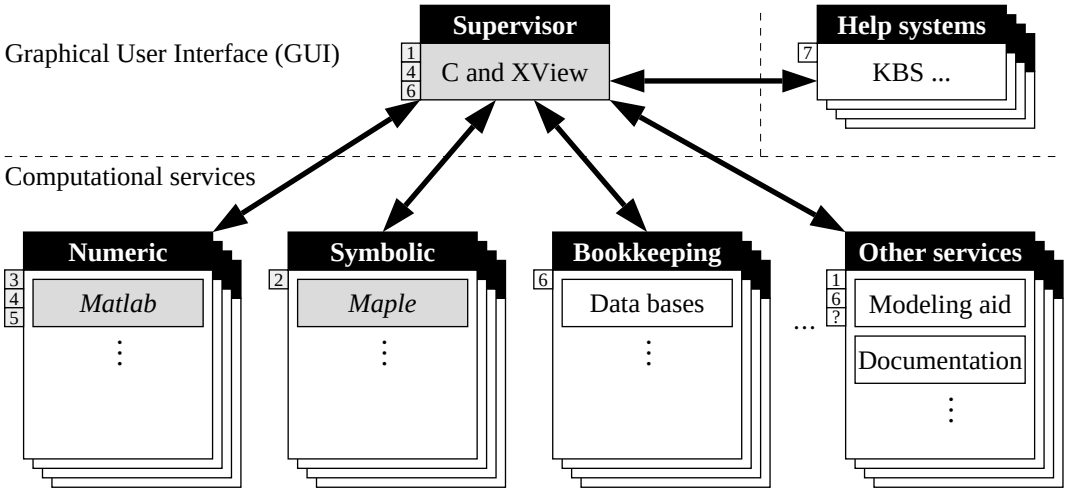

Figure 6. Software structure of an open semiphysical modelling tool. Grey boxes indicate pieces currently integrated in SEMI. Several boxes on top of each other illustrate parallel computations, possibly in a distributed environment. Numbers within boxes are used to connect the tool with tasks (from the specification list) for which it is primarily intended for

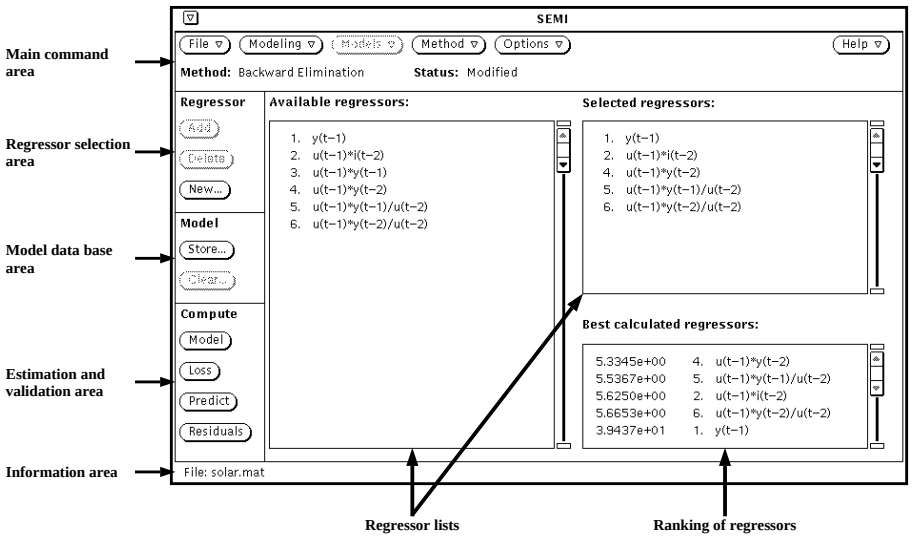

Figure 7. The SEMI interface. The system under investigation is the solar-heated house

2. Impose an ordering (ranking) amongst the variables:

   `ORDERING := [x[1](t), x[2](t), y(t), u(t), I(t)]:`

   The ordering means that `x[1](t)` is less desirable than `x[2](t)`, which in turn is less desirable than the output `y(t)` and so forth.

3. Enter equations as *Maple* expressions:

```
EQUATIONS := [ x[1](t+1)-x[1](t) = eta[1]*I(t)-eta[2]*x[1](t)
                                   -eta[3]*x[1](t)*u(t),
               x[2](t+1)-x[2](t) = eta[3]*x[1](t)*u(t)-eta[4]*x[2](t),
               y(t) = x[2](t) ]:
```

4. Eliminate the unknown variables (Requirement 1). Since the system is given in discrete-time polynomial state space form, Gröbner basis calculations can be carried out right away:

   ```
   IO := gbio(EQUATIONS, ORDERING):
   ```

   The computer now returns a polynomial, `IO`, in measured variables only. (When applicable, Ritt's algorithm is available through `rittio(EQUATIONS,RANKING)`.)

5. Find linear regression formulation (Requirement 4):

   ```
   LR := lrform(IO, y(t)):
   ```

6. In this case it is possible to arrive at an ordinary predictor formulation. Then, after `lrform` has been executed, the corresponding regressors are lined up in the **Available regressors** subwindow according to Figure 7. After having loaded estimation and validation data, it remains to judge the predictive power of the regressors. This interactive work can here be carried out using stepwise regression and ordinary least squares estimation as was discussed in Section 4.2. Notice that in using an interactive not fully automated selection procedure, the choice of regressors to include in the model can be based not only on different model validation tests (e.g. simulations, residual tests, etc.) but also on hard-to-formalize personal judgments. In this way it is rather straightforward to arrive at structure (11) for the solar-heated house.

6. CONCLUSIONS

In this article we have considered the identification problem of finding suitable non-linear transformations of the available measurements by utilizing prior physical knowledge about the underlying plant. The proposed semiphysical modelling procedure combines symbolic and numeric software tools. From the prior knowledge, assumed to be given as a set of differential algebraic equations, the main idea is to use general symbolic algorithms stemming from commutative and differential algebra in order to come up with a reasonable input-output model structure. The significance of the so-derived regressors is then tested against data using ordinary regressor selection procedures.

The major benefit with the procedure lies in the automated way a set of candidate regressors is derived. Instead of testing non-linear regressors of, say, a certain maximum degree in an ad hoc manner, the prior knowledge is used to restrict the search to certain, typically much fewer, input-output combinations. In practice, though, the most severe limitation of the method is the computational complexity of the symbolic algorithms. Another difficulty is that a linear-in-the-parameters model structure cannot always be constructed owing to the structure of the original equations.

In summary, it is our belief that semiphysical modelling constitutes a reasonable trade-off between the time to arrive at and the quality of a model structure for many identification applications. There still remains the fact that an increasing use of such procedures will require better and more general software tools. The SEMI software environment discussed in Section 5 is an attempt in this direction.

ACKNOWLEDGMENT

This work was supported by the Swedish National Board for Industrial and Technical Development (NUTEK), which is gratefully acknowledged.

REFERENCES

1. Ljung, L., *System Identification: Theory for the User*, Prentice-Hall, 1987.
2. Ljung, L., 'Perspectives on the process of identification', *Preprints of the 12th IFAC World Congress*, Sydney, Australia, vol. 5, July 1993, pp. 197–205.

3. Ritt, J. F., *Differential Algebra*, Dover Publications, 1950.
4. Buchberger, B., 'Gröbner bases: an algorithmic method in polynomial ideal theory', in Bose, N. K. (ed), *Multidimensional Systems Theory*, Dordrecht Reidel, 1985, pp. 184–232.
5. Forsman, K., *Constructive Commutative Algebra in Nonlinear Control Theory*, Phd thesis 261, Department of Electrical Engineering, Linköping University, Linköping, Sweden, December 1991.
6. Rubel, L. A. and M. F. Singer, 'A differentially algebraic elimination theorem with application to analog computability in the calculus of variations', *Proceedings of the American Mathematical Society*, **94**(4), 653–658 (August 1985).
7. Lazard, D., 'Systems of algebraic equations (algorithms and complexity)', Notes from a tutorial at the ISSAC'91, Bonn, July 1991.
8. Dahlquist, G. and Å. Björk, *Numerical Methods*, Automatic computation, Prentice-Hall, 1974.
9. Draper, N. and H. Smith, *Applied Regression Analysis*, John Wiley & Sons, 2nd ed., 1981.
10. Golub, G. H. and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 2nd ed., 1989.
11. Wold, S., A. Ruhe, H. Wold and W. J. Dunn III, 'The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses', *SIAM Journal of Scientific and Statistical Computing*, **5**(3), 735–742 (September 1984).
12. Ljung, L., 'A graphical user interface (GUI) to the system identification toolbox', in Blanke, M. and T. Söderström (eds), *Preprints of the 10th IFAC Symposium on System Identification*, Copenhagen, Denmark, vol. 4, July 1994, pp. 29.
13. Van Overschee, P., B. De Moor, H. Aling, R. Kosut and S. Boyd, 'ISID II: a fully interactive identification module for Xmath', in Blanke, M. and T. Söderström (eds), *Preprints of the 10th IFAC Symposium on System Identification*, Copenhagen, Denmark, vol. 4, July 1994, pp. 1.