On Differential/Algebraic Systems

Mattsson, Sven Erik

1986

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7327)/1-026/(1986)

On Differential/Algebraic Systems

Sven Erik Mattsson

Department of Automatic Control
Lund Institute of Technology
September 1986

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Report

Date of issue
1986-09-29

Document Number
CODEN:LUTFD2/(TFRT-7327)/1-026/(1986)

Author(s)
Sven Erik Mattsson

Supervisor

Sponsoring organisation
The National Swedish Board of Technical Development (STU contract 84-5069)

Title and subtitle
On Differential/Algebraic Systems

Abstract

It is of a vital interest that software for model development and simulation supports models given as differential/algebraic equations (DAE), since many models for physical systems are naturally given as sets of ordinary differential and algebraic equations and since, as motivated in the report, the equation form is the only reasonable for model libraries. Unfortunately, most simulation packages of today do not allow models given as DAE systems, but require assignment statements for derivatives and algebraic variables.

There are numerical solvers for DAE systems, but they are not as reliable and robust as numerical solvers for ordinary differential equations on state space form. Unfortunately, DAE systems may exhibit bad properties, which make them difficult to solve numerically today. The purpose of the report is to give an overview of important properties of DAE systems, indicate possible ways to handle difficulties and to motivate others to do research in this area.

By means of examples it is illustrated that even small, simple DAE systems can exhibit bad properties. The examples are by no means pathological. They arise naturally during model development.

There are many open questions to answer. The questions cover a broad range from theoretical to practical problems. To encourage research in this area, interesting issues are listed.

Key words
Computer Aided Control Engineering, Simulation, Differential/Algebraic Systems

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
26

Recipient's notes

Security classification

Contents

1. Introduction

When deriving models from first principles the result is often a system of differential algebraic equations (DAE). Consequently, it is natural to require that interactive software for model development and simulation supports DAE systems. Unfortunately, this is hardly the case today. Most simulation packages of today require assignment statements for derivatives and algebraic variables.

This report discusses the possibilities of supporting use of DAE systems in interactive software for model development and simulation. This introductory chapter is organized as follows. First, the motivations for supporting DAE systems are reviewed and then difficulties and possibilities of solving DAE problems are reviewed. Third, the organization of the report is described.

Motivation

Many models for physical systems are naturally given as sets of ordinary differential equations and algebraic equations. When DAE systems are supported, the model becomes more readable since the user can recognize fundamental relations as mass and energy balances and other phenomenological equations. It is easier to check that the model is entered correctly and the risk of introducing errors during manual transformation is reduced.

To handle models for large and complex systems the model developer must be able to decompose the model into submodels. Modularization simplifies modelling and makes the model more flexible and easier to adapt and manage. We can also build libraries of models. Technical systems are often built in a modular way composed of standard components. Their behaviour may be well-known. Good, generally accepted models may already exist.

Furthermore, as thoroughly motivated by Elmqvist (1978), the equation form is the only reasonable representation for model libraries. With models on assignment form, it must be decided for each submodel which of its variables that are inputs (in other words are known) and which of its variables that are outputs (defined by the model). As a simple example consider a resistor. Ohm's law states $V_1 - V_2 = RI$, where V_1 and V_2 are the voltages at the ends of the resistor, I is the current through the resistor and R is the resistance. The model has three variables V_1, V_2 and I. The resistance R is in this model a given parameter. If we should write the model on assignment form there are three possibilities

$$I := (V_1 - V_2)/R$$
$$V_1 := V_2 + RI$$
$$V_2 := V_1 - RI$$

The first variant assumes that V_1 and V_2 are inputs and defines I. This model

is appropriate if for example one end of the resistor is connected to a voltage source and the other to ground. The second and third variants assume that the current and the voltage at one end is known. These models are appropriate if the resistor is connected to a current source and ground. Consequently, for models on assignment form we need several different models for a resistor, depending on how it is connected to the environment. This makes both use and maintenance of a model library messy. Furthermore, other environments may result in algebraic loops so that equation systems with equations from several submodels must be solved to transform the model into assignment form. Two resistors connected in series between a voltage source and ground is a simple example of this. Submodels cannot be transformed into assignment form individually, but the transformation is a global problem.

How Can DAE Systems Be Solved?

As motivated above simulators should be able to deal with systems, which can be described by sets of ordinary differential equations and algebraic equations. This means that they should be able to solve DAE systems of the form

$$g(t, \dot{x}, x, v, p, c) = 0 \tag{1}$$

where t is the time, x and v vectors of unknown variables, p a vector of known parameters and c a vector of known constants. The unknown variables are here for convenience split into the two vectors x and v. The vector v contains the unknown variables which do not appear differentiated in the equations whereas the components of x appear differentiated. We will in the following assume that there is the same number of unknown variables and equations. To make the problem well-posed we must also for example specify initial values. We will in the following for the most not mention the initial values explicitly, but assume that they are given.

A now classical paper discussing the solution of (1) is Petzold (1982a). One approach is to use a numerical DAE solver which accepts problems on the form (1) directly. It requires a routine for calculating the residual vector $\Delta = g(t, \dot{x}, x, v, p, c)$ when all arguments of g are known. One example is the differential/algebraic system solver DASSL (Petzold, 1982b) written in Fortran. DASSL has a reputation of being one of the best and most robust solvers for differential/algebraic systems. Another approach is to transform the problem into the form

$$\dot{y} = f(t, y, p, c) \tag{2a}$$
$$x = X(t, y, p, c) \tag{2b}$$
$$v = V(t, y, p, c) \tag{2c}$$

and use a numerical ODE solver to solve (2a). Both methods have their cons and pros.

Difficulties

Unfortunately, the numerical DAE solvers of today have basic shortcomings. There are classes of "well-posed" DAE systems which they cannot solve. These difficulties can in principle be removed by symbolic formula manipulation. The

theory for deciding when a general problem of form (1) has a unique solution is incomplete.

Most numerical solvers for ordinary differential equations of today solve explicit problems of the type (2). There are good efficient and robust computer procedures available. The theory for explicit problems of the type (2) is well-established. There are nice theorems about existence and uniqueness of solutions (e.g. Coddington and Levinson, 1955). Unfortunately, numerical ODE solvers cannot be used directly to solve DAE systems on the form (1). The problem has to be transformed into the form (2). This is in general a nontrivial problem. In some cases the transformation from (1) into the form (2) can be done by symbolic manipulation, but nonlinearities can make it impossible to give an analytic expression for $f(t, y, p, c)$.

What Can Be Done?

This introduction may give a pessimistic impression. Certainly, much remains to be done before we can say that there are good, efficient and robust computer procedures for solving DAE systems. However, we should take an optimistic approach. By discussing the properties of DAE systems, we get insight into what is hard and can take advantage of that and avoid some difficulties during the modelling. Also if the properties of DAE systems become commonly known, the possibilities would increase that a person comes up with a bright idea. By showing that classes of DAE systems, which the numerical DAE solvers of today cannot handle, arise "naturally" when modelling, the need and importance of a DAE-solving capability will be more widely recognized. This will hopefully inspire numerics people to invent new algorithms.

We must also keep in mind that many simple things can be done to facilitate the numerical solution. For example, it is advisable to make some formula manipulation before using a numerical DAE solver, since simple manipulations may drastically reduce the order and the complexity of the problem. For example connections between submodels lead to identities of the form $A = B$, and it is of course stupid to let the numerical DAE solver handle these. It is so easy to eliminate them, and let identical variables have just one representative. Furthermore, when defining models it is also natural to introduce auxiliary variables as $A = \langle$expression independent of $A\rangle$ for common expressions to improve the readability. It is often simple to remove these variables from the vector of unknown variables passed to the numerical DAE solver and calculate them internally in the routine for calculating the residual.

Outline of the Report

Our aim is to discuss the difficulties and the possibilities to solve them. The report is organized as follows. In Chapter 2 there is a short discussion on existence and uniqueness. The possibilities to transform a problem on the form (1) into (2) are considered in Chapter 3. In the simplest case this transformation can be done by algebraic manipulations. In more complex cases differentiation of equations may be needed. It turns out that then the numerical DAE solvers of today are also in trouble. In this case there are algebraic constraints on components of the x-vector. This is a basic problem and is discussed in Chapter 4. Chapter 5 considers the handling of discontinuities. The possibilities of symbolic formula manipulation when using a numerical DAE solver are

discussed in Chapter 6. A summary of interesting issues is given in Chapter 7. However, before discussing the difficulties and possibilities to solve them, some notation will be introduced.

Notation

The Jacobian of $g(v)$ with respect to v is denoted $\partial g/\partial v$. It is a $dim\, g \times dim\, v$ matrix whose element i, j is $\partial g_i/\partial v_j$. We will also work with a matrix $\delta g/\delta v$. It is a $dim\, g \times dim\, v$ matrix whose element i, j is zero if v_j does not appear in the expression g_i, otherwise it is one. The matrix $\delta g/\delta v$ can be viewed as a structure Jacobian or incidence matrix of variable dependencies. The concept that a variable v_j appears in an expression should be interpreted literally. A mathematical definition of 'appear' could have been like this: A variable v appears in $h(x, v)$ if there exist x, v_1 and v_2 so that $h(x, v_1) \neq h(x, v_2)$. However, this definition of appear makes it much more difficult to check if a variable appears in an expression or not. The variable x thus 'appears' in the expression $(a - a)x$ from our point of view.

2. Existence and Uniqueness

As indicated above there is no general theory for deciding when a DAE system of type (1) has a unique solution. However, the "simple" problem $E\dot{y} = Fy + f(t)$ where E and F are constant square matrices is quite well understood. This problem is considered further in Section 4. For example März (1982a,b) has been working with existence and uniqueness issues.

From a more pragmatic view, we can reformulate the question as "Can we put any not too restrictive demands on $g(t, \dot{x}, x, v, p, c)$ to guarantee that the problem is well-posed?". For special applications like electrical circuits and mechanics of rigid bodies the formulation will be well-posed if special rules are followed during the model development. It is natural to demand some skill of the model developer. Consider the following example.

EXAMPLE 2.1 — The mathematical pendulum
Consider a mathematical pendulum having the length l and unit mass. Introduce a Cartesian coordinate system with origin at the mounting point of the pendulum and with the x-axis pointing downwards. For an ideal mathematical pendulum the energy is preserved

$$\dot{x}^2 + \dot{y}^2 - 2gx = \dot{x}_0^2 + \dot{y}_0^2 - 2gx_0$$

where $\dot{x}_0$ denotes the value of $\dot{x}$ at the time t_0 etc. The parameter g denotes the gravitational acceleration. We also have the constraint that the length of the pendulum is l;

$$x^2 + y^2 = l^2$$

To be a well-posed problem, the initial values of x_0 and y_0 must fulfil the length constraint. However, even if we supply proper initial values so that the length constraint is fulfilled, this is not a well-posed problem having a unique solution. For example, the model allows the speed to change signs at every instance. To see that, introduce instead polar coordinates

$$x = l\cos\varphi, \quad y = l\sin\varphi$$

We can then easily remove the length constraint and get

$$l^2\dot{\varphi}^2 - 2gl\cos\varphi = l^2\dot{\varphi}_0^2 - 2gl\cos\varphi_0$$

The speed $\dot{\varphi}$ appears squared in the model so we can have two solutions; the pendulum swinging up or down. We can get the well-known model

$$\ddot{\varphi} + g/l\sin\varphi = 0$$

by assuming that $\ddot{\varphi}$ exists and differentiating the previous equation, but, that is another story. A proper model can be obtained directly by using Newton's equation of motion or Lagrange's equation. □

3. Use of Numerical ODE Solvers

Let us consider the DAE system (1). What possibilities are there to use a numerical ODE solver which can solve problems on the form (2)? The simulator should then first transform the DAE system into the form (2) before invoking the numerical ODE solver. We will assume that the transformation into the form (2) can be done without differentiation. The assumption implies that we can take $y = x$. A numerical ODE solver for the problem (2) requires a routine for calculating $\dot{y}$ when t, y, p, c are known. If we set $z = \dot{x}\,(= \dot{y})$, it is easy to see that what is needed is a routine that solves the algebraic equation system $g(t, z, x, v, p, c) = 0$ when t, x, p and c are considered to be known and $(z^T\ v^T)^T$ are considered to be the vector of unknowns. In many cases this equation system is sparse and a standard technique then is to partition the problem into smaller and simpler problems. We will now consider the block lower triangular partition.

The Block Lower Triangular Partition

If we want to solve the algebraic system

$$h(x) = 0 \tag{3}$$

it is simplified if we can split it up into smaller problems which can be solved sequentially. A standard partition is to permute x and h giving $\bar{x} = Px$ and $\bar{h}(\bar{x}) = Qh(P^{-1}\bar{x})$ so that $\delta\bar{h}/\delta\bar{x}$ becomes block lower triangular with minimal diagonal blocks and non-zero diagonal elements. This partition will be referenced as the block lower triangular partition or the BLT-partition. The problem $\bar{h}(\bar{x}) = 0$ can be solved by first solving the problem associated with the first block, then with this solution available the problem with the second block is solved and so on. There are efficient procedures for construction of the permutations P and Q giving a BLT-partition (e.g. Tarjan, 1972, Wiberg, 1977 and Elmqvist, 1978).

The BLT-Partition and Numerical ODE Solvers

In the work to solve $z\,(= \dot{x})$ from $g(t, z, x, v, p, c) = 0$ when t, x, p and c are considered to be known a BLT-partition procedure can be used as a first step to decompose the equation system into smaller ones. Let $\bar{y} = P(\dot{x}^T\ v^T)^T$ denote the permuted unknowns and consider the partitioned problem $\bar{g}(t, \bar{y}, x, p, c) = 0$. Note that it is trivial to determine $\dot{x}$ and v, if $\bar{y}$ is known, since $\bar{y}$ is just a permutation of the vector $(\dot{x}^T\ v^T)^T$.

A very nice case is when the partition procedure gives a problem where $\delta\bar{g}/\delta\bar{y}$ is lower triangular and where $\bar{g}_i$ for all i is affine in y_i i.e.

$$\bar{g}_i(t, \bar{y}, x, p, c) = \bar{y}_i\bar{g}_{i1}(t, \bar{y}_1, ..., \bar{y}_{i-1}, x, p, c) + \bar{g}_{i2}(t, \bar{y}_1, ..., \bar{y}_{i-1}, x, p, c)$$

with $\bar{g}_{i1}(t, \bar{y}_1, ..., \bar{y}_{i-1}, x, p, c) \neq 0$ for all t, $\bar{y}$, x and p. For this case it is simple to calculate $\bar{y}$, by calculating the elements $\bar{y}_i$ in sequential order using the formula

$$\bar{y}_i = -\bar{g}_{i2}(t, \bar{y}_1, ..., \bar{y}_{i-1}, x, p, c)/\bar{g}_{i1}(t, \bar{y}_1, ..., \bar{y}_{i-1}, x, p, c)$$

If we take a model accepted by for example Simnon (Elmqvist, 1975, Åström 1983, 1985) or ACSL (ACSL, 1986) and interpret the assignment statements as equations, the partition procedure will give this desirable result.

In the singular case when $\bar{g}_{i1}(t, \bar{y}_1, ..., \bar{y}_{i-1}, x, p, c) = 0$ for some t, $\bar{y}$, x and p, the problem still may be well-posed having a unique solution. For example the function $\bar{g}_{i1}$ may be zero for some t, $\bar{y}$, x and p that cannot be obtained, or differentiation of some equations is needed to transform the problem into the form (2). A third possibility is that $\partial g/\partial \dot{x}_j = 0$ for some j. Examples of these cases will show up further below.

If $\delta\bar{g}/\delta\bar{y}$ is diagonal, but there is a $\bar{g}_i$ which is not affine in y_i the simulator might solve this nonlinear scalar equation by symbolic manipulation or numerically by using for example Newton's method. If $\delta\bar{g}/\delta\bar{y}$ has nonscalar diagonal blocks which all are small and linear the simulator can solve these symbolically. But in the general case there may appear large, nasty non-linear blocks that have to be solved numerically. However, there are efficient and robust solvers (Chambers and Whitmarsh-Everiss, 198?, Söderlind, 1980) for DAE systems of the form

$$\dot{x} = f(t, x, v, p, c)$$
$$0 = g(t, x, v, p, c)$$

where the Jacobian of g with respect to v is nonsingular. This is an important case. It arises frequently and it is simpler to transform analytically a problem of type (1) into this form than into the more restricted form (2).

It may happen that some of the blocks are singular for some t, $\bar{y}$, x and p. As indicated above for the simple case, the original problem may still be a well-posed problem having a unique solution.

Let us consider some examples.

Examples

EXAMPLE 3.1 — Two rotating masses connected with a slip coupling
Newton's equation of motion gives the model

$$J\dot{\omega} = Q_l + Q_r$$

for a rotating mass with rotational speed ω and inertia J under influence of a torque Q_l on the left side and a torque Q_r on the right side. Assume that we want to model two such rotating masses which are connected with a slip coupling having linear characteristics with the damping coefficient d. We then get the following model:

$$J_1\dot{\omega}_1 = Q_{l1} + Q_{r1} \quad (a)$$
$$J_2\dot{\omega}_2 = Q_{l2} + Q_{r2} \quad (b)$$
$$Q_{r1} = -Q_{l2} \quad (c)$$
$$Q_{r1} = d(\omega_2 - \omega_1) \quad (d)$$

We will in the following assume that Q_{l1} and Q_{r2} are known functions of time and that J_1, J_2 and d are parameters.

If we want to try the approach discussed above we should assume that $\dot{\omega}_1$, $\dot{\omega}_2$, Q_{r1} and Q_{l2} are unknown and make a partition. The partition procedure gives:

1. Compute Q_{r1} using (d) (assuming that ω_1 and ω_2 are known)

$$Q_{r1} := d(\omega_2 - \omega_1)$$

2. Compute Q_{l2} using (c) and the in step 1 computed Q_{r1}

$$Q_{l2} := -Q_{r1}$$

3. Solve $\dot{\omega}_1$ from (a) using the in step 1 computed Q_{r1}

$$\dot{\omega}_1 := (Q_{l1} + Q_{r1})/J_1$$

4. Solve $\dot{\omega}_2$ from (b) using the in step 2 computed Q_{l2}

$$\dot{\omega}_2 := (Q_{l2} + Q_{r2})/J_2$$

This computational procedure for calculating $\dot{\omega}_1$ and $\dot{\omega}_2$ works if $J_1 \neq 0$ and $J_2 \neq 0$ and it is easy to verify that the problem has a unique solution when initial values for ω_1 and ω_2 are defined.

Let us consider the case $J_1 \neq 0$ and $J_2 = 0$. The BLT-partition calculated above is useless. It is singular, since it implies that we should calculate $\dot{\omega}_2$ as $\dot{\omega}_2 := (Q_{l2} + Q_{r2})/J_2$. However, it easy to see that $\dot{\omega}_2$ disappears from the equations (a, b, c, d) when $J_2 = 0$. Consequently, we should now view ω_2 as unknown when performing the BLT-partition. It gives the following sequence of calculation:

$$\begin{aligned} Q_{l2} &:= -Q_{r2} && \text{from } (b) \\ Q_{r1} &:= -Q_{l2} && \text{from } (c) \\ \dot{\omega}_1 &:= (Q_{l1} + Q_{r2})/J_1 && \text{from } (a) \\ \omega_2 &:= \omega_1 + Q_{r1}/d && \text{from } (d) \end{aligned}$$

The problem (a, b, c, e) is as seen mathematically well-posed even for $J_2 = 0$, if we do not specify initial value for ω_2. This example shows us that the structure of a BLT-partition is discontinuous. We need two BLT-partitions; one when $J_2 \neq 0$ and another when $J_2 = 0$. □

Let us now consider another system of two rotating masses.

EXAMPLE 3.2 — Two rigidly coupled rotating masses
A model for two rotating masses which are connected rigidly to each other is given by (a, b, c) and

$$\omega_1 = \omega_2 \qquad (e)$$

It is easy to realize that the BLT-partition procedure described above will fail if it is applied on the problem (a, b, c, e). We have four unknown variables

($\dot{\omega}_1$, $\dot{\omega}_2$, Q_{r1} and Q_{l2}) and four equations (a, b, c, e). However, the equation (e) stating $\omega_1 = \omega_2$ does not contain any unknown variable, since we have assumed that ω_1 and ω_2 are known. If $J_1 + J_2 \neq 0$ and the initial values for ω_1 and ω_2 are equal, the problem (a, b, c, e) is mathematically well-posed as seen by the following procedure:

1. Add (a) to (b);

$$J_1\dot{\omega}_1 + J_2\dot{\omega}_2 = Q_{l1} + Q_{r1} + Q_{l2} + Q_{r2}$$

Use (e) to substitute ω_2 with ω_1 and $\dot{\omega}_2$ with $\dot{\omega}_1$ and use (c) to eliminate $Q_{r1} + Q_{l2}$;

$$(J_1 + J_2)\,\dot{\omega}_1 = Q_{l1} + Q_{r2} \qquad (f)$$

2. Solve $\dot{\omega}_1$ from (f) (recall that we assumed $J_1 + J_2 \neq 0$) and substitute this expression into (a);

$$J_1(Q_{l1} + Q_{r2})/(J_1 + J_2) = Q_{l1} + Q_{r1}$$

Multiply with $(J_1 + J_2)$;

$$J_1Q_{l1} + J_1Q_{r2} = J_1Q_{l1} + J_1Q_{r1} + J_2Q_{l2} + J_2Q_{r1}$$

which is equivalent with

$$(J_1 + J_2)Q_{r1} = J_1Q_{r2} - J_2Q_{l1} \qquad (g)$$

Consequently, the model defined by (c, e, f, g) is equivalent to the model (a, b, c, e) if $J_1 + J_2 \neq 0$, and it is trivial to see that the system (c, e, f, g) has a unique solution if ω_1 is given an initial value. The BLT-partition procedure applied on the problem (c, e, f, g) will be successful. We may for instance calculate $\dot{\omega}_1$ from (f), calculate ω_2 from (e), calculate Q_{r1} from (g) and use this value to calculate Q_{l2} from (c).

A more careful study of (c, e, f, g) shows that this problem contains just one first order differential equation, whereas the problem (a, b, c, e) contains two different variables which appear differentiated namely ω_1 and ω_2. However, there is an algebraic relation between these variables; $\omega_1 = \omega_2$. This relation causes as seen a reduction of the degree. If we want to use an ODE solver, the simulator must be able to differentiate. One possibility is to eliminate $\dot{\omega}_2$ by realizing that (e) implies $\dot{\omega}_1 = \dot{\omega}_2$ as we did when we made the transformation into (c, e, f, g). Another approach is to exchange the relation (e) with $\dot{\omega}_1 = \dot{\omega}_2$ and $\omega_1(0) = \omega_2(0)$. The first approach has the best numerical properties. For the second approach numerical imperfections may cause drift so that (e) will not be fulfilled within desired numerical accuracy. □

4. Algebraic Constraints on the Vector x

We demonstrated above that differentiation of equations is sometimes needed to transform a problem of type (1) into the form (2). Unfortunately, those problems which cannot be transformed into form (2) without differentiation may also be very difficult to solve for a numerical DAE solver. Consequently, it is important to get insight into this problem.

Consider the special case when we can write the problem as

$$E\dot{y} = Fy + f(t) \tag{4}$$

where E and F are constant square matrices with regular matrix pencil $sE-F$. A matrix pencil $sE - F$ is regular if E and F are square and $\det(sE - F)$ is not identical to zero. If the matrix pencil $sE-F$ is singular the problem (4) is ill-posed. It is natural to demand that we have as many linearly independent equations as unknown variables.

The Kronecker Canonical Form

Problems of the form (4) with regular pencil can be put on a canonical form, Kronecker canonical form, which reveals structural properties. It can be shown (Gantmacher, 1959) that there are non-singular matrices P and Q such that by taking $y = Qz$ and $g = Pf$ we get from (4)

$$\dot{z}_1 + Cz_1 = g_1 \tag{5a}$$
$$N\dot{z}_2 + z_2 = g_2 \tag{5b}$$

where N is a nilpotent matrix of degree n (i.e. $N^n = 0$ and $N^{n-1} \neq 0$). The degree n defines what is called the index (or nil-potency) of the problem (4). It is well-known that the solution to (5a) can be written explicitly as

$$z_1(t) = e^{-(t-t_0)C} z_1(t_0) + \int_{t_0}^{t} e^{-(t-\tau)C} g_1(\tau) d\tau \tag{6a}$$

There is also an explicit solution for (5b)

$$z_2(t) = \sum_{i=0}^{n-1} (-N)^i \frac{d^i g_2(t)}{dt^i} \tag{6b}$$

The explicit solution (6b) for the problem (5b) reveals some demands on the problem (4) to be well-posed. The initial values must be consistent with (6b)

and the function $f(t)$ must be properly differentiable if the solution should be a function and not only a distribution. If the solution to a problem is a distribution but not a function the problem cannot be said to be well-behaved in a numerical sense.

What Does the Index Imply?

If the index is zero, the Kronecker canonical form is just $(5a)$ which means that E is non-singular. If E is non-singular, we can put the problem (4) on form (2) without differentiation; $\dot{y} = E^{-1}(Fy + f(t))$. The problem is more intricate if E is singular. The Kronecker canonical form has then also a $(5b)$ part. If E is non-singular it is well-known that the poles s of this linear and time invariant system are given by $\det(sI - E^{-1}F) = 0$, where I is a unity matrix of appropriate dimension. Equivalently we can write the equation as $\det(sE - F) = 0$. This formulation as a generalized eigenvalue problem has the advantage that it also applies for singular E. If E is singular there are infinite eigenvalues and the properties of their eigenvectors decide the index of the problem. Fortunately, we can avoid working with infinite eigenvalues by studying the eigenvalues at zero of the generalized eigenvalue problem $\det(E - \mu F) = 0$. If $\det(E - \mu F) = 0$ for all μ the pencil is singular. The index of (4) is equal to the size of the largest Jordan block for the eigenvalue zero of $E - \mu F$ (if there are no eigenvalues at zero, the index is zero).

If the index is one, $(5b)$ has the form $z_2 = g_2$ and the problem can obviously be transformed to the form (2) without differentiation. If the index is greater than one, the matrix N is nonzero and nonsingular and differentiation of equations is needed when transforming it into the form (2).

To sum up, the problem (4) can be transformed into the form (2) without differentiation if and only if the matrix pencil $E - \mu F$ is regular and has a full, linearly independent set of eigenvectors for the eigenvalues $\mu = 0$ (the index is then zero or one).

All known numerical DAE solvers have trouble with solving problems of form (4) with index greater than one, because the error does not necessarily decrease with the step length. The difficulties are nicely explained in Petzold (1982a). Barrlund (1985) gives examples of how DASSL (Petzold, 1982b) behaves for high index problems. The approach discussed above can be extended to time varying F, but if E is time varying or the system is non-linear there are no extensions or results available.

Reduction of the Index

The index of problem (4) can be decreased to zero by differentiation in the following way (e.g. Gear and Petzold (1984)):

1. If E is non-singular go to 6.
2. Find non-singular matrices P and Q such that

$$PEQ = \begin{pmatrix} E_{11} \\ 0 \end{pmatrix}$$

with E_{11} having full rank.

3. Make the variable substitution $y = Qz$ and multiply the equations from the left with P giving

$$\begin{pmatrix} E_{11} \\ 0 \end{pmatrix} \dot{z} = \begin{pmatrix} F_{11} \\ F_{21} \end{pmatrix} z + \begin{pmatrix} g_1(t) \\ g_2(t) \end{pmatrix}$$

4. Differentiate the lower part of the system

$$\begin{pmatrix} E_{11} \\ F_{21} \end{pmatrix} \dot{z} = \begin{pmatrix} F_{11} \\ 0 \end{pmatrix} z + \begin{pmatrix} g_1(t) \\ \dot{g}_2(t) \end{pmatrix}$$

5. If the "E-matrix" for the new problem is singular, consider the new problem as your problem and go to 2.
6. The index of the problem is now zero and the index of the original problem is equal to the number of times the sequence 2-3-4 was performed.

The reduction procedure is another method for deciding the index of a problem on the form (4). It also reduces the index of the problem to zero, possibly making it more suitable for numerical solution. However, numerical imperfections may introduce drift so that the algebraic relations that we differentiated will not be fulfilled within desired numerical accuracy. It would of course be better to make the partition (5) first.

Examples

An important question is "Do problems with indices greater than one arise 'naturally'?". The answer to this question is undoubtedly yes. Let us consider some examples and we will start by revisit the examples with rotating masses coupled in different ways.

EXAMPLE 4.1 — Two rotating masses connected with a slip coupling
The problem of to rotating masses connected with a slip coupling formulated as (a, b, c, d) can be written on the form (4) with

$$y = (\omega_1 \quad \omega_2 \quad Q_{r1} \quad Q_{l2})^T, \qquad f(t) = (Q_{l1} \quad Q_{r2} \quad 0 \quad 0)^T,$$

$$E = \begin{pmatrix} J_1 & 0 & 0 & 0 \\ 0 & J_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad F = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ d & -d & 1 & 0 \end{pmatrix}$$

We get

$$\det(E - \mu F) = -\mu^2(J_1 J_2 + \mu d(J_1 + J_2))$$

If $J_1 \neq 0$ and $J_2 \neq 0$ the matrix pencil has two eigenvalues at zero and a full eigenvector set e.g. $(0\ 0\ 1\ 0)^T$ and $(0\ 0\ 0\ 1)^T$. If one of J_1 or J_2 is zero (implying that $J_1 J_2 = 0$), say $J_2 = 0$, but $d(J_1 + J_2) \neq 0$, the problem has three eigenvalues at zero and a full eigenvector set e.g. $(0\ 1\ 0\ 0)^T$, $(0\ 0\ 1\ 0)^T$ and $(0\ 0\ 0\ 1)^T$. If $J_1 J_2 = 0$ and $d(J_1 + J_2) = 0$ then $\det(E - \mu F)$ is zero for all μ implying that the problem is ill-posed. Physically it means that we apply the torques Q_{l1} or Q_{r2} to objects having zero inertia. Consequently, the problem (a, b, c, d) has the index 1 provided $J_1 J_2 \neq 0$ or $d(J_1 + J_2) \neq 0$ otherwise it is ill-posed. Since the index is one, when the problem is well-posed, we should

be able to transform the DAE system (a, b, c, d) into the form (2) without differentiation. Recall that we have already done this above. □

EXAMPLE 4.2 — Two rigidly coupled rotating masses
The problem of two rigidly coupled, rotating masses formulated as (a, b, c, e) can also be written on the form (4):

$$y = (\omega_1 \quad \omega_2 \quad Q_{r1} \quad Q_{l2})^T, \qquad f(t) = (Q_{l1} \quad Q_{r2} \quad 0 \quad 0)^T,$$

$$E = \begin{pmatrix} J_1 & 0 & 0 & 0 \\ 0 & J_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad F = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \end{pmatrix}$$

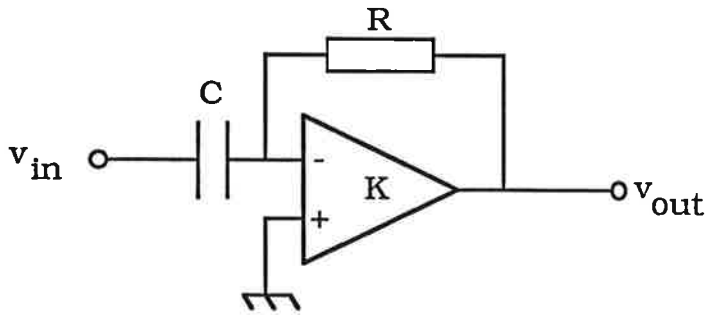
We get

$$\det(E - \mu F) = -\mu^3(J_1 + J_2)$$

which shows that the problem is ill-posed if $J_1 + J_2 = 0$. Simple calculations show that this implies that $Q_{l1} + Q_{r2} = 0$ which is not generally fulfilled. If $J_1 + J_2 \neq 0$ the matrix pencil has three eigenvalues at zero. However, there is a full set of three linearly independent eigenvectors only if J_1 or J_2 is zero and the index is for this case one. If $J_1 J_2 \neq 0$ and $J_1 + J_2 \neq 0$ the index is two. Recall that to put this DAE problem formulated as (a, b, c, d) on the form (2) one differentiation was needed. Note that the index depends on the formulation of the problem and that two mathematically equivalent problems may have different indices. For example, the formulation (c, e, f, g) is as we found above mathematically equivalent with the formulation (a, b, c, e). Both formulations are ill-posed when $J_1 + J_2 = 0$. The index of the formulation (c, e, f, g) is one whereas the index of (a, b, c, e) is one only when $J_1 J_2 = 0$ otherwise it is two. □

Let us consider an even nastier problem.

EXAMPLE 4.3 — A differentiator
Consider the figure



Let v_c denote the voltage over the capacitor and let i be the current through the capacitor (and the resistor) then a simple model of form (4) for this system is

$$y = (v_c \quad v_{out} \quad i)^T, \qquad f(t) = (0 \quad v_{in} \quad v_{in})^T,$$

$$E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad F = \begin{pmatrix} 0 & 0 & 1/C \\ -1 & 1/K & 0 \\ -1 & -1 & -R \end{pmatrix}$$

where v_{in} is considered to be a known function of t and C, K and R to be parameters.

For this problem we have

$$\det(E - \mu F) = -\mu^2(R/K + \mu(1 + 1/K)/C)$$

implying that this is a nice problem for finite and nonzero parameters R, C, and K. However, it is very common to assume infinite gain K when making calculations with operational amplifiers; meaning that $1/K = 0$. If $1/K = 0$ then $\det(E - \mu F) = -\mu^3/C$ implying that there are three eigenvalues at zero, but just two linearly independent eigenvectors. Simple calculations show that $v_c = v_{in}$ and $v_{out} = -Ri = -(R/C)\dot{v}_{in}$. Consequently, v_{out} is proportional to the time derivative of v_{in}, so if v_{in} is e.g. a step, the simulator is in trouble if we try to solve the system numerically. You may argue that it nonphysical to assume infinite gain, but it is usual to make this idealization when making calculations with operational amplifiers. The motive of including this example is to show that nasty systems may easily arise. □

5. Discontinuities

We will now consider discontinuities in $g(t, \dot{x}, x, v, p, c)$. Let us start by an example.

EXAMPLE 5.1 — Switching between rigid coupling and slip coupling
We will continue to consider rotating masses and study what happens when the coupling switches between a rigid coupling and a slip coupling. If we assume that the switches can be done instantly, the model will be discontinuous.

The formulation (a, b, c, d) modelling two rotating masses connected by a slip coupling has index one and if transformed into the form (2), we get a second order differential equation. The formulation (a, b, c, e) modelling two rigidly connected masses has index two and if formulated as (c, e, f, g) the index is one and can be transformed into a first order differential equation. If we now take two rotating masses and let the coupling switch between a rigid coupling and a slip coupling, we can get different variants of index and order variations.

Let us first consider a switch from rigid coupling to slip coupling at time t_1. The model (a, b, c, e) is valid in the time interval $0 \leq t < t_1$ and the model (a, b, c, d) is valid for $t_1 < t$. The problem (a, b, c, e) with given, equal initial values for ω_1 and ω_2 is a well-posed problem with a unique solution at least if $J_1 \neq 0$ and $J_2 \neq 0$. In the time interval up to t_1 our problem is equivalent to the problem of two rigidly coupled masses. This problem was shown to have a unique solution, if the initial values for ω_1 and ω_2 are equal. After the time t_1 our problem is identical to the problem with two rotating masses connected by a slip coupling, since the behaviour at the time t_1 is smooth. We have for t_1 that $\omega_1(t_1) = \omega_2(t_1)$. But what are the torques $Q_{l1}(t_1)$ and $Q_{r2}(t_1)$? Just before the switch we have according to (g) that

$$Q_{r1}(t_1-) = (J_1 Q_{r2}(t_1-) - J_2 Q_{l1}(t_1-))/(J_1 + J_2)$$

and just after the switch we have according to (d) that

$$Q_{r1}(t_1+) = d(\omega_2(t_1+) - \omega_1(t_1+)) = 0$$

Consequently, the torques Q_{r1} and Q_{l2} may be discontinuous at t_1. The value at t_1 depends on how the switch is done and we have here idealized the switch and assumed that it can be done instantly. Fortunately, the behaviour of the system after the switch is independent of how the switch is performed as long as it does not imply exchange of impulse with the environment. However, we cannot answer the question "What are the torques Q_{r1} and Q_{l2} during the transition?". If we accept this, the idealization can be motivated. We can also specify $Q_{r1}(t_1)$ ourselves. For example we can say that $Q_{r1}(t_1) = 0$ indicating

that the switch has been performed at t_1. With this choice we can formulate the model as (a, b, c) and

$$\Big(\texttt{if } t < t_1 \texttt{ then } \omega_1 - \omega_2 \texttt{ else } Q_{r1} - d(\omega_2 - \omega_1)\Big) = 0 \qquad (h)$$

The modelling of two rotating masses coupled first with a slip coupling and then rigidly is much more intricate. Just before the transition, the speeds $\omega_1(t_1-)$ and $\omega_2(t_1-)$ may have different values but after the transition they should be equal according to (e). Consequently, the speeds ω_1 and ω_2 may be discontinuous at t_1. Preservation of the impulse implies that

$$J_1\omega_1(t_1-) + J_2\omega_2(t_1-) = (J_1 + J_2)\omega_1(t_1+) \qquad (i)$$

What are the torques Q_{r1} and Q_{l2} during the transition? Our assumption of an infinitely short transition implies that the speed is discontinuous. To achieve a step in the speed, the torque Q_{r1} must contain a term

$$\frac{J_1 J_2}{J_1 + J_2}\Big(\omega_2(t_1-) - \omega_1(t_1-)\Big)\delta(t - t_1)$$

where $\delta(t)$ is the Dirac delta function. Consequently, numeric solution including the time t_1 will be troublesome. However, we need not calculate $Q_{r1}(t_1)$. Integrate the equations up to t_1 using (a, b, c, d) then modify ω_1 (and ω_2) according to (i) and continue the integration using the model (a, b, c, e) or (c, e, f, g). This approach indicates that new language elements are needed to describe the model. □

Dirac Delta Pulses

As seen above, discontinuities may imply that the solution could contain Dirac delta pulses. Such Dirac pulses must be removed before numerical solution. One possibility is to let the user handle the problem, because he is in most cases aware of it. Vaguely formulated, when considering modelling of physical objects, discontinuities in the model are often due to idealizations made by the model developer.

Recall the example above. There the discontinuities were due to the assumption that the switches between different couplings could be done instantly. This resulted in some cases that the torques could contain Dirac pulses. Another example is the classical way of modelling a bouncing ball assuming an infinitely short bounce. The assumption leads to forces having Dirac delta components. When the model developer makes this idealization of the bounce, he is implicitly saying that he is not really interested to model the bounce itself and the reaction force from the floor. Consequently, it is reasonable to take the stand point that then he cannot ask questions about the reaction force. It should not be allowed to appear in the model. To allow this kind of idealizations, the user could be given a possibility to define how the state variables should be recalculated when an event occurs. If he really is interested in the reaction torque, he must model the bounce. One possible model is to consider the ball and floor to be elastic and let the reaction force from the floor on the ball depend on the distance from the floor.

Detection of Discontinuities

There is also another trouble which also appears for problems of the form (2). Cellier (1979) shows (for problems on the form (2)) that discontinuities cannot be handled by step length control, since the integration routines can miss fast switches and the integration routine can give a completely false result. His approach is to view the problem as a continuous problem and to view a discontinuity as an event which means that we should switch to a new continuous problem. A discontinuity that only depends on time is easy to detect. It can be handled by integrating the equations up to that time and then restart the integration routine. Thus the discontinuity is handled by solving two continuous problems in sequence. This means that it is almost straightforward to simulate models consisting of both continuous time submodels and sampled submodels.

If the discontinuity depends on x and v, the problem becomes more complex since we also have detect if and when events occur. Cellier's approach is to tell the integration via indicator functions when events may occur. The integration goes on until an indicator function indicates an event. The step-size control mechanism of the integration routine is then disabled and the step is repeated with a new step-size computed by a special iteration scheme to establish if and when an event has occurred. It is important that the indicator functions are calculated with enough accuracy and resolution in time so that events are detected. One way to get some automatic control of the calculation of the indicator functions is to introduce the indicator functions as extra states. Thus making the integration routine explicitly aware of the indicators, so it adjusts the step-size also with respect to the behaviour of the indicators. However, the method does not guarantee that fast multiple switches are detected. If the model is supposed to model something and the behaviour of the model exhibits multiple fast switches, the user must really consider whether important dynamics has been neglected. It may then be important to model the transitions in more detail.

In Cellier's approach the integration routines must not switch to the new set of equation until it has been established that an event has occured. Note that this implies, that the numerical integration routine must then evaluate the residual $\Delta = g(t, \dot{x}, x, v, p, c)$ outside the validity range of the equation system. Consequently, this implies that the the set of equations must be formulated so that this evaluation does not cause numerical overflow or division by zero.

Symbolic analysis can be used to set up the indicator functions automatically. If the model is given in symbolic form, it is rather easy to detect discontinuities by scanning through the equations.

6. Numerical DAE Solvers and Symbolic Manipulation

We have above indicated that there are many reasons for symbolic formula manipulation before using a numerical DAE solver. It is of interest to detect and eliminate algebraic constraints on the vector x, since they cause trouble for numerical DAE solvers. A basic difficulty with algebraic relations is that they might not be detectable by local analysis of a submodel. It is necessary to make a global analysis. It is also of interest to detect discontinuities and treat them in special ways. All these aspects are very important, but we will not consider them further. We will instead consider aspects of symbolic formula manipulation to make the numerical solution more efficient and reliable.

Partitioning of the Problem

When solving the DAE problem of the form (1) numerically, the residual $\Delta = g(t, \dot{x}, x, v, p, c)$ and its Jacobian are evaluated a number of times and linear equation systems with the Jacobian as coefficient matrix are solved. The possibilities to do this reliable and fast depends on the number of unknowns, but also on the complexity of g and the structure of the Jacobian. It may be a good idea to try to partition the problem to split it up into smaller subproblems and with simple interaction. It is an open question which partition that is most favorable. There must exist efficient partition algorithms. The possibilities that the partition gives small subproblems must be good. If the desired partition is block diagonal form, the partition will mostly give large blocks. If we consider the BLT-partition, each subproblem can be smaller to the cost of a more complex interaction structure.

Elimination of Variables

Elimination of components of the vector v must be done with care not to destroy structure and not to increase the computational work. As an example assume that the model developer has introduced the variable α to denote a complex expression that appears in many places of the model. The computations of the residual then becomes more laborious if the simulator eliminates α by symbolic substitutions before invoking a numerical DAE solver. If the simulator decides to remove α from the vector of unknowns passed to the numerical solver, the computations of the residual becomes faster if the routine for calculating the residual keeps the variable α internally.

It is a basic demand that the symbolic formula manipulation procedures do not introduce divisions by zero. A basic issue is how we should consider

parameters. It may be correct to make a certain manipulation for most parameter values, but there may be certain combinations that cause division by zero. However, in many cases those combinations never will show up, because they are for example non-physical. This indicates that it could be useful to let the user supply ranges for parameters.

Another demand is that the system should be able to give clear error message. It cannot just print out transformed equations and complain. In some way or other, it must be able to relate the error to the original equations.

Various kinds of symbolic formula manipulation are possible. Permutation of variables must be considered as being of the simplest type. Permutation is nice, since it is easy to implement and preserves basic properties. It is also easy to give clear error messages when something happens since permutations does not change individual equations.

Let us consider the possibilities to eliminate components of the vector v. The BLT-partition can be used to find components that may be eliminated. However, as we found above it is not a general method in our case. Note that it can be assumed that $\dot{x}$ and x are known since they are calculated by the numerical DAE solver. This fact gives additional freedom when eliminating components of v. How can this freedom be explored? For example the bookkeeping can be simplified if simultaneously elimination of variables are avoided and that only sequential elimination is allowed as done in the connection with the BLT-partition. However, there are no good algorithms available for finding these sequences of variables and associated equations. The problem is closely related to what is called tearing. The idea is as follows. Consider the problem of solving

$$\begin{pmatrix} 1 & 0 & \dots & 0 & b_1 \\ a_{21} & 1 & & 0 & b_2 \\ \vdots & & \ddots & & \vdots \\ a_{n1} & a_{n2} & & 1 & b_n \\ c_1 & c_2 & \dots & c_n & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ z \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \\ e \end{pmatrix}$$

If we knew z, it would be straightforward to solve for x. Consequently, if we want to solve the problem by an iterative procedure, we just have to iterate over z. The basic idea of tearing is to divide the unknown variables into two vectors so that it is in some sense easy to solve for the variables in the first vector x, if we know the values of the variables in the second vector z. An iterative procedure to solve the problem then just has to iterate over z, and the dimensionality of the problem is decreased. Unfortunately, there are no efficient general tearing algorithms available. Our problem has special properties. In our case the numerical DAE solver handles the iterative part and it should also solve $\dot{x}$ and x. A user often introduces a sequence of auxiliary variables to make the model more readable. It may be a good idea to try to explore this fact and look for such sequences. The simulator may consider all equations whose left part consists of a simple variable.

7. Conclusions

It is of a basic interest to be able to solve DAE systems, since many models for physical systems are naturally given as sets of ordinary differential equations and algebraic equations and since the equation form is the only reasonable representation for model libraries. We have discussed the possibilities to solve DAE systems. There are numerical solvers for DAE systems, but they are not as reliable and robust as numerical solvers ordinary differential equations on state space form. Unfortunately, DAE systems may exhibit bad properties, which make them difficult to solve numerically today. By means of examples it has been illustrated that even small, simple DAE systems can exhibit bad properties. We have also shown that they are by no means pathological. They arise naturally during model development. Possible ways to handle difficulties have been indicated.

We strongly encourage research in this area, since it is of a vital interest for simulation that these difficulties are solved. There are many open questions to answer. The questions cover a broad range from theoretical to practical problems. Interesting issues are

1. When can we decide whether a problem is well-posed, i.e. having a unique solution? Can we put any not too restrictive demands on $g(t, \dot{x}, x, v, p, c)$, which can be checked, to guarantee that the problem is well-posed? For special applications the problem will be well-posed if special application dependent rules are followed.

2. In simple cases it is possible to solve $\dot{x}$ and v from $g(t, \dot{x}, x, v, p, c) = 0$ directly. In more complex cases differentiations of equations are needed. It turns out that then the numerical DAE solvers of today are also in trouble. There are algebraic constraints on the components of the x-vector. Can the constraints be detected and eliminated by symbolic formula manipulation? They may put restrictions on how the starting vector $x(t_0)$ can be chosen. Can these restrictions be determined by the simulator?

3. Discontinuities cause also difficulties, especially if they depend on x and v. Can they be detected and can changes be flagged for the numerical solver? Discontinuities may imply that the solution could contain Dirac delta pulses. In many cases they are results of idealizations made by the model developer to make the model simpler. How can such be detected and eliminated for the numerical solver? Since the model developer often is aware of these effects, it may be of interest to introduce new language constructs so he can handle and eliminate Dirac delta pulses explicitly. You may take the stand point that idealizations implies that the model developer is not really interested in those variables containing Dirac delta

pulses, so they should not appear in the model. The user should not be allowed to pose questions about their behaviour.

4. Besides dealing with algebraic constraints on the vector x and discontinuities, how could symbolic formula manipulation be used? An interesting question is "How should we partition the problem to make numerical solution more efficient and reliable?".

5. One possible way to reduce the complexity of the problem by formula manipulation is to eliminate components of the vector v from the vector of unknowns passed to the numerical solver. However, the elimination must be made with care not to destroy the structure or make the problem ill-posed. For example, division by zero must not be introduced.

   Connection of submodels gives simple equations of the type $A = B$ which are simple to eliminate. When defining models it is also natural to introduce sequences of auxiliary variables to improve the readability. If we can find those sequences, it is easy to eliminate the associated variables from the vector of unknown passed to the numerical solver. It may, however, be an good idea to keep them as internal variables in the routine for calculating the residual. Note, also that when searching for such sequences of auxiliary variables it can be assumed that $\dot{x}$ and x are known since they are calculated by the numerical DAE solver. Unfortunately, there are no good algorithms available for this kind of problem, so it is an interesting research topic.

6. The man-machine interface is important. The system must be able to give clear error messages, explaining the errors in terms of the original equations not in transformed ones unrecognizable for the user.

   The user also probably has more information available about the problem than what he has stated. It may for example be very useful for the simulator to know that certain combinations of parameter values will never or should never be tried. It is desirable that the simulator could be able to pose such questions. When modelling, it is often not obvious what is critical.

7. How can the numerical DAE solvers themselves be improved?

Acknowledgements

This work has been a part of the project Computer Aided Control Engineering (CACE) at Lund Institute of Technology. We are grateful to the National Swedish Board of Technical Development (STU) who has supported this project under contract 84-5069.

The author would like to thank Professor Per Hagander for many useful and stimulating discussions. Many thanks to Professor Gustav Söderlind for valuable comments on the manuscript.

References

ACSL (1986): *Advanced Continuous Simulation Language (ACSL) - Reference Manual*, Fourth Edition, Mitchell and Gauthier Associates, Concord, Mass. 01742, USA.

ÅSTRÖM, K.J. (1983): "Computer Aided Modelling, Analysis and Design of Control Systems - A perspective," *IEEE Control Systems Magazine*, Vol. 3, No. 2, May 1983, pp. 4-16.

ÅSTRÖM, K.J. (1985): "Computer Aided Tools for Control System Design - A perspective," In Jamshidi and Herget (1985).

BARRLUND, A. (1985): "Numerisk behandling av differentialalgebraiska system (Numeric solution of differential/algebraic systems)," UMNAD-16.85, Institute of Information Processing, University of Umeå, Umeå, Sweden.

CODDINGTON, E.A. and N. LEVINSON (1955): *The Theory of Ordinary Differential Equations*, McGraw-Hill Book Company, Inc., New York.

CELLIER, F.A. (1979): *Combined Continuous/Discrete System Simulation by Use of Digital Computers: Techniques and Tools*, Ph.D-thesis, Diss ETH No 6483, The Swiss Federal Institute of Technology, ETH, Zürich.

CHAMBERS, T.H.E. and M.J. WHITMARSH-EVERISS (198?): "A Methodology for the design of Plant Analysers," PKR/SE/201. Central Electricity Generating Board, U.K. (the report is not dated).

ELMQVIST, H. (1975): "SIMNON - User's Manual," Rept. TFRT-3091. Department of Automatic Control, Lund Institute of Technology, Sweden.

GANTMACHER, F.R. (1959): *The Theory of Matrices, Vol. I-II*, Chelsea Publishing Company, New York, N.Y., USA.

GEAR, C.W. and L.R. PETZOLD (1984): "ODE Methods for the Solution of Differential/Algebraic Systems," *SIAM J. Numerical Analysis*, Vol. 21, No. 4, Aug. 1984, pp. 716-728.

JAMSHIDI M. and C.J. HERGET (Eds.) (1985): *Computer-Aided Control Systems Engineering*, North-Holland.

MÄRZ, R. (1982a): "Multistep Methods for Initial Value Problems in Implicit Differential-Algebraic Equations," Preprint 22, Sektion Mathematik, Humbolt-University, Berlin.

MÄRZ, R. (1982b): "On Difference and Shooting Methods for Boundary Value Problems in Differential-Algebraic Equations," Preprint 24, Sektion Mathematik, Humbolt-University, Berlin.

PETZOLD, L.R. (1982a): "Differential-Algebraic Systems are not ODE's," *SIAM Journal on Scientific and Statistical Computing*, Vol. 3, No. 3, Sept. 1982, pp. 367-384.

PETZOLD, L.R. (1982b): "A Description of DASSL: A Differential/Algebraic System Solver," Rept. SAND 82-8637, Sandia National Laboratories or Proceedings of IMACS World Congress, Montreal, Canada, 1982.

SÖDERLIND, G. (1980): "DASP3 - A Program for the Numerical Integration of Partitioned Stiff ODEs and Differential-Algebraic Systems," Rept. TRITA-NA-8008. Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, Stockholm, Sweden.