

# Intelligent DAE Solvers and User Friendly Design, Simulation, and Analysis Packages

S. L. Campbell  
Department of Mathematics  
North Carolina State University  
Raleigh, NC 27695-8205 USA

## ABSTRACT

Simulation and design packages are an integral part of many computational systems. However, the current capabilities of many of these systems are limited by their ability to handle naturally posed questions in a user friendly manner. This paper examines what would be required to handle those problems that naturally lead to differential algebraic equations.

## 1. INTRODUCTION

There is an ever widening use of simulation, design, and analysis packages. One feature of many of these packages is that the user describes the system in a graphical manner using some type of block diagrams. This software is often used by individuals who are far from expert in differential equations and numerical analysis. Frequently the underlying mathematical models are a mixed system of algebraic and differential equations or what is called a differential algebraic equation (DAE). This has been one of the driving forces behind the work on DAEs over the last decade.

Considerable progress has been made on understanding DAEs and their numerical solution. However, this progress, some advertising to the contrary, is frequently not fully exploited in many simulation or analysis packages. Part of the problem concerns open problems in the theory of DAEs. However, some of the difficulties could be handled for many practical problems by an intelligent user interface that works with users who may have never even heard of a DAE much less appreciate how they differ from ODEs. A lot of the needed software diagnostics are already available. Some of the needed software routines are not currently available. The purpose of this paper is not to survey the latest results on DAEs and their numerical simulation [4]. Nor is it to discuss the design of mixed symbolic numeric approaches to working with DAEs [9].

In some cases the needed software and supporting analysis seems to run counter to current practice in numerical analysis but the need for such software will be carefully described. We will also give our opinion on how the information from the diagnostics should be presented to the user. While the developments we describe will be of even greater use to the expert user, the emphasis in this paper is on the special demands placed by the "average user."

The situation we are interested in is the following. A user is interested in a particular system. We shall assume that the system is modeled with some sort of relational model. That is, there is a time variable  $t$ , a physical set of variables  $x(t)$  and relationships between  $x$  and its time derivatives  $x'$ .

$$B(x', x, t) = 0 \quad (1)$$

Since this model is computer generated, the user does not have access to  $B$  and probably only knows of some of the variables  $x$ , namely those used to describe the components specified by the user. The rest of  $x$  and  $B$  are created by the software. For example, some of  $x$  may be part of internal device models.

Suppose the user wishes to ask some questions. To be specific suppose the user chooses a subset of  $x$ , call it  $u$ , and wonders what  $u$  would have to be in order to achieve some desired behavior. For example, the goal might be exact trajectory tracking by some other subset of  $x$  called  $y_1$ . Let  $x = [y_1, y_2, u]$ . Let  $y = [y_1, y_2]$ . Then the software must solve the DAE

$$B(y', u, y, u', t) = 0 \quad (2a)$$

$$R(y_1, t) = 0 \quad (2b)$$

Note that system (2) is a DAE even if (2a) is an ordinary differential equation (ODE),  $y' = Q(y, t, u)$ . Many current software systems cannot solve the system (2) and those that do solve special cases of (2) often require considerable user expertise. The rest of this paper will focus on the design of software that

can answer this type of question in a helpful way. A guiding principle is that in an ideal world the software should *always* help the user even if it cannot solve the requested problem. Failure to integrate, for example, should ideally be accompanied by an indication of what the difficulty might be. Another principle is that there should be defaults for most options but that the user should be told when certain defaults are exercised.

The needed software tools are varied. By graphical methods we mean those that are computed from the graph that describes the relationships between the variables. That is, which equations appear in which equations. In discussing DAEs, the word structural is used in two ways. One deals with the form of the equations. The second refers to a property that is true except on a set of measure zero. For example a square matrix that has all nonzero entries is structurally invertible. To distinguish the two types we shall refer to the first as a structured system. Other options are numerical and symbolic. We shall not consider symbolic algorithms here although they can be useful [9, 18].

## 2. HOW DAEs ARE DIFFERENT

First we need to briefly review how DAEs are different from ODEs. Consider the system (3).

$$w' + w = f(t) \quad (3a)$$

$$\beta(t)y' + x = g(t) \quad (3b)$$

$$\alpha(t)z' + y = h(t) \quad (3c)$$

$$z = k(t) \quad (3d)$$

which has the solution

$$w = c_1 e^{-t} + \int_0^t e^{(s-t)} f(s) ds \quad (4a)$$

$$x = f(t) - \beta(t)[g'(t) - \alpha'(t)h'(t) - \alpha(t)h''(t)] \quad (4b)$$

$$y = g(t) - \alpha(t)h'(t) \quad (4c)$$

$$z = h(t) \quad (4d)$$

This example illustrates some of the many ways that DAEs are different from ODEs. These differences impact in a number of ways on DAE based simulators.

- Not all initial conditions lead to smooth solutions. Equivalently solutions reside on lower dimension manifolds.
- There may be hidden constraints such as (4b), (4c) in addition to the explicit constraints (3d).

- The solution can depend on derivatives of both equations and forcing functions  $(f, g, h, k)$ .

One measure of the singularity of a DAE is a nonnegative integer called the *index* [4]. There are a number of definitions of the index which are identical for simple systems but can differ on more complex systems. An ODE is index zero. One definition is the number of times that one has to differentiate in order to get an ODE in all the state variables. Example (3) is index three.

The classical numerical methods cannot be applied to all DAEs. For index one DAEs, backward differentiation formulas (BDF) and implicit Runge-Kutta (IRK) methods can be used. But for DAEs of index two or above these classical methods only work for certain special structured systems such as Hessenberg systems. An index three Hessenberg system looks like (5).

$$y' = g(x, y, z, t) \quad (5a)$$

$$z' = h(y, z, t) \quad (5b)$$

$$0 = q(z, t) \quad (5c)$$

with  $q_z h_y q_x$  nonsingular. For general DAEs a general purpose integrator needs to be used. The current integrator theory while useful has some important limitations that will be discussed later

Finally, linearization is a major tool in working with nonlinear systems. There are positive results [7] but the usual linearizations are not always appropriate.

## 3. DIAGNOSTICS

We begin with a set of  $m$  equations in the  $n$  dimensional vector  $x$

$$F(x', x, z, t) = 0, \quad Px(t_0) = q_0 \quad (6)$$

where  $z$  are source terms or parameters and  $q_0$  are a user specified set of initial values. The matrix  $P$  indicates the user may only know some initial values. We suppose that the software is being used for a variety of types of systems so that it is not known a priori that the system is mechanical, fluid, electrical, etc.

### Is It an ODE?

Some problems are initially formulated as ODEs. Other software, by restricting the users' options, forces the user to specify a problem that can be formulated as an ODE. However, often in areas ranging from circuit theory to mechanics (6) begins as an implicit model. The software then either has to generate

an ODE, solve the DAE by a DAE method, or return to the user.

If graph theoretic methods, such as are common in circuit theory, are able to determine an ODE model, than it will usually be in terms of a reduced set of variables. The software then needs to determine if the initial specification in (6) is consistent with this set of variables. The users initial specification may not be consistent with relationships given by  $F = 0$ . In this case the software should inform the user that not all variables can be specified as given. User options can vary from trying a different formulation to having the variables projected to a consistent set of initial conditions. The later choice is particularly useful if the user is doing initial simulations and does not actually have a good feel for what some initial values should be.

We have supposed that the original system in a partially prescribed initial value problem (IVP). However, the solution of the questions asked by the user may involve DAE boundary value problems (BVP). For example, if (6) is the process and an optimal control problem is posed, the software may have to solve a BVP.

For a more general system, the least obvious case is when  $m > n$ . This problem is either overdetermined or has redundant equations. The redundancy can be examined by symbolic algorithms or estimated by numerical rank checks on submatrices of  $[F_{x'}, F_x, F_z]$ . If it is determined that the problem is overdetermined by  $k$  equations, there are several options. Options might include: user removes  $k$  restrictions if they have specified an equation, software is told to pick an independent set of equations and integrate, software is told to integrate as an overdetermined system. Various software exists for the later by integrating in a least squares since or adding additional slack variables [12]. Overdetermined systems are used to impose conserved quantities on numerical integrations both when solving PDEs and in mechanics.

If  $m < n$ , then the problem is undetermined. There are no obvious defaults. In this case, the user should be told this and asked to either specify additional relationships and constraints, or to specify values for certain subsets of  $x$ . The user should be able to ask whether specifying a given variable is reasonable. This can be difficult to do completely but if a user wants to specify say  $\{x_5, x_9, x_{23}\}$  then the software can attempt to determine if there are relationships between these variables. Such software could be based on the ideas in [15].

Finally, if  $m = n$ , then the question is whether  $F_{x'}$  is nonsingular. Nonsingularity can be examined by graphical, symbolic, or numerical means.

### Simple DAEs

The most widely studied systems are index one systems. These can be characterized by the requirement that the pencil  $\{F_{x'}, F_x\}$  is a regular pencil and  $F_{x'}$  has constant rank. These properties can be quickly checked by graph based or numerical algorithms. Graph algorithms can never be sufficient conditions. If the system is index one, then there are a variety of numerical integrators that can be used. Most are either based on BDF or IRK methods.

Many simulation packages have been expanded in recent years to have the capability to integrate index one systems. The only general concerns with index one DAEs are the problem of consistent initial conditions at boundary points and events since the initial conditions cannot be specified arbitrarily. Some user dialogue may be necessary where the user is asked to specify a number  $m$  of initial values, it is checked whether this is an acceptable choice, and then the given initial values are extended to a set of initial conditions. This is not always an easy task and must often be done by some nonlinear method. Upon determination of the full initial values for  $x$  and  $x'$ , the software should check with the user as to whether these are acceptable values. In some cases, physically correct ranges are known, and the user may wish to not accept certain initial conditions. For example, in [4] the correct, and a spurious control history are close during part of the maneuver.

A second special class of “simple DAEs” consists of linear time invariant DAEs

$$Ex' + Fx = f(t) \quad (7)$$

Provided  $E, F$  are of moderate size, and the generalized eigenstructure is well conditioned, numerical algorithms can be applied based on the matrix pencil  $\{E, F\}$ . Note that it is not necessary to compute the full Kronecker structure.

### Higher Index DAEs

If it is determined that the DAE is not index one, then the situation becomes much more complex since none of the standard numerical methods work on all index two DAEs, much less all DAEs.

One option is to return to the user and say that the problem as formulated is a higher index DAE and needs to be reformulated. Unfortunately, that rules out a number of areas of applications including many problems in constrained mechanics. The other two

options are to either try to integrate the DAE directly or to have the software try to reformulate the problem.

Suppose that the decision is to directly integrate. There are two types of integrators that might be used. One is one of the BDF or IRK methods. The second is one of the general integrators that are under development [5]. The general nonlinear integrators are in the prototype stage at this time. They also tend to be computationally intensive. We shall assume then that the first option is to see whether one of the existing index two or index three integrators can be used. Graph based structural analysis algorithms have been developed [20, 22]

These methods usually assume that the DAE is in Hessenberg form. A simple Hessenberg form can be checked using a graph algorithm followed by a numerical linear algebra routines to verify that the Jacobians are nonsingular. The Jacobian check would need to be repeated during the integration. If the system is Hessenberg and the  $x, y, z$  split of the state variables determined, then an integrator like RADAU5 called. One such system determines the index and structure and then generates Jacobians and calls DASSL for index one system and generates the appropriate data and calls RADAU5 for index two or three Hessenberg systems [18]

The problem is that many applications are not just a Hessenberg system. They may even be Hessenberg but not in the coordinate system given by the user.

Generally one thinks of variable order, variable step integrators as being more robust. However, in this setting that is not the case. For higher index DAEs one generally gets different variables having different orders of convergence. If the software does not know how to partition the state, then the error control strategy will not interact well with the step size selection and the method can fail. We have seen several examples where IRK codes easily solved a problem in a fixed step mode but it took a lot of trial and error to get the tolerances set up right to run the variable step version. Thus a useful option is to return to the user and say that the problem is not a classically structured DAE and ask if the user wants to attempt a fixed step integration. Here fixed step means the step size is decided ahead of time and not that all steps are the same length. If the answer is yes, an integration can be done. It may be desirable to run the integration at more than one stepsize and to check whether the integration appears to be giving a reasonable answer. Note that integration at two stepsizes can be done more cheaply if the integrations

are done simultaneously since the Jacobians from one integration and be used for the other one.

**Problem Reformulation** Given that the problem is higher index there are options other than just solving the higher index problem. One is to return to the user and ask if they are interested in reformulation. For example, a different set of state and independent variables may have a lower index. The other option is to use one of the index reduction approaches [1, 17]. Some of these are graphically based so that they only work for some systems. They then often perform differentiations of state equations. Others, such as those that underlie the observer work in [3], are based on the ability of the software to return another system which is then the basis of subsequent algorithms. For example, for solvable linear systems

$$E(t)x' = F(t)x(t) + f(t) \quad (8)$$

algorithms exist which return an ODE and a set of constraints

$$x' = Q(t)x + q(t) \quad (9a)$$

$$C(t)x = h(t) \quad (9b)$$

such that the solutions of (8) satisfy (9a) and (9b) and in addition the dimension of the solution manifold of (8) is the nullity of the full rank matrix  $C$ .

### Singularity & Event Detection

Many simulation packages have the possibility of events [19]. Events also occur in optimization when the set of active constraints changes. Regardless of whether the events are specified by the user or whether they occur during the simulation, it is not possible to always handle this without some input from the user.

The difficult case is when the event results in a new DAE with either a different index or a different number of degrees of freedom resulting in a discontinuity in the solution manifold. The obvious default is to ask for continuity in the state but this is not always physically correct. An example is a short which results in a spark. Realistically the software will have to notify the user that some types of events require information in order to tell how they should be resolved. This is especially true when properties such as the dimension of the solution manifold change at the event. Many papers in the discussion of the optimization of DAEs, assume that the user has provided state transition functions that describe how to handle discontinuities. Sometimes this is easy. However, it often is highly nontrivial and will require considerable physical insight on the part of the user.

## System Analysis

Given a system (6) there are a variety of questions that one can ask. Some of these, such as determining equilibria, can be done without extensive diagnostics since it would just involve solving

$$F(0, c, t) = 0 \text{ for all } t$$

However, other questions such as analyzing the stability of the equilibria, determining the degrees of freedom, and deciding whether the equilibrium lies on a well defined solution manifold requires knowing something about the structure of the DAE. There have been some studies [11]. Sensitivity analysis has been studied in [16] and other papers. But considerable additional work is needed on the extension of the user nonlinear analysis capabilities to DAE systems. This is especially true of higher index DAEs.

## 4. NEEDED CAPABILITIES

As shown in the cited references and [4] there has been a considerable amount of work done. However, more work is needed on several topics.

One is the general purpose integrators mentioned earlier. Prototype codes exist. But a variety of technical problems need to be resolved and production codes written.

Another problem concerns the results on BDF and IRK methods. We shall focus on the IRK methods but the issues are similar. The theory is proven for systems in Hessenberg form. There are positive convergence estimates that usually show that some variables are found to less accuracy than others. This is referred to as order reduction. This order reduction must be taken into account when doing variable step-size implementations. As noted, many applications are not in a standard form such as Hessenberg. Note that an operation as simple as multiplying the system (5) on the left by an invertible constant matrix can destroy the Hessenberg structure. Such an operation would not affect convergence for IRK or BDF methods. The outcome of this is that many problems which are apparently not in Hessenberg form can be solved by IRK methods with an a priori step size selection. However, one cannot do error control. This means that variable step methods will experience failures or excessive work on systems that they can integrate. Similar statements apply to optimization of DAEs where error based mesh refinement strategies are used as in [2].

There is a need for more work on accurately described larger classes. In company with this there is a need

for a development for heuristics for fixed step integrators or, what is the same thing, integration without access to the usual error control strategies. Note that some of the usual points of view change in a fixed mesh integration. For example, the usual practice with standard methods is to not use damped Newton iterations since slowness of the Newton iteration is used as one criteria for reducing the stepsize. However, in a fixed mesh integrator, it may be desirable to use a more global iteration strategy.

Another basic problem is parameter identification or model estimation. There has been some work on parameter ID for DAEs [21]. However, this work presupposes that the user has some predetermined form for the equations and their remains only some physical parameters that need to be identified.

In the ODE case, the desire might be to fit a model of the form

$$x' = (A + Q)x + q(t, \beta) \quad (10)$$

where  $A$  is a known constant matrix (possibly zero),  $q$  is a known function of  $t, \beta$ . Here  $Q, \beta$  are the parameters to be fit. There are questions to be asked, such as what should be the size of  $x$  but many good pieces of software exist.

The problem with fitting a DAE is more complex. Given data how does one fit

$$Ex' = Qx + q(t, \beta) \quad (11)$$

where  $E, Q, \beta$  are parameters. The first problem is that the answer is not unique since left multiplication gives another system with the same behavior. Even when this is taken care of, additional difficulties remain. Chief among these is that the dimension of the solution manifold and the index is unknown. Even if the dimension of  $x$  is fixed, then it still remains open how to determine the dimension of the solution manifold.

## 5. CONCLUSION

We have see that the “complete DAE package” will need to have the ability to perform a variety of graphical algorithms. It will need to have a variety of numerical integrators and the capability of automatically calling a reasonable integrator. To do this, it will need to determine the index and some of the systems structure. Many of the general diagnostic algorithms are based on various numerical linear algebra calculations of matrices which are Jacobians with respect

to higher derivatives. These could be computed symbolically in theory, but in practice some automatic differentiation capability is probably necessary. Finally, the software needs to have different options, to explain to the user when default options are exercised and what these assumptions mean, and provide software tools to assist the user in their decision making.

## ACKNOWLEDGEMENT

Research supported in part by the National Science Foundation under Grant No. DMS-9423705, DMS-9714811, and ECS-9500589.

## 6. REFERENCES

- [1] R. Bachman, L. Brüll, Th. Mrziglod, and U. Pallaske, "On methods for reducing the index of differential algebraic equations," *Computers Chem. Engng*, Vol. 14, 1990, pp. 1271-1273.
- [2] J. T. Betts and W. P. Huffman, "Mesh refinement in direct transcription methods for optimal control," *Optimal Control Applications & Methods*, Vol. 19, 1998, pp. 1-21.
- [3] N. Biehn, S. L. Campbell, F. Delebecque, and R. Nikoukhah, "Observer Design for Linear Time Varying Descriptor Systems: Numerical Algorithms," preprint, 1998.
- [4] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Philadelphia, PA: SIAM, 1996.
- [5] S. L. Campbell, "Numerical methods for unstructured higher index DAEs," *Annals of Numerical Mathematics*, Vol. 1, 1994, pp. 265-278.
- [6] S. L. Campbell, "High index differential algebraic equations," *J. Mech. Struct. & Machines*, Vol. 23, 1995, pp. 199-222.
- [7] S. L. Campbell, "Linearization of DAEs along trajectories," *Z. angew Math. Phys. (ZAMP)*, Vol. 46, 1995, pp. 70-84.
- [8] S. L. Campbell and C. W. Gear, "The index of general nonlinear DAEs," *Numerische Mathematik*, Vol. 72, 1995, pp.173-196.
- [9] S. L. Campbell, R. Hollenbeck, K. Yeomans, and Y. Zhong, "Mixed symbolic-numerical computations with general DAEs I: system properties," *Numerical Algorithms.*, to appear.
- [10] I. Duff and C. W. Gear, "Computing the structural index," *SIAM J. Alg. Discrete Systems*, Vol. 7, 1986, pp. 594-603.
- [11] S. L. Campbell and W. Marszalek, "DAEs arising from the traveling wave solution of PDEs," *J. Computational and Applied Mathematics*, Vol. 82, 1997, pp. 41-58.
- [12] C. W. Gear, "Maintaining solution invariants in the numerical solution of ODEs," *SIAM J. Sci. Stat. Comp.*, Vol. 7, 1986, pp. 734-743.
- [13] E. Hairer, C. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods.*, New York: Springer-Verlag, 1989.
- [14] P. Kunkel and V. Mehrmann, "Canonical forms for linear differential-algebraic equations with variable coefficients," *J. Comp. Appl. Math.*, Vol. 56, 1994) pp. 225-251.
- [15] A. Lefkopoulos and M. A. Stadtherr, "Index analysis of unsteady-state chemical process systems-I. An algorithm for problem formulation," *Computers Chem. Engng*, Vol. 17, 1993, pp. 399-413.
- [16] T. Maly and L. R. Petzold, "Numerical methods and software for sensitivity analysis of differential-algebraic systems," *Appl. Num. Math.*, Vol. 20, 1996, pp. 57-79.
- [17] S. Mattsson and G. Söderlind, "Index reduction in differential-algebraic equations using dummy derivatives," *SIAM J. Sci. Comp.*, Vol. 14, 1993, pp. 677-692.
- [18] V. V. Murata and E. C. Biscaia, Jr., "Structural and symbolic techniques for automatic characterization of differential-algebraic equations," Vol. 21, 1997, pp. S829-S834
- [19] T. Park and P. I. Barton, "State event location in differential-algebraic models," *ACM Transactions on Modeling and Computer Simulation*, Vol. 6, 1996, pp. 137-165.
- [20] K. J. Reinschke, "Graph-theoretic approach to symbolic analysis of linear descriptor systems," *Linear Alg. & Its Appl.*, Vol. 197, 1994, pp.217-244.
- [21] R. von Schwerin, M. Winckler, and V. Shulz, "Parameter estimation in discontinuous descriptor systems," preprint.
- [22] J. Unger, A. Kröner, and W. Marquardt, "Structural analysis of differential-algebraic equation systems – theory and applications," *Computers Chem. Engng*, Vol. 19, 1995, pp. 867-882.