Consistent Initial Conditions for Unstructured Higher Index DAEs: A Computational Study

S. L. Campbell[1] C. T. Kelley[2] K. D. Yeomans

Department of Mathematics
North Carolina State University
Raleigh, NC 27695-8205 USA

ABSTRACT

Differential algebraic equations (DAEs) are implicit systems of ordinary differential equations, $F(x', x, t) = 0$, for which the Jacobian $F_{x'}$ is always singular. DAEs arise in many applications. Significant progress has been made in developing numerical methods for solving DAEs. Determination of consistent initial conditions remains a difficult problem especially for large higher index DAEs. This paper looks at one approach for computing consistent initial conditions for these systems. The focus is on initializing higher index DAE integrators but the observations are relevant to the general problem of initialization of DAE integrators.

1. INTRODUCTION

Many physical problems can be characterized as a nonlinear implicit system of differential and algebraic equations (DAE),

$$F(x', x, t) = 0 \tag{1}$$

with $F_{x'} = \partial F/\partial x'$ identically singular [1]. The index ν, which will be defined later, is one measure of how singular a DAE is. An ordinary differential equation is index zero. A DAE is higher index if $\nu > 1$. Many of the problems in constrained mechanics are initially formulated as index two and three DAEs. Systems of index up to six naturally occur in mechanics if actuator dynamics, joint flexibility, and other effects are included [3]. Higher index DAEs also occur in several other areas [1, 3].

A variety of numerical approaches have been suggested for numerically integrating various classes of DAEs. However, it is generally agreed that the problem of initialization remains a difficult problem. With a DAE not all x can be used for initial conditions. It is often necessary to have initial values of x', and sometimes estimates for higher derivatives of x.

Most of the numerical methods for DAEs require that the systems have special structure, or have indices of only one or two. More general approaches have been introduced. One approach for the numerical integration of general unstructured higher index DAEs is being developed based on the derivative array [8].

There has been some previous work on the initialization of special classes of DAEs [2, 14, 15]. In this paper we examine the problem of computing initial values for the derivative array based general integrators. These methods are described briefly in Section 3 and will be referred to as general DAE integrators. There are other applications for these same computations but they will be examined elsewhere. However, these other applications will influence our choice of approaches to examine.

In initialization we will not have good predictors for the starting values so we need to consider a more global iterative scheme. The globalization of iterative schemes has been well studied in the optimization literature and several choices are available [10, 13]. However, the equations we consider, and the intended application to initialization, has several special features. The equations we must solve are now well understood near a solution.

In this paper we report on some computational studies designed to guide us in developing robust initialization software. We are interested in the practicality of the proposed approach, examining different computational strategies, and in investigating the types of challenges posed by the special types of nonlinear equations that we must solve. Section 2 establishes needed basic facts about higher index DAEs. Section

[1]Research supported in part by the National Science Foundation under DMS-9423705 and ECS-9500589.

[2]Research supported in part by the National Science Foundation under DMS-9321938.

3 describes the various types of general integrators and how their initialization problems differ. Section 4 describes the special features of the initialization and integration problems that impact on choice of equation solving method. The results of the computational tests to date are in Section 5.

2. HIGHER INDEX DAES

In general, the solution $x(t)$ of (1) is known to depend on derivatives of the equations F. Suppose the DAE (1) is a system of m equations in the $(2m+1)$-dimensional variable (x', x, t) and that $F_{x'}$ is always singular. We also assume that F is sufficiently differentiable in the variables (x', x, t) so that all necessary differentiations can be carried out. If (1) is differentiated k times with respect to t, we get

$$\begin{aligned} F(x', x, t) &= 0 \\ &\vdots \\ \frac{d^k}{dt^k}F(x', x, t) &= 0. \end{aligned}$$

These $(k+1)m$ equations are called the *derivative array equations* [5], and denoted

$$G(z) = G(x', w, x, t) = 0, \qquad (2)$$

where $w = [x^{(2)}, \cdots, x^{(k+1)}]$, $z = (x', w, x, t)$.

The *(differentiation) index* ν of the DAE (1) is often taken to be the least integer k for which (2) uniquely determines x' for consistent (x, t). If such a k exists, then x' is a function of just (x, t) (for consistent (x, t)) so that $x' = g(x, t)$. For general unstructured DAEs, the index is a more subtle concept than this definition suggests [4]. A DAE is called solvable if there exists a smooth manifold of solutions [1, 4]. Many problems have a structure which allows one to differentiate some of the equations less than other equations thereby reducing the size of (2). While we do so in practice we will not indicate this reduced differentiation with our notation.

3. GENERAL DAE INTEGRATORS

We shall assume that the DAE is a solvable DAE in a moderate number of variables and that formulas are known for the equations making up the DAE. In this section we describe two general approaches.

A matrix C is *1-full* with respect to a component x_1 of x if $Cx = b$ uniquely determines x_1 for consistent b. We assume throughout this paper that there is an integer k such that the following assumptions hold.

(A1) Sufficient smoothness of G.

(A2) Consistency of $G = 0$ as an algebraic equation.

(A3) $\overline{J} = [\ G_{x'} \quad G_w\ ]$ is 1-full with respect to x' and has constant rank independent of (x', w, x, t).

(A4) $J = [\ G_{x'} \quad G_w \quad G_x\ ]$ has full row rank independent of (x', w, x, t).

Conditions (A1)–(A4) frequently hold in practice and are numerically verifiable using a combination of symbolic and numeric software [5].

Implicit Coordinate Partitioning (ICP)
It is often desirable to preserve constraints. One cannot just use the obvious projection to stay on the solution manifold if the problem is unstructured. An alternative is developed in [8] that is similar in principle to the generalized coordinate partitioning used in mechanics. However, we do not assume that any of the constraints are known explicitly. Thus the coordinates must be picked in an implicit manner. Also we do not assume that the equations have any specific structure such as being Euler-Lagrange equations. A brief outline of this approach is the following.

We take a partition of the state variables $x = (x_1, x_2)$ so that

$$\widetilde{J} = [G_v \ G_w \ G_{x_1}] \qquad (3)$$

has full row rank where $v = x'$. The variables x_2 are used to parameterize the solution manifold locally. Define the variables $\widetilde{z}, \omega$ by $\widetilde{z} = ([x', w], x_1) = (\omega, x_1)$. Thus $\widetilde{z}$ is everything but x_2 and ω is everything but x. Then $G_{\widetilde{z}}$ is not only 1-full with respect to x' but it is also full row rank by the choice of x_1. Thus $G_{\widetilde{z}}(\widetilde{z}, x_2, t)^T G(\widetilde{z}, x_2, t) = 0$ is equivalent to $G(\widetilde{z}, x_2, t) = 0$. We may now use any method of finding $\widetilde{z}$ given (x_2, t) which minimizes $C(\widetilde{z}) = \frac{1}{2}G(\widetilde{z}, x_2, t)^T G(\widetilde{z}, x_2, t)$. We have chosen to use a Gauss-Newton iteration.

Suppose then that we have a point $(\widetilde{z}_0, x_{20}, t_0)$ where $G(\widetilde{z}_0, x_{20}, t_0) = 0$ and our assumptions hold. Then by [8] there is a partitioning such that the limit $({x'_1}^*, {x'_2}^*, w^*, x_1^*)$ of the Gauss Newton iteration satisfies the fundamental equations

$$\begin{aligned} {x'_1}^* &= f_1(x_2, t) & (4a) \\ {x'_2}^* &= f_2(x_2, t) & (4b) \\ x_1^* &= h(x_2, t) & (4c) \end{aligned}$$

where f_1, f_2, h are defined by the limit of the iteration.

One step of the integration of the DAE now goes as follows. Given $x_{n-1} = (x_{1,n-1}, x_{2,n-1})$ we apply a multistep integrator to (4a)–(4b) to get $\widehat{x}_n = (\widehat{x}_{1,n}, \widehat{x}_{2,n})$. A final function evaluation gives $x'_n = (x'_{1,n}, x'_{2,n})$ and $\overline{x}_{1,n} = h(\widehat{x}_{2,n}, t_n)$. Then the value for x_n is taken to be $(\overline{x}_{1,n}, \widehat{x}_{2,n})$. Thus x_n lies on the solution manifold and satisfies all constraints. This approach will be referred to here as the *implicit coordinate partitioning approach* (ICP).

Explicit Integration (EI)
The explicit approach differs in several aspects from ICP. At a given value of (t, x) we hold x fixed and do a Gauss-Newton with the Jacobian $[G_v,\ G_w]$ instead of $[G_v, G_w, G_{x_1}]$. This gives an ODE $x' = h(x, t)$ which can now be integrated by a standard integrator.

Since the Jacobian is no longer full rank the linear algebra requires a rank determination. Also there are some subtleties with Jacobian reuse during integration. The constraints are not explicitly reimposed so that there can be drift off the manifold. On the other hand, the explicit approach does not require the monitoring of the local coordinates and thus is simpler to implement and somewhat more robust than ICP.

4. ALGORITHMIC ISSUES

Problem Features
The initialization of higher index DAEs has several features that need to be kept in mind in the developing of numerical methods.

One option of many ODE and DAE integrators is for the user to specify an absolute error tolerance that the solution is to meet. In order to meet this error tolerance it is essential that the initial conditions be found to a given tolerance. This means that stopping criteria must both guarantee that we are near a suitable initial condition and that the approximation is within a prescribed error tolerance. In particular, our stopping criteria must insure both small residual and small steps.

With initialization we expect to have very poor initial estimates of many of the initial values. Thus we need a robust global algorithm. We assume we are dealing with equations which are known explicitly. By using automatic differentiation codes if necessary we can assume that exact up to round off error Jacobians are available at substantially less computational cost than that of one matrix factorization [6].

It is fairly rare in our experience to just want an initial condition. Usually there is a subset of the variables which are known, or for which we have good estimates. The number of these variables could be the same, less than, or greater than the actual degrees of freedom in the solution. If these quantities are allowed to vary during the iteration, then the final initial condition may have greatly changed in these components. Thus we want to allow for the user to specify a subset of the variables. We would like to get a single method that works for any reasonable specification by the user.

With any DAE integrator one usually needs not only initial values of x but also x'. However with either the EI approach or the ICP approach we have to initialize the entire w vector for the nonlinear equation solver. For a general unstructured problem, parts of the w vector will be arbitrary. We shall consider three scenarios. The assumptions given on the Jacobians are based on the theoretical results used in developing the general integrators [5]. To simplify our notation we let u denote whichever variables are allowed to vary and we write $G(u)$. The limit of the iteration is u^* and $J(u) = G_u(u)$.

Scenario 1 (S1): We merely seek a solution of the derivative array equations. No components of z are assumed known. The Jacobian is full row rank in a neighborhood of u^*.

Scenario 2 (S2): A subset z_2 of z is specified. We assume that $\dim(z_2)$ is less than the dimension of the solution manifold of the DAE and that the z_2 variables are a subset of local coordinates for the solutions of $G(z) = 0$. The Jacobian is full row rank in a neighborhood of u^*.

Scenario 3 (S3): A subset z_2 of z is specified. We assume that $\dim(z_2)$ is greater than the dimension of the solution manifold of the DAE. The Jacobian is assumed constant rank in a neighborhood of the limit point. However the Jacobian is neither full row nor full column rank. This scenario is also important during the integration of a DAE by the EI approach. We expect $G(u^*)$ will be small but possibly nonzero.

There is a hybrid scenario where first a portion z_1 of the variables are kept constant and the remaining variables determined by an iteration. Then all of the variables are allowed to vary. This could be modified where in the final stage some of the z_1 are still fixed. The idea is to first reduce the error in the less well known variables so that their "error" is the same as the error of the better known variables. Prior experience has suggested that this is sometimes advantageous [9]. However, we will not discuss this approach further here.

We assume the Jacobian has constant rank in a neighborhood of u^*. There is no reason to assume that J has constant rank during the iteration. The typical situation that we consider is that the Jacobian is constant rank throughout its domain except on certain lower dimensional manifolds. During the iteration we may pass near, or land on, these manifolds.

Approaches Considered

There are three things to be decided for an iteration; what direction to move, Δu_n, how far to move in that direction, and how to terminate the iteration. Our iterations will take the form

$$u_{n+1} = u_n + \rho_n \Delta u_n \tag{5}$$

where $0 < \rho_n \leq 1$. In this paper we will terminate if Δu and the residual $\|G\|$ are less than tolerances E_X and E_R. E_X is to insure sufficient accuracy in the solution. E_R is to prevent prematurely stopping the iteration.

Gauss-Newton: The *plain Gauss-Newton* iteration uses $\Delta u_n = -J^\dagger(u_n)G(u_n)$ and $\rho_n = 1$:

$$u_{n+1} = u_n - J^\dagger(u_n)G(u_n) \tag{6}$$

The theory for Gauss-Newton (like ordinary Newton's method) requires the starting value to be near the solution. The plain Gauss-Newton has performed reasonably well in our prior experiments with integrators. However, as to be expected, it performed very poorly during our initialization tests with even moderately poor initial guesses.

There are a variety of (minimization) algorithms and some are being examined for use with DAEs. For example, Biegler is currently examining sequential linear programming (SLP) for chemical engineering problems [11]. However, there are several reasons for wanting to utilize a method based on a variant of the Gauss-Newton iteration. One of the most important for us is that such a method is based on information of the type that we are using in the DAE integrator. Also, in the numerical integrator it would be advantageous to have a more robust iteration. A good initialization strategy might also prove beneficial in our integrators if step size is being limited by the iteration needing good starting values. This could permit us to take larger time steps or to use lower order integrators on higher index DAEs.

There are many ways to pick ρ_n in the *damped Gauss Newton*:

$$u_{n+1} = u_n - \rho_n J^\dagger(u_n)G(u_n) \tag{7}$$

We want the method to revert to plain Gauss Newton near u^*. We choose ρ_n so that the value of a scalar test function $T_n(u)$ decreases on the nth iteration. There are a variety of line search methods. We initially take $\rho_n = 1$ and then halve ρ_n until

$$\|T_n(u_{n+1})\| \leq (1 - \alpha\rho_n)\|T_n(u_n)\| \tag{8}$$

for a fixed α. The norm squared of the residual is one natural choice of T_n

$$T_n^{[1]}(u) = G^T(u)G(u) = \|G(u)\|^2 \quad (\mathbf{T1}) \tag{9}$$

There are several reasons to expect that a different test function might be better. First, because of the absolute error tolerances on u^* we do not want to terminate the iteration just on small residuals $G(u_n)$. Also, we expect that the derivative array equations will sometimes have nontrivial condition numbers and there may be ill conditioning encountered during the iteration. Condition numbers of 10^4 or higher can be routinely expected. The limit of the Gauss-Newton iteration satisfies $J^\dagger(u^*)G(u^*) = 0$. Bock and Deuflhard have suggested the test function

$$T_n^{[2]}(u) = \|J^\dagger(u_n)G(u)\|^2 \quad (\mathbf{T2}) \tag{10}$$

The gradient of $T_n^{[2]}(u)$ is

$$\nabla T_n^{[2]}(u) = G(u)^T(J^\dagger(u_n))^T J^\dagger(u_n)J(u) \tag{11}$$

At the current point u_n we have that $\nabla T_n^{[2]}(u_n)$ is the Gauss-Newton direction. Thus there is always a $\rho \leq 1$ which will cause $T_n^{[2]}$ to decrease.

Truncated Gauss Newton: If $J = U\Sigma V$ is the SVD of J, let Σ_δ be Σ but with all singular values below δ set to zero. Define $J_\delta = U\Sigma_\delta V$ and $J_\delta^\dagger = (J_\delta)^\dagger$. Often one wants δ greater than machine roundoff and below the smallest nonzero singular value of $J(u^*)$. This leads to the iteration

$$u_{n+1} = u_n - \rho_n J_\delta^\dagger(u_n)G(u_n) \tag{12}$$

In all our calculations involving singular values the small singular values are set to zero so we are always using a J_δ instead of J.

Steepest Descent Residual Minimization: One could try to minimize the residual $\|G(u)\|^2$ in the direction of its gradient.

$$u_{n+1} = u_n - \rho_n J^T(u_n)G(u_n) \tag{13}$$

Levenberg-Marquardt: During the iteration one can encounter or pass close to singularities in the rank of J. A classical way to handle singularities during an iteration is Levenberg-Marquardt

$$u_{n+1} = u_n - \rho_n(\epsilon_n I + J^T J)^{-1} J^T G(u_n) \tag{14}$$

with J evaluated at u_n and $\epsilon \to 0$ as $u_n \to u^*$. If $J(u^*)$ has full row rank, then this method acts like Gauss Newton near u^*. Away from u^* it acts like $J^\dagger G$ in the direction of large singular values of J and like $\frac{1}{\epsilon}J^T$ in the direction of small singular values of J. Because we have full row rank in Scenarios 1 and 2 instead of column rank, the variant

$$u_{n+1} = u_n - \rho_n J^T(\epsilon_n I + JJ^T)^{-1} G(u_n) \tag{15}$$

is more appropriate. In the direction of large singular values of J, (14) acts like (13).

Proposition 1 *Given the above definitions and assumptions:*

1. $J^\dagger(u)G(u) \neq 0$ *if and only if* $J^T(u)G(u) \neq 0$.
2. $J^\dagger_\delta(u)G(u) \neq 0$ *implies that* $J^\dagger(u)G(u) \neq 0$.
3. $J^\dagger(u_n)G(u_n)$, $J^T(\epsilon + JJ^T)^{-1}G$, $J^T(u_n)G(u_n)$ *are descent directions for (T1) and (T2) if any one of them is nonzero.*
4. $J^\dagger_\delta G$ *is a descent direction for (1) and (T2) if it is nonzero.*

It is worth noting that Proposition 1 need not be true if other types of generalized inverses are used.

5. COMPUTATIONAL EXPERIMENTS

In order to get a feel for the behavior of these globalization methods we have conducted a series of computational experiments. We will consider three test problems which are taken from [6, 9, 8]. They represent different types of difficulties that can occur in applications. The testing of additional larger examples is underway.

Test Problem 1. (Chemical Reactor): This is an index three DAE in 4 variables. Many, but not all of the terms are linear. All equations are differentiated 3 times so that G has 16 equations.

Test Problem 2. (Shuttle): This is an index three DAE in 7 variables [1]. It has a Hessenberg structure which generally helps the iteration. The equations are highly nonlinear. In this example we exploit the Hessenberg structure of the equations and differentiate equations from 1 to 3 times each. G has 20 equations. It is known that there are two values of the control variable (bank angle β) that are close to each other. Only $\beta > 0$ is physically correct.

Test Problem 3. (Torus): This is an index 3 DAE in 7 variables [8]. It is fully implicit. In addition, at different times different subsets of initial conditions can be taken fixed since the solution winds around a torus. All equations are differentiated 3 times so that G has 28 equations.

Implementation

The Jacobians were given by MAPLE generated FORTRAN subroutines [6]. Our interest at this point is in finding which methods are the most robust and in examining problem behavior. Accordingly we compute $J^\dagger G$ using an SVD. The cost of integrator initialization is amortized over the cost of performing the integration which reduces the cost somewhat. However, it is important to speed up the linear algebra in any final code. We took $\alpha = 10^{-4}$ and $\delta = 10^{-8}$ unless stated otherwise. δ is designed to ignore numerically zero singular values rather than to regularize an iteration with small, but nonzero, singular values. Also, $E_X = E_R = 10^{-9}$. The tolerances were set so that we could examine the long run behavior of the iteration. For initialization one might well want different E_X and E_R values.

In our tests we started at a solution u^*. We then generated several random directions w scaled in a similar manner to u^* and took starting values at different distances in those directions. The initial point was $u^* + 10^{\gamma-1}w$. Our intention was to examine the behavior of the iteration and the effect of increasingly poor initial guesses. We experimented with several versions of Levenberg-Marquardt. Having $\epsilon = 0$ for small $\|G\|$ worked best. In what follows we made the simple choice of $\epsilon = 10^{-4}$ if $\|G\| > 10^{-4}$ otherwise $\epsilon = 0$. We also considered a test function $T_n(u) = \|J^T(\epsilon + JJ^T)^{-1}J^TG\|$ as well as T2. The test functions performed the same.

Computational Results

We have run a large number of experimental runs with Scenario 1. Plain Gauss Newton, as expected, performed very poorly when we had poor initial starting values. It clearly is not practical for initialization. Not truncating small singular values when computing $J^\dagger G$ led to convergence problems. On the other hand an aggressive truncation strategy designed to perform regularization near singularities was also less reliable. A small but numerically nonzero value worked best.

We found that implementing Levenberg-Marquardt by forming $(\epsilon I + JJ^T)$, solving $(\epsilon I + JJ^T)y = G$, and then letting $\Delta u = -J^Ty$ frequently converged much more slowly (or not at all) then computing an SVD of $J = U\Sigma V$ and then letting $\Delta u = -V\Sigma^T(\epsilon I + \Sigma\Sigma^T)^{-1}U^TG$. One explanation is that there is too much loss of accuracy in the "normal equations" version of Levenberg-Marquardt.

When the iterations converged the residual usually met its stopping criteria a few iterations before Δu did. On the other hand, particularly with poor initial guesses, there were several examples where Δu met the tolerance but we did not have small residuals. A combined stopping criteria appears to be required.

In the remaining discussion P is undamped, D is damped, GN is Gauss-Newton, LM is Levenberg-Marquardt, R is test function T1 and B is function T2. An * T1 or T2. GN is any of the tested Gauss-Newton methods. Similarly for LM.

On the chemical reactor problem for $\gamma = 0, 1, 2$, GN converged typically in 3, 4, and 5 iterations. LM also converged but in 2 to 3 times as many iterations. However for $\gamma = 3$, PGN often failed. DGN* converged more often. The same held for PLM and DLM*. For some starting values DGN* converged while DLM* did not. The situation was reversed for other starting values.

The shuttle problem is much less well conditioned with singular values ranging from approximately 10^{-4} to 10^4. LM often did not converge even for $\gamma = 0$ or 1. When it did converge it took 150-300 iterations. On the other hand GN took 5-7 iterations for $\gamma = 0$ and 15-35 iterations for $\gamma = 1$. At $\gamma = 2$, PGN often failed while DGN* worked some times.

For the torus problem, all the methods were essentially the same for $\gamma = 0, 1, 2$, taking 4, 4-5, and 6-9 iterations respectively. PGN failed at $\gamma = 3, 4$. DGNR failed at $\gamma = 4$. DGNB frequently converged at $\gamma = 4$ but failed sometimes. On the other hand DLM* converged for $\gamma = 3, 4$.

Finally we considered S2 where some of the variables were fixed. In this test we fixed β, β' and V_R in the shuttle problem. For technical reasons LM has not been run yet. At $\gamma = 1$, GN converged in 11-13 iterations. However, DGNR failed at $\gamma = 2$. The fixing of some entries had the desired affect of making the converged to value closer to the perturbed from value. However, the Jacobian now has few columns. While the Jacobian may still be full row rank it may be less well conditioned. An examination of the singular values of J shows that this in fact occurs during this iteration.

Considerably more testing is needed with more examples, especially larger dimensional ones, and at additional u^* values. However, the tests reported here suggest several conclusions.

6. CONCLUSIONS

Initialization by solving the derivative array appears to be a practical approach for moderately sized problem especially if G and J are computed by automatic differentiation codes. As to be expected, some damping strategy is needed. A damped Gauss Newton method appears attractive. The fixing of certain known values is advantageous but can have the affect of increasing the condition number of Jacobians and increasing the likelihood of passing near singularities. The stopping criteria must enforce both small steps and small residuals.

An examination of the Jacobians during the iterations suggests that scaling may be important in some problems, especially those where a large number of variables are fixed. This will be examined elsewhere.

References

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold, "Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations", SIAM, 1996.

[2] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, "Consistent initial condition calculation for differential-algebraic systems", preprint, 1995.

[3] S. L. Campbell, "High index differential algebraic equations", Mech. Structures and Machines, 23 (1995), 199-222.

[4] S. L. Campbell and C. W. Gear, "The index of general nonlinear DAEs", Numer. Math., 72 (1995), 173-196.

[5] S. L. Campbell and E. Griepentrog, "Solvability of general differential algebraic equations", SIAM J. Sci. Stat. Comp., 16 (1995), 257-270.

[6] S. L. Campbell and R. Hollenbeck, "Automatic Differentiation and Implicit Differential Equations", Proc. Second International Workshop on Computational Differentiation, SIAM, 1996, to appear.

[7] S. L. Campbell and C. D. Meyer, Jr., "Generalized Inverses of Linear Transformations", Dover Press, New York, 1991.

[8] S. L. Campbell and E. Moore, "Constraint preserving integrators for general nonlinear higher index DAEs", Numer. Math., 69 (1995), 383-399.

[9] S. L. Campbell and E. Moore, "A Coordinate Free Approach for Constraint Preserving Higher Index DAE Integrators", Proc. 94 IMACS World Congress on Scientific Computation, 65-68.

[10] J. E. Dennis, Jr., and R. B. Schnabel, "Numerical Methods for Unconstrained Optimization and Nonlinear Equations", SIAM, 1996.

[11] V. Gopal and L. T. Biegler, "An optimization approach to consistent initialization and reinitialization after discontinuities of differential algebraic equations", preprint, 1995.

[12] E. Hairer, C. Lubich, and M. Roche, "The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods", Springer-Verlag, New York, 1989.

[13] C. T. Kelley, "Iterative Methods for Linear and Nonlinear Equations", SIAM, 1995.

[14] B. J. Leimkuhler, L. R. Petzold, and C. W. Gear, "Approximation methods for the consistent initialization of differential-algebraic systems of equations", SIAM J. Numer. Anal, 28 (1991), 205-226.

[15] C. C. Pantelides, "The consistent initialization of differential-algebraic systems", SIAM J. Sci. Stat. Comput., 9 (1988), 213-231.