REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 18 Dec 97	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE INITIALIZATION ISSUES IN GENERAL DIFFERENTIAL ALGEBRAIC EQUATION INTEGRATORS	5. FUNDING NUMBERS
6. AUTHOR(S) KEVIN DEAN YEOMANS	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NORTH CAROLINA STATE UNIVERSITY	8. PERFORMING ORGANIZATION REPORT NUMBER 97-037D
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) THE DEPARTMENT OF THE AIR FORCE AFIT/CIA, BLDG 125 2950 P STREET WPAFB OH 45433	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION AVAILABILITY STATEMENT Unlimited Distribution In Acdordance With AFI 35-205/AFIT Sup 1	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

14. SUBJECT TERMS			15. NUMBER OF PAGES 111
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

DTIC QUALITY INSPECTED 5

Abstract

YEOMANS, KEVIN DEAN. Initialization Issues in General Differential Algebraic Equation Integrators. (Under the direction of Dr. Stephen L. Campbell.)

The past several years have yielded much research into the development of general numerical integrators for nonlinear unstructured higher index differential algebraic equations (DAEs) of the form $F(y', y, t) = 0$ where $F_{y'}$ is identically singular. These methods are based on integrating an implicitly defined ODE produced by forming the derivative array equations $G(y', w, y, t) = 0$. This larger nonlinear system may have components that are not uniquely determined. Prior work has examined the theoretical aspects of these methods. There has also been considerable work done on the efficient implementation of these integrators.

This thesis will examine two initialization problems that arise when numerically solving DAEs. One problem is the computation of the consistent initial conditions required to begin the integration of DAEs. Various line search strategies will be compared and examples provided showing where these strategies are needed.

The second problem is the initialization of the iterative solvers used during the integration of a DAE. Integration of the implicitly defined ODE depends on solving the nonlinear system $G = 0$ at each time step. For fully implicit nonlinear systems the possible effects of polynomial prediction on the nonunique components will be examined. A complete analysis is given in the case of higher index linear time varying DAEs. It is shown that the standard ODE theory does not hold and a different prediction strategy must be used.

INITIALIZATION ISSUES IN GENERAL DIFFERENTIAL ALGEBRAIC EQUATION INTEGRATORS

BY

KEVIN DEAN YEOMANS

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF

NORTH CAROLINA STATE UNIVERSITY

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF MATHEMATICS

RALEIGH

DECEMBER 1997

APPROVED BY:

S. L. CAMPBELL	C. T. KELLEY
CHAIR OF ADVISORY COMMITTEE	
J. S. SCROGGS	H. T. TRAN

Biography

Kevin Dean Yeomans was born July 19, 1962 in Fort Walton Beach, Florida. His family was in the Air Force and moved frequently during his younger years. He graduated from Telfair County High School in McRae, Georgia in 1980.

He received a B.S. degree with a major in Mathematics and a commission in the Air Force from the United States Air Force Academy in 1986. He received a M.S. degree in 1987 from North Carolina State University. From 1987 to 1991, he was assigned to Eglin AFB in Fort Walton Beach, Florida where he worked as an analyst and performed contract management for conventional weapons concepts. Additionally he performed tests of high explosives. From 1992 to 1994, he served as an instructor at the Air Force Academy in Colorado Springs, Colorado. He returned to N.C. State in 1994 to pursue a PhD in Applied Mathematics. He is currently serving on active duty in the Air Force.

The author is married to the former Krista Eileen Carter, and they have four children, Kenny, Kari, Katelyn and Kyle.

Acknowledgements

None of this would have been possible without the patient guidance of my advisor, Dr. Stephen L. Campbell. I have been honored to work with him these past several years. His mathematical knowledge and research expertise have left me in awe. It is hard to convey in words just how much I appreciate all that he has done to help me in this endeavor.

I am also grateful to Dr. Carl T. Kelley, Dr. Jeffrey S. Scroggs and Dr. Hien T. Tran for serving on my committee. Dr. Kelley's course in numerical methods for nonlinear equations and unconstrained optimization was a highlight of the time I spent at N.C. State.

I wish to express my gratitude to the Department of Mathematical Sciences at the United States Air Force Academy for supporting me.

Finally, I must mention the love and support of my family. My wife Krista has sacrificed much for my success. Her efforts in raising and educating our children - Kenny, Kari, Katelyn and Kyle - has allowed me the freedom to pursue this academic dream to broaden my mathematical knowledge.

Table of Contents

List of Tables

List of Figures

Chapter 1

Differential Algebraic Equations

Recently much research has been done in developing general numerical methods for solving differential algebraic equations (DAEs)

$$F(y', y, t) = 0 \tag{1.1}$$

where $F_{y'}$ is identically singular. These systems of differential and algebraic equations arise in a variety of applications which includes constrained variational problems, prescribed path control problems, network modelling problems, model reduction for problems with small parameters, and discretization of partial differential equations by the method of lines or the method of moving grids. A detailed survey of applications and examples can be found in [13, 25, 50].

These general methods discussed in Section 1.2 integrate an implicitly defined ordinary differential equation (ODE) and have the advantage that (1.1) does not require any special structure. Additionally these methods have been demonstrated to work on higher index DAEs. Briefly the index is a measure of how singular a DAE is and will be defined shortly. An ordinary differential equation (ODE) is index zero. The higher the index the more complex behavior of the DAE. Most of the literature on numerical methods for DAEs is confined to systems whose index is three or less.

These general approaches [22, 24, 35] require the solution of an enlarged system of nonlinear equations. The iterative solution of these equations can have several consequences. First is the need for sufficiently accurate initial values. This can force the integrator to either use higher order methods or take smaller steps than the solution of the DAE would appear to need. In this thesis we investigate ways to globalize the iterative solver. This has important consequences for the consistent initialization of higher index DAEs. Additionally more robust step size strategies are possible. We also investigate the use of polynomial interpolation to provide more accurate prediction as an aid in solving these systems. The goal of these efforts is to aid in the development of these more general DAE solvers.

1.1 Introductory Survey

We begin by surveying the current knowledge about the theory and numerical solution of DAEs.

1.1.1 Basic Theory

To understand and illustrate some properties of DAEs, we begin with the following simple example

$$z\,z' + y = f(t) \tag{1.2a}$$

$$z = g(t). \tag{1.2b}$$

Equation (1.2) consists of a differential equation (1.2a) and an algebraic equation (1.2b), which can be thought of as an algebraic constraint. The Jacobian of (1.2) with respect to y', z' is

$$\begin{bmatrix} 0 & z \\ 0 & 0 \end{bmatrix}$$

which is identically singular for all t. By differentiating (1.2a) once and (1.2b) twice to obtain the enlarged nonlinear system

$$\begin{aligned} z\,z' + y &= f(t) && (1.3a)\\ z\,z'' + (z')^2 + y' &= f'(t) && (1.3b)\\ z &= g(t) && (1.3c)\\ z' &= g'(t) && (1.3d)\\ z'' &= g''(t) && (1.3e) \end{aligned}$$

we can solve for y' and z' to obtain the ODE

$$\begin{aligned} y' &= f' - g\,g'' - (g')^2 && (1.4a)\\ z' &= g'. && (1.4b) \end{aligned}$$

The solution to (1.2) is

$$\begin{aligned} y &= f - g\,g' && (1.5a)\\ z &= g. && (1.5b) \end{aligned}$$

It is interesting to note that if g is not continuously differentiable, then the solution component y may in fact be discontinuous. From this simple example we can observe a few properties of DAEs:

(1) solutions to DAEs reside on and form manifolds;

(2) solutions depend on derivatives;

(3) not all initial conditions of the DAE (1.2) admit a smooth solution, those initial values which do are referred to as *consistent initial conditions*;

(4) there are hidden constraints.

Properties (1) and (2) are illustrated by the solution (1.5). In fact, the solution to a DAE may depend on derivatives of the coefficients of the state variables. However, that is not illustrated by this example. It is well known that solutions to DAEs form submanifolds of the state space [77]. The theory for initial value problems applied to ODEs says that any initial condition for which the right hand side of (1.4) is Lipschitz continuous will have a unique solution [61]. However, not every initial condition to (1.4) will admit a solution to the DAE (1.2). By property (3) it is meant that initial conditions must satisfy both the ODE (1.4) and the algebraic constraints (1.5) defined by the DAE. Since solutions reside in and form manifolds, only points on these surfaces can reside on solution curves. Such points are called *consistent.* If we multiply (1.3d) by z and subtract this result from (1.3a) we get (1.5a) which is a constraint along with (1.5b) that explicitly defines the solution manifold to the DAE, illustrating property (4). Later, the *index* of a DAE will be defined. It is known that higher index DAEs will always include hidden constraints. These properties illustrate some of the important differences between solving ODEs and DAEs.

Next, we define what we mean for a DAE to be *solvable.* Basically $y(t)$ is a solution of (1.1) on an interval I if it is continuously differentiable and satisfies (1.1) for all $t \in I$. The following definition is found in [13, 27].

Definition 1.1.1 *Let I be an open subinterval of $\mathbb{R}$, Ω a connected open subset of $\mathbb{R}^{2s+1}$, and F a differentiable function from Ω to $\mathbb{R}^s$. Then the DAE* (1.1) *is* solvable *on I in Ω if there is an r-dimensional family of solutions $\phi(t,c)$ defined on a connected open set $I \times \tilde{\Omega}$, $\tilde{\Omega} \subset \mathbb{R}^r$, such that:*

1. *$\phi(t,c)$ is defined on all of I for each $c \in \tilde{\Omega}$;*
2. *$(\phi'(t,c), \phi(t,c), t) \in \Omega$ for $(t,c) \in I \times \tilde{\Omega}$;*
3. *If $\psi(t)$ is any other solution with $(\psi'(t), \psi(t), t) \in \Omega$, then $\psi(t) = \phi(t,c)$ for*

some $c \in \tilde{\Omega}$;

4. *The graph of ϕ as a function of (t, c) is an $r + 1$-dimensional manifold.*

Intuitively the DAE (1.1) is *solvable* in an open set $\Omega \subset \mathbb{R}^{2s+1}$ if the graphs (y', y, t) of the solutions form a smooth manifold Ω called the *solution manifold* and solutions are uniquely determined by their value y_0 at any t_0 such that $(y'_0, y_0, t_0) \in \Omega$ [27]. It is possible that $r = 0$ in which case the DAE has exactly one solution as in (1.2).

We have already observed that the solution to DAEs depends on differentiation. The next key definition is that of the *uniform differentiation index* [13, 26].

Definition 1.1.2 *The minimum number of times that all or part of* (1.1) *must be differentiated with respect to t in order to determine y' as a* continuous *function of y, t is the* index, *ν, of the DAE* (1.1).

Example (1.2) is an index two DAE. Higher index problems are traditionally more difficult to solve numerically. There are many equivalent definitions for the index of a linear time invariant DAE. For nonlinear DAEs there are several definitions of index that are not equivalent [26]. The uniform differentiation index is a computable quantity for moderately sized problems and is closely related to establishing solvability [27].

The following special structural forms for DAEs are frequently cited in the literature

- Fully Implicit DAE

$$F(y', y, t) = 0 \tag{1.6}$$

- Semi-Explicit DAE

$$y' = f(x, y, t) \tag{1.7a}$$

$$0 = g(x, y, t) \tag{1.7b}$$

- Linear Time Invariant DAE

$$Ey' + Fy = f(t) \tag{1.8}$$

- Linear Time Varying DAE

$$E(t)y' + F(t)y = f(t) \tag{1.9}$$

- Hessenberg Index r DAE

$$\begin{aligned}
y_1' &= f_1(y_1, y_2, \ldots, y_r, t) && \text{(1.10a)}\\
y_2' &= f_2(y_1, y_2, \ldots, y_{r-1}, t) && \text{(1.10b)}\\
&\vdots\\
y_i' &= f_i(y_{i-1}, y_i, \ldots, y_{r-1}, t) && \text{(1.10c)}\\
&\vdots\\
y_{r-1}' &= f_{r-1}(y_{r-2}, y_{r-1}, t) && \text{(1.10d)}\\
0 &= f_r(y_{r-1}, t) && \text{(1.10e)}
\end{aligned}$$

where $y_i \in \mathbb{R}^{n_i}$, for $i = 1, \ldots, r$ and

$$\frac{\partial f_r}{\partial y_{r-1}} \frac{\partial f_{r-1}}{\partial y_{r-2}} \cdots \frac{\partial f_2}{\partial y_1} \frac{\partial f_1}{\partial y_r}$$

is nonsingular.

The theory for (1.8) is well established. In [13] necessary and sufficient conditions for (1.8) to be solvable are expressed in terms of a matrix pencil. Given matrices E, F and $\lambda \in \mathbb{C}$, then $\lambda E + F$ is called a *matrix pencil.* $\lambda E + F$ is said to be a *regular* pencil if the determinant is not identically zero as a function of λ. (1.8) is solvable if and only if $\lambda E + F$ is a regular pencil [13]. If (1.8) is solvable then there exists nonsingular matrix P, Q so that we can rewrite

$$Ey' + Fy = f(t)$$

as

$$PEQx' + PFQx = Pf(t) = g(t) \tag{1.11}$$

where

$$PEQ = \begin{bmatrix} I & 0 \\ 0 & N \end{bmatrix}, \quad PFQ = \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix}.$$

N is a nilpotent matrix whose index is the same as the uniform differentiation index defined earlier. The decoupled system (1.11)

$$x_1' + Cx_1 = g_1(t) \tag{1.12a}$$

$$Nx_2' + x_2 = g_2(t) \tag{1.12b}$$

can then be easily solved. Equation (1.12a) is an explicit ODE that has a unique solution for any initial value of x_1 and differentiable forcing function $g_1(t)$. The unique solution to (1.12b) is

$$x_2 = (ND + I)^{-1} g_2(t) = \sum_{i=0}^{k-1} (-1)^i N^i g_2^{(i)}(t)$$

where k is the index, or degree of nilpotency, of N, and D is the differentiation operator. Again, we see that initial values for x_2 are completely determined.

For (1.9), we have that if $E(t), F(t)$ are real analytic, then (1.9) is solvable if and only if we can rewrite the system using linear time varying coordinate changes as

$$x_1' + C(t)x_2 = g_1(t) \tag{1.13a}$$

$$N(t)x_2' + x_2 = g_2(t). \tag{1.13b}$$

The unique analytic solution to (1.13b) is

$$x_2(t) = \sum_{i=0}^{k-1} (-N(t)D)^i g_2(t). \tag{1.14}$$

The difficulty is computing the time varying coordinate transformations that will transform (1.9) into the equivalent system (1.13). In [8, 21, 75], more results are given which establish the existence and uniqueness of solutions to (1.9). In Section 1.2 we give assumptions (A1)–(A4) that are computationally verifiable and equivalent to solvability for every sufficiently smooth function $f(t)$.

Additional existence and uniqueness results for solutions to (1.6) and other structural forms are given in [72, 73, 76, 77]. Most of these results are established from a differential–geometric approach. This approach is based on the observation that DAEs are locally equivalent to ODEs defined on a constraint manifold.

1.1.2 Numerical Methods

Most of the numerical literature to date discusses numerical methods for DAEs whose index is less than three or DAEs with special structure such as Hessenberg. Some of the earliest numerical methods for DAEs were backward differentiation formulas (BDF) applied to semi-explicit index one systems [45]. Later BDF methods were extended to fully implicit index one systems

$$F(y', y, t) = 0. \tag{1.15}$$

The k-step constant step size BDF replaces y' by a polynomial which interpolates the computed solution at $t_n, t_{n-1}, \ldots, t_{n-k}$ to yield

$$F\left(\frac{1}{h}\sum_{i=0}^{k} \alpha_i y_{n-i}, y_n, t_n\right) = 0. \tag{1.16}$$

Equation (1.16) is then solved by Newton's, or other iterative methods, for y_n. Theory for the convergence of the k-step ($k < 7$) constant step size BDF methods has been proven in the case of linear constant coefficient, fully implicit index one, semi-explicit index two and Hessenberg index two and three systems [13]. The results are also true for variable step BDF methods except in the Hessenberg index three case. The BDF code DASSL [13, 71] is one of the most widely recognized DAE production codes for solving index one systems. Several variants of DASSL have recently been developed. These extensions include DASPK [16] for solving large-scale DAE systems and DASSLSO [66] for sensitivity analysis of DAE systems. The linear system at each step of the integration is solved by the use of a preconditioned GMRES [55] iterative solver in the PK variants. Another well known BDF code for index one DAEs is LSODI [53].

Implicit Runge-Kutta (IRK) methods have been studied extensively in the case of Hessenberg index one, two and three systems [50]. An M-stage method applied to (1.15) is given by

$$F\left(Y_i', y_{n-1} + h\sum_{j=1}^{M} a_{ij}Y_j', t_{n-1} + c_i h\right) = 0, \quad i = 1, 2, \cdots, M$$

$$y_n = y_{n-1} + h\sum_{i=1}^{M} b_i Y_i'$$

where $h = t_n - t_{n-1}$, Y_i' are estimates for $y'(t_{n-1} + c_i h)$ and are called state derivatives, and a_{ij}, c_i, b_i are the coefficients for the method [13]. Theory for the fully implicit index one, semi-explicit index one and two cases can be found in [13, 50]. RADAU5 is an implicit Runge-Kutta code that will solve Hessenberg systems of index one to three [50].

Linear multistep, one-leg and implicit Runge-Kutta methods are extensively examined and analysed in [49, 67]. However, it is assumed the nullspace of $F_{y'}$ in (1.1) depends only on t or is constant.

Extrapolation methods have also been applied to the solution of index one DAEs. LIMEX [42] is one example of a code that has been developed using extrapolation.

MEXX [64] is another extrapolation method based on half-explicit Euler or half-explicit midpoint formulas for solving the equations of motion of constrained multi-body systems. More discussion can be found in [13].

The numerical solution of algebraically explicit DAEs is examined in [78, 79]. It is assumed that all the algebraic constraints or the algebraic variables are explicitly defined. Existence proofs for several types of DAEs of index one to three are given.

A specialized algorithm for the numerical solution of the Euler–Lagrange equations is given in [74]. The problem is reduced to a second order ODE on the constraint manifold.

Finally, we mention a code that has recently been released for solving higher index linear time varying DAEs. GELDA [60] is based on the index one integration method described in Section 1.2.3. It can deal with systems that do not have unique solutions or inconsistencies in the initial values. A nonlinear version of this approach is currently being developed.

1.1.3 Applications

In this section we briefly outline some applications where DAEs occur. In control theory applications, it is frequently the case that we have a differential equation of the form $F(y', y, t, u) = 0$ where u represents a set of controls. The controls are applied so that the solution satisfies some constraints $g(y, u) = 0$. Such problems naturally give rise to DAEs even if the differential equation is explicit, $F(y', y, t, u) = y' - f(y, u)$. DAEs whose index is between four and seven frequently arise in control and mechanics applications [25].

Our first example models the planar motion of a ship loading crane as shown in Figure 1.1. A discussion of this model is found in [25, 44] and is a classical object of study. The model can be derived by applying Newton's law to obtain the

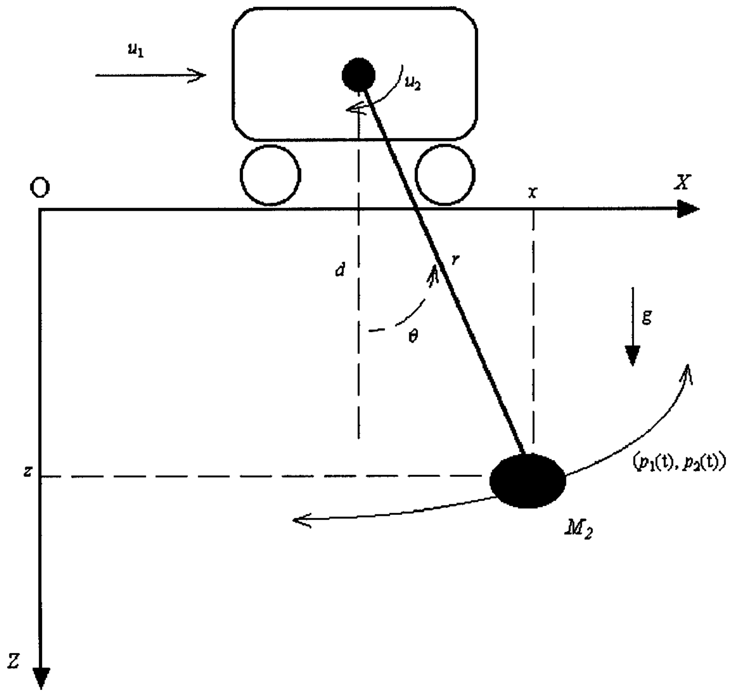

Figure 1.1: Two Dimensional Crane.

differential equations and the geometric constraints of the system give rise to the algebraic equations. The crane consists of a trolley of mass M_1 that moves along a linear horizontal track. A cable of length r, and tension τ, connects a winch on the trolley to a mass M_2. The controls consist of an external force u_1 which moves the trolley and a torque u_2 that is applied to the winch. The trolley location is d while the load location is (x, z). The angle of the cable with the vertical is θ and J is the moment of inertia of the winch. The load is required to follow a prescribed path $(p_1(t),\ p_2(t))$. The resulting index five DAE in $\{u_1, u_2, x, z, \tau, \theta, x', z', r, d, r', d'\}$ is

$$M_2\, x'' = -\tau \sin\theta \tag{1.17a}$$

$$M_2\, z'' = -\tau \cos\theta + mg \tag{1.17b}$$

$$M_1\, d'' = -C_1\, d' + u_1 + \tau \sin\theta \tag{1.17c}$$

$$J\,r'' = -C_2\,r' - C_3\,u_2 + C_3^2\,\tau \tag{1.17d}$$

$$0 = r\,\sin\theta + d - x \tag{1.17e}$$

$$0 = r\,\cos\theta - z \tag{1.17f}$$

$$0 = x - p_1(t) \tag{1.17g}$$

$$0 = z - p_2(t). \tag{1.17h}$$

System (1.17) will be an index six DAE if actuator dynamics are included. Note that this DAE system is not in Hessenberg form since the variable θ in the constraints does not appear in differentiated form.

The next example is a slight simplification of the equations for the prescribed path control of a two-link, flexible joint, planar robotic arm as depicted in Figure 1.2.

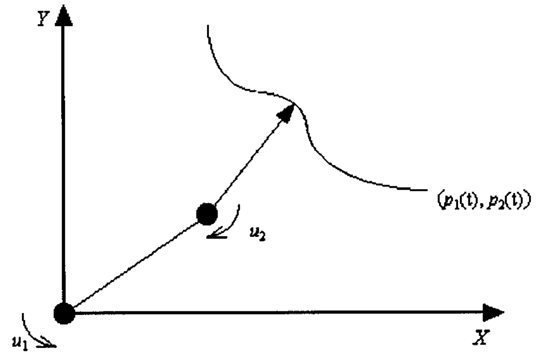

Figure 1.2: Two-Link, Flexible Joint, Planar Robotic Arm.

$$x_1' = x_4 \tag{1.18a}$$

$$x_2' = x_5 \tag{1.18b}$$

$$x_3' = x_6 \tag{1.18c}$$

$$x_4' = 2c(x_3)(x_4+x_6)^2 + d(x_3)x_4^2 + (2x_3 - x_2)(a(x_3) + 2b(x_3)) + a(x_3)u_1$$

$$-a(x_3)u_2 \tag{1.18d}$$

$$\begin{aligned} x_5' &= -2c(x_3)(x_4+x_6)^2 - d(x_3)x_4^2 + (2x_3 - x_2)(1 - 3a(x_3) - 2b(x_3)) \\ &\quad -a(x_3)u_1 + (a(x_3)+1)u_2 \end{aligned} \tag{1.18e}$$

$$\begin{aligned} x_6' &= -2c(x_3)(x_4+x_6)^2 - d(x_3)x_4^2 + (2x_3 - x_2)(a(x_3) - 9b(x_3)) - 2c(x_3)x_4^2 \\ &\quad -(x_4+x_6)^2 d(x_3) - (a(x_3)+b(x_3))u_1 + (a(x_3)+b(x_3))u_2 \end{aligned} \tag{1.18f}$$

$$0 = \cos x_1 + \cos(x_1 + x_3) - p_1(t) \tag{1.18g}$$

$$0 = \sin x_1 + \sin(x_1 + x_3) - p_2(t) \tag{1.18h}$$

where

$$p_1(t) = \cos(e^t - 1) + \cos(t-1) \tag{1.19a}$$

$$p_2(t) = \sin(1 - e^t) + \sin(1-t) \tag{1.19b}$$

$$a(s) = \frac{2}{2-\cos^2 s}, \quad b(s) = \frac{\cos s}{2-\cos^2 s} \tag{1.19c}$$

$$c(s) = \frac{\sin s}{2-\cos^2 s}, \quad d(s) = \frac{\cos s \sin s}{2-\cos^2 s}. \tag{1.19d}$$

System (1.18,1.19) is also an index five DAE in $\{x_1, \ldots, x_6, u_1, u_2\}$ that is not in Hessenberg form. Equations (1.18a)–(1.18f) model the dynamics of the robot arm while equations (1.18g)–(1.18h) specify the moving end of the robot arm to be on the prescribed path $(p_1(t), p_2(t))$ [22, 25]. The angle of the first link with respect to the x axis is x_1, the angle of the second link to the first is x_3, and x_2 is the rotor angle with respect to the second link which is present because of the flexible joint [22]. The torques applied to the first and second joints are the control variables u_1, u_2 respectively.

In Chapter 2, there are a few more examples of index three DAEs that arise in applications and will be discussed there. These are a chemical reactor problem, a trajectory prescribed path control problem (TPPC), and the torus problem [68]. However, we wanted to illustrate here that models of unstructured higher index DAEs do arise in applications. Methods for directly solving these systems is an area of active research.

1.2 General DAE Integrators

The general DAE integrators described in this thesis are based on integrating an implicit ODE defined by the *derivative array equations* [24]. Suppose the DAE (1.1) is a system of s equations in the $(2s+1)$-dimensional variable (y', y, t). We assume that F is sufficiently differentiable in the variables (y', y, t) so that all necessary differentiations can be carried out. In general, the solution $y(t)$ of (1.1) is known to depend on derivatives of F. If (1.1) is differentiated k times with respect to t, we get

$$\begin{aligned} F(y',y,t) &= 0 && (1.20a)\\ F_{y'}(y',y,t)y'' + F_y(y',y,t)y' + F_t(y',y,t) &= 0 && (1.20b)\\ &\vdots && \\ \frac{d^k}{dt^k}F(y',y,t) &= 0. && (1.20c) \end{aligned}$$

These $s(k+1)$ equations are called the derivative array equations and denoted by

$$G(y', w, y, t) = 0 \tag{1.21}$$

where

$$w = \left(y^{(2)}, \ldots, y^{(k+1)}\right).$$

We have

$$G : \Theta \subset \mathbb{R}^{s(k+2)+1} \to \mathbb{R}^{s(k+1)} \tag{1.22}$$

where it is assumed that Θ is open. Often in practice all of the equations are not differentiated ν times. If k is the maximum number of times that any one equation is differentiated, then

$$G : \Theta \subset \mathbb{R}^n \to \mathbb{R}^m \tag{1.23}$$

where $n = s(k+2)+1$ and $m \leq s(k+1)$. The derivative array equations provide information about the solvability of the DAE (1.1), the index, and the dimension of the solution manifold [27]. In Definition 1.1.2 we defined the index ν of the DAE (1.1) to be the least integer k for which (1.21) uniquely determines y' for consistent (y, t). If such a k exists, then y' is just a function of (y, t) so that

$$y' = f(y, t). \tag{1.24}$$

In this thesis we assume that the DAE (1.1) is solvable in a moderate number of variables and that formulas are explicitly given for the equations making up the DAE.

Before proceeding, we need one final definition.

Definition 1.2.1 *The matrix A in the linear system $Ax = b$ is said to be* 1-full *with respect to x_1 if there is a nonsingular matrix B such that*

$$BA = \begin{bmatrix} I & 0 \\ 0 & C \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

The following proposition is proven in [68].

Proposition 1.2.1 *The following are equivalent:*

(i) *A is one full with respect to x_1;*

(ii) *there is a nonsingular matrix B such that*

$$BA = \begin{bmatrix} I & 0 \\ 0 & C \end{bmatrix}$$

where I is an identity matrix having the same size as x_1; and

(iii) *if*

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathcal{N}(A)$$

then $x_1 = 0$.

Finally, we assume throughout this thesis that an integer ν in (1.20) exists so that the following assumptions hold.

(A1) Sufficient smoothness of G.

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

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

(A4) $J_I = [G_{y'}\ G_w\ G_y]$ has full row rank independent of (y', w, y, t)

Conditions (A1)–(A4) frequently hold in practice and are verifiable using a combination of symbolic and numeric software [27, 30]. Additionally, (A1)–(A4) are directly in terms of the original equations and their derivatives and do not require any sort of coordinate changes. (A1) can often be shown to hold on an open set by examining the functions which define the DAE. (A4) can be verified using standard numerical linear algebra routines. If (A4) holds at a point, then it will hold in a neighborhood of that point by continuity [30]. Assumption (A2) is easily checked by evaluating G. Assumption (A4) is the most difficult to verify requiring both numerical and symbolic computations. More discussion can be found in [30]. These assumptions are almost equivalent to a type of uniform solvability as discussed in [26].

Currently three different general integrators are being developed. All are based on the derivative array equations but differ substantially in their approach.

1.2.1 Explicit Integration Method

The first general DAE method we examine is discussed in [22]. The explicit integration method (EI) approach consists of integrating (1.24) by an ODE integration method. In order to do this one needs to be able to evaluate $f(y_n, t_n)$ for a given y_n, t_n. This is done by solving

$$G(z_n, y_n, t_n) = 0 \tag{1.25}$$

for z_n where $z = (y', w)$. Currently we solve $G(z, y, t) = 0$ via a damped Gauss–Newton iteration

$$z^{[m+1]} = z^{[m]} - \rho_m \, G_z^{\dagger}(z^{[m]}, y, t) \, G(z^{[m]}, y, t) \tag{1.26}$$

where $A^{\dagger}b$ is the minimum norm least squares solution of $Ax = b$. Under assumptions (A1)–(A4), it has been shown [23] that the iteration (1.26) converges to a limit z^*. This limit satisfies the *least squares equation* (1.27)

$$G_z^{\dagger}(z, y, t) \, G(z, y, t) = 0. \tag{1.27}$$

Note that (1.27) is not equivalent to $G = 0$ but has additional solutions since G_z is not full row rank. However, y' is uniquely determined due to the 1-fullness assumption (A3). Also y' depends only on (y, t). Thus y' defines a smooth and unique *completion*

$$y' = \overline{f}(y, t). \tag{1.28}$$

This vector field (1.28) will be called a *least squares completion.* However in the remainder of this thesis we will just refer to (1.28) as a completion.

The next theorem is proven in [23]. Define

$$\widehat{G}(z, y, t) = G_z^T(z, y, t) G(z, y, t).$$

Theorem 1.2.1 *Let*

$$G(z_0, y_0, t_0) = 0, \tag{1.29}$$

and let the following conditions be satisfied on an open neighborhood of (z_0, y_0, t_0)*:*

(i) *G_z has constant rank κ;*

(ii) *G_z is 1-full with respect to y'; and*

(iii) *$[G_z \mid G_y]$ has full row rank.*

Define $\mu = rank[G_z \mid G_y] - \kappa$. *Then the following conditions are satisfied:*

(a) $\widehat{G}_z$ is 1-full with respect to y' for solutions of $\widehat{G} = 0$;

(b) $\widehat{G} = 0$ determines locally a unique $\overline{f}$ such that $y' = \overline{f}(y,t)$ and (y'_0, y_0, t_0) lies on the graph of f;

(c) there exists $\Phi : \mathbb{R}^{s+1} \to \mathbb{R}^{\mu}$ such that solutions of $G = 0$ satisfy $\Phi(y,t) = 0$ and $\Phi_y(y_0, t_0)$ has rank μ; and

(d) both $\overline{f}$ and Φ are no more than one order of smoothness less in (y,t) than G is in (z,y,t).

Theorem 1.2.1 states, for a fixed t, $\{y \in \mathbb{R}^s : \Phi(y,t) = 0\}$ is a manifold of dimension $s - \mu$ and $\Phi^{-1}(\{0\})$ is a manifold of dimension $s - \mu + 1$, which is the solution manifold. Φ is theoretically determined by the least squares completion and a subset of the derivative array equations.

Prior research efforts [86] examined the use of a different inverse other than $G_z^{\dagger}$ in (1.26). In order to speed up the method the reuse of Jacobians was also investigated. It is shown in [38] that Jacobian reuse results in discontinuous completions. However, if the Gauss–Newton iteration is solved with sufficient accuracy the integration can proceed and leads to good numerical approximations to the solution of the DAE. A comparison of the EI method with RADAU5 is done in [54]. Unlike implicit Runge-Kutta methods there is no loss of order in the higher index variables with the EI method. The use of automatic differentiation to form G and G_z at each time step has also been investigated. Substantial savings in evaluation time and memory requirements are reported in [28, 36] for moderately sized DAEs. Automatic differentiation will likely provide a means to solve higher dimensional problems.

One of the problems with the EI method is the tendency to drift off the solution manifold since constraints are not all preserved. This led to the development of the next general method for unstructured higher index DAEs.

1.2.2 Implicit Coordinate Partitioning Method

The implicit coordinate partitioning method (ICP) was developed in order to preserve both explicit and implicit constraints that occur in higher index DAEs [35, 68]. A subset of the state variables y are used to set up a local set of coordinates for the solution manifold. Within this local set of coordinates, an explicit integration approach is used.

We take a partition of the state variables

$$y = (y_1, y_2)$$

so that

$$\widetilde{J} = [G_z \; G_{y_1}]$$

has full row rank. The variables y_2 are used to parameterize the solution manifold locally.

Define the variable $\widetilde{z}$ by

$$\widetilde{z} = (z, y_1).$$

Thus $\widetilde{z}$ is everything but y_2 and z is everything but y. Then $G_{\widetilde{z}}$ is not only 1-full with respect to y' but it is also full row rank by the choice of y_1. Thus

$$G_{\widetilde{z}}^T(\widetilde{z}, y_2, t)G(\widetilde{z}, y_2, t) = 0$$

is equivalent to $G(\widetilde{z}, y_2, t) = 0$. Unlike with the explicit approach, we may now use any method of finding $\widetilde{z}$ given (y_2, t) which minimizes

$$C(\widetilde{z}) = \frac{1}{2} G^T(\widetilde{z}, y_2, t) G(\widetilde{z}, y_2, t).$$

Again we use a damped Gauss–Newton iteration,

$$\widetilde{z}^{[m+1]} = \widetilde{z}^{[m]} - \rho_m G^{\dagger}_{\widetilde{z}}(\widetilde{z}^{[m]}, y_2, t) G(\widetilde{z}^{[m]}, y_2, t) = 0.$$

Suppose then that we have a point $(\widetilde{z}_0, y_{20}, t_0)$ where $G(\widetilde{z}_0, y_{20}, t_0) = 0$ and our assumptions (A1)-(A4) hold. Then there is a partitioning such that $\widetilde{z} = (y', [\xi, \eta], [y_1, y_2])$ and open neighborhoods $\widetilde{N}$ and $\overline{N}$ such that within these neighborhoods we get that the limit $(y_1'^*, y_2'^*, w^*, y_1^*)$ satisfies the fundamental equations

$$y_1'^* = f_1(y_2, t) \tag{1.30a}$$

$$y_2'^* = f_2(y_2, t) \tag{1.30b}$$

$$y_1^* = g(y_2, t) \tag{1.30c}$$

where f_1, f_2, h are defined by the limit of the integration [35].

One step of the integration of the DAE is as follows. Given $y_{n-1} = (y_{1,n-1}, y_{2,n-1})$ we apply an ODE integrator to (1.30a)-(1.30b) to get $\widehat{y}_n = (\widehat{y}_{1,n}, \widehat{y}_{2,n})$. A final function evaluation gives $y_n' = (y_{1,n}', y_{2,n}')$ and $\overline{y}_{1,n} = g(\widehat{y}_{2,n}, t_n)$. Then the value for y_n is taken to be $(\overline{y}_{1,n}, \widehat{y}_{2,n})$. Thus y_n lies on the solution manifold and satisfies all constraints. The implementation of the ICP method is discussed in detail in [68].

Specific computational issues such as Jacobian reuse and partitioning strategies are further addressed in [86]. The issue of when to terminate the Gauss–Newton iteration is also examined in [86]. If using a k^{th} order integrator on the completion, then the iteration should be terminated at $O(h^k)$ or $O(h^{k+1})$ accuracy when applying the EI and ICP methods respectively in order to obtain k^{th} order accurate numerical solutions.

Figure 1.3 summarizes the steps necessary to solve a given DAE by the EI or ICP methods.

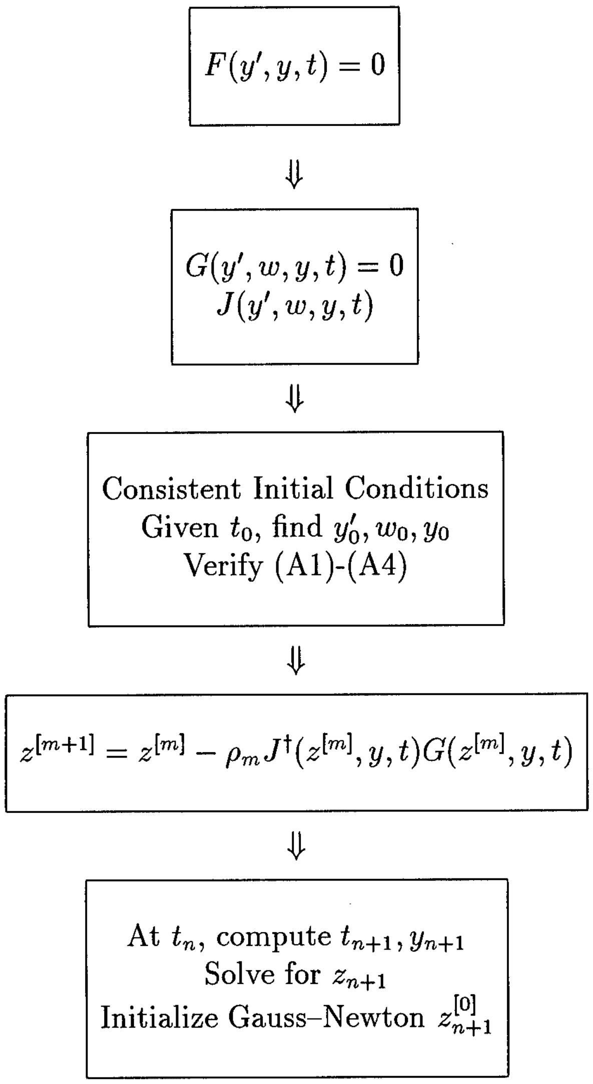
$F(y', y, t) = 0$
$G(y', w, y, t) = 0$
$J(y', w, y, t)$
Consistent Initial Conditions
Given t_0, find y'_0, w_0, y_0
Verify (A1)-(A4)
$z^{[m+1]} = z^{[m]} - \rho_m J^{\dagger}(z^{[m]}, y, t) G(z^{[m]}, y, t)$
At t_n, compute t_{n+1}, y_{n+1}
Solve for z_{n+1}
Initialize Gauss–Newton $z_{n+1}^{[0]}$

Figure 1.3: Summary of General DAE Integrators

1.2.3 Index One Integration

The final method that is related to the EI and ICP methods is being developed by Kunkel and Mehrmann [58, 60]. However, it is a computationally different approach. To understand their approach, suppose that we have a DAE $F(y', y, t) = 0$ and the system of equations

$$g(y, t) = 0 \tag{1.31}$$

describes the constraint manifold of $F(y', y, t) = 0$ and g_y has full row rank. Then locally there is a partition of $y = (y_1, y_2)$ and a subset

$$\widetilde{F}(y', y, t) = 0 \tag{1.32}$$

of the equations $F(y', y, t) = 0$ such that g_{y_2} and $\widetilde{F}_{y_1}$ are nonsingular. Then the DAE composed of (1.31) and (1.32) is index one and has the same solutions as $F(y', y, t) = 0$.

The method proceeds as follows. Let D_h be the k-step BDF operator and suppose t is the current time. A matrix Z is computed from the Jacobian of G. Then the solution y of

$$\begin{aligned} ZF(D_h y, y, t + h) &= 0 \qquad (1.33a) \\ G(z, y, t + h) &= 0 \qquad (1.33b) \end{aligned}$$

is taken as the estimate for $y(t + h)$. Note that the equations (1.33b) implicitly determine a relationship (1.31). Z is such that

$$\begin{aligned} ZF(y', y, t) &= 0 \qquad (1.34a) \\ g(y, t) &= 0 \qquad (1.34b) \end{aligned}$$

is index one. Here (1.34a) is the same as (1.32). Therefore solving (1.33) is equivalent to solving (1.34) by the BDF method. This method does not require constant rank of the Jacobian of G in a neighborhood of a solution to the DAE but only on a submanifold [58].

1.2.4 Constrained Least Squares

There is another numerical approach based on the derivative array. However, it is not a general method since it requires some numerical structure to the DAE.

The Constrained Least Squares (CLS) method by Barrlund [5, 6] applies BDF to the derivatives appearing in the derivative array equations and minimizes this discretization subject to a subset of the derivative array equations. At each step of the integration the DAE and some of its derivatives are used as constraints to a least squares problem that corresponds to a multistep, multiderivative formula for the solution. It is required that the user be able to identity all necessary constraints. An analysis of the stability properties of the CLS method for linear DAEs is given in [7].

1.3 Outline of Thesis

The purpose of this thesis to examine two initialization problems that arise when numerically solving DAEs by general methods. Resolution of these problems is needed before production quality codes can be developed. The first problem is the computation of the consistent initial conditions required to begin the integration of DAEs. In Chapter 2, various line search strategies are examined and compared for solving the nonlinear system $G(y', w, y, t) = 0$. 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. Results of this effort have been published in [31].

The second problem that is examined is the initialization of the iterative solvers used during the integration of a DAE. In Chapter 3, we discuss how linear multistep methods that are used in the integration of the completion are implemented to provide initial iterates used in the Gauss–Newton iteration.

We examine the case where previous converged values of z are used to fit a polynomial that is used for prediction in the Gauss–Newton solve. It is shown in the linear time varying case that the z components are actually approximating the solution to another DAE which we label the auxiliary DAE or ADAE. The results of our analysis indicate that constant and linear prediction appear feasible to implement.

Chapter 4 summarizes the main conclusions of this research effort.

Chapter 5 discusses areas of future investigation.

1.4 Contributions of Thesis

The research in this thesis has appeared in the following publications.

- S. L. CAMPBELL, C. T. KELLEY AND K. D. YEOMANS, *Consistent Initial Conditions for Unstructured Higher Index DAEs: A Computational Study,* in Proc. Computational Engineering in Systems Applications, Lille, France, 1996, pp. 416-421.

- S. L. CAMPBELL, R. HOLLENBECK, K. YEOMANS AND Y. ZHONG, *Mixed Symbolic–Numerical Computations with General DAEs I: System Properties.* Preprint, 1997.

- S. L. CAMPBELL AND K. D. YEOMANS, *Behavior of the Nonunique Terms in General DAE Integrators.* To appear in Applied Numerical Mathematics, 1997.

Chapter 2

Consistent Initialization of General DAE Integrators

Obtaining a consistent set of initial conditions is perhaps the most difficult part in determining a numerical solution of a DAE by the general DAE methods described in Chapter 1. Initialization is also important in dealing with discontinuities of the solution that frequently occur in applications [65]. There has been work on the initialization of special classes of DAEs [17, 20, 57, 63, 69]. However in this chapter we examine the problem of computing consistent initial conditions for the derivative array based general integrators (EI and ICP methods).

In initialization we will not have good predictors for the starting values particularly for the higher derivative variables w. Therefore we will need to consider more global iterative schemes. The globalization of iterative schemes has been extensively studied and several approaches considered [39, 55]. In this chapter we examine various line search algorithms that are applied in the context of our general DAE integrators. Unlike Chapter 3 which includes theoretical results, this chapter consists primarily of computational studies.

2.1 Algorithmic Issues

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 the 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 Jacobians, up to round off error, are available at substantially less computational cost than that of one matrix factorization [29, 36].

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 many have greatly changed in these components. Thus we want to allow for the user to specify a subset of the variables which are considered known. 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 y but also y'. 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 [27]. Let $z = (y', w, y)$, to simplify our notation we let u denote whichever variables in z are allowed to vary and we write the derivative array as $G(u)$. The limit of the iteration will be denoted by 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 the $\dim(z_2)$ 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 $G(u) = 0$. 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 the $\dim(z_2)$ is greater than the dimension of the solution manifold of the DAE. The Jacobian is assumed to have constant rank in a neighborhood of the limit point u^*. 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 the nonlinear residual $\|G(u^*)\|$ 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 in the final stage so that some of the z_1 components remain 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 [33].

We assume that 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.

2.2 Line Search Algorithms

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

$$u_{n+1} = u_n + \rho_n \, \Delta u_n \tag{2.1}$$

where $0 < \rho_n \leq 1$. We will terminate the iteration if Δu and the residual $\|G(u)\|$ are less than tolerances E_X and E_R respectively. 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 $\rho_n = 1$:

$$u_{n+1} = u_n - J^{\dagger}(u_n)G(u_n). \tag{2.2}$$

The theory for Gauss–Newton (like ordinary Newton's method) requires the starting value to be near the solution [9, 10, 52]. 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, sequential linear programming methods (SLP) are currently being examined for chemical engineering problems [48]. However, there are several reasons for wanting to utilize a method based on a variant of the Gauss–Newton method. 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.

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

$$u_{n+1} = u_n - \rho_n J^{\dagger}(u_n)G(u_n). \tag{2.3}$$

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 n^{th} iteration. There are a variety of line search methods. One is to initially take $\rho_n = 1$ and then halve ρ_n until

$$T_n(u_{n+1}) \leq (1 - \alpha\rho_n)\, T_n(u_n) \tag{2.4}$$

for a fixed $\alpha = 10^{-4}$ [39, 55, 56].

The norm squared of the residual is one natural choice of T_n

$$T_n^{[1]}(u) = G^T(u)G(u) = \|G(u)\|^2. \qquad \mathbf{(T1)} \tag{2.5}$$

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 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 $J^{\dagger}(u^*)G(u^*) = 0$. For ill conditioned problems the test function

$$T_n^{[2]}(u) = \|J^{\dagger}(u_n)G(u)\|^2. \qquad \mathbf{(T2)} \tag{2.6}$$

has been suggested [40, 41]. The gradient of $T_n^{[2]}(u)$ is

$$\nabla T_n^{[2]}(u) = 2G^T(u)(J^{\dagger}(u_n))^T J^{\dagger}(u_n)J(u). \tag{2.7}$$

At the current point u_n we have that $\nabla^T T_n^{[2]}(u_n)$ 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 $J = U\Sigma V^T$ is the singular value decomposition (SVD) of J, then define Σ_δ to be Σ with all singular values below δ set to zero. Define $J_\delta = U\Sigma_\delta V^T$ and $J_\delta^\dagger = (J_\delta)^\dagger$. This leads to the iteration

$$u_{n+1} = u_n - \rho_n J_\delta^\dagger(u_n)G(u_n). \tag{2.8}$$

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. However, our value of δ is small so that only singular values which are theoretically zero but numerically nonzero are ignored. One could also use larger values of δ as a way to try and counteract ill conditioning of J.

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

$$u_{n+1} = u_n - \rho_n J^T(u_n)G(u_n). \tag{2.9}$$

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 the *Levenberg–Marquardt* iteration [39, 51]

$$u_{n+1} = u_n - \rho_n(J^T(u_n)J(u_n) + \varepsilon I)^{-1}J^T(u_n)G(u_n) \tag{2.10}$$

with $\varepsilon \to 0$ 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 likes $J^\dagger(u_n)G(u_n)$ in the direction of large singular values of J and like $\frac{1}{\varepsilon}J^T$ in the direction of small singular values of J. Because we have the full row rank assumption in scenarios **S1** and **S2**, the variant

$$u_{n+1} = u_n - \rho_n J^T(u_n)(J(u_n)J^T(u_n) + \varepsilon I)^{-1}G(u_n) \tag{2.11}$$

is more appropriate for our intended application. In the direction of large singular values of J, (2.10) acts like (2.9).

Proposition 2.2.1 *Given the preceding definitions and assumptions:*

(a) $J^\dagger(u_n)G(u_n) \neq 0$ *if and only if* $J^T(u_n)G(u_n) \neq 0$.

(b) $J_\delta^\dagger(u_n)G(u_n) \neq 0$ *implies that* $J^\dagger(u_n)G(u_n) \neq 0$.

(c) *If any of* $-J^\dagger G$, $-J^T(JJ^T + \varepsilon I)^{-1}G$ *and* $-J^T G$ *are nonzero at a point* u_n, *then they all are descent directions for* **(T1)** *and* **(T2)**.

(d) *If* $-J_\delta^\dagger G$ *is nonzero at a point* u_n, *it is a descent direction for* **(T1)** *and* **(T2)**.

Proof: Part (a) follows from the fact that $\mathcal{N}(J^\dagger(u_n)) = \mathcal{N}(J^T(u_n))$ where $\mathcal{N}$ represents the null space of a matrix.

Part (b) is obvious since if $G(u_n) \notin \mathcal{N}(J_\delta^\dagger(u_n))$, then it is immediate that $G(u_n) \notin \mathcal{N}(J^\dagger(u_n))$ since $\mathcal{N}(J_\delta^\dagger(u_n)) \subset \mathcal{N}(J^\dagger(u_n))$.

To demonstrate (c), we need to show that the inner product of the gradient of either **(T1)** or **(T2)** with any of the given steps is negative. We note that

$$\nabla^T T_n^{[1]}(u_n) = 2J^T(u_n)G(u_n) \tag{2.12}$$

$$\nabla^T T_n^{[2]}(u_n) = 2J^T(u_n)(J^\dagger(u_n))^T J^\dagger(u_n)G(u_n). \tag{2.13}$$

In order to simplify the notation, it will be understood that the function G and its Jacobian J are evaluated at the point u_n.

Assume that $-J^\dagger G$ is nonzero, then using properties of $J^\dagger$ [32] we have

$$\begin{aligned}\left(\nabla T_n^{[1]}\right)\left(-J^\dagger G\right) &= -2G^T J J^\dagger G \\ &= -2G^T(JJ^\dagger J)J^\dagger G \\ &= -2G^T(JJ^\dagger)^T(JJ^\dagger)G \\ &= -2\|JJ^\dagger G\|^2 < 0.\end{aligned}$$

Also we have

$$\left(\nabla T_n^{[2]}\right)\left(-J^\dagger G\right) = -2G^T J^{\dagger^T} J^\dagger J J^\dagger G = -2G^T J^{\dagger^T} J^\dagger G = -2\|J^\dagger G\|^2 < 0.$$

Therefore $-J^\dagger G$ is a descent direction using either **(T1)** or **(T2)**.

Assume that $-J^T(JJ^T+\varepsilon I)^{-1}G$ is nonzero, then for **(T1)**

$$\begin{aligned}\left(\nabla T_n^{[1]}\right)\left(-J^T(JJ^T+\varepsilon I)^{-1}G\right) &= -2G^T JJ^T(JJ^T+\varepsilon I)^{-1}G \\ &= -2G^T U\tilde{\Sigma}U^T G^T\end{aligned}$$

where $\tilde{\Sigma}$ is a diagonal matrix whose elements are given by

$$\frac{\sigma_i^2}{\sigma_i^2+\varepsilon} \text{ if } \sigma_i \neq 0, \text{ and 0 otherwise,}$$

and σ_i are the singular values of J computed from its SVD, $J = U\Sigma V^T$. Therefore

$$\left(\nabla T_n^{[1]}\right)\left(-J^T(JJ^T+\varepsilon I)^{-1}G\right) = -2\|U\hat{\Sigma}U^T G\|^2 < 0$$

where $U\hat{\Sigma}U^T$ is the positive semi-definite square root of $JJ^T(JJ^T+\varepsilon I)^{-1}$ [62].

For the test function **(T2)** we have

$$\begin{aligned}\left(\nabla T_n^{[2]}\right)\left(-J^T(JJ^T+\varepsilon I)^{-1}G\right) &= -2G^T J^{\dagger^T} J^\dagger J J^T(JJ^T+\varepsilon I)^{-1}G \\ &= -2G^T JJ^\dagger(JJ^T+\varepsilon I)^{-1}G \\ &= -2G^T U\bar{\Sigma}U^T G \\ &= -2\|U\check{\Sigma}U^T G\|^2 < 0\end{aligned}$$

where $\bar{\Sigma}$ is a diagonal matrix with diagonal entries

$$\frac{1}{\sigma_i^2+\varepsilon} \text{ if } \sigma_i \neq 0, \text{ and 0 otherwise,}$$

and $U\check{\Sigma}U^T$ is the positive semi-definite square root of $JJ^\dagger(JJ^T+\varepsilon I)^{-1}$. Therefore $-J^T(JJ^T+\varepsilon I)^{-1}G$ is a descent direction using either **(T1)** or **(T2)**.

If $-J^TG \neq 0$ we have

$$\left(\nabla T_n^{[1]}\right)\left(-J^TG\right) = -2G^TJJ^TG = -2\|J^TG\|^2 < 0$$

and

$$\begin{aligned}
\left(\nabla T_n^{[2]}\right)\left(-J^TG\right) &= -2G^TJ^{\dagger^T}J^{\dagger}JJ^TG \\
&= -2G^TJ^{\dagger^T}(J^{\dagger}J)^TJ^TG \\
&= -2G^TJ^{\dagger^T}J^TJ^{\dagger^T}J^TG \\
&= -2G^T(JJ^{\dagger})^T(JJ^{\dagger})^TG \\
&= -2G^T(JJ^{\dagger})^TJJ^{\dagger}G \\
&= -2\|JJ^{\dagger}G\|^2 < 0.
\end{aligned}$$

Again we conclude that $-J^TG$ is a descent direction using either **(T1)** or **(T2)**.

For part (d) assume $-J_\delta^{\dagger}G \neq 0$, then

$$\begin{aligned}
\left(\nabla T_n^{[1]}\right)\left(-J_\delta^{\dagger}G\right) &= -2G^TJJ_\delta^{\dagger}G \\
&= -2G^TJ(J_\delta^{\dagger}JJ_\delta^{\dagger})G \\
&= -2G^T(JJ_\delta^{\dagger})^TJJ_\delta^{\dagger}G \\
&= -2\|JJ_\delta^{\dagger}G\|^2 < 0.
\end{aligned}$$

A similar computation as in part (c) for the LM step $-J^T(JJ^T+\varepsilon I)^{-1}G$ holds for the test function $T_n^{[2]}$. □

2.3 Numerical Examples

In the examples that follow, the Jacobians were computed analytically in MAPLE. The Moore–Penrose inverse was computed by an SVD of J. Additionally we set

$\alpha = 10^{-4}$ and $\delta = 10^{-8}$ unless noted otherwise. δ is designed to ignore numerically zero singular values rather than to regularize an iteration with small, but nonzero, singular values. Also, $E_X = E_R = 10^{-10}$. 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.

We first started our initialization tests by perturbing a known solution $u(t_0)$. We generated several random directions w scaled in a similar manner to $u(t_0)$ by

$$(w)_i = (-1)^{p_i}\, r_i\, (u(t_0))_i$$

where $p_i = 1, 2$ and $.5 < r_i < 1$. We then took starting values at different distances in those directions. The initial point was $u_0 = u(t_0) + 10^{\gamma-1}w$, $\gamma = 0, 1, \ldots, 4$. Our intention is to examine the behavior of the iteration and the effect of increasingly poor initial guesses. We experimented with several variants of the Levenberg–Marquardt method. Setting $\varepsilon = 0$ when the residual $\|G\|$ became sufficiently small appeared to be the best strategy. In what follows we made the simple choice of $\varepsilon = 10^{-4}$ if $\|G\| > 10^{-4}$, otherwise $\varepsilon = 0$. We also considered the test function $T_n(u) = \|J^T(u)(J(u)J^T(u) + \varepsilon I)^{-1}J^T(u)G(u)\|$ as well as (T2). The test functions performed the same.

We did a large number of experimental runs with Scenario 1 (S1). Plain Gauss–Newton performed very poorly when we had poor initial guesses. It is clearly 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 $(JJ^T + \varepsilon I)$, solving $(JJ^T + \varepsilon I)y = G$, and then letting $\Delta u = -J^T y$ frequently converged much more slowly (or not at all), then computing an SVD of $J = U\Sigma V^T$ and setting $\Delta u =$

$-V\Sigma^T(\Sigma\Sigma^T+\varepsilon I)^{-1}U^TG$. One explanation is that there is too much loss of accuracy in the "normal equations" version of Levenberg–Marquardt for the problems considered here.

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

In the remaining discussion we will adopt the following notation when discussing various methods and damping strategies:

PGN - Plain Gauss–Newton

DGNR - Damped Gauss–Newton, test function (**T1**)

DGNB - Damped Gauss–Newton, test function (**T2**)

PLM - Plain Levenberg–Marquardt

DLMR - Damped Levenberg–Marquardt, test function (**T1**)

DLMB - Damped Levenberg–Marquardt, test function (**T2**).

GN is any of the tested Gauss–Newton methods, and LM is any of the Levenberg–Marquardt methods.

2.3.1 Chemical Reactor Problem

The first example is taken from [33] and is an index three DAE in the four variables C, R, T and T_c

$$C' + C + R = 4 + t + t^3 \tag{2.14a}$$

$$T' + 2T + R + T_c = 1 + e^{-t} \tag{2.14b}$$

$$T^{-1} + \ln(R/C) = 0 \tag{2.14c}$$

$$C = \cosh(t-1). \tag{2.14d}$$

System (2.14) models the situation where C is a specified product concentration and we want to determine the temperature T_c (open loop control) that will produce this C. Each of the equations in (2.14) is differentiated three times so that G consists of 16 equations. Many of the equations are linear in the unknown variables. The solution to (2.14) is given by

$$C = \cosh(t-1) \tag{2.15a}$$

$$R = 4 + t + t^3 - C - C' \tag{2.15b}$$

$$T = -1/\ln(R/C) \tag{2.15c}$$

$$T_c = 1 + e^{(-t)} - T' - 2T - R. \tag{2.15d}$$

MAPLE was used to differentiate the solution and evaluate all the equations at time $t = 0$ to obtain $u(0)$. For $\gamma = 0, 1, 2$ the GN methods converged typically in 3, 4 and 5 iterations. The LM methods also converged but in 2 to 3 times as many iterations. However for $\gamma = 3$, PGN usually failed. DGNR and DGNB converged more often. There did not appear to be any difference with the choice of test function in determining the damping parameter. The same results occurred for the LM methods. However the damped LM methods required substantially more iterations when far from a solution. Figure 2.1 is a plot of the iteration histories for the damped GN and LM methods for an initial condition where $\gamma = 3$. The undamped GN methods did not converge in this test.

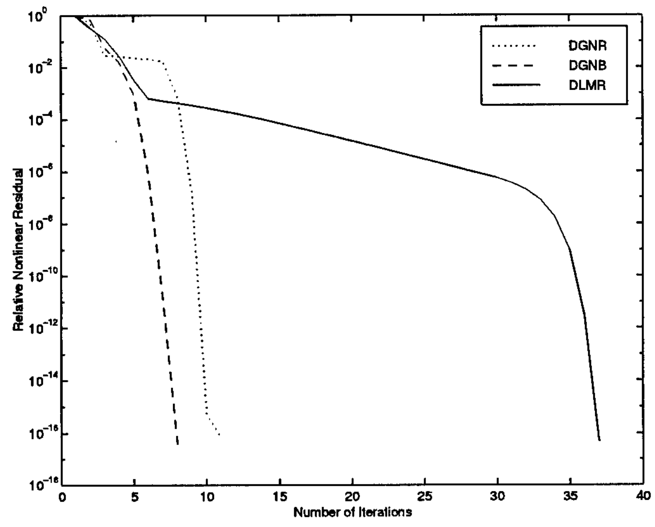

Figure 2.1: Comparison of Damping Strategies for Chemical Problem

Table 2.1 shows the results of a test which converged after 86 iterations for DLMB. The GN methods did not converge. We observe that the converged values of some of the higher order derivatives are not close to the initial values which is expected since these components are not uniquely determined. The first column is the initial iterate u_0 given when $\gamma = 3$. The second column is the limit of the iteration u^*. The third column is the unperturbed solution $u(0)$. The fourth column is the absolute error componentwise between u^* and $u(0)$. The nonlinear residual was $\|G(u^*)\| = .151518D - 14$.

2.3.2 Shuttle Trajectory Problem

The next example is the shuttle trajectory (TPPC) problem described in [13]. This is also an index three DAE in seven variables. Additionally, it is a Hessenberg system which allows us to reduce the number of differentiations to some of the equations. The equations are highly nonlinear. We exploit the Hessenberg structure and differentiate equations (2.16a)-(2.16d) twice, equations (2.16e)-(2.16f) once and the

	u_0	u^*	$u(0)$	|Error|
C'	-.211161D+01	-.117520D+01	-.117520D+01	.444089D-15
R'	.952947D+00	.632121D+00	.632121D+00	.111022D-15
T'	.458495D+00	.127679D+01	.127679D+01	.444089D-15
T'_C	-.118081D+00	-.642026D+00	-.642026D+00	.352052D-12
$C^{(2)}$	.307968D+01	.154308D+01	.154308D+01	.222045D-15
$R^{(2)}$	-.734643D+00	-.367879D+00	-.367879D+00	.555112D-16
$T^{(2)}$	-.614925D+01	-.354368D+01	-.354368D+01	.133227D-14
$T_C^{(2)}$	-.136720D+02	.458258D+01	-.696186D+01	.115444D+02
$C^{(3)}$	-.300881D+00	-.117520D+01	-.117520D+01	.444089D-15
$R^{(3)}$	.123989D+01	-.250947D+02	.563212D+01	.307268D+02
$T^{(3)}$	.290901D+01	.387265D+01	.154171D+02	.115444D+02
$T_C^{(3)}$	.374820D+01	.147837D+02	.520078D+02	.372241D+02
$C^{(4)}$	.436052D+00	.322699D+02	.154308D+01	.307268D+02
$R^{(4)}$	-.567714D+00	-.567714D+00	-.367879D+00	.199834D+00
$T^{(4)}$	-.946989D+01	.156564D+01	-.894741D+02	.910398D+02
$T_C^{(4)}$	-.873182D+03	-.873182D+03	-.466743D+03	.406439D+03
C	.263811D+01	.154308D+01	.154308D+01	.222045D-15
R	.192360D+00	.363212D+01	.363212D+01	.444089D-15
T	-.188381D+01	-.116818D+01	-.116818D+01	.222045D-15
T_C	-.947791D+00	-.572563D+00	-.572563D+00	.111022D-15

Table 2.1: Results of Initialization of Chemical Problem using DLMB Method

constraint (2.16g) three times. The initial conditions are taken from [12, 34] where $t_0 = 332.867734542$. G has 20 equations in 35 unknowns. 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.

$$H' = V_R \sin(\gamma) \tag{2.16a}$$

$$\xi' = \frac{V_R \cos(\gamma)\sin(A)}{r\cos(\lambda)} \tag{2.16b}$$

$$\lambda' = \frac{V_R}{r}\cos(\gamma)\cos(A) \tag{2.16c}$$

$$\begin{aligned} V_R' &= \frac{-D}{m} - g\sin(\gamma) - \\ &\quad \Omega_E^2 r\cos(\lambda)(\sin(\lambda)\cos(A)\cos(\gamma) - \cos(\lambda)\sin(\gamma)) && (2.16\text{d}) \\ \gamma' &= \frac{L\cos(\beta)}{mV_R} + \frac{\cos(\gamma)}{V_R}\left(\frac{V_R^2}{r} - g\right) + 2\Omega_E\cos(\lambda)\sin(A) \\ &\quad + \frac{\Omega_E^2 r\cos(\lambda)}{V_R}(\sin(\lambda)\cos(A)\cos(\gamma) + \cos(\lambda)\ sin(\gamma)) && (2.16\text{e}) \\ A' &= \frac{L\sin(\beta)}{mV_R\cos(\gamma)} + \frac{V_R}{r}\cos(\gamma)\sin(A)\tan(\lambda) \\ &\quad -2\Omega_E(\cos(\lambda)\cos(A)\tan(\gamma) - \sin(\lambda)) \\ &\quad + \frac{\Omega_E^2 r\cos(\lambda)\sin(\lambda)\sin(A)}{V_R\cos(\gamma)} && (2.16\text{f}) \end{aligned}$$

along with a path constraint

$$\frac{D}{m} - \left[C_0 + C_1(V_R - V_0) + C_2(V_R - V_0)^2 + C_3(V_R - V_0)^3\right] = 0. \tag{2.16g}$$

Some of the variables are given in terms of the state variables [33] as:

$$\begin{aligned} \rho(H) &= .002378\exp(-H/23800) \\ C_L(\alpha) &= .84 - .48(38 - \alpha C_{rd})/26 \\ D &= .5\rho C_D S V_R^2 \\ C_D(\alpha) &= .78 - .58(38 - \alpha C_{rd})/26 \\ r &= H + a_e \\ g &= \mu/r^2 \\ L &= .5\rho C_L S V_R^2. \end{aligned}$$

The following parameters and constants are also given:

$$\mu = 0.1407653916 \times 10^{17}\ \text{ft}^3/\text{s}^2$$

$$\begin{aligned}
\Omega_E &= .72921159 \times 10^{-4} \text{ rads/sec} \\
C_0 &= D(t_0)/m = 3.974960446019 \\
C_1 &= -0.01448947694635 \\
C_2 &= -0.2156171551995 \times 10^{-4} \\
C_3 &= -0.1089609507291 \times 10^{-7} \\
a_e &= 20902900\text{ft} \\
m &= 5964.4965 \text{ slugs} \\
S &= 2690 \text{ ft}^2 \\
C_{rd} &= 360/2\pi \\
\alpha &= 40^\circ.
\end{aligned}$$

The shuttle problem is not 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 convergence occurred, it usually required 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 usually failed while DGN* worked some times. However, the limit u^* frequently had the component $\beta < 0$.

To illustrate **S2** we fixed β, β' and V_R in this example. However, the Jacobian now has fewer columns. While the Jacobian may still be full row rank it may be less well conditioned. An examination of the singular values of J show that this in fact occurs during the iteration with singular values ranging from 10^{-9} to 10^5. Table 2.2 shows the result of a test where the DGNB worked requiring 27 iterations. Note that the components β, β' and V_R did not move from their initial values. For this example and initial iterate, the DGNR did not converge and flagged that the Jacobian experienced a rank drop. The LM methods all stagnated at a local minimum. Again, the DGNB method as discussed in [40] appeared the most attractive in this scenario.

	u_0	u^*	$u(t_0)$	$\lvert$Error$\rvert$
H'	-.292419D+03	-.318278D+03	-.318295D+03	.169781D-01
ξ'	.129469D-02	.121420D-02	.120518D-02	.902030D-05
λ'	.496299D-03	-.517715D-03	.525288D-03	.104300D-02
V_R'	-.337139D+01	-.358773D+01	-.358792D+01	.191383D-03
γ'	.945038D-04	.101053D-03	.101068D-03	.151924D-07
A'	.908818D-03	.845165D-03	.833641D-03	.115243D-04
β'	-.881555D-02	-.881555D-02	-.881555D-02	.000000D+00
H	.243808D+06	.264039D+06	.264039D+06	.349246D-09
ξ	.287535D+01	.287535D+01	.310177D+01	.226412D+00
λ	.517299D+00	-.370668D+01	.559232D+00	.426591D+01
V_R	.243171D+05	.243171D+05	.243171D+05	.000000D+00
γ	-.120586D-01	-.130890D-01	-.130897D-01	.698258D-06
A	.119394D+01	-.203833D+01	.109586D+01	.313419D+01
β	.717332D+00	.717332D+00	.717332D+00	.000000D+00

Table 2.2: Results of Initialization of Shuttle Problem using DGNB Method

2.3.3 Torus Problem

The following is an index three DAE from [35] in the seven state variables $\{x_1, x_2, x_3, u_1, u_2, u_3, \lambda\}$

$$x_1' = u_1 \tag{2.17a}$$

$$x_2' = u_2 \tag{2.17b}$$

$$x_3' = u_3 \tag{2.17c}$$

$$u_1' = u_3\cos(t) - x_3\sin(t) - u_2 + 2x_1\left[1 - r(x_1^2 + x_2^2)^{-1/2}\right]\lambda \tag{2.17d}$$

$$u_2' = u_3\sin(t) + x_3\cos(t) + u_1 + 2x_2\left[1 - r(x_1^2 + x_2^2)^{-1/2}\right]\lambda \tag{2.17e}$$

$$u_3' = -x_3 + 2x_3\lambda \tag{2.17f}$$

$$0 = x_1^2 + x_2^2 + x_3^2 - 2r(x_1^2 + x_2^2)^{1/2} + r^2 - \rho^2. \tag{2.17g}$$

The solution is given by

$$x_1 = [\rho\cos(2\pi - t) + r]\cos t \tag{2.18a}$$

$$x_2 = [\rho\cos(2\pi - t) + r]\sin t \tag{2.18b}$$

$$x_3 = \rho\sin(2\pi - t) \tag{2.18c}$$

which lies on a torus. The solution manifold is four dimensional. In the tests we set $\rho = 5, r = 10$. All equations are differentiated three times so that G has 28 equations. Additionally a nonlinear transformation is applied to the problem so that it is fully implicit [35, 68]. Initial conditions were taken from [68].

For this example, 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 usually failed at $\gamma = 4$ while DGNB converged more frequently for this value of γ. The damped LM methods often converged when all the GN methods failed. Figure 2.2 displays the results for convergence of the LM methods when $\gamma = 4$. However none of the GN methods converged. The conditioning of the problem was extremely poor as the singular values ranged from 10^{-5} to 10^{9} during the iteration.

2.4 Continuous Gauss–Newton Analogues

In has been noted [11, 43, 70] that solving the square nonlinear system $G(u) = 0$ by Newton's method is equivalent to integrating

$$u' = -J^{-1}(u)G(u) \tag{2.19}$$

$$u(t_0) = u_0 \tag{2.20}$$

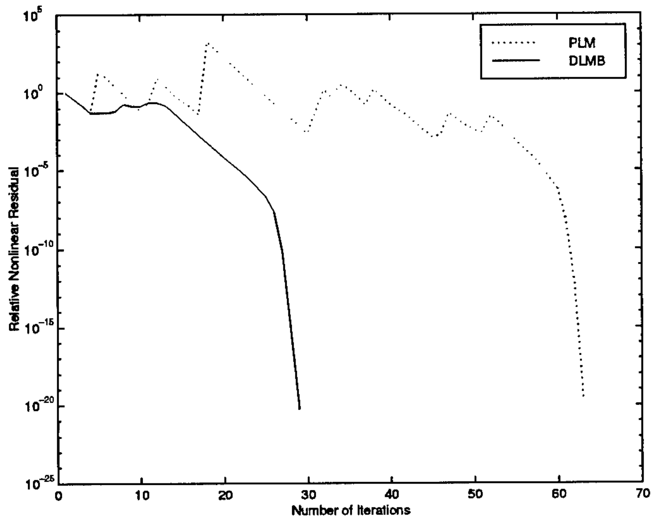

Figure 2.2: Iteration History of LM Methods for Torus Problem

with Euler's method and a suitable choice of stepsize. Tanabe [84] proved that in the underdetermined case, trajectories to the differential equation

$$u' = -J^{\dagger}(u)G(u) \tag{2.21}$$

$$u(t_0) = u_0, \tag{2.22}$$

where $J(u_0)$ is full row rank, will either approach a stationary point, diverge, or stagnate at a point where a drop in the rank of J occurs. A stationary point u^* will be a solution to $G(u) = 0$ that we seek. In [85], Tanabe extended his analysis to a continuous analogue of the Levenberg–Marquardt method

$$u' = -J^T(u)\left(J(u)J^T(u) + \varepsilon I\right)G(u). \tag{2.23}$$

It is noted that the Levenberg–Marquardt method (2.23) seems to have a better chance to converge to a solution u^* because it is able to resolve rank drops that might occur.

We can also pose the problem in terms of unconstrained optimization by minimizing the function

$$f(u) = \frac{1}{2}G^T(u)G(u). \tag{2.24}$$

Convergence results for ODE methods applied to (2.24) can be found in [1, 2, 3, 80, 81, 82]. Additionally some numerical tests comparing ODE methods with sequential quadratic programming algorithms (SQP) is done in [14, 15].

We have not implemented integration methods to control the damping parameter. However, results cited in these articles indicate that further study of continuous methods for initialization of our general DAE integrators might have merit.

We will use this relationship between the Gauss–Newton and continuous Gauss–Newton methods in Section 3.4 of Chapter 3.

2.5 Conclusions

Initialization by solving the derivative array equations appears to be a practical approach for moderately sized problems. As expected, some damping strategy is needed. The damped Gauss–Newton appears most attractive. However the fixing of certain known values, while advantageous, can have the affect of increasing the condition number of the Jacobians and increasing the likelihood of passing near singularities. The Levenberg–Marquardt appears to handle near singularities better but requires considerably more iterations. The stopping criteria must enforce both small steps and small residuals.

The issue of scaling of the Jacobians has not been examined and remains a topic of further research. Considerable more testing is needed with more examples, especially larger dimensional ones, and at additional u^* values.

Chapter 3

Prediction Strategies in General DAE Integrators

The general DAE integrators (EI, ICP and IOI methods) require the solution of the nonlinear system of equations

$$G(y', w, y, t) = 0 \tag{3.1}$$

at each time step. For fully implicit problems this enlarged nonlinear system will have components that are not uniquely determined. This poses questions about the effects of predictors and a possible instability in the growth of these terms during a numerical integration. In this chapter it is shown that the nonunique components are actually the numerical solution of an auxiliary DAE which depends not only on the original DAE but also the predictor being used in the Gauss–Newton iteration. The behavior of these nonunique components could interfere with the convergence of the nonlinear equation solver. This is radically different from the case for ODEs and DAEs with special structure where the predictor used for iterative solvers does not directly affect the limit of the iteration. Finally, the"true solution" that is being sought at each time step is a vector of Taylor coefficients, except that only some of them are found correctly. In [23, 35, 36, 37] in order to guarantee at least a first order prediction to start the Newton iteration it was necessary to assume that the integrator being used had order higher than the index of the DAE. For index one or two systems this is not a difficulty. However, this requirement becomes a severe restriction when considering DAEs whose index is greater than three. The results on predictors will be applied to the question of developing low order integrators for higher index DAEs.

We begin this chapter by reviewing linear multistep methods. For the EI and ICP methods, linear multistep methods have been applied to the ODE which is implicitly defined by the derivative array equations. The justification for these methods is based on their capability of achieving reasonable accuracy using very few function evaluations per forward step. The function evaluations are obtained by solving the derivative array equations.

Next we illustrate how an Adams-Bashforth-Moulton (ABM) method (predictor-corrector pair) is implemented as a one step method via the Nordsieck transformation. This implementation is used to provide an approximation to the higher order derivatives at each step of the integration. ABM methods are used in the current versions of the EI and ICP methods under development.

We briefly review polynomial interpolation and its role in providing predictors for the nonunique components in the system $G = 0$. The linear time varying case is analyzed in detail with examples given to illustrate the results derived. We briefly discuss the nonlinear case and the reuse of Jacobians. Finally we both establish a basis for the design of low order integrators for high index DAEs and develop guidelines for the use of predictors in integrating general high index DAEs.

3.1 Linear Multistep Methods

We begin our studies with a k-step method applied to the scalar ODE

$$y' = f(y,t). \tag{3.2}$$

We assume that the step size h is fixed. Let y_{n-i} and hy'_{n-i} denote the numerical approximations for $y(t_{n-i})$ and $hy'(t_{n-i})$ respectively, for $i = 0, \dots, k-1$. Arrange these values in the vector

$$Y_n = \begin{bmatrix} y_n & y_{n-1} & \cdots & y_{n-k+1} & hy'_n & hy'_{n-1} & \cdots & hy'_{n-k+1} \end{bmatrix}^T. \tag{3.3}$$

Multistep methods find a numerical approximation for Y_{n+1} from Y_n. The components of Y_n are typically generated by a Runge-Kutta method in order to begin the iteration. For notational convenience, we assume that the index of a vector or a matrix starts from 0. Define the matrix B by

$$B = \left[\begin{array}{cccc|cccc} \alpha_1 & \alpha_2 & \cdots & \alpha_k & \beta_1 & \beta_2 & \cdots & \beta_k \\ 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ \hline \gamma_1 & \gamma_2 & \cdots & \gamma_k & \delta_1 & \delta_2 & \cdots & \delta_k \\ & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \end{array}\right]_{2k\times 2k} \tag{3.4}$$

where the omitted entries are zeros. Let

$$Y_{n,(0)} \;=\; BY_{n-1} \tag{3.5}$$

$$= \begin{bmatrix} y_{n,(0)} \\ y_{n-1} \\ \vdots \\ y_{n-k+1} \\ hy'_{n,(0)} \\ hy'_{n-1} \\ \vdots \\ hy'_{n-k+1} \end{bmatrix} \tag{3.6}$$

so that the parameters $\alpha_i, \beta_i, \gamma_i, \delta_i$ are constants such that

$$y_{n,(0)} = \sum_{i=1}^{k} (\alpha_i \, y_{n-i} + h \, \beta_i \, y'_{n-i}) \tag{3.7}$$

is an approximation to $y(t_n)$, and

$$hy'_{n,(0)} = \sum_{i=1}^{k} (\gamma_i \, y_{n-i} + h \, \delta_i \, y'_{n-i}) \tag{3.8}$$

is an approximation to $hy'(t_n)$. The equation (3.7) represents an explicit multistep method that will be referred to as the *predictor* (P).

Let $(Y_n)_i$ denote the i^{th} component of Y_n. Define

$$\begin{aligned} G(Y_{n,(0)}, t_n) &= -(Y_{n,(0)})_k + hf((Y_{n,(0)})_0, t_n) && (3.9) \\ &= -hy'_{n,(0)} + hf(y_{n,(0)}, t_n). && (3.10) \end{aligned}$$

Recall that y'_n denotes the numerical approximation to $y'(t_n)$. We see immediately that (3.9) will be identically zero if the computed value of y and f at the current mesh point satisfies the differential equation (3.2). The computation of $f(y_n, t_n)$ is the *evaluation* (E) step.

Subsequent iterations are given by

$$Y_{n,(m+1)} = Y_{n,(m)} + \mathbf{c}\, G(Y_{n,(m)}, t_n) \tag{3.11}$$

for $m = 0, 1, \ldots$, where

$$Y_{n,(m)} = \left[y_{n,(m)} \;\; y_{n-1} \;\; \cdots \;\; y_{n-k+1} \;\; hy'_{n,(m)} \;\; hy'_{n-1} \;\; \cdots \;\; hy'_{n-k+1} \right]^T . \tag{3.12}$$

and $\mathbf{c}$ is a $2k$-dimensional vector defined as

$$\mathbf{c} = [\beta_0^* \; 0 \; \cdots \; 0 \; 1 \; 0 \; \cdots \; 0]^T \tag{3.13}$$

and 1 is the k^{th} component of $\mathbf{c}$. Equation (3.11) will be referred to as the functional iteration and will involve an evaluation (E) of f. This iteration may be repeated for a fixed number of steps or until $|G| << 1$. It will be helpful to note that after the first iteration, we have

$$\begin{aligned} hy'_{n,(1)} &= hy'_{n,(0)} + (-hy'_{n,(0)} + hf(y_{n,(0)}, t_n)) \\ &= hf(y_{n,(0)}, t_n) \end{aligned} \tag{3.14}$$

and

$$\begin{aligned} y_{n,(1)} &= \sum_{i=1}^{k} (\alpha_i - \beta_0^* \gamma_i) y_{n-i} + h \sum_{i=1}^{k} (\beta_i - \beta_0^* \delta_i) y'_{n-i} \\ &+ h\beta_0^* f(y_{n,(0)}, t_n). \end{aligned} \tag{3.15}$$

After m iterations, we obtain

$$\begin{aligned} hy'_{n,(m)} &= hy'_{n,(m-1)} + (-hy'_{n,(m-1)} + hf(y_{n,(m-1)}, t_n)) \\ &= hf(y_{n,(m-1)}, t_n) \end{aligned} \tag{3.16}$$

and

$$\begin{aligned} y_{n,(m)} &= \sum_{i=1}^{k} (\alpha_i - \beta_0^* \gamma_i) y_{n-i} + h \sum_{i=1}^{k} (\beta_i - \beta_0^* \delta_i) y'_{n-i} \\ &+ h\beta_0^* f(y_{n,(m-1)}, t_n). \end{aligned} \tag{3.17}$$

If h is chosen sufficiently small so that $|h\beta_0^*(\partial f/\partial y)| < 1$, then (3.17) converges. Also if $y_{n,(m)} \to y_n$, then (3.16) implies that $y'_{n,(m)} \to y'_n$ where y_n, y'_n satisfy

$$y'_n = f(y_n, t_n). \tag{3.18}$$

Thus we have

$$y_n \;=\; \sum_{i=1}^{k} \alpha_i^* y_{n-i} + h\sum_{i=1}^{k} \mathbf{y}'_{n-i} + \; h\beta_0^* f(y_n, t_n) \tag{3.19}$$

where $\alpha_i^* = \alpha_i - \beta_0^*\gamma_i$ and $\beta_i^* = \beta_i - \beta_0^*\delta_i$ for $i = 1, \ldots, k$. Equation (3.19) represents an implicit multistep method that will be referred to as the *corrector* (C). The implementations that have been incorporated into the general DAE integrators being developed at North Carolina State University use a k^{th} order Adams–Bashforth predictor and a k^{th} order Adams–Moulton corrector which will be referred to as an ABM k^{th} order pair. In practice only a fixed number of functional iterations are implemented and are summarized by the notation $P(EC)^M E$. The final function evaluation is

$$Y_{n,(M+1)} \;=\; Y_{n,(M)} + \mathbf{e}_k\, G(Y_{n,(M)}, t_n) \tag{3.20}$$

where $\mathbf{e}_k$ is a $2k$-dimensional vector defined as

$$\mathbf{e}_k = [0\;0\;\cdots\;0\;1\;0\;\cdots\;0]^T$$

and 1 is the k^{th} component. If

$$Y_{n,(M)} = \begin{bmatrix} y_{n,(M)} \\ y_{n-1} \\ \vdots \\ y_{n-k+1} \\ hy'_{n,(M)} \\ hy'_{n-1} \\ \vdots \\ hy'_{n-k+1} \end{bmatrix}, \tag{3.21}$$

then

$$Y_{n,(M+1)} = \begin{bmatrix} y_{n,(M)} \\ y_{n-1} \\ \vdots \\ y_{n-k+1} \\ hy'_{n,(M+1)} \\ hy'_{n-1} \\ \vdots \\ hy'_{n-k+1} \end{bmatrix} \tag{3.22}$$

and

$$\begin{aligned} hy'_{n,(M+1)} &= hy'_{n,(M)} + (-hy'_{n,(M)} + hf(y_{n,(M)}, t_n)) && (3.23) \\ &= hf(y_{n,(M)}, t_n). && (3.24) \end{aligned}$$

The first component $y_{n,(M)}$ is unchanged. Linear multistep methods are frequently derived from Taylor expansions or interpolation formulas. The elements of B and $\mathbf{c}$ for various ABM methods can be found in [61].

3.1.1 Nordsieck Implementation

The idea behind the Nordsieck implementation is to be able to convert a linear multistep step into a one-step method. This transformation makes it easier to change the step size. An example of the Nordsieck vector for a k^{th} order ABM pair is given by

$$Z_n = \begin{bmatrix} y_n \\ hy'_n \\ \frac{h^2}{2!}y''_n \\ \vdots \\ \frac{h^k}{k!}y_n^{(k)} \end{bmatrix}. \tag{3.25}$$

The k^{th} order ABM method upon which our implementation is based can be formulated using the vector

$$Y_n = \begin{bmatrix} y_n \\ hy'_n \\ hy'_{n-1} \\ \vdots \\ hy'_{n-k+1} \end{bmatrix} \tag{3.26}$$

which is a special case of (3.3). It turns out that (3.25) and (3.26) are related by a nonsingular linear transformation

$$Z_n = Q\,Y_n \tag{3.27}$$

where Q is chosen so that

$$Z(t_n) = QY(t_n) + O(h^{k+1}). \tag{3.28}$$

Q turns out to be independent of h [61]. $Z(t_n)$ and $Y(t_n)$ represent the exact values of Z_n, Y_n respectively. Using the formula

$$hy'(t_n - jh) = \sum_{i=1}^{k} (-j)^{i-1}\, i \left(\frac{h^i y^{(i)}(t_n)}{i!} \right) + O(h^{k+1}) \tag{3.29}$$

we get the expression

$$Q^{-1} = \left[\begin{array}{cc|cccc} 1 & & & & & \\ & 1 & & & & \\ \hline & 1 & -1 \cdot 2 & (-1)^2 \cdot 3 & \cdots & (-1)^{k-1} \cdot k \\ & 1 & -2 \cdot 2 & (-2)^2 \cdot 3 & \cdots & (-2)^{k-1} \cdot k \\ & \vdots & \vdots & \vdots & & \vdots \\ & 1 & -k \cdot 2 & (-k)^2 \cdot 3 & \cdots & (-k)^{k-1} \cdot k \end{array} \right]_{k+1 \times k+1} \tag{3.30}$$

where the omitted entries are zeros. Using the transformation (3.27) we have

$$\begin{aligned} Z_{n,(0)} &= Q\,Y_{n,(0)} \\ &= QBY_{n-1} \\ &= QBQ^{-1}Z_{n-1}, \end{aligned}$$

and

$$\begin{aligned} Z_{n,(m+1)} &= Q\,Y_{n,(m+1)} \\ &= QY_{n,(m)} + Q\mathbf{c}G(Y_{n,(m)}, t_n) \\ &= QY_{n,(m)} + Q\mathbf{c}G(Q^{-1}Z_{n,(m)}, t_n) \\ &= Z_{n,(m)} + \mathbf{d}G(Q^{-1}Z_{n,(m)}, t_n) \end{aligned}$$

where $\mathbf{d} = Q\mathbf{c}$. The final function evaluation is given by

$$Z_{n,(M+1)} = Q\,Y_{n,(M+1)}$$

$$
\begin{aligned}
&= QY_{n,(M)} + Q\mathbf{e}_1 G(Y_{n,(M)}, t_n) \\
&= QY_{n,(M)} + Q\mathbf{e}_1 G(Q^{-1}Z_{n,(M)}, t_n) \\
&= Z_{n,(M)} + \mathbf{e}_1 G(Q^{-1}Z_{n,(M)}, t_n).
\end{aligned}
$$

The matrix QBQ^{-1} turns out to be the Pascal triangle matrix given by

$$QBQ^{-1} = \begin{bmatrix} \binom{0}{0} & \cdots & \binom{k}{0} \\ & \ddots & \vdots \\ & & \binom{k}{k} \end{bmatrix}. \tag{3.31}$$

The implementation of these methods for a system of ODEs will have

$$Z_{n,(0)} = (QBQ^{-1} \otimes I)Z_{n-1}$$

as the prediction step,

$$Z_{n,(m+1)} = Z_{n,(m+1)} + (\mathbf{d} \otimes I)G(Q^{-1}Z_{n,(m)}, t_n)$$

for the correction steps, and

$$Z_{n,(M+1)} = Z_{n,(M)} + (\mathbf{e}_1 \otimes I)G(Q^{-1}Z_{n,(M)}, t_n)$$

as the final function evaluation. I is an identity matrix whose dimension is the same as y, and $\otimes$ denotes the Kronecker product. If $A \in \mathbb{R}^{m\times l}$ and $B \in \mathbb{R}^{n\times k}$, then the (right) Kronecker product [62] $A \otimes B$ is defined to be the partitioned matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1l}B \\ a_{21}B & a_{22}B & \cdots & a_{2l}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{ml}B \end{bmatrix} \in \mathbb{R}^{mn\times lk}. \tag{3.32}$$

3.1.2 Prediction using Nordsieck Vector

The goal of the general DAE methods is to integrate the implicitly defined ODE given by the derivative array equations

$$G(y', w, y, t) = 0$$

where $w = [y'', \ldots, y^{(\nu+1)}]$ and ν is the index of the DAE. For convenience, this section will assume that we are using the EI method for obtaining a solution to the original DAE $F(y', y, t) = 0$. Let $z = [y', w]$ so that we are considering the nonlinear system

$$G(z, y, t) = 0 \tag{3.33}$$

and $J = G_z$.

Integrating the implicitly defined ODE

$$y' = f(y, t) \tag{3.34}$$

requires that we be able to solve for y'_{n+1} given y_{n+1}, t_{n+1}. This is done by solving (3.33) via a damped Gauss–Newton algorithm

$$z^{[m+1]}_{n+1} = z^{[m]}_{n+1} - J^{\dagger}(z^{[m]}_{n+1}, y_{n+1}, t_{n+1})G(z^{[m]}_{n+1}, y_{n+1}, t_{n+1}) \tag{3.35}$$

as described in [22, 23].

Prior work [68, 86] assumed the the order of the integrator k was such that $k \geq \nu + 1$. By using the Nordsieck transformation described in the previous section to integrate the completion, we have after each step the vector

$$Z_n = \begin{bmatrix} y_n \\ hy'_n \\ a_n \\ b_n \end{bmatrix} \tag{3.36}$$

where

$$a_n = \begin{bmatrix} \frac{h^2}{2!}\, y_n'' \\ \vdots \\ \frac{h^{\nu+1}}{(\nu+1)!}\, y_n^{(\nu+1)} \end{bmatrix} \tag{3.37}$$

and

$$b_n = \begin{bmatrix} \frac{h^{\nu+2}}{(\nu+2)!}\, y_n^{(\nu+2)} \\ \vdots \\ \frac{h^k}{k!}\, y_n^{(k)} \end{bmatrix}. \tag{3.38}$$

Define

$$T(h) = \begin{bmatrix} \frac{2!}{h^2} I & & & \\ & \frac{3!}{h^3} I & & \\ & & \ddots & \\ & & & \frac{(\nu+1)!}{h^{\nu+1}} I \end{bmatrix}. \tag{3.39}$$

We can obtain the approximations of the first $\nu + 1$ derivatives of the solution by premultiplying (3.36) by a matrix as shown in (3.40)

$$\begin{bmatrix} y_n \\ hy_n' \\ w_n \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & \frac{1}{h} I & 0 & 0 \\ 0 & 0 & T(h) & 0 \end{bmatrix} \begin{bmatrix} y_n \\ hy_n' \\ a_n \\ b_n \end{bmatrix}. \tag{3.40}$$

The initial iterate in (3.35) is then given by

$$z_{n+1}^{[0]} = \begin{bmatrix} y_{n+1}' \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} y_n' \\ T(h)a_n \end{bmatrix}. \tag{3.41}$$

The benefits of this approach are that we have at least $O(h)$ approximations to the "true" derivatives of the solution prior to beginning the Gauss–Newton solve. It is hoped that this will allow the numerical solution to remain within the region of attraction of the solution manifold to the DAE. However for high index problems ($\nu > 3$), this approach will require using high order ABM methods. It is well known that the stability regions for ABM methods shrink as the order increases [61]. The use of high order methods for high index problems may force the integrator to take small steps which may not be appropriate for the problem. We would like the capability of using low order integrators on higher index problems when it appears feasible.

3.2 Polynomial Interpolation

One traditional way to improve the iterative solvers is through the use of better predictors. We begin this section by examining the use of polynomial interpolation in initializing the iteration (3.35). Given values $z_{n-i}, \ldots, z_n$ at times $t_{n-i}, \ldots, t_n$, define

$$L_{i,k}(t) = \prod_{\substack{j=0 \\ j \neq k}}^{i} \frac{(t - t_{n-j})}{(t_{n-k} - t_{n-j})}, \tag{3.42}$$

then the Lagrange form of the interpolating polynomial is given by

$$p_i(t) = \sum_{k=0}^{i} L_{i,k}(t) z_{n-k}. \tag{3.43}$$

The starting value of our iteration at time t_{n+1} is given by $p(t_{n+1})$. For the cases $i = 0, 1, 2$ we have from (3.43)

$$\begin{aligned}
p_0(t) &= z_n \\
p_1(t) &= L_{1,0}(t) z_n + L_{1,1}(t) z_{n-1} \\
&= \frac{(t - t_{n-1})}{(t_n - t_{n-1})} z_n + \frac{(t - t_n)}{(t_{n-1} - t_n)} z_{n-1} \\
p_2(t) &= L_{2,0}(t) z_n + L_{2,1}(t) z_{n-1} + L_{2,2}(t) z_{n-2} \\
&= \frac{(t - t_{n-1})(t - t_{n-2})}{(t_n - t_{n-1})(t_n - t_{n-2})} z_n + \frac{(t - t_n)(t - t_{n-2})}{(t_{n-1} - t_n)(t_{n-1} - t_{n-2})} z_{n-1} \\
&\quad + \frac{(t - t_n)(t - t_{n-1})}{(t_{n-2} - t_n)(t_{n-2} - t_{n-1})} z_{n-2}.
\end{aligned}$$

For a fixed step size h, these equations reduce to

$$\begin{aligned}
z_{n+1} &= p_0(t_{n+1}) = z_n \\
z_{n+1} &= p_1(t_{n+1}) = 2z_n - z_{n-1} \\
z_{n+1} &= p_2(t_{n+1}) = 3z_n - 3z_{n-1} + z_{n-2}.
\end{aligned}$$

If $z(t)$ is sufficiently differentiable, then it is easy to show that for small enough h that $\|z(t_{n+1}) - p_i(t_{n+1})\| = O(h^q)$ where q is the minimum of $i+1$ and the error in the already computed values of z_m.

Before proceeding further, we will define a couple of operators that will prove useful. Let $\{(z_n, t_n) \mid z_n \in \mathbb{R}^s,\ n = 0, 1, 2, \ldots \text{ and } t_n \in \mathbb{R}\}$ be a set of *equally spaced* data points in $\mathbb{R}^{s+1}$ ($t_n = t_0 + nh$). Then the *forward shift operator* E is defined by

$$Ez_n = z_{n+1}, \quad E^2 z_n = E(Ez_n) = z_{n+2}, \quad \text{etc.}$$

We similarly define E for negative exponents, e.g. $E^{-3}z_n = z_{n-3}$. The *backward difference operator* ∇ is defined by $\nabla = 1 - E^{-1}$. This leads to

$$\begin{aligned}
\nabla z_n &= z_n - z_{n-1} \\
\nabla^2 z_n &= \nabla(z_n - z_{n-1}) = z_n - 2z_{n-1} + z_{n-2}.
\end{aligned}$$

or, in general,

$$\begin{aligned}
\nabla^k z_n &= (1 - E^{-1})^k z_n \\
&= \sum_{i=0}^{k} (-1)^i \binom{k}{i} E^{-i} z_n \\
&= \sum_{i=0}^{k} (-1)^i \binom{k}{i} z_{n-i}. \qquad (3.44)
\end{aligned}$$

If $z(t) \in C^{k+1}$, we have the following result from [61]

$$\nabla^k z(t_n) = h^k z^{(k)}(t_n) + O(h^{k+1}). \qquad (3.45)$$

Let $p_m(t)$ be the predictive polynomial of degree m. We will also assume that the step size h is fixed so that the predictor is given by

$$p_m(t_{n+1}) = \sum_{i=0}^{m}(-1)^i \binom{m+1}{i+1} z_{n-i}.$$

Lemma 3.2.1 *Assume z is sufficiently smooth. Then $\left(z_{n+1} - p_m(t_{n+1})\right)/h^{m+1}$ will converge to $z_{n+1}^{(m+1)}$.*

Proof: This result follows almost immediately from (3.44) and (3.45).

$$\begin{aligned} z_{n+1} - p_m(t_{n+1}) &= z_{n+1} - \sum_{i=0}^{m}(-1)^i \binom{m+1}{i+1} z_{n-i} \\ &= \sum_{i=0}^{m+1}(-1)^i \binom{m+1}{i} z_{n+1-i} \\ &= \nabla^{m+1} z_{n+1} \\ &= h^{m+1} z_{n+1}^{(m+1)} + O(h^{m+2}). \end{aligned}$$

□

3.3 Analysis of Linear Time Varying Case

Given the linear time varying (LTV) DAE

$$E(t)y' + F(t)y = f(t) \tag{3.46}$$

where $E(t)$ is identically singular for all $t \in [t_0, t_f]$, we wish to approximate the solution $y(t)$ to (3.46) on $I = [t_0, t_f]$. It is known [21] that conditions (A1)-(A4) are equivalent to (3.46) being solvable for every sufficiently smooth $f(t)$. The theory for numerically solving (3.46) by either the EI or ICP methods is reasonably complete [19, 21]. Note that prediction is not actually needed because the equations are linear.

However, our goal in carefully examining the effect of prediction on (3.46) is to develop insight on what can be expected for general nonlinear problems. As we shall see the analysis of (3.46) sheds considerable light on the more general case.

The derivative array equations for (3.46) are

$$\begin{aligned}
Ey' + Fy &= f \\
(E' + F)y' + Ey'' + F'y &= f' \\
(E'' + 2F')y' + (2E' + F)y'' + Ey''' + F''y &= f'' \\
(E''' + 3F'')y' + (3E'' + 3F'')y'' + (3E' + F)y''' + Ey^{(4)} + F'''y &= f''' \\
\vdots \quad & \quad \vdots \\
\frac{d^\nu}{dt^\nu}(Ey' + Fy) &= f^{(\nu)}
\end{aligned}$$

which produces the singular system

$$\mathcal{A}\begin{bmatrix} y' \\ w \end{bmatrix} + \mathcal{B}y = g \tag{3.47}$$

where

$$\begin{aligned}
(\mathcal{A})_{i,j} &= \binom{i-1}{j-1} E^{(i-j)} + \binom{i-1}{j} F^{(i-j-1)}, \quad j \le i, && (3.48a) \\
(\mathcal{A})_{i,j} &= 0, \quad j > i, && (3.48b) \\
(w)_i &= y^{(i+1)}, \quad i = 1, \dots, \nu, && (3.48c) \\
(\mathcal{B})_i &= F^{(i-1)}, \quad i = 1, \dots, \nu + 1, && (3.48d) \\
(g)_i &= f^{(i-1)}, \quad i = 1, \dots, \nu + 1 && (3.48e)
\end{aligned}$$

and $\binom{m}{n} = 0$ if $n > m$.

The derivative array equations in the time varying case then take the form

$$\mathcal{C}z = \mathcal{D}u + g \tag{3.49}$$

where z is y', w and maybe some part of y and u are any fixed state variables. In the EI method, $u = y$, $\mathcal{C} = \mathcal{A}$, and $\mathcal{D} = -\mathcal{B}$, while for the ICP method $u = y_2$. Let the time values for the integrator be t_n and the computed values of $z, u, g, \mathcal{C}, \mathcal{D}$ at t_n be denoted by use of the subscript n. Thus the equation to be solved by the Gauss–Newton method is

$$\mathcal{C}_{n+1} z_{n+1} = \mathcal{D}_{n+1} u_{n+1} + g_{n+1}. \tag{3.50}$$

But the u_n are a subset of the state values of the integrator. Thus we know that

$$u_n = a_n + O(h^{k+1}) \tag{3.51}$$

where $a(t)$ is a smooth function. Therefore we can rewrite (3.50) as

$$\mathcal{C}_{n+1} z_{n+1} = b_{n+1} + O(h^{k+1}). \tag{3.52}$$

where $b(t) = D(t)a(t) + g(t)$ is also smooth.

Given a prediction $\widehat{z}_{n+1}$, the Gauss–Newton applied to (3.52) converges in one step to

$$z_{n+1} = (I - P_{n+1})\widehat{z}_{n+1} + \mathcal{C}^{\dagger}_{n+1} b_{n+1} + \mathcal{C}^{\dagger}_{n+1} O(h^{k+1}) \tag{3.53}$$

where $P = \mathcal{C}^{\dagger}\mathcal{C}$ so that $I - P$ is an orthogonal projector onto $\mathcal{N}(\mathcal{C}) = \mathcal{R}(\mathcal{C}^{\dagger})^{\perp}$, where $\mathcal{N}$ and $\mathcal{R}$ represent the null space and range space of a matrix. From (3.53), we can see that the component of z that lies in $\mathcal{R}(\mathcal{C}^T)$ is uniquely determined and the prediction $\widehat{z}$ is projected onto $\mathcal{N}(\mathcal{C})$. It is instructive to note that

$$P = \begin{bmatrix} I & 0 \\ 0 & D(t) \end{bmatrix}$$

due to the 1-fullness of $\mathcal{C}$ and the size of I is at least equal to the dimension of y. Therefore from (3.53) we have

$$\begin{bmatrix} y'_{n+1} \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & I - D_{n+1} \end{bmatrix} \begin{bmatrix} \widehat{y}'_{n+1} \\ \widehat{w}_{n+1} \end{bmatrix} + \mathcal{C}^{\dagger}_{n+1} b_{n+1} + O(h^{k+1})$$

if we are using the EI method. The point is that y' is not affected by prediction, but the w components are affected by prediction and $\mathcal{C}^{\dagger}b$.

Proposition 3.3.1 *Suppose that the Gauss–Newton iteration is applied to* (3.52) *and the underlying numerical integrator of the completion is of order $\kappa \geq 1$, the interpolation polynomial is p_m, and the sequence z_n converges as $h \to 0^+$ for fixed t_n to an $m+1$ times differentiable function $z(t)$. Then $z(t)$ is the solution of the auxiliary differential algebraic equation* (ADAE)

$$(I-P)z^{(m+1)} = 0 \qquad (3.54a)$$

$$Pz = \mathcal{C}^{\dagger}b \qquad (3.54b)$$

where b depends on derivatives of f and the solution y of the LTV DAE (3.46).

Proof: If we multiply (3.53) by $I - P_{n+1}$ and P_{n+1} we get the following pair of equations

$$(I-P_{n+1})z_{n+1} = (I-P_{n+1})\hat{z}_{n+1} \qquad (3.55)$$

$$P_{n+1}\, z_{n+1} = \mathcal{C}^{\dagger}_{n+1}b_{n+1} + \mathcal{C}^{\dagger}_{n+1}O(h^{\kappa}). \qquad (3.56)$$

Equation (3.54b) follows immediately from (3.56). From (3.55) we have

$$(I-P_{n+1})(z_{n+1}-\hat{z}_{n+1}) = 0$$

so that

$$(I-P_{n+1})(z_{n+1}-p_m(t_{n+1})) = 0.$$

Dividing by $h^{-(m+1)}$ results in

$$(I-P_{n+1})\frac{z_{n+1}-p_m(t_{n+1})}{h^{m+1}} = 0.$$

Holding t_n fixed and taking the limit as $h \to 0^+$ gives (3.54a) by Lemma 3.2.1. □

Proposition 3.3.1 tells us that if the Gauss–Newton iterates are converging, then the limit must be the ADAE (3.54). This DAE has no recognizable structure such as semi-explicit or Hessenberg.

Proposition 3.3.2 *System* (3.54) *is an index* $m+1$ *solvable DAE. Its solutions satisfy the auxiliary ODE* (AODE)

$$z^{(m+1)} = \left(\mathcal{C}^{\dagger}b\right)^{(m+1)} - \sum_{j=0}^{m} \binom{m+1}{j} P^{(m+1-j)} z^{(j)}. \tag{3.57}$$

Proof: This follows by differentiating (3.54b) $m+1$ times to get

$$P\, z^{(m+1)} = \left(\mathcal{C}^{\dagger}b\right)^{(m+1)} - \sum_{j=0}^{m} \binom{m+1}{j} P^{(m+1-j)} z^{(j)}. \tag{3.58}$$

Adding (3.54a) and (3.58) gives (3.57). □

In the rest of this chapter, we focus only on the cases where $m = 0, 1$ or 2. For later reference we will be referring to the following ODEs (3.57) given by Proposition 3.3.2

$$z' = \left(\mathcal{C}^{\dagger}b\right)' - P'z \tag{3.59}$$

$$z'' = \left(\mathcal{C}^{\dagger}b\right)'' - 2P'z' - P''z \tag{3.60}$$

$$z''' = \left(\mathcal{C}^{\dagger}b\right)''' - 3P'z'' - 3P''z' - P'''z. \tag{3.61}$$

3.3.1 Growth of $(I-P)z$

Recall that in the general DAE integrators the Pz component of z is uniquely determined by the derivative array equations. The $(I-P)z$ component is arbitrary. The actual performance of general DAE integrators will be affected by how $(I-P)z$ varies since the size of this component impacts convergence of the Gauss–Newton iteration and scaling of variables if they become too large. Therefore it is of considerable interest to know how $(I-P)z$ evolves during the integration.

Proposition 3.3.3 *Let z be the solution of the ADAE* (3.54). *Then $\theta = (I - P)z$ satisfies the differential equation*

$$\theta^{(m+1)} = -\sum_{j=0}^{m} \binom{m+1}{j} P^{(m+1-j)} \left(\theta + \mathcal{C}^{\dagger}b\right)^{(j)} . \tag{3.62}$$

Proof: From (3.54b) we have $z = \theta + \mathcal{C}^{\dagger}b$. Substituting for z in (3.57) gives

$$\theta^{(m+1)} + (\mathcal{C}^{\dagger}b)^{(m+1)} = (\mathcal{C}^{\dagger}b)^{(m+1)} - \sum_{j=0}^{m} \binom{m+1}{j} P^{(m+1-j)} \left(\theta + \mathcal{C}^{\dagger}b\right)^{(j)} \tag{3.63}$$

which gives (3.62). □

Equation (3.62) is a linear differential equation. The forced response due to $\mathcal{C}^{\dagger} b$ arises from the solution y and the forcing function f of (3.46), or equivalently, the u, g in (3.49). The solution of the associated homogeneous equation then reflects how θ changes in response to the varying of P and the choice of predictor $\hat{z}$.

Consider the homogeneous DAE

$$(I - P)z^{(m+1)} = 0 \tag{3.64a}$$

$$Pz = 0. \tag{3.64b}$$

Proposition 3.3.4 *Let Θ be a matrix solution of* (3.64) *of maximum rank. Let $\|\cdot\|$ be the Frobenius matrix norm, $\|Q\|^2 = \mathrm{trace}(Q^TQ)$. Then*

$$\frac{d}{dt}\|\Theta\| = 0 \quad \textit{for } m = 0 \tag{3.65}$$

$$\frac{d^2}{dt^2}\|\Theta\|^2 = 2\|\Theta'\|^2 \quad \textit{for } m = 1 \tag{3.66}$$

$$\frac{d^2}{dt^2}\|\Theta\|^2 = 3\|\Theta'\|^2 + c \quad \textit{for } m = 2, \quad c \textit{ constant.} \tag{3.67}$$

Proof: Equations (3.64a,3.64b) imply that $\Theta' \in \mathcal{N}(I-P)$ and $\Theta \in \mathcal{N}(P)$. Therefore the columns of Θ' are orthogonal to the columns of Θ or

$$\Theta'^T\Theta = \Theta^T\Theta' = 0.$$

Therefore

$$\left(\|\Theta\|^2\right)' = \text{trace}(\Theta^T\Theta' + (\Theta')^T\Theta) = 0$$

which proves (3.65).

For $m = 1$, (3.64) implies that the columns of Θ'' and Θ are mutually orthogonal. Hence

$$(\Theta^T\Theta)'' = (\Theta'')^T\Theta + 2(\Theta')^T\Theta' + \Theta^T\Theta'' = 2(\Theta')^T\Theta'.$$

Taking the trace gives the result (3.66).

For $m = 2$, we have

$$(\Theta^T\Theta)''' = (\Theta''')^T\Theta + 3(\Theta'')^T\Theta'' + 3(\Theta')^T\Theta'' + \Theta^T\Theta'''.$$

Since the columns of Θ''' and Θ are orthogonal we have

$$\begin{aligned}\frac{d^3}{dt^3}\left(\|\Theta\|^2\right) &= 3\,(\text{trace}\,((\Theta'')^T\Theta' + (\Theta')^T\Theta'') \\ &= 3\,\frac{d}{dt}\,(\text{trace}\,((\Theta')^T\Theta').\end{aligned}$$

Integration gives the result (3.67) with $c = -2\,\text{trace}\,(\Theta(t_0)^T\,\Theta''(t_0)) + \|\Theta'(t_0)\|^2$. □

Similar equations for higher m can be derived but become more complicated and are not as easy to interpret.

We have demonstrated that if the sequence z_n converges as $h \to 0^+$, to a smooth limit, then the limit satisfies the ADAE (3.54). We now show that the limiting behavior takes place. To address this issue, note that if we have an i^{th} order DAE with no lower order derivative terms

$$F(x^{(i)}, x, t) = 0 \tag{3.68}$$

and rewrite it as a first order DAE

$$
\begin{aligned}
F(v_i', x, t) &= 0 && (3.69a)\\
v_{i-1}' - v_i &= 0 && (3.69b)\\
&\vdots\\
v_2' - v_3 &= 0 && (3.69c)\\
v_1' - v_2 &= 0. && (3.69d)
\end{aligned}
$$

where $v_1 = x$, then we get the following result.

Lemma 3.3.1 *Applying a backward Euler* (BDF-1) *approximation to the first order DAE* (3.69) *is the same, for $n \geq m$, as using the approximation*

$$F(h^{-i}(x_{n+1} - p_{i-1}(t_{n+1})), x_{n+1}, t_{n+1}) = 0 \qquad (3.70)$$

on the i^{th} order DAE (3.68).

Proof: We begin by applying backward Euler's to (3.69)

$$
\begin{aligned}
F\left(\frac{v_{i,n+1} - v_{i,n}}{h}, x_{n+1}, t_{n+1}\right) &= 0 && (3.71a)\\
\frac{v_{i-1,n+1} - v_{i-1,n}}{h} - v_{i,n+1} &= 0 && (3.71b)\\
&\vdots\\
\frac{v_{2,n+1} - v_{2,n}}{h} - v_{3,n+1} &= 0 && (3.71c)\\
\frac{v_{1,n+1} - v_{1,n}}{h} - v_{2,n+1} &= 0. && (3.71d)
\end{aligned}
$$

We begin with (3.71d) which we rewrite as

$$\frac{x_{n+1} - x_n}{h} - v_{2,n+1} = 0 \qquad (3.72)$$

or

$$\frac{\nabla x_{n+1}}{h} - v_{2,n+1} = 0. \tag{3.73}$$

Thus (3.71c) is equivalent to

$$\frac{\nabla^2 x_{n+1}}{h^2} - v_{3,n+1} = 0. \tag{3.74}$$

Continuing in this manner we get for (3.71b)

$$\frac{\nabla^{i-1} x_{n+1}}{h^{i-1}} - v_{i,n+1} = 0 \tag{3.75}$$

and (3.71a)

$$F\left(\frac{\nabla^i x_{n+1}}{h^i}, x_{n+1}, t_{n+1}\right) = 0.$$

From Lemma 3.2.1 we have

$$\nabla^i x_{n+1} = x_{n+1} - p_{i-1}(t_{n+1})$$

which gives the result. □

Therefore, the sequence z_n is a numerical approximation to the solution of the solvable DAE (3.69). The $m = 0$ case gives us an index one DAE. It is known that index one solvable DAEs can be integrated with BDF-1 [13].

3.3.2 Convergence to the ADAE for $m = 1, 2$

We will now prove that BDF-1 on (3.54) converges for the cases $m = 1, 2$. Our interest is in the long term dynamic behavior of z_k so that we will not be concerned with the first few values.

We have observed in computational tests that if the DAE is solvable on a compact interval, and z_0 is fixed, then the z_k sequence stays bounded independent of h for h small. This suggests the boundedness assumption on y_n in Lemma 3.3.3 is reasonable. We will use the following result from [86] in the proof of Lemma 3.3.3.

Lemma 3.3.2 *If $\|e_{n+1}\| \leq (1+hL)\|e_n\| + D$, and $0 \leq nh \leq b$, then*

$$\begin{aligned} \|e_n\| &\leq (1+hL)^n\|e_0\| + D\frac{(1+hL)^n - 1}{hL} \\ &\leq e^{Lb}\|e_0\| + \frac{D}{hL}(e^{Lb} - 1). \end{aligned} \tag{3.76}$$

Lemma 3.3.3 *Suppose that we have a linear time varying ODE, $x' = A(t)x + f(t)$ where A, f are continuously differentiable on $I = [a,b]$, and consider two different schemes*

$$\frac{x_{n+1} - x_n}{h} = A_n x_n + f_n$$

which is forward Euler's and

$$\frac{y_{n+1} - y_n}{h} = (A_n + hC(t_n,h))\, y_n + f_n + O(h)$$

where C is smooth in t, h. Suppose that for a fixed y_0, there is a constant h_0 such that y_n is bounded for $0 \leq nh \leq b$ and $0 < h < h_0$. Let $e_n = y_n - x_n$. Then

$$e_n = O(\|e_0\|) + O(h).$$

A similar result holds for backward Euler.

Proof: The difference of the two schemes gives

$$\frac{e_{n+1} - e_n}{h} = A_n\, e_n + h\, C(t_n,h)y_n + O(h) = A_n\, e_n + O(h),$$

or

$$e_{n+1} = (I + hA_n)e_n + hO(h). \tag{3.77}$$

Since $A(t)$ is continuous on I, we have $\|A(t)\| \leq L$. By taking norms of both sides of (3.77) we have

$$\|e_{n+1}\| \leq (I + hL)\|e_n\| + Kh^2$$

where K is a constant given by the O-term. From Lemma (3.3.2) we have

$$\begin{aligned}\|e_n\| &\leq e^{Lb}\|e_0\| + \frac{Kh^2}{hL}(e^{Lb}-1)\\ &= e^{Lb}\|e_0\| + \frac{K}{L}(e^{Lb}-1)h.\end{aligned}$$

□

We are now ready to prove the following.

Theorem 3.3.1 *Let $m = 1, 2$. Suppose that* (3.46) *is a uniformly solvable DAE on the interval $\mathcal{I}$ of length L. Assume that the Gauss–Newton iteration is being used with a κ order integrator and fixed step h. Suppose that for z_0 in a neighborhood of a fixed $\hat{z}_0$, there is a constant h_0 such that z_n is bounded for $|nh| \leq L$. Then the sequence z_n is an $O(h)$ approximation to a solution of the ADAE.*

Proof: From the proof of Proposition 3.3.1 and Lemma 3.3.1, we note that the z_n are the same as a backward Euler applied to the ADAE. We now show that the backward Euler on the ADAE can be viewed as a forward Euler on a perturbation of an ODE. Lemma 3.3.3 then gives the needed result. We show in detail the result for $m = 1$. The case for $m = 2$ is outlined.

Since P is a self-adjoint projection we can write it as

$$P = V^T V$$

where V is a full row rank matrix with orthonormal columns. Similarly

$$I - P = U^T U$$

where the rows of U are orthonormal and also orthogonal to the rows of V. We may assume that U, V are smooth [18, 59]. The following relationships are immediate

$$UU^T = I$$

$$\begin{aligned} V\,V^T &= I \\ U\,V^T &= 0 \\ \begin{bmatrix} U \\ V \end{bmatrix}^{-1} &= \begin{bmatrix} U^T \; V^T \end{bmatrix}. \end{aligned}$$

For convenience let $g = V\,\mathcal{C}^\dagger\,b$ so that the (3.54) can be written, after a permutation of the equations, as

$$z' - y = 0 \tag{3.78a}$$

$$U\,y' = 0 \tag{3.78b}$$

$$V\,z = g. \tag{3.78c}$$

Let

$$z = V^T\,r + U^T\,s = q + p \tag{3.79}$$

so that r, s are independent variables. Equation (3.78c) gives $r = Vz = g$ which is solved uniquely at each time step. The key is what happens to the other component s of z. Note that (3.78b) is

$$U\,\left((U^T\,s)'' + q''\right) = 0$$

or

$$s'' + 2\,U\,(U')^T\,s' + U\,(U'')^T\,s + U\,q'' = 0. \tag{3.80}$$

Applying BDF-1 to (3.78) we have

$$\frac{z_{n+1} - z_n}{h} - y_{n+1} = 0 \tag{3.81}$$

$$U_{n+1}\,\frac{y_{n+1} - y_n}{h} = 0. \tag{3.82}$$

Using (3.81) for y_{n+1} in (3.82) gives

$$U_{n+1}\left(\frac{\frac{z_{n+1}-z_n}{h} - \frac{z_n - z_{n-1}}{h}}{h}\right) = 0.$$

If we substitute (3.79) for z and use the fact that $q = V^T g = V^T r$ is known, then we have

$$U_{n+1}\left(\frac{\frac{p_{n+1}-p_n}{h} - \frac{p_n - p_{n-1}}{h}}{h} + q''_{n+1} + O(h)\right) = 0$$

or equivalently

$$U_{n+1}\left(U_{n+1}^T s_{n+1} - 2U_n^T s_n + U_{n-1}^T s_{n-1}\right) = -h^2\, U_{n+1}\, q''_{m+1} + O(h^3). \tag{3.83}$$

Working with the left side of (3.83)

$$\begin{aligned}
& s_{n+1} - 2\,U_{n+1}U_n^T s_n + U_{n+1}U_{n-1}^T s_{n-1} \\
&= s_{n+1} - 2\,s_n + s_{n-1} + 2\,U_{n+1}(U_{n+1}^T - U_n^T)s_n + U_{n+1}(-U_{n+1}^T + U_{n-1}^T)s_{n-1} \\
&= s_{n+1} - 2\,s_n + s_{n-1} + 2\,U_{n+1}(U_{n+1}^T - U_n^T)(s_n - s_{n-1}) + \\
&\qquad U_{n+1}(U_{n+1}^T - 2\,U_n^T + U_{n-1}^T)s_{n-1}.
\end{aligned}$$

Dividing by h^2 we get

$$\frac{s_{n+1} - 2s_n + s_{n-1}}{h^2} + 2\,U_{n+1}\left((U_{n+1}^T)' + O(h)\right)\frac{s_n - s_{n-1}}{h} +$$
$$U_{n+1}\left((U''_{n+1})^T + O(h)\right)s_{n-1} = -U_{n+1}\,q''_{n+1} + O(h)$$

or

$$\frac{s_{n+1} - 2s_n + s_{n-1}}{h^2} + 2\,U_{n-1}\left((U_{n-1}^T)' + O(h)\right)\frac{s_n - s_{n-1}}{h} +$$
$$U_{n-1}\left((U''_{n-1})^T + O(h)\right)s_{n-1} = -U_{n-1}\,q''_{n+1} + O(h)$$

which is an $O(h)$ perturbation of a forward Euler's applied to (3.80). Thus if our original iterative sequence is bounded independent of h, then the s_n sequence is bounded independent of h. By Lemma 3.3.3 this sequence must approximate the numerical solution of (3.80).

For $m = 2$, a similar analysis leads to

$$z' - w = 0 \tag{3.84a}$$

$$w' - y = 0 \tag{3.84b}$$

$$Uy' = 0 \tag{3.84c}$$

$$Vz = g. \tag{3.84d}$$

and the ODE

$$s''' + 3\,U\,(U')^T s'' + 3\,U\,(U'')^T s' + U(U''')^T s + Uq''' = 0. \tag{3.85}$$

Applying BDF-1 to (3.84) gives

$$U_{n+1}(U_{n+1}^T - 3U_n^T s_n + 3U_{n-1}^T s_{n-1} - U_{n-2}^T s_{n-2}) = -h^3 U_{n+1} q_{n+1}''' + O(h^4). \tag{3.86}$$

Again, we first work only with the left side of (3.86)

$$s_{n+1} - 3\,U_{n+1}U_n^T s_n + 3\,U_{n+1}U_{n-1}^T s_{n-1} - U_{n+1}U_{n-2}^T s_{n-2}$$

$$= s_{n+1} - 3\,s_n + 3\,s_{n-1} - s_{n-2} + 3\,U_{n+1}(U_{n+1}^T - U_n^T)s_n +$$

$$3\,U_{n+1}(-U_{n+1}^T + U_{n-1}^T)s_{n-1} + U_{n+1}(U_{n+1}^T - U_{n-2}^T)s_{n-2}$$

$$= s_{n+1} - 3\,s_n + 3\,s_{n-1} - s_{n-2} + 3\,U_{n+1}\left(U_{n+1}^T - U_n^T\right)(s_n - 2\,s_{n-1} + s_{n-2}) +$$

$$3\,U_{n+1}\left(U_{n+1}^T - 2\,U_n^T + U_{n-1}^T\right)s_{n-1} + U_{n+1}\left(-2\,U_{n+1}^T + 3\,U_n^T - U_{n-2}^T\right)s_{n-2}$$

$$= s_{n+1} - 3\,s_n + 3\,s_{n-1} - s_{n-2} + 3\,U_{n+1}\left(U_{n+1}^T - U_n^T\right)(s_n - 2\,s_{n-1} + s_{n-2}) +$$

$$3\,U_{n+1}\left(U_{n+1}^T - 2\,U_n^T + U_{n-1}^T\right)(s_{n-1} - s_{n-2}) +$$

$$U_{n+1}\left(U_{n+1}^T - 3\,U_n^T + 3\,U_{n-1}^T - U_{n-2}^T\right)s_{n-2}.$$

Dividing by h^3 we get

$$\frac{s_{n+1} - 3s_n + 3s_{n-1} - s_{n-2}}{h^3} + 3\,U_{n+1}\left((U_{n+1}^T)' + O(h)\right)\frac{s_n - 2s_{n-1} + s_{n-2}}{h^2} +$$

$$3\,U_{n+1}\left(((U''_{n+1})^T + O(h)\right)\frac{s_{n-1}-s_{n-2}}{h} + U_{n+1}\left(((U'''_{n+1})^T + O(h)\right)s_{n-2}$$
$$= -U_{n+1}\,q'''_{n+1} + O(h)$$

or

$$\frac{s_{n+1}-3s_n+3s_{n-1}-s_{n-2}}{h^3} + 3\,U_{n-2}\left((U^T_{n-2})' + O(h)\right)\frac{s_n-2s_{n-1}+s_{n-2}}{h^2} +$$
$$3\,U_{n-2}\left(((U''_{n-2})^T + O(h)\right)\frac{s_{n-1}-s_{n-2}}{h} + U_{n-2}\left(((U'''_{n-2})^T + O(h)\right)s_{n-2}$$
$$= -U_{n-2}\,q'''_{n-2} + O(h)$$

and the result follows as with the $m = 1$ case. □

3.3.3 Index 2 Example

In order to illustrate the previous analysis we consider the following index two LTV system [46]

$$\begin{bmatrix} 0 & 0 \\ 1 & \eta t \end{bmatrix}\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} + \begin{bmatrix} 1 & \eta t \\ 0 & 1+\eta \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} f(t) \\ 0 \end{bmatrix} \qquad (3.87)$$

with solution

$$y_1(t) = f(t) + \eta\,t\,f'(t) \qquad (3.88)$$
$$y_2(t) = -f'(t). \qquad (3.89)$$

BDF-1 applied to (3.87) results in

$$y_{1,n} + \eta\,t_n\,y_{2,n} = f_n \qquad (3.90)$$
$$\frac{y_{1,n}-y_{1,n-1}}{h} + \eta\,t_n\frac{y_{2,n}-y_{2,n-1}}{h} + (1+\eta)\,y_{2,n} = 0. \qquad (3.91)$$

After solving (3.90) for $y_{1,n}$ and substituting this expression into (3.91) for $y_{1,n}$ and $y_{1,n-1}$, then solving for $y_{2,n}$ we have

$$y_{1,n} = f_n - \eta\,t_n\,y_{2,n} \qquad (3.92)$$

$$y_{2,n} = \frac{\eta}{1+\eta} y_{2,n-1} - \frac{f_n - f_{n-1}}{h(1+\eta)}. \tag{3.93}$$

Assume $f = 0$ in (3.87), then we have

$$y_{1,n} = -\eta\, t_n\, y_{2,n} \tag{3.94}$$

$$y_{2,n} = \frac{\eta}{1+\eta} y_{2,n-1}. \tag{3.95}$$

As noted in [46] it is required that $|\frac{\eta}{1+\eta}| < 1$ in order for $y_{2,n}$ to be stable (i.e. $-1/2 < \eta$), otherwise $y_{2,n}$ will oscillate and become unbounded as $n \to \infty$. This example illustrates where BDF methods applied to an index two linear time varying DAE can fail. However, the general DAE methods (EI or ICP) can integrate (3.87) without difficulty.

The derivative array equations for (3.87) are

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & \eta t \\ 1 & \eta t & 0 & 0 & 0 & 0 & 0 & 1+\eta \\ 1 & \eta t & 0 & 0 & 0 & 0 & 0 & \eta \\ 0 & 1+2\eta & 1 & \eta t & 0 & 0 & 0 & 0 \\ 0 & 2\eta & 1 & \eta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1+3\eta & 1 & \eta t & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1' \\ y_2' \\ y_1'' \\ y_2'' \\ y_1''' \\ y_2''' \\ y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} f \\ 0 \\ f' \\ 0 \\ f'' \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

We will use the EI method in this example. The system that we will solve is

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \eta t & 0 & 0 & 0 & 0 \\ 1 & \eta t & 0 & 0 & 0 & 0 \\ 0 & 1+2\eta & 1 & \eta t & 0 & 0 \\ 0 & 2\eta & 1 & \eta t & 0 & 0 \\ 0 & 0 & 0 & 1+3\eta & 1 & \eta t \end{bmatrix}}_{\mathcal{C}} \underbrace{\begin{bmatrix} y_1' \\ y_2' \\ y_1'' \\ y_2'' \\ y_1''' \\ y_2''' \end{bmatrix}}_{z} = \underbrace{\begin{bmatrix} f - x_1 - \eta t x_2 \\ -(1+\eta)x_2 \\ f' - \eta x_2 \\ 0 \\ f'' \\ 0 \end{bmatrix}}_{g} \tag{3.96}$$

We can find a completion to the least squares equations by premultiplying the equation $\mathcal{C}z = g$ (3.96) by $\mathcal{C}^\dagger$ to obtain

$$\mathcal{C}^\dagger \mathcal{C} = \mathcal{C}^\dagger g \tag{3.97}$$

$$\begin{bmatrix} I & 0 \\ 0 & D(t) \end{bmatrix} \begin{bmatrix} y' \\ w \end{bmatrix} = \mathcal{C}^\dagger g. \tag{3.98}$$

When numerically solving (3.96) the computation of $\mathcal{C}^\dagger$ can easily be obtained by either an SVD or QR. In order to examine the solutions given by the EI method we will compute an analytic generalized inverse of $\mathcal{C}$. Let $\mathcal{C} = BC$ be a full rank factorization of $\mathcal{C}$. Then it is well known [47] that the Moore–Penrose inverse of $\mathcal{C}$ is given by

$$\mathcal{C}^\dagger = C^T(CC^T)^{-1}(B^TB)^{-1}B^T. \tag{3.99}$$

MAPLE was used to compute

$$\mathcal{C} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & \eta t & 0 & 0 \\ 1 & \eta t & 0 & 0 \\ 0 & 1+2\eta & 1 & \eta t \\ 0 & 2\eta & 1 & \eta t \\ 0 & 0 & 0 & 1+3\eta \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -\dfrac{\eta t}{1+3\eta} & -\dfrac{\eta^2 t^2}{1+3\eta} \\ 0 & 0 & 0 & 1 & \dfrac{1}{1+3\eta} & \dfrac{\eta t}{1+3\eta} \end{bmatrix}}_{C}.$$

The orthogonal projector $P = \mathcal{C}^\dagger \mathcal{C}$ is given by

$$\begin{aligned} P &= C^T(CC^T)^{-1}C \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2+6\eta+9\eta^2+\eta^2t^2}{D} & \frac{\eta t(1+\eta^2t^2)}{D} & \frac{-\eta t(1+3\eta)}{D} & \frac{-\eta^2t^2(1+3\eta)}{D} \\ 0 & 0 & \frac{\eta t(1+\eta^2t^2)}{D} & \frac{1+6\eta+9\eta^2+\eta^2t^2+\eta^4t^4}{D} & \frac{1+3\eta}{D} & \frac{\eta t(1+3\eta)}{D} \\ 0 & 0 & \frac{-\eta t(1+3\eta)}{D} & \frac{1+3\eta}{D} & \frac{1+\eta^2t^2}{D} & \frac{\eta t(1+\eta^2t^2)}{D} \\ 0 & 0 & \frac{-\eta^2t^2(1+3\eta)}{D} & \frac{\eta t(1+3\eta)}{D} & \frac{\eta t(1+\eta^2t^2)}{D} & \frac{\eta^2t^2(1+\eta^2t^2)}{D} \end{bmatrix} \end{aligned} \tag{3.100}$$

where $D = \eta^4t^4 + 9\eta^2 + 2\eta^2t^2 + 6\eta + 2$. Notice that $P \to P^*$ as $t \to \infty$ where P^* is the diagonal matrix whose elements are $\{1, 1, 0, 1, 0, 1\}$.

We also have

$$\mathcal{C}^{\dagger}g = \begin{bmatrix} -\frac{y_2}{2} - \eta y_2 + \frac{f'}{2} + \eta t f'' \\ -f'' \\ \frac{(1+2\eta)(2+6\eta+9\eta^2+\eta^2t^2)f''}{\eta^4t^4+9\eta^2+2\eta^2T^2+6\eta+2} \\ \frac{\eta t(1+2\eta)(1+\eta^2t^2)f''}{\eta^4t^4+9\eta^2+2\eta^2t^2+6\eta+2} \\ \frac{-\eta t(1+3\eta)(1+2\eta)f''}{\eta^4t^4+9\eta^2+2\eta^2t^2+6\eta+2} \\ \frac{-\eta^2t^2(1+3\eta)(1+2\eta)f''}{\eta^4t^4+9\eta^2+2\eta^2t^2+6\eta+2} \end{bmatrix}. \tag{3.101}$$

The numerical results that follow integrate (3.87) on $[0,20]$, $\eta = -3/4$ and $f(t) = \cos(t)$.

The z_n are given by the Gauss–Newton iteration

$$z_n = \begin{bmatrix} y_n' \\ w_n \end{bmatrix} = (I-P)\widehat{z}_n + \mathcal{C}_n^{\dagger}g_n. \tag{3.102}$$

The y' components are uniquely determined (3.101)

$$\begin{aligned} y_1' &= \frac{y_2}{4} - \frac{\sin(t)}{2} + \frac{3}{4}t\cos(t) \\ y_2' &= \cos(t). \end{aligned}$$

The other components of z are given by

$$\begin{aligned} w_1 &= \frac{9t^2(16+9t^2)\widehat{w}_1 + 12t(16+9t^2)\widehat{w}_2 + 240t\widehat{w}_3 - 180t^2\widehat{w}_4 + 8\cos(t)(41+9t^2)}{656+288t^2+81t^4} \\ w_2 &= \frac{12t(16+9t^2)\widehat{w}_1 + 16(16+9t^2)\widehat{w}_2 + 320\widehat{w}_3 - 240t\widehat{w}_4 - 6t\cos(t)(16+9t^2)}{656+288t^2+81t^4} \\ w_3 &= \frac{240t\widehat{w}_1 + 320\widehat{w}_2 + (400+144t^2+81t^4)\widehat{w}_3 + 12t(16+9t^2)\widehat{w}_4 - 120t\cos(t)}{656+288t^2+81t^4} \\ w_4 &= \frac{-180t^2\widehat{w}_1 - 240t\widehat{w}_2 + 12t(16+9t^2)\widehat{w}_3 + 16(41+9t^2)\widehat{w}_4 + 90t^2\cos(t)}{656+288t^2+81t^4}. \end{aligned}$$

All computations were done on a CTX PC with an Intel Pentium 120 MHz processor. The EI code was written in FORTRAN 77 and LAPACK [4] was used to performed numerical computations of $\mathcal{C}^\dagger$.

Figure 3.1 shows the result of using polynomial interpolation for the cases $m = 0, 1, 2$. A second order ABM method with fixed step size $h = .05$ is used in integrating the completion. The top two plots are the graphs of y_1' and y_2' which are unaffected by prediction. The next four plots of $w_1, \ldots, w_4$ demonstrates essentially bounded behavior for $m = 0$, slow linear growth for $m = 1$ and much stronger growth in some of these components for $m = 2$. We observe this in Figure 3.1 for $m = 0, 1, 2$.

Convergence of BDF-1 for the index one case ($m = 0$) is clearly seen in Figure 3.2. The y' components of the z vector are omitted since they are not affected by prediction (assumption (A3)); only the w components are displayed since these values change depending on different prediction strategies. The AODE (3.59) is solved using the numerical code ODE.F written by Shampine [83] (obtained via NETLIB). The absolute and relative error tolerances were set to 10^{-6} in ODE and the step size was fixed at $h = .05$. Figure 3.2 shows that the z components given by integrating the AODE

$$z' = (\mathcal{C}^\dagger g)' - P'z$$

and the z components given by the Gauss–Newton iteration

$$z = (I - P)\hat{z} + \mathcal{C}^\dagger g$$

are approximating the same differential equation.

For the higher order predictors, $m = 1, 2$, we initialized the AODEs (3.60,3.61) at $t = 5h$ using the values of z given by the Gauss–Newton iteration at that time. Figures 3.3 and 3.4 show the values of w components given by integrating (3.60) and the Gauss–Newton method respectively. Again we observe that the AODE is correctly capturing the behavior of the z sequence. These figures also illustrate convergence as $h \to 0^+$ of z to the AODE solution. Figures 3.5 and 3.6 demonstrate the same conclusion for the $m = 2$ case.

The previous graphs dealt with the solutions themselves. The theory developed earlier states that it is the $(I - P)z$ term whose growth depends on the choice of the predictor. For time invariant DAEs the governing differential equation is $\theta^{(m+1)} = 0$ and polynomial growth results. However, the example considered here is not constant coefficient. An examination of the derivative array shows that the coefficients of the the solutions of the derivative array equations are asymptotically constant.

Figures 3.7-3.9 show the graphs of $\|(I-P)z\|$ for various step sizes and $m = 0, 1, 2$. Several points are worth noting. First, asymptotically the graph looks constant, linear, and quadratic for $m = 0, 1, 2$ respectively, as expected. Note the initial oscillation. While Proposition 3.3.4 asserts that $\|\Theta\|$ is constant, $\|(I - P)z\|$ is not initially constant. However we observed earlier that last four components of the vector $\mathcal{C}^{\dagger}g$ (3.101) will decay to zero as t increases. Additionally the matrix P will converge to a constant matrix.

Finally, we compare the global error in each component of the solution at time t_n by computing

$$\max_n |e_{j,n}| = \max_n |y_{j,n} - y_j(t_n)|. \tag{3.103}$$

Table 3.1 shows that the prediction strategy did not affect the global error. The numerical solution had the same accuracy regardless of the prediction strategy for this example.

	$h = .1$	$h = .05$	$h = .025$	$h = .0125$
$\max_n \lvert e_{1,n}\rvert$	5.5669D-02	1.4329D-02	3.6292D-03	9.1918D-04
$\max_n \lvert e_{2,n}\rvert$	4.4144D-03	1.0732D-03	2.6479D-04	6.6067D-05

Table 3.1: Global Errors for Index 2 LTV, $m = 0, 1, 2$.

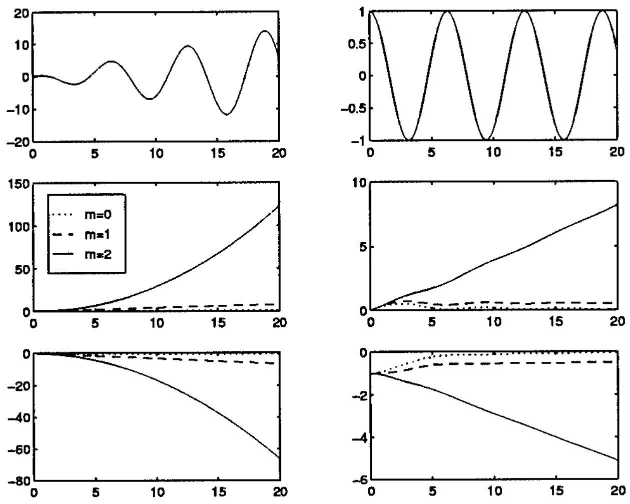

Figure 3.1: z from Gauss–Newton given by $m = 0, 1, 2$ prediction.

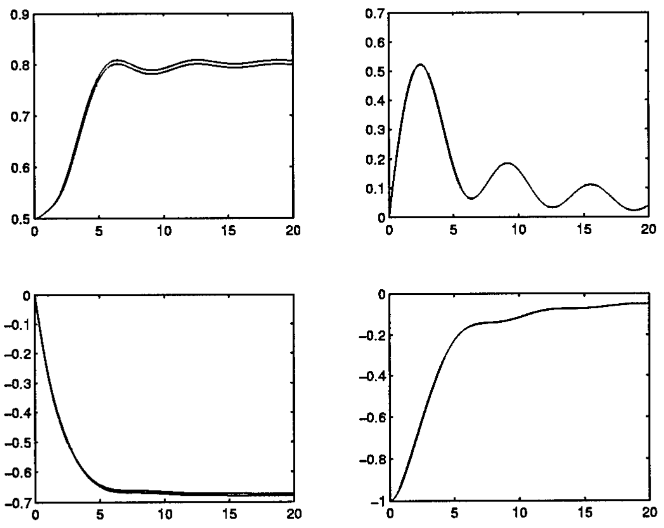

Figure 3.2: $z' = (C^{\dagger}b)' - P'z$ and w from Gauss–Newton, $m = 0$.

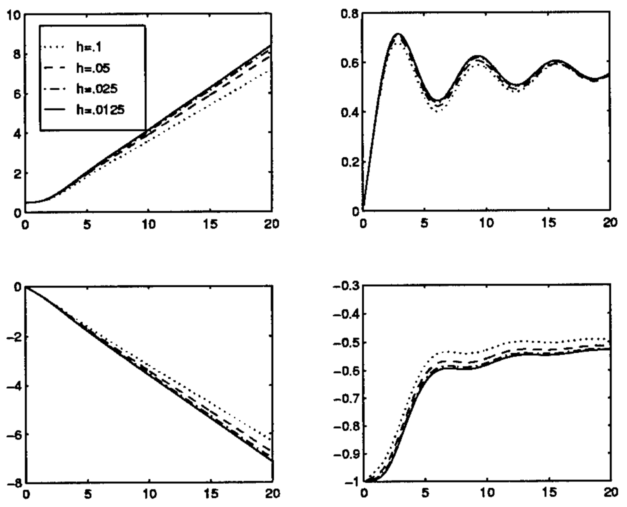

Figure 3.3: $z'' = (C^{\dagger}b)'' - 2P'z' - P''z$.

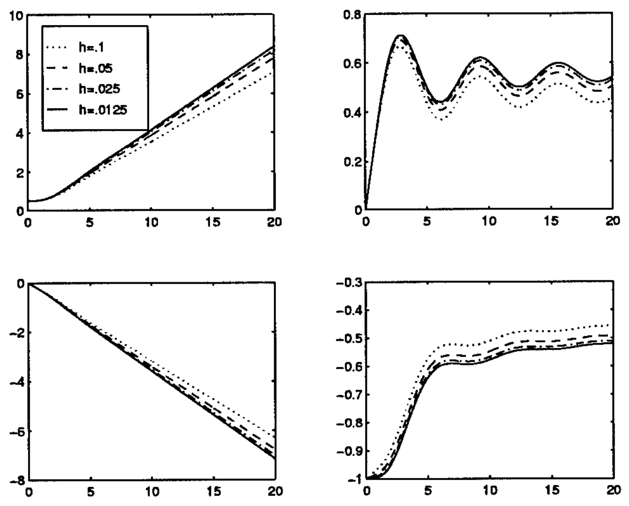

Figure 3.4: w from Gauss–Newton, $m = 1$.

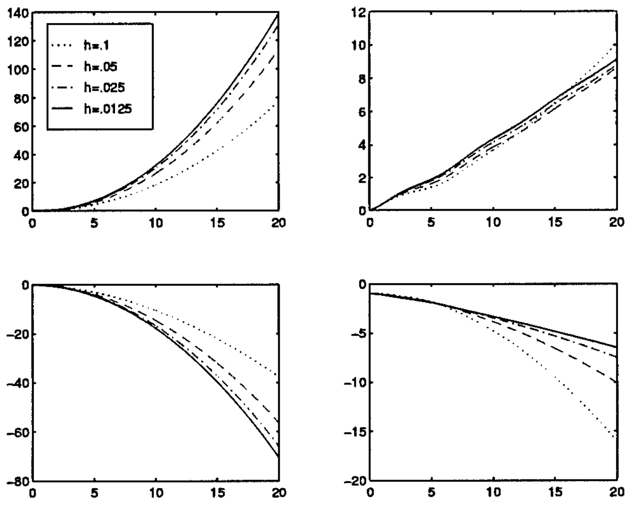

Figure 3.5: $z''' = (\mathcal{C}^{\dagger}b)''' - 3P'z'' - 3P''z' - P'''z$.

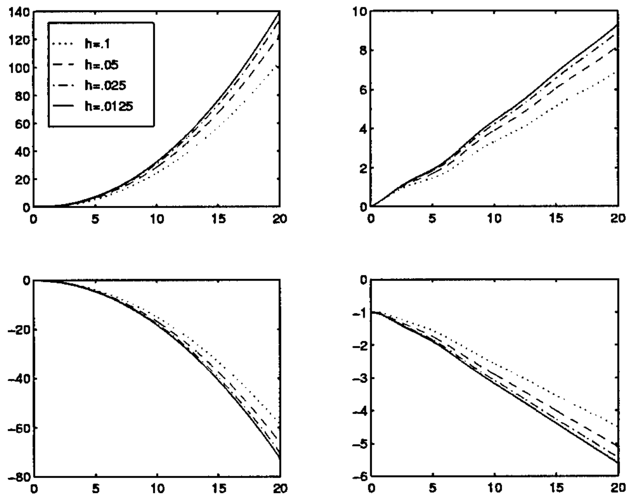

Figure 3.6: w from Gauss–Newton, $m = 2$.

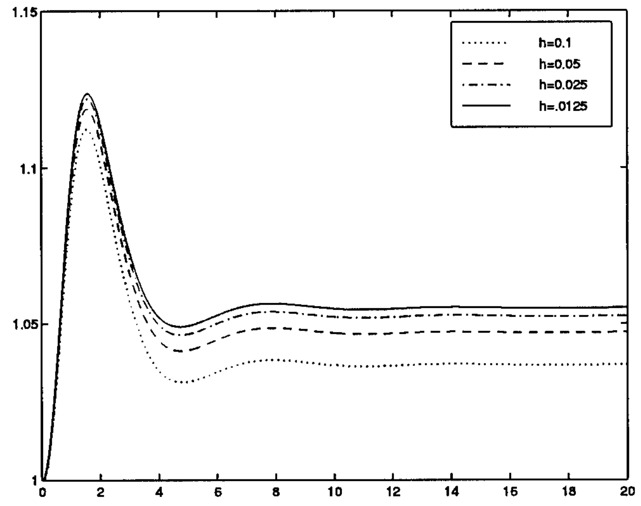

Figure 3.7: $\|(I - P)z\|$ for Index 2 LTV, $m = 0$.

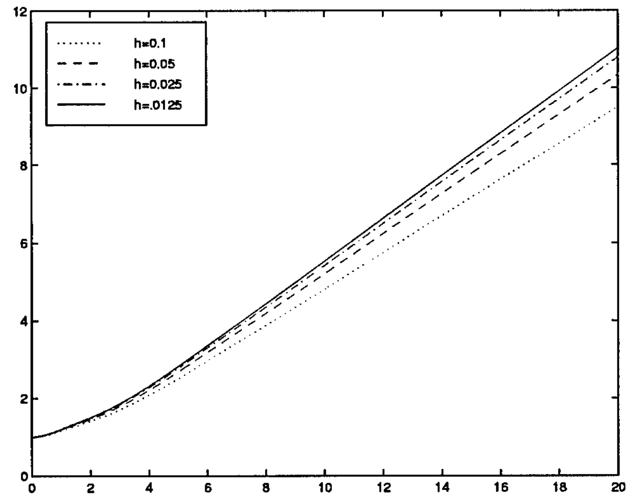

Figure 3.8: $\|(I - P)z\|$ for Index 2 LTV, $m = 1$.

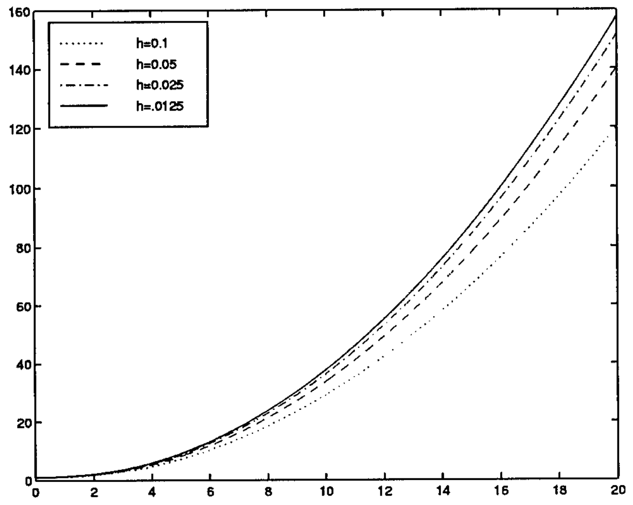

Figure 3.9: $\|(I-P)z\|$ for Index 2 LTV, $m=2$.

3.3.4 Index 4 Example

Earlier we discussed that prediction could affect the convergence of the Gauss–Newton iteration. This next example illustrates integrating an ill–conditioned problem and the effects of prediction. We also examine the use of a low order integrator on a higher index DAE. We begin by considering the linear time invariant DAE

$$Nx' - x = f(t) \tag{3.104}$$

where $(\alpha, \beta \neq 0)$

$$N = \left[\begin{array}{cc|cccc} \alpha & & & & & \\ & \beta & & & & \\ \hline & & 0 & 1 & & \\ & & & 0 & 1 & \\ & & & & 0 & 1 \\ & & & & & 0 \end{array}\right], \quad f(t) = \begin{bmatrix} t \\ \sin(t) \\ e^{-t} \\ t^2 \\ te^{-t} \\ \cos(t) \end{bmatrix}.$$

The block diagonal structure of N consists of a nilpotent matrix whose index is four [32] so that (3.104) is an index four DAE. A solution is given by

$$\begin{aligned} x_1 &= e^{t/\alpha} - t - \alpha && (3.105a) \\ x_2 &= e^{t/\beta} - \frac{1}{1+\beta^2}\sin(t) - \frac{\beta}{1+\beta^2}\cos(t) && (3.105b) \\ x_3 &= -\sin(t) - te^{-t} + e^{-t} - 2t && (3.105c) \\ x_4 &= \cos(t) - e^{-t} + te^{-t} - t^2 && (3.105d) \\ x_5 &= \sin(t) - te^{-t} && (3.105e) \\ x_6 &= -\cos(t). && (3.105f) \end{aligned}$$

By applying an orthogonal time-varying change of coordinates $x = U(t)y$ where

$$U(t) = \begin{bmatrix} \sin^2(\omega t) & L & \cos^2(\omega t) & -L & 0 & 0 \\ L & -\sin^2(\omega t) & 0 & 0 & L & \cos^2(\omega t) \\ 0 & 0 & \sin^2(\omega t) & L & \cos^2(\omega t) & -L \\ L & \cos^2(\omega t) & -L & \sin^2(\omega t) & 0 & 0 \\ \cos^2(\omega t) & -L & 0 & 0 & -\sin^2(\omega t) & -L \\ 0 & 0 & L & \cos^2(\omega t) & -L & \sin^2(\omega t) \end{bmatrix}$$

and $L = \sin(\omega t)\cos(\omega t)$, we construct the index four linear time varying DAE

$$NU(t)y' + (NU'(t) - U(t))y = f. \qquad (3.106)$$

whose solution is

$$y = U^T(t)x \qquad (3.107)$$

and is displayed in Figure 3.10.

We differentiated each equation in (3.106) four times so that the derivative array equations consisted of 30 equations in $\{y', y'', y''', y^{(4)}, y^{(5)}, y\}$. The parameters α, β and ω can be altered to change the conditioning of the problem. The numerical results that follow are for the case where $\alpha = \beta = -1$ and $\omega = 2$. For these values of α, β and ω, the singular values of $\mathcal{C}$ varied from 10^{-4} to 10^{3} at each step of the integration.

Second and third order ABM methods, denoted by ABM2 and ABM3 respectively, were used to integrate the completion. Figure 3.11 show the growth of the nonunique components $\|(I - P)z\|$ for various step sizes and ABM2. The graphs for ABM3 were similar. Linear interpolation ($m = 0$) was used in predicting all values of z at the next time step. We conclude that the EI method correctly computed a numerical solution to the index four DAE (3.106) using a lower order integrator on the completion and the prediction strategy $m = 0$. Tables 3.2-3.3 are the global errors using ABM2 and ABM3. The numerical solution is displayed in Figure 3.12.

However, the result was much different for the $m = 1$ case. Figure 3.13 shows that $\|(I - P)z\|$ blows up around $t = 3$. As h decreased we observe that the Gauss–Newton failed to converge earlier in the integration. We also attempted to use higher order integrators, Figure 3.14. Convergence still did not occur when applying $m = 1$ prediction to all of the z components.

In order to examine the use of the Nordsieck vector to initialize the z components, we integrated the completion with different ABM schemes. We initialized as many components of z as possible given by the integrator. Remaining components were initialized using prediction. For example, if we used the integrator ABM3, then we would initialize the z components corresponding to $\{y', y'', y'''\}$ using the values contained in the Nordsieck vector. We fixed the step size at $h = .025$. Figure 3.15 shows again that the $m = 0$ case could be successfully applied to components for which the Nordsieck vector did not provide enough information to initialize the entire z vector. It is important to note that the numerical solution is not effected by prediction. Whether we initialized all the z components by prediction or part using the Nordsieck vector, the numerical solution returned the same global errors.

However, we still had failure in the $m = 1$ case when we used the lower order integrator ABM2. Figure 3.16 shows that $m = 1$ did work for ABM3, ABM4 and ABM5. Again, it should be emphasized that a fourth order ABM method using a Nordsieck implementation provides approximations to derivatives of the true solution up to the fourth order. In this case the only components in the z vector that are initialized by polynomial prediction are those corresponding to $y^{(5)}$. ABM5 provided approximations to all the derivatives appearing in the derivative array equations. ABM5 also controlled the growth of $||(I - P)z||$ better than any of the other methods. Our conjecture that using the Nordsieck vector for initialization helps the initial iterate for the Gauss–Newton solve to remain close to the solution manifold of the DAE appears valid in this example. Prediction did not affect the numerical solution where convergence occured. However, higher order prediction caused failure. Using the Nordsieck vector for initialization provides a safeguard for convergence in ill conditioned problems.

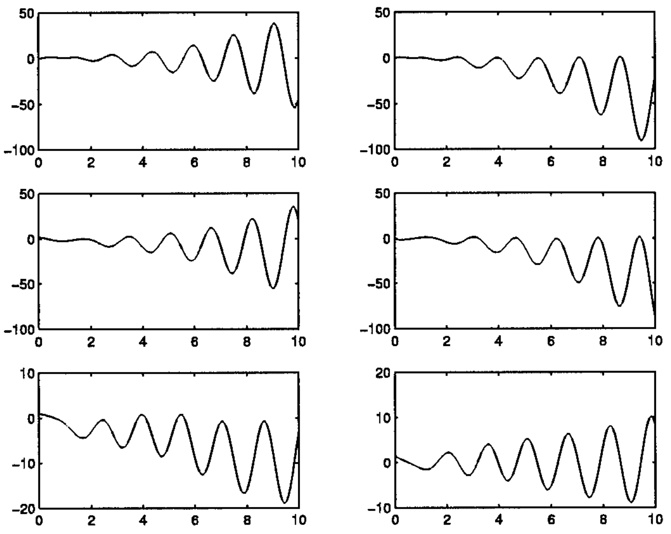

Figure 3.10: True Solution to Index 4 LTV (3.106).

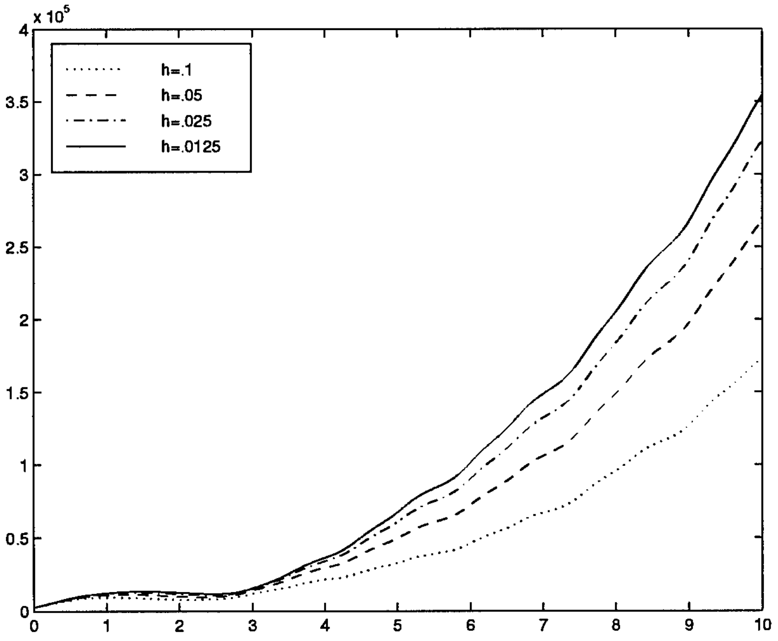

Figure 3.11: $\|(I-P)z\|$ for Index 4 LTV, $m=0$, ABM2.

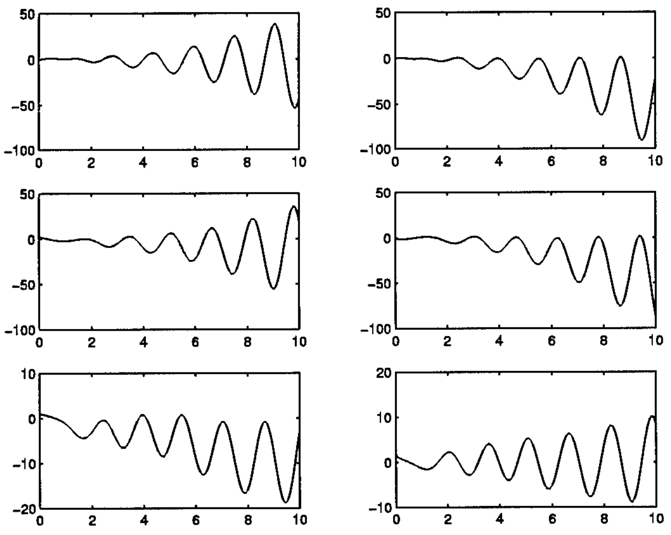

Figure 3.12: Numerical Solution to Index 4 LTV (3.106), $m = 0$, ABM2.

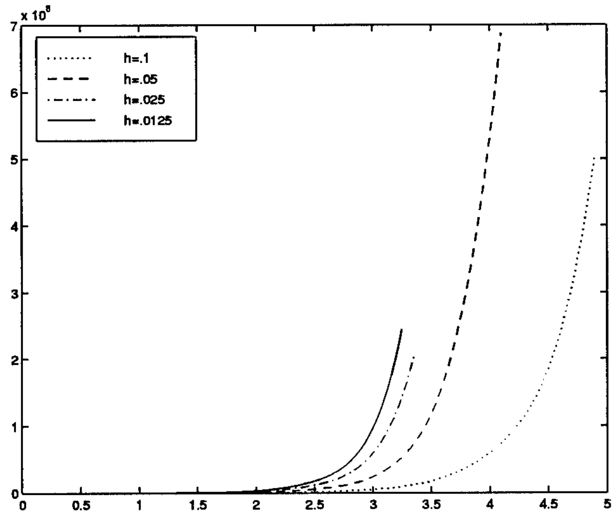

Figure 3.13: $\|(I - P)z\|$ for Index 4 LTV, $m = 1$, ABM2.

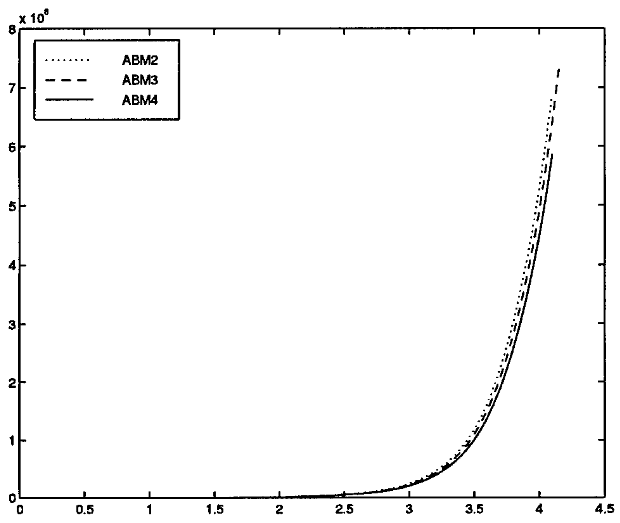

Figure 3.14: $\|(I-P)z\|$ for Index 4 LTV, $m = 1$.

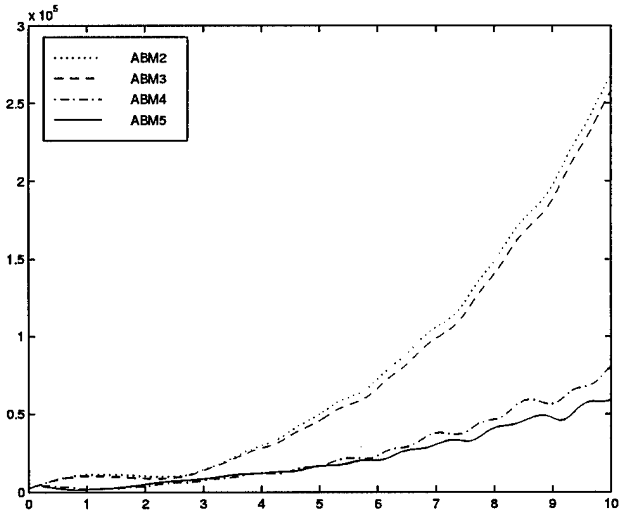

Figure 3.15: $\|(I-P)z\|$ for Index 4 LTV, $m = 0$ and Nordsieck.

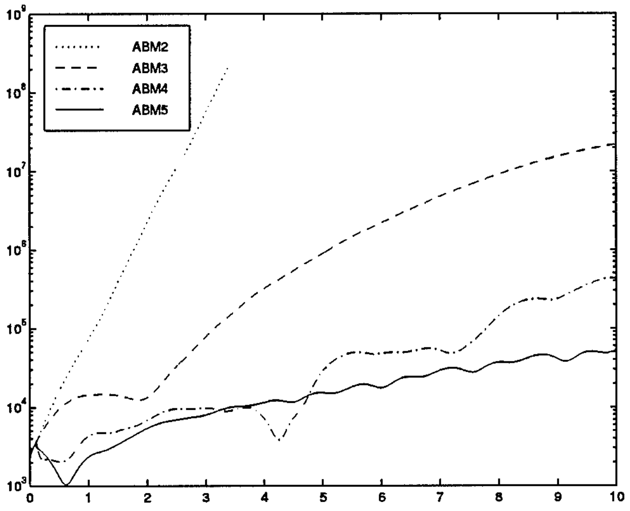

Figure 3.16: $\|(I-P)z\|$ for Index 4 LTV, $m=1$ and Nordsieck.

	$h=.1$	$h=.05$	$h=.025$	$h=.0125$
$\max_n \lvert e_{1,n}\rvert$	2.3302D+01	6.3993D+00	1.4535D+00	3.3323D-01
$\max_n \lvert e_{2,n}\rvert$	1.6569D+01	4.5085D+00	1.0223D+00	2.3430D-01
$\max_n \lvert e_{3,n}\rvert$	3.4310D+00	1.7658D+00	4.9174D-01	1.2429D-01
$\max_n \lvert e_{4,n}\rvert$	1.7861D+01	5.6336D+00	1.3829D+00	3.3104D-01
$\max_n \lvert e_{5,n}\rvert$	2.7166D+01	8.0756D+00	1.9080D+00	4.4649D-01
$\max_n \lvert e_{6,n}\rvert$	1.6160D+01	5.0587D+00	1.2323D+00	2.9364D-01

Table 3.2: Global Errors for Index 4 LTV, $m=0$, ABM2.

	$h = .1$	$h = .05$	$h = .025$	$h = .0125$
$\max_n \|e_{1,n}\|$	5.5484D+00	1.5554D-01	1.8873D-02	4.3434D-03
$\max_n \|e_{2,n}\|$	3.9334D+00	1.1373D-01	1.3470D-02	3.0437D-03
$\max_n \|e_{3,n}\|$	2.3633D+00	1.7231D-01	1.0350D-02	8.1917D-04
$\max_n \|e_{4,n}\|$	5.7546D+00	2.8564D-01	1.1906D-02	2.7959D-03
$\max_n \|e_{5,n}\|$	7.4058D+00	2.6152D-01	1.3183D-02	4.4531D-03
$\max_n \|e_{6,n}\|$	4.8886D+00	2.2686D-01	6.9777D-03	2.1837D-03

Table 3.3: Global Errors for Index 4 LTV, $m = 0$, ABM3.

	$h = .1$	$h = .05$	$h = .025$	$h = .0125$
$\max_n \|e_{1,n}\|$	5.9281D-01	8.4212D-02	5.3819D-03	3.1340D-04
$\max_n \|e_{2,n}\|$	4.4651D-01	5.9521D-02	3.7725D-03	2.2247D-04
$\max_n \|e_{3,n}\|$	3.6539D-01	1.2031D-02	1.4115D-03	1.0623D-04
$\max_n \|e_{4,n}\|$	3.5716D-01	6.3850D-02	4.6371D-03	2.9836D-04
$\max_n \|e_{5,n}\|$	3.5284D-01	9.8719D-02	6.8081D-03	4.1312D-04
$\max_n \|e_{6,n}\|$	2.8794D-01	5.6071D-02	4.2017D-03	2.6566D-04

Table 3.4: Global Errors for Index 4 LTV, $m = 1$ and Nordsieck, ABM4.

	$h = .1$	$h = .05$	$h = .025$	$h = .0125$
$\max_n \|e_{1,n}\|$	7.5466D-01	1.2311D-02	1.2340D-04	6.1310e-006
$\max_n \|e_{2,n}\|$	5.3438D-01	8.5953D-03	8.9314D-05	5.7579e-006
$\max_n \|e_{3,n}\|$	1.9715D-01	5.8098D-03	1.2323D-04	6.9728e-006
$\max_n \|e_{4,n}\|$	6.4097D-01	1.3592D-02	2.1520D-04	7.1199e-006
$\max_n \|e_{5,n}\|$	9.3373D-01	1.7077D-02	2.0312D-04	6.7074e-006
$\max_n \|e_{6,n}\|$	5.5461D-01	1.1753D-02	1.7759D-04	3.9276e-006

Table 3.5: Global Errors for Index 4 LTV, $m = 1$ and Nordsieck, ABM5.

3.4 Nonlinear Case

In the nonlinear case we have the derivative array equations given by the nonlinear system

$$G(z,t) = 0 \tag{3.108}$$

where $J = G_z$ is assumed to have constant rank. In Section 2.4 it was demonstrated that the Gauss–Newton method applied to (3.108) can be viewed as the integration of an ODE with a suitable step size. Given a fixed t, solving (3.108) for z would involve integrating

$$\frac{\partial z}{\partial \tau} = -G_z^{\dagger}(z(\tau,t),t)G(z(\tau,t),t). \tag{3.109}$$

We shall view the generation of the z_{n+1} iterates as a two step process. First, we predict $\widehat{z}_{n+1}$ by polynomial interpolation. Second, we follow the Gauss–Newton flow to steady state z_{n+1}. Then in a neighborhood of the solution of the DAE there exists a function $\Phi(\widehat{z},t)$ which is the limiting value of the continuous Gauss–Newton starting at $\widehat{z}$ and time t. By definition $\Phi(\Phi(\widehat{z},t),t) = \Phi(\widehat{z},t)$. Differentiating with respect to $\widehat{z}$ we note that Φ_z is a projection. In the linear case we had $\Phi(\widehat{z},t) = (I-P)\widehat{z} + \mathcal{C}^{\dagger}b$ where $P = \mathcal{C}^{\dagger}\mathcal{C}$ and $\Phi_z = I - P$. Another property of Φ by definition is

$$J^{\dagger}(\Phi(z,t),t)G(\Phi(z,t),t) = 0. \tag{3.110}$$

In the linear time varying case this is precisely equation (3.54b).

One step of predict and iterate then becomes

$$z_{n+1} = \Phi(p_m(t_{n+1}),t_{n+1}).$$

Thus

$$\begin{aligned}
0 &= z_{n+1} - \Phi(p_m(t_{n+1}),t_{n+1}) \\
&= \Phi(z_{n+1},t_{n+1}) - \Phi(p_m(t_{n+1}),t_{n+1}) \\
&= \Phi_z(z_{n+1},t_{n+1})(z_{n+1} - p_m(t_{n+1})) + O(\|z_{n+1} - p_m(t_{n+1})\|^2).
\end{aligned}$$

Assuming the limit exists, we divide by h^{m+1} and get that the limit as $h \to 0^+$ must then satisfy

$$\Phi_z(z,t)\, z^{(m+1)}(t) = 0$$

which is (3.54a). We also have that z_{n+1} is the limit of the Gauss–Newton iteration so that

$$J^{\dagger}(z_{n+1}, t_{n+1}) G(z_{n+1}, t_{n+1}) = 0$$

which is the nonlinear analogue of (3.54b). Thus we have the following

Proposition 3.4.1 *Suppose that the continuous Gauss–Newton iteration is being used, the underlying numerical integrator of the completion is of order $\kappa \geq 1$, the interpolation polynomial is p_m, and the sequence z_n converges as $h \to 0^+$ for fixed t_n to an $m+1$ times differentiable function $z(t)$. Then $z(t)$ is the solution of the index $m+1$ auxiliary differential algebraic equation* (ADAE)

$$\begin{aligned} \Phi_z(z(t),t)\, z^{(m+1)}(t) &= 0 && (3.111a)\\ J^{\dagger}(z(t),t) G(z(t),t) &= 0. && (3.111b) \end{aligned}$$

Proof: It only remains to show that (3.111) is index $m+1$. Equation (3.111b) says that $z(t)$ is a fixed point of the Gauss–Newton iteration and thus $z(t) = \Phi(z(t), t)$. Differentiating with respect to t we get $z' = \Phi_z\, z' + \Phi_t$ so that

$$(I - \Phi_z) z' = -\Phi_t.$$

Differentiating this equation m times with respect to t yields

$$(I - \Phi_z) z^{(m+1)} = \Gamma(t, z, \ldots, z^{(m)}). \qquad (3.112)$$

Adding (3.112) and (3.111a) gives $z^{(m+1)} = \Gamma(t, z, \ldots, z^{(m)})$ and (3.111) is index $m+1$.
□

Proposition 3.4.1 concludes that if the continuous Gauss–Newton converges, then the limit will again satisfy an auxiliary DAE. However, (3.111) is nonlinear and computing an explicit AODE that solutions to this DAE will also satisfy cannot be done here. We saw in the linear time varying case that the nonunique components $(I-P)z$ satisfied a linear ODE (3.62) whose dynamics varied with the prediction strategy. We should also expect that the nonunique components will change based on the choice of predictor $\hat{z}$ in the nonlinear case.

3.5 Implications

So far in this chapter we have carefully examined what happens for continuous updating of the the Jacobians. In practice, numerical integrators try to reuse Jacobians as much as possible. This can sometimes introduce difficulties with DAEs that do not occur with ODE integrators [38]. This section will briefly discuss this problem along with the implications of the preceding analysis.

An important special case is when the range of P is constant. In this case, the Θ equation becomes $\Theta^{(m+1)} = 0$ so that Θ is an m^{th} degree polynomial. Predictors given by $m = 0, 1$ are safe to use in this instance. The initial conditions for the z equation can be made smaller by solving the Gauss–Newton iteration to higher accuracy. For moderate length time intervals and well conditioned problems $m = 2$ could possibly be used.

In actual implementations of a general integrator one wants to reuse Jacobians as much as possible [37, 38]. A slight modification of the preceding analysis can be applied during the time that the Jacobian is constant to again get $\Theta^{(m+1)} = 0$. However the size of the initial values of Θ are based not only on the iteration but also the updating [38] so that these values cannot be assumed to be very small. Accordingly $m = 0, 1$ are better choices.

The analysis in this chapter has another important consequence. In previous discussions of general DAE integrators it was assumed that the order of the integrator was greater than the index. This was done so that the Nordsieck vector could be used for a predictor in the Gauss–Newton iteration. This allows for the integration of many important DAEs but has the disadvantage that it requires the use of high order integrators for moderately indexed DAEs and rules out the integration of DAEs of indices much above four.

The fact that the arbitrary terms terms also satisfy a differential equation means that one can relax this assumption. This is important. One can use low order integrators on higher index DAEs provided that polynomial prediction is used for the w terms.

We have shown that the growth of the nonunique terms in the the Gauss–Newton solver as part of a general DAE integrator satisfies an auxiliary DAE which depends on the predictor. The choice $m = 1$ appears safest, but $m = 0$ may have to be used on hard problems. The choice $m = 1$ will speed up the Newton iteration somewhat and avoids too rapid a growth of the Θ terms as described for the first time in this chapter. The use of integrators of order less than the index has been justified provided they are combined with predictors in the EI and ICP methods. These conclusions are also expected to hold for IOI based integrators.

Chapter 4

Conclusion

Current methods available in codes for the numerical integration of DAEs are limited to systems of low index ($\nu \leq 1$) or systems having a special structural form such as being Hessenberg. These methods are also known to suffer from a loss of accuracy in the higher index variables. Methods discussed in this thesis are being developed to integrate fully implicit, higher index, unstructured nonlinear DAEs. While these general DAE integration methods will not likely replace existing codes for index one problems, they can be applied to problems where existing methods are known to fail.

The goals of this thesis were to examine ways to make these general methods more numerically robust. One area of investigation was obtaining consistent initial conditions for the derivative array equations. We cannot expect to have good approximations to the higher order derivatives at the beginning of the integration. We investigated line search algorithms as a means to globalize the nonlinear equation solvers. We conducted computational studies which conclude that damping strategies are appropriate in determining consistent initial conditions.

A second area of investigation was the initialization of the Gauss-Newton method by polynomial interpolation at each step in the integration of the completion. Using a fixed step size, we have proven that the undetermined components in the derivative array equations will satisfy an auxiliary DAE that depends on the predictor. Additionally we examined the growth of these nonunique components in the linear time varying case. We have demonstrated the use of low order integrators when combined with polynomial prediction in the integration of higher index DAEs. The development of both the EI and ICP methods will incorporate the reuse of Jacobians due to computational cost and efficiency. In this case, prediction can be safely applied.

In summary, we have examined various globalization strategies for computing consistent initial conditions. It is anticipated that these strategies can also be applied during the integration of the DAE in order to provide a more robust code by allowing larger step sizes when appropriate. However this issue is still to be examined. Additionally we have examined polynomial interpolation applied to the initialization of the Gauss–Newton iteration during the integration of the completion.

A detailed list of the contributions of this research effort are

- investigated various line search strategies to globalize iterative solver for computing consistent initial conditions (Chapter 2);
- implemented these methods in FORTRAN and conducted numerical tests to verify utility of various strategies;
- implemented polynomial interpolation to provide different predictors for iterative solver (Section 3.2);
- for a fixed step size and using Lagrange polynomials, proved that limit of the Gauss–Newton solve satisfied an auxiliary DAE (Sections 3.3 and 3.4);
- proved solutions to ADAE satisfied a linear auxiliary ODE in the linear time varying case (Proposition 3.3.2);
- proved nonunique components from Gauss–Newton satisfied differential equation whose behavior is affected by predictor used (Propositions 3.3.3 and 3.3.4);

- proved convergence of BDF-1 applied to the auxiliary DAE in the $m = 1, 2$ cases (Theorem 3.3.1);
- conducted numerical tests to verify analysis of predictors applied to linear time varying DAEs (Sections 3.3.3 and 3.3.4);
- examined a mixed strategy using Nordsieck vector and prediction (Section 3.3.4).

Chapter 5

Future Research

While much progress has been made towards the development of general numerical methods for the integration of fully implicit higher index DAEs, there remains several areas of investigation. This section will briefly describe some of these important research issues.

It is anticipated that the final versions of the EI and ICP codes will be variable step, variable order versions. Work has been done on a variable step version of the EI code. It has been observed elsewhere that the local truncation error of this method is not identical to that of the numerical integration scheme on which it is based. Additional research is required to estimate the local truncation error. Strategies for determining when to change step size and order to preserve the stability of the method need to be developed. Incorporating the globalization strategies that we used in computing consistent initial conditions also needs to be investigated.

The EI and ICP methods currently employ linear multistep methods for integrating the completion. These methods are known to have bounded regions of absolute stability and are generally unsuitable for solving stiff ODEs. There needs to be a mechanism for determining when the completion is stiff. If stiff methods are employed in the integration of the completion, then the Jacobian or an approximation of the right hand side of the differential equation implicitly defined by the derivative array equations is required. To date, no other methods for integrating the completion have been implemented.

The linear algebra required to solve the least squares problem in the Gauss–Newton iteration is a major computational cost in these methods. The Jacobian of the derivatives array equations have a block triangular structure that can be exploited by faster least squares solvers. The theoretical work that would need to be completed would include showing that the solver works, the Gauss–Newton iteration converges, and meaningful completions can be obtained.

The cost of constructing the derivative array equations with computer algebra packages is high and limited to systems of moderate dimension. Although automatic differentiation has been studied for evaluating the derivative array equations and its Jacobian, this technology has not been implemented into any of the general methods for integrating DAEs. This might just involve rewriting the codes in C and incorporating ADOL-C for obtaining the derivative array equations and the associated Jacobian. Parallel implementation of these methods for large problems may also be required and needs to be investigated.

Several other possibilities can be examined which might give estimates on the higher derivatives contained in the derivative array equations. One is to fit a polynomial to converged values of the state variables. By differentiating this interpolatory polynomial we can obtain approximations to the derivatives of the solution. Another possibility is to use differencing to obtain such approximations. Additional differentiations of the original DAE so that it is one full with respect to some of the higher order derivatives might also be beneficial in this area. Again, further research is needed in this area.

List of References

[1] J. Amaya, *On the Convergence of Curvilinear Search Algorithms in Unconstrained Optimization*, Operations Research Letters, 4 (1985), pp. 31–34.

[2] ——, *A Differential Equations Approach to Function Minimization*, Rev. Mat. Apl., 1 (1987), pp. 3–6.

[3] ——, *Convergence of Curvilinear Search Algorithms to Second Order Points*, Rev. Mat. Apl., 10 (1989), pp. 71–79.

[4] E. Anderson and et. al., *LAPACK Users' Guide*, SIAM, second ed., 1995.

[5] A. Barrlund, *User Guide to CLS - A Fortran Code for the Numerical Solution of Higher Index Differential Algebraic Equation Systems*, Dept. of Computing Science, Umeå University, Sweden.

[6] ——, *Constrained Least Squares Methods for Linear Time Varying DAE Systems*, Numer. Math., 60 (1991), pp. 145–161.

[7] ——, *Comparing Stability Properties of Three Methods in DAEs or ODEs with Invariants*, BIT, 35 (1995), pp. 1–18.

[8] A. Barrlund and B. Kågström, *Analytical and Numerical Solutions to Higher Index Linear Variable Coefficient DAE Systems*, J. Comput. Appl. Math., 30 (1990), pp. 305–330.

[9] A. Ben-Israel, *A Newton–Raphson Method for the Solution of Systems of Equations*, J. Math. Anal. Appl., 15 (1966), pp. 243–252.

[10] P. T. Boggs, *The Convergence of the Ben–Israel Iteration for Nonlinear Least Squares Problems*, Math. Comp., 30 (1976), pp. 512–522.

[11] F. H. Branin, *Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations*, IBM J. Res. Develop., 16 (1972), pp. 504–522.

[12] K. E. Brenan, *Numerical Simulation of Trajectory Prescribed Path Control Problems by the Backward Differentiation Formulas*, IEEE Trans. Automat. Control, AC-31 (1986), pp. 266–269.

[13] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial–Value Problems in Differential–Algebraic Equations*, Classics in Applied Mathematics, SIAM, 1996.

[14] A. A. Brown and M. C. Bartholomew-Biggs, *ODE versus SQP Methods for Constrained Optimization*, J. Optim. Appl., 62 (1989), pp. 371–386.

[15] ——, *Some Effective Methods for Unconstrained Optimization Based on the Solution of Systems of Ordinary Differential Equations*, J. Optim. Appl., 62 (1989), pp. 211–224.

[16] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, *Using Krylov Methods in the Solution of Large–Scale Differential–Algebraic Systems*, SIAM J. Sci. Comput., 15 (1994), pp. 1467–1488.

[17] ——, *Consistent Initial Condition Calculation for Differential–Algebraic Systems.* Preprint, 1995.

[18] A. BUNSE-GERSTNER, R. BYERS, V. MEHRMANN, AND N. K. NICHOLS, *Numerical Computation of an Analytic Singular Value Decomposition of a Matrix Valued Function*, Numer. Math., 60 (1991), pp. 1–39.

[19] S. L. CAMPBELL, *The Numerical Solution of Higher Index Linear Time Varying Singular Systems of Differential Equations*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 334–348.

[20] ——, *Consistent Initial Conditions for Linear Time Varying Singular Systems*, Frequency Domain and State Space Methods for Linear Systems, (1986), pp. 313–318.

[21] ——, *A General Form for Solvable Linear Time Varying Singular Systems of Differential Equations*, SIAM J. Math. Anal., 18 (1987), pp. 1101–1115.

[22] ——, *A Computational Method for General Higher Index Nonlinear Singular Systems of Differential Equations*, 1989 IMACS Transactions Scientific Computing, 12 (1989), pp. 555–560.

[23] ——, *Least Squares Completions for Nonlinear Differential Algebraic Equations*, Numer. Math., 65 (1993), pp. 77–94.

[24] ——, *Numerical Methods for Unstructured Higher Index DAEs*, Annals of Numerical Mathematics, 1 (1994), pp. 265–277.

[25] ——, *High–Index Differential Algebraic Equations*, Mech. Struct. & Mach., 23 (1995), pp. 199–222.

[26] S. L. CAMPBELL AND C. W. GEAR, *The Index of General Nonlinear DAEs*, Numer. Math., 72 (1995), pp. 173–196.

[27] S. L. Campbell and E. Griepentrog, *Solvability of General Differential Algebraic Equations*, SIAM J. Sci. Comput., 16 (1995), pp. 257–270.

[28] S. L. Campbell and R. Hollenbeck, *Automatic Differentiation and Implicit Differential Equations*, in Computational Differentiation: Techniques, Applications and Tools, M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds., SIAM, Philadelphia, 1996, pp. 215–227.

[29] ——, *Automatic Differentiation and Implicit Differential Equations*, in Computational Differentiation: Techniques, Applications, and Tools, M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds., SIAM, Philadelphia, 1996, pp. 215–227.

[30] S. L. Campbell, R. Hollenbeck, K. Yeomans, and Y. Zhong, *Mixed Symbolic–Numerical Computations with General DAEs I: System Properties.* Preprint, 1997.

[31] S. L. Campbell, C. T. Kelley, and K. D. Yeomans, *Consistent Initial Conditions for Unstructured Higher Index DAEs: A Computational Study*, in Proc. Computational Engineering in Systems Applications, Lille, France, 1996, pp. 416–421.

[32] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*, Dover Publications, New York, 1991.

[33] S. L. Campbell and E. Moore, *Progress on a General Numerical Method for Nonlinear Higher Index DAEs*, in Proc. Sec. Int. Symp. Implicit and Robust Systems, Warsaw, 1991, pp. 55–59.

[34] ——, *Progress on a General Numerical Method for Nonlinear Higher Index DAEs II*, Circuits Systems Signal Process, 13 (1994), pp. 123–138.

[35] ——, *Constraint Preserving Integrators for General Nonlinear Higher Index DAEs*, Numer. Math., 69 (1995), pp. 383–399.

[36] S. L. CAMPBELL, E. MOORE, AND Y. ZHONG, *Utilization of Automatic Differentiation in Control Algorithms*, IEEE Trans. Automat. Contr., 39 (1994), pp. 1047–1052.

[37] S. L. CAMPBELL AND Y. ZHONG, *Computational Speedup for General Nonlinear Higher Index DAE Integrators*, in Proc. 94 IMACS World Congress on Scientific Computation, 1994, pp. 69–73.

[38] ——, *Jacobian Reuse in Explicit Integrators for Higher Index DAEs.* To appear in Appl. Num. Math, 1997.

[39] J. E. DENNIS AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics in Applied Mathematics, SIAM, 1996.

[40] P. DEUFLHARD, *A Modified Newton Method for the Solution of Ill–Conditioned Systems of Nonlinear Equations with Applications to Multiple Shooting*, Numer. Math., 22 (1974), pp. 289–315.

[41] P. DEUFLHARD AND V. APOSTOLESCU, *A Study of the Gauss–Newton Algorithm for the Solution of Nonlinear Least Squares Problems.* Preprint Nr. 51, January 1980.

[42] P. DEUFLHARD, E. HAIRER, AND J. ZUGCK, *One Step and Extrapolation Methods for Differential–Algebraic Systems*, Numer. Math., 51 (1987), pp. 501–516.

[43] I. DIENER, *On the Global Convergence of Path–Following Methods to Determine All Solutions to a System of Nonlinear Equations*, Mathematical Programming, 39 (1987), pp. 181–188.

[44] M. FLIESS, J. LÉVIN, P. MARTIN, AND P. ROUCHON, *Flatness and Defect of Non-linear Systems: Introductory Theory and Examples*, Int. J. Control, 61 (1995), pp. 1327–1361.

[45] C. W. GEAR, *Simultaneous Numerical Solution of Differential–Algebraic Equations*, IEEE Trans. Circuit Theory, 18 (1971), pp. 89–95.

[46] C. W. GEAR AND L. R. PETZOLD, *ODE Methods for the Solution of Differential/Algebraic Systems*, SIAM J. Numer. Anal., 21 (1984), pp. 716–728.

[47] G. H. GOLUB AND C. F. V. LOAN, *Matrix Computations*, John Hopkins University Press, 1996.

[48] V. GOPAL AND L. T. BIEGLER, *A Successive Linear Programming Approach for Initialization and Reinitialization after Discontinuities of Differential Algebraic Equations.* Preprint, 1996.

[49] E. GRIEPENTROG AND R. MÄRZ, *Differential–Algebraic Equations and Their Numerical Treatment*, Teubner–Texte zur Mathematik, Band 88, 1986.

[50] E. HAIRER AND G. WANNER, *Solving Ordinary Differential Equations II*, Springer Series in Computational Mathematics, Springer-Verlag, 1991.

[51] W. M. HÄUSSLER, *The Convergence of the Damped Levenberg–Morrison–Marquardt resp. Gauss–Newton Methods in Adequate Nonlinear Regression*, Methods of Operations Research, 51 (1984), pp. 13–22.

[52] W. M. HÄUSSLER, *A Kantorovich–type Convergence Analysis for the Gauss–Newton–Method*, Numer. Math., 48 (1986), pp. 119–125.

[53] A. C. HINDMARSH, *ODEPACK, A Systematized Collection of ODE Solvers*, in Scientific Computing, R. S. Stepleman and et al., eds., North-Holland, Amsterdam, 1983, pp. 55–64.

[54] R. HOLLENBECK, *A Computational Comparison of RADAU5 and Explicit Integration of the Completion of Hessenberg DAE Systems*, Master's thesis, N. C. State University, 1997.

[55] C. T. KELLEY, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers in Applied Mathematics, SIAM, 1995.

[56] O. KNOTH, *A Globalization Scheme for the Generalized Gauss–Newton Method*, Numer. Math., 56 (1989), pp. 591–607.

[57] A. KRÖNER, W. MARQUARDT, AND E. D. GILLES, *Computing Consistent Initial Conditions for Differential–Algebraic Equations.* Institut für Systemdynamik und Regelungstechnik.

[58] P. KUNKEL AND V. MEHRMANN, *Regular Solutions of Nonlinear Differential–Algebraic Equations and their Numerical Determination.* To appear in Numer. Math.

[59] ——, *Smooth Factorizations of Matrix Valued Functions and Their Derivatives*, Numer. Math., 60 (1991), pp. 115–131.

[60] P. KUNKEL, V. MEHRMANN, W. RATH, AND J. WEICKERT, *A New Software Package for Linear Differential–Algebraic Equations*, SIAM J. Sci. Comput., 18 (1997), pp. 115–138.

[61] J. D. LAMBERT, *Numerical Methods for Ordinary Differential Systems*, John Wiley & Sons, New York, 1993.

[62] P. LANCASTER AND M. TISMENETSKY, *The Theory of Matrices: Second Edition with Applications*, Academic Press, Inc., New York, 1985.

[63] B. LEIMKUHLER, L. R. PETZOLD, AND C. W. GEAR, *Approximation Methods for the Consistent Initialization of Differential–Algebraic Equations*, SIAM J. Numer. Anal., 28 (1991), pp. 205–226.

[64] C. LUBICH, *Extrapolation Integrators for Constrained Multibody Systems*, Impact Comput. Sci. Engrg., 3 (1991), pp. 213–234.

[65] C. MAJER, W. MARQUARDT, AND E. D. GILLES, *Reinitialization of DAE's after Discontinuities*, tech. rep., Lehrstuhl für Prozeßtechnik, February 1995.

[66] T. MALY AND L. R. PETZOLD, *Numerical Methods and Software for Sensitivity Analysis of Differential–Algebraic Systems*, Appl. Numer. Math., 20 (1996), pp. 57–79.

[67] R. MÄRZ, *Numerical Methods for Differential–Algebraic Equations*, Acta Numerica, (1992), pp. 141–198.

[68] E. L. MOORE, *Constraint Preserving Multistep Integrators for Differential Algebraic Equations*, PhD thesis, N. C. State University, 1994.

[69] C. C. PANTELIDES, *The Consistent Initialization of Differential–Algebraic Systems*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 213–231.

[70] H. O. PEITGEN, M. PRÜFER, AND K. SCHMITT, *Global Aspects of the Continuous and Discrete Newton Method: A Case Study*, Acta Applicandae Mathematicae, 13 (1988), pp. 123–202.

[71] L. R. PETZOLD, *A Description of DASSL: A Differential/Algebraic System Solver*, in Scientific Computing, R. S. Stepleman and et al., eds., North-Holland, Amsterdam, 1983, pp. 65–68.

[72] P. J. RABIER AND W. C. RHEINBOLDT, *A General Existence and Uniqueness Theory for Implicit Differential–Algebraic Equations*, SIAM J. Numer. Anal., 4 (1991), pp. 563–582.

[73] ——, *A Geometric Treatment of Implicit Differential–Algebraic Equations*, J. Differential Equations, 109 (1994), pp. 110–146.

[74] ——, *On the Numerical Solution of the Euler–Lagrange Equations*, SIAM J. Numer. Anal., 32 (1995), pp. 318–329.

[75] ——, *Classical and Generalized Solutions of Time-Dependent Linear Differential-Algebraic Equations*, Linear Algebra Appl., 245 (1996), pp. 259–293.

[76] S. REICH, *On an Existence and Uniqueness Theory for Nonlinear Differential-Algebraic Equations*, Circuits Systems Signal Process, 10 (1991), pp. 343–359.

[77] W. C. RHEINBOLDT, *Differential–Algebraic Systems as Differential Equations on Manifolds*, Math. Comp., 43 (1984), pp. 473–482.

[78] ——, *MANPAK: A Set of Algorithms for Computations on Implicitly Defined Manifolds*, Computers Math. Applic., 32 (1996), pp. 15–28.

[79] ——, *Solving Algebraically Explicit DAEs with the MANPAK-Manifold-Algorithms*, Computers Math. Applic., 33 (1997), pp. 31–43.

[80] J. SCHROPP, *A note on minimization problems and multistep methods.* To Appear in Numer. Math.

[81] ——, *One and Multistep Procedures for Constrained Minimization Problems.* Submitted to SIAM Journal of Numerical Analysis.

[82] ——, *Using Dynamical Systems Methods to Solve Minimization Problems*, Applied Numerical Mathematics, 18 (1995), pp. 321–335.

[83] L. F. SHAMPINE, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.

[84] K. TANABE, *Continuous Newton–Raphson Method for Solving an Underdetermined System of Nonlinear Equations*, Nonlinear Analysis, Theory, Methods & Applications, 3 (1979), pp. 495–503.

[85] ——, *Global Analysis of Continuous Analogues of the Levenberg–Marquardt and Newton–Raphson Methods for Solving Nonlinear Equations*, Ann. Inst. Statist. Math., 37 (1985), pp. 189–203.

[86] Y. ZHONG, *Efficient Numerical Solutions for Nonlinear Higher Index Differential Algebraic Equations*, PhD thesis, N. C. State University, 1997.