AN ALGORITHM FOR SOLVING INDEX 1 DIFFERENTIAL ALGEBRAIC EQUATIONS IN BOUNDARY VALUE PROBLEMS

By

PAT S.Y. CHAN

B.Sc., University of Toronto, Canada, 1987

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
(DEPARTMENT OF COMPUTER SCIENCE)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
August 1989
© Pat S.Y. Chan, 1989
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science
The University of British Columbia
Vancouver, Canada

Date Aug. 11, 84
Abstract

In this thesis we consider the numerical solution of boundary value problems for differential algebraic equations (DAEs) of index 1. It was shown in Ascher [1], [2], that if symmetric schemes are used for linear index 1 DAEs, the stability is controlled by the stability of an auxiliary (ghost) differential problem which is not necessarily stable even if the given problem is. We examine an algorithm which preserves the stability of the ghost problem by arranging the auxiliary boundary conditions properly. The algorithm has been implemented in a code (DAESOLVE) and tested on numerous examples. This code has been compared with a code which chooses the boundary conditions arbitrarily, showing favourable results.
# Contents

Abstract .................................................. i  

Contents .................................................. ii  

Acknowledgement ......................................... iii  

1 Introduction ........................................... 1  

2 Background Material For BVP ....................... 6  
   2.1 Projecting Boundary Conditions .................. 13  

3 Background Material For DAE ....................... 17  
   3.1 Classification of DAEs ............................ 17  
   3.2 Symmetric Schemes for DAEs in BVPs ............. 20  
      3.2.1 Analysis .................................... 23  

4 Algorithm (linear index 1 DAEs in BVPs) .......... 28  
   4.1 Details of Step 1 ................................ 30  
   4.2 Details of Step 2 ................................ 33  
   4.3 Details of Step 3 ................................ 35  
   4.4 Details of Step 4 ................................ 39  

5 Numerical Examples .................................... 40  
   5.1 Example A ......................................... 41  
   5.2 Example B ......................................... 43  
   5.3 Example C ......................................... 46  

6 Conclusion and Future Research ...................... 51  

Bibliography .............................................. 53
Acknowledgement

First, I would like to extend my special thanks to my supervisor Dr. Uri Ascher for suggesting my thesis topic, for the many helpful "remote" e-mail discussions, and for the financial support. Thanks also go to Dr. Feng Gao who acts as the second reader of my thesis.

Thanks to all of my friends in the department and in my church.

Last but not the least, my deepest thanks go to my parents, brother, sisters, brother-in-law, and sister-in-law, for their endless help, support and love without which none of this would have been possible. I dedicate this thesis to my whole family.
Chapter 1
Introduction

In recent years there has been a lot of interest in numerical treatment of differential algebraic equations (DAEs). DAEs occur in some modelling problems in which the state variables (representing the state of the system) are related by both differential equations and algebraic constraints (e.g. the simulation of electrical networks, the solution of certain equations in fluid dynamics, and the simulation of mechanical systems). They are often characterized by their index (see discussion in Section 3.1). In this thesis, we present and study an algorithm used to solve linear boundary value problems (BVPs) for DAEs of (global) index 1.

In general, a system of ordinary differential equations (ODE) can be written as

\[ F(x, x', t) = 0 \]  

\( a \leq t \leq b. \) Here \( x \in \mathbb{R}^n \) and \( x' \in \mathbb{R}^n. \) A special case is when (1.1) is linear

\[ E(t)x' = A(t)x + q(t) \]  

(1.2)
CHAPTER 1. INTRODUCTION

If $E(t)$ is non-singular, then (1.2) can be written as an explicit ODE

$$x' = E^{-1}(t)A(t)x + E^{-1}(t)q(t), \quad a \leq t \leq b$$

Likewise, (1.1) can be converted, at least in principle, to an explicit ODE form when $J(t)$ is nonsingular on $a < t < b$, where

$$J(t) = \frac{\partial F(x(t), x'(t), t)}{\partial x'}$$

(1.3)

However this conversion may not always be simple or possible in practice. Equation (1.1) is related to (1.2) in the sense that at each iteration of a quasilinearization of (1.1) we have a linear DAE of the form (1.2) with $E(t) = J(t)$.

When $E(t)$ is singular, it means that the differential equations are mixed together with algebraic constraints. It is the so-called DAE.

A simple remark is that any ODE is a DAE. But not every DAE is an ODE. A simple example is as follows: [11]

Example 1.1

$$x_1 = g(t) \quad x'_1 - x_2 = 0 \quad x'_2 - x_3 = 0$$

where $a \leq t \leq b$. Obviously the solution is $x_1 = g(t)$, $x_2 = g'(t)$ and $x_3 = g''(t)$.

However, if initial values are given at $x(a) = [x_1(a), x_2(a), x_3(a)]$, then one has to make
sure that \( x(a) \) is compatible with \([g(a), g'(a), g''(a)]\), or else there is no solution. This simple example shows how easily one can get into trouble when one does not look at DAEs carefully.

Much effort has been put in studying DAE for initial value problems (IVPs); see for instance [11], [7], [17] and [12]. Gear [7] was the first to propose using BDF (Backward Differentiating Formulas) to solve DAEs. März [15] shows that BDF methods converge if \( J(t) \) of (1.3) has constant rank and the index is one. It is proved in [20] that if a \( k \)-step, with \( k < 7 \), constant stepsize BDF method is applied to a linear constant coefficient DAE (with index \( m \)) the solution is \( O(h^k) \) accurate globally after a maximum of \( (m - 1)k + 1 \) steps. But BDF methods may fail when the step size is not constant [10]. Further, and more importantly, when BDF methods are applied to more general higher index problems, instability could occur, unless certain restrictive conditions hold [19], [4], [1], [9].

Petzold [18] studies order and stability properties for implicit Runge-Kutta methods applied to index 1 linear constant coefficient DAEs. There is a reduction of order under some conditions. This order reduction effect also occurs for some of the implicit Runge-Kutta methods in the stiff ODE literature. Kvaerno [13] has extended Petzold’s work for implicit Runge-Kutta methods for index 1 problems in which \( J(t) \) has a constant rank.

Campbell [5] gives a very interesting algorithm applied to general linear DAEs
(1.2). However this method involves differentiating the coefficient matrices and therefore is impractical for certain problems.

The existing theory is well-developed for index 1 DAE in IVP and a general purpose code DASSL (by Petzold [16]) is available. But less work has been done on BVP. The theory is less satisfactory, and no general purpose code has been developed for even index 1 problems, let alone for higher index ones. It is, therefore, the purpose of this thesis to present an algorithm to solve linear index 1 BVPs using symmetric schemes. Symmetric schemes are used because, as is well-known in numerical BVP literature, they weakly preserve dichotomy, whereas those one-sided schemes do not (see, e.g. [3]). The analysis is based on Ascher [1], [2]. It was shown that if symmetric schemes are used for linear index 1 DAEs, the stability is controlled by the stability of an auxiliary (ghost) ODE problem which is not necessarily stable even if the given problem is. The algorithm we study attempts to preserve the stability of the ghost problem by arranging the auxiliary boundary conditions properly.

In Chapter 2 we provide the background material necessary for understanding the relevant theory of BVP. The ideas of well-conditioning and dichotomy are introduced. It also discusses using eigenvalues to approximate kinematic eigenvalues. This discussion is important since it constitutes a crucial step in the proposed algorithm of this thesis. In Chapter 3 we discuss the theory for using symmetric schemes in solving linear index 1 DAEs. In Chapter 4 we give the details of the proposed algorithm. In Chapter 5
experimental results are presented based on the algorithm in Chapter 4, and in the last chapter we draw conclusions and suggest potential future improvement.
In this chapter we study the concept of well-conditioning for linear BVPs. The concept of dichotomy is introduced as it plays a crucial role in the study. In particular, the relationship between dichotomy and kinematic eigenvalues (and also Lyapunov equation) is investigated. Since it is very difficult to find the kinematic eigenvalues in practice, usual eigenvalue analysis is used to approximate the kinematic eigenvalues. A crucial step for the algorithm proposed in Chapter 4 uses eigenvalue analysis to study the behaviour of the difference scheme, which involves a ghost problem. If the ghost problem is found not to be well-conditioned, the proposed algorithm will quit. Therefore it is very important to investigate how good this approximation is.

Consider the following BVP

\[ y' = A(t)y + q(t) \quad a \leq t \leq b \quad (2.1) \]

\[ B_a y(a) + B_b y(b) = \beta \quad (2.2) \]
where \( y(t), q(t) \in \mathcal{R}^n \), and \( A(t) \in \mathcal{R}^{n \times n} \). Intuitively the BVP (2.1), (2.2) is \textit{well-conditioned} if a small change in \( q(t) \) and \( \beta \) should produce a “small” change in the solution \( y \). In contrast, it is \textit{ill-conditioned} if a small change produces a “large” change in \( y \). It does not make too much sense to solve a very ill-conditioned problem because starting with inexact data, the solution will be arbitrarily inexact no matter how it is computed. (Remark: an algorithm is stable if it yields a solution that is the exact solution of a slightly perturbed problem). It is the purpose of this chapter to further quantify and characterize this important idea.

Assume that

\[
\max(||B_a||, ||B_b||) = 1
\]

The norm || · || used in this chapter is assumed to be a Hölder norm. The induced matrix norm will also be denoted by || · ||. Furthermore, assume that the BVP (2.1) and (2.2) has a unique solution \( y(t) \). This solution can be written as

\[
y(t) = Y(t)B^{-1} \beta + \int_a^b G(t, s)q(s)ds
\]  

(2.3)

\[
B = B_a Y(a) + B_b Y(b)
\]  

(2.4)

\[
G(t, s) = \begin{cases} 
Y(t)B_a Y(a)Y^{-1}(s) & \text{if } s \leq t \\
-Y(t)B_b Y(b)Y^{-1}(s) & \text{if } s > t
\end{cases}
\]  

(2.5)

where \( Y(t) \) is a fundamental solution and \( G(t, s) \) is the \textit{Green's function}. Taking the norm of (2.3) we obtain

\[
||y||_\infty \leq \kappa_1 ||\beta||_\infty + \kappa_2 ||q||_p
\]  

(2.6)
where

\[ \kappa_1 = \|YQ^{-1}\|_\infty \]
\[ \kappa_2 = \sup_x (\int_a^b ||G(t,s)||^q dt)^{1/q}, \quad 1/p + 1/q = 1 \]

The conditioning constant is defined as

\[ \kappa = \max\{\kappa_1, \kappa_2\} \]

Equation (2.6) becomes

\[ ||y||_\infty \leq \kappa (||\beta||_\infty + ||q||_p) \]  (2.7)

One can easily see from (2.7) that \( \kappa \) provides a bound on the perturbations of \( \beta \) and \( q \), i.e. it measures how sensitive the change of \( y(t) \) with respect to the perturbation of \( \beta \) and \( q(s) \) is. The linear BVP (2.1), (2.2) is said to be well-conditioned if \( \kappa \) is a constant of moderate size. One can show that \( \kappa_1 \leq 2\kappa_2 \). (see [3], Chapter 3).

Actually, for a well-conditioned BVP, the ODE must have a dichotomy [3].

**Definition 2.1 (Dichotomy)** Let \( Y(t) \) be the fundamental solution of

\[ y' = A(t)y \]  (2.8)

The ODE (2.8) has an exponential dichotomy if there exist a constant orthogonal projection matrix \( P \in \mathbb{R}^{n \times n} \) of rank \( p \), \( 0 \leq p \leq n \), and positive constants \( K, \lambda, \mu \) with \( K \) of “moderate” size, such that

\[ ||Y(t)PY^{-1}(s)|| \leq Ke^{-\lambda(t-s)} \quad \text{for} \; t \geq s \]  (2.9)
CHAPTER 2. BACKGROUND MATERIAL FOR BVP

\[ ||Y(t)PY^{-1}(s)|| \leq Ke^{-\mu(t-s)} \quad \text{for } s \leq t \]  \hspace{0.5cm} (2.10)

for \( a \leq t, s \leq b \). It has an ordinary dichotomy if \( \lambda = 0 \) or \( \mu = 0 \) in (2.9) or (2.10).

It is said to be dichotomic if it has either an ordinary dichotomy or an exponential dichotomy.

Moreover, the ideas of well-conditioning, dichotomy and boundary conditions are closely related with each other. A particularly simple, but important, case is when the boundary conditions (2.2) are separated, i.e.

\[ \begin{align*}
B_a y(a) &= \beta_0 \\
B_b y(b) &= \beta_1
\end{align*} \quad (2.12)\]

where \( B_a \in \mathcal{R}^{m \times n} \), \( B_b \in \mathcal{R}^{(n-m) \times n} \). If the BVP is well-conditioned, then it must have a dichotomy. The BC \( B_a y(a) = \beta_0 \) and \( B_b y(b) = \beta_1 \) must control the nonincreasing modes and nondecreasing modes, respectively. This idea is stated formally in the following theorem.

**Theorem 2.1** Suppose the BVP (2.1) with separated BC (2.12) is well-conditioned. Let \( Y(t) \) be any fundamental solution, \( B \) be as defined in (2.4), \( I_p \) be a \( p \times p \) identity matrix, and \( P \) be a projection matrix where

\[ P := B_a Y(a)B^{-1} = \begin{bmatrix} I_p & 0 \\ 0 & 0 \end{bmatrix} \]

Then
(a) $m = p$;

(b) $\text{range}(B^T_{a1})^\perp \cap \text{range}(Y(a)P) = \{0\}$, i.e., no nonzero vector in $\text{range}(Y(a)P)$ is orthogonal to $\text{range}(B^T_{a1})$;

(c) $\text{range}(B^T_{b2})^\perp \cap \text{range}(Y(b)(I - P)) = \{0\}$;

(d) If the system has a dichotomy with $(b - a)\min\{\lambda, \mu\} \to \infty$, then the rank of the projection matrix must be $m$.

**Proof:** (see [3], Chapter 3, Theorem 107.).

This theorem implies that the number of BC's at $t = a$ must be at least the number of rapidly decreasing modes. Likewise the number of BC's at $t = b$ must be at least the number of rapidly increasing modes. One of the crucial steps in the proposed algorithm in Chapter 4 is based on this theorem, especially (b) and (c).

However it is impractical to use Definition (2.1) to check for dichotomy and then conclude well-conditioning, since we have to compute a fundamental solution. It is natural to ask if we can characterize the ideas of dichotomy and stability involving $A(t)$. Consider the ODE (2.8). Suppose that there exists a transformation matrix $Q(t)$, $Q(t)$ is differentiable and $||Q(t)||||Q^{-1}(s)||$ is uniformly bounded for $t \geq s$, such that $\mathcal{V}(t)$ is upper-triangular, where

$$\mathcal{V}(t) := Q^{-1}(t)(A(t)Q(t) - Q'(t))$$

(2.13)
CHAPTER 2. BACKGROUND MATERIAL FOR BVP

Let

\[ w(t) = Q^{-1}(t)y(t) \]  

(2.14)

Substituting (2.14) into (2.8) (note: \( y' = Q'(t)w(t) + Q(t)w'(t) \)), we obtain a new ODE

\[ w' = V(t)w \]

(2.15)

Let \( W(t) \) be the fundamental solution for (2.15) satisfying

\[ W(a) = I \]

\[ W := \begin{bmatrix} W^{11} & W^{12} \\ W^{21} & W^{22} \end{bmatrix} \]  

(2.16)

where \( W^{11} \in \mathbb{R}^{p \times p}, W^{12} \in \mathbb{R}^{p \times n}, W^{21} \in \mathbb{R}^{(n-p) \times p}, \) and \( W^{22} \in \mathbb{R}^{(n-p) \times (n-p)} \). Then one can show that \( W(t) = Q^{-1}Y(t) \) and \( W(t) \) is block upper-triangular, i.e. \( W^{21} = 0 \).

Moreover, both \( W^{11} \) and \( W^{22} \) are nonsingular.

**Definition 2.2 (Kinematic Eigenvalues)** The system (2.8) and (2.15) are kinematically similar. The diagonal elements of \( V(t) \) are called the kinematic eigenvalues corresponding to \( Q(t) \).

Equation (2.13) is the famous Lyapunov equation. The following theorem relates kinematic eigenvalues with dichotomy.

**Theorem 2.2** Consider the ODE (2.8). Suppose that there exists a transformation matrix \( Q(t) \) satisfying (2.13), (2.14) and (2.15). Denote the kinematic eigenvalues by \( \lambda_1, \lambda_2, \ldots, \lambda_n \). Then (2.8) has a dichotomy iff there exist positive constants \( c \) and \( \hat{\lambda} \) such
that for \( p \) kinematic eigenvalues

\[
\text{Re}(\int_s^t \lambda_i(\tau)\,d\tau) < -\dot{\lambda}(t-s) \quad \text{if } t-s > c
\]

(2.17)

and for the other \( n - p \) kinematic eigenvalues

\[
\text{Re}(\int_s^t \lambda_i(\tau)\,d\tau) > \dot{\lambda}(t-s) \quad \text{if } t-s > c
\]

(2.18)

Proof: (see [3], Chapter 3, Theorem 101).

Equations (2.17) and (2.18) imply that there are \( p \) nonincreasing modes and \( n - p \) non-decreasing modes. The transformation matrix \( Q(t) \) actually decouples the system (2.8) into two modes, namely nonincreasing modes and nondecreasing modes. The diagonal elements of \( V(t) \) (i.e. the kinematic values) can be used to determine if the system has a dichotomy. Let

\[
V(t) = Q^T(t)A(t)Q(t)
\]

(2.19)

where \( V(t) \) is upper triangular, and its diagonal elements are the eigenvalues of \( A(t) \).

A simple observation on (2.13) is that \( Q'(t) = 0 \) if \( A(t) \) is a constant matrix. The eigenvalues are the same in this case as the kinematic eigenvalues. One could then ask if it is always possible to use eigenvalues to approximate the kinematic eigenvalues for a general \( A(t) \). The answer is negative. We demonstrate this by a simple example.

**Example 2.1**

\[
A(t) = \begin{bmatrix}
\beta \cos(2\alpha t) & \beta \sin(2\alpha t) + \alpha \\
\beta \sin(2\alpha t) - \alpha & -\beta \cos(2\alpha t)
\end{bmatrix}
\]

(2.20)
With the following transformation matrix $Q(t)$ we obtain the kinematic eigenvalues

$$Q = \begin{bmatrix} \cos(\alpha t) & \sin(\alpha t) \\ -\sin(\alpha t) & \cos(\alpha t) \end{bmatrix} \quad V = \begin{bmatrix} \beta & 0 \\ 0 & -\beta \end{bmatrix}$$

(2.21)

However a straightforward computation shows that the eigenvalues of (2.20) are

$$\pm \sqrt{\beta^2 - \alpha^2}$$

(2.22)

The eigenvalues are only close to the kinematic eigenvalues when $|\alpha|$ is small compared to $|\beta|$. The example is pathological because a large $|\alpha|$ indicates that modes rotate fast. This is rare in practice.

It should be noted that often it is the number of nonincreasing modes and the number of nondecreasing modes we are interested in, and not the exact magnitude. Even though eigenvalue analysis may fail, it is the only simple tool we have to approximate the kinematic eigenvalues. Besides if $A(t)$ does not change rapidly it is often good enough.

### 2.1 Projecting Boundary Conditions

In this section we make use of the material presented in this chapter to answer a question we face in Chapter 4 (when performing the projection in Section 4.2). The question is as follows: Consider the BVP (2.1) with separated BC (2.12). Let $V$, $Q$, $W$ and $w$ be defined as in (2.13), (2.14), (2.15) and (2.16). Assuming that the ODE has a dichotomy, there are $n^-$ nonincreasing modes and $n^+$ nondecreasing modes ($n^- + n^+ = n$). Furthermore,
the real parts of kinematic eigenvalues of $V$ are arranged from large negative to large positive. Suppose that we are given $n_a$ BC's at $t = a$ and $n_b$ BC's at $t = b$, with $n_a \geq n^-$, $n_b \geq n^+$ and $n_a + n_b > n$. The BC are consistent, i.e. there is a unique solution. However the numerical algorithm used only requires $n$ BC's. We have more BC's than we need. How should we choose $n$ BC's from these $n_a + n_b$ BC's so as to obtain a well-conditioned BVP if possible?

From Theorem 2.1, we know that there should be $n^-$ BC's at $t = a$ and $n^+$ BC's at $t = b$. It implies that $n^-$ BC's and $n^+$ BC's should be chosen from $B_{a1}$ and from $B_{b2}$, respectively (cf. (2.12)). Denote these $n^-$ BC's at $t = a$ and $n^+$ BC's at $t = b$ as

$$\hat{B}_{a1} y(a) = \hat{\beta}_0 \quad \hat{B}_{b2} y(b) = \hat{\beta}_1$$

(2.23)

Then

$$\hat{B}_{a1} Q^{-1}(a) y(a) = \hat{\beta}_0 \quad \hat{B}_{b2} Q^{-1}(b) y(b) = \hat{\beta}_1$$

$$\hat{B}_{a1} := \hat{B}_{a1} Q(a) =: [\hat{B}_{a1}^1, \hat{B}_{a1}^2] \quad \hat{B}_{b2} := \hat{B}_{b2} Q(b) =: [\hat{B}_{b2}^1, \hat{B}_{b2}^2]$$

(2.24)

where $\hat{B}_{a1} \in \mathbb{R}^{n^- \times n}$, $\hat{B}_{b2} \in \mathbb{R}^{n^+ \times n}$, $\hat{B}_{a1}^1 \in \mathbb{R}^{n^- \times n^-}$, and $\hat{B}_{b2}^2 \in \mathbb{R}^{n^+ \times n^+}$. Under what conditions will the BVP (2.1), (2.23) be well-conditioned?

Let $P$ be the projection matrix defined as in Theorem 2.1. By part (b) of Theorem 2.1, we must have $\text{range}(\hat{B}_{a1}^T) \perp \cap \text{range}(Y(a) P) = \{0\}$. Let the vector $z \in \text{range}(Y(a) P)$, i.e. $z = Y(a)[c, 0]^T$ for some $c \in \mathbb{R}^{n^-}$. If also $z \in \text{range}(\hat{B}_{a1}^T) \perp$, then for any $x \in \mathbb{R}^{n^-}$

$$x^T \hat{B}_{a1} Y(a)[c, 0]^T = 0$$
So using \( W(t) = Q^{-1}(t)Y(t) \)

\[
x^T \hat{B}_{a1} Q^{-1}(a)Y(a)[c,0]^T = 0
\]

\[
x^T [\hat{B}_{a1}^1, \hat{B}_{a1}^2]W(a)[c,0]^T = 0
\]

\[
x^T \hat{B}_{a1}^1 W^{11}(a)c = 0
\]

If \( \hat{B}_{a1}^1 \) is nonsingular, then \( \hat{B}_{a1}^1 W^{11}(a) \) is also nonsingular, since \( W^{11} \) is nonsingular. This implies that \( c = 0 \). It means that there does not exist a \( z \neq 0 \) such that \( z \in \text{range}(Y(a)P) \) and also \( z \in \text{range}(\hat{B}_{a1}^T)^\perp \). Hence part (b) of Theorem 2.1 is satisfied. Therefore among the \( n_a \) BC's given by the user at \( t = a \), we should choose a combination of \( n^- \) of them such that \( \hat{B}_{a1}^1 \) is nonsingular.

By part (c) of Theorem 2.1, we also must have that \( \text{range}(\hat{B}_{b2}^T)^\perp \cap \text{range}(Y(b)(I - P)) = \{0\} \). Using the same technique, let the vector \( z \in \text{range}(Y(b)(I - P)) \), i.e. \( z = Y(b)[0, c]^T \) for some \( c \in \mathcal{R}^{n^+} \). If also \( z \in \text{range}(\hat{B}_{b2}^T)^\perp \), then for any \( x \in \mathcal{R}^{n^+} \)

\[
x^T \hat{B}_{b2} Y(b)[0, c]^T = 0
\]

So using \( W(t) = Q^{-1}(t)Y(t) \)

\[
x^T \hat{B}_{b2} Q^{-1}(b)Y(a)[0, c]^T = 0
\]

\[
x^T [\hat{B}_{b2}^1, \hat{B}_{b2}^2]W(b)[0, c]^T = 0
\]

\[
x^T (\hat{B}_{b2}^1 W^{12}(b) + \hat{B}_{b2}^2 W^{22}(b))c = 0
\]
If $\tilde{B}_{b_2}^2$ is nonsingular and $\tilde{B}_{b_2}^1 = 0$, then $(\tilde{B}_{b_2}^1 W^{12}(b) + \tilde{B}_{b_2}^2 W^{22}(b))$ is nonsingular, since $W^{22}$ is nonsingular. This implies that $c = 0$. Hence part (c) of Theorem 2.1 is satisfied.

However, the nonsingularity of $\tilde{B}_{b_2}^2$ in itself is not sufficient to guarantee that $(\tilde{B}_{b_2}^1 W^{12}(b) + \tilde{B}_{b_2}^2 W^{22}(b))$ is nonsingular (so it is a bit different from the previous case).

In practice, one may replace $Q^{-1}$ by $Q^{-1}$ (see (2.19)). The $n^{-}$ BC's at $t = a$ and the $n^{+}$ BC's at $t = b$ are chosen such that $\tilde{B}_{a_1}^1$ and $\tilde{B}_{b_2}^2$ are nonsingular. Of course, $\tilde{B}_{a_1}^1$ and $\tilde{B}_{b_2}^2$ should not be ill-conditioned too. Even though these two conditions may not guarantee that part (c) of Theorem 2.1 holds, it is often good enough in practice.
Chapter 3

Background Material For DAE

In this chapter we review some fundamental concepts of DAE. In Section 3.1 we discuss the classification of DAEs. In Section 3.2 we consider the use of symmetric schemes for linear index 1 DAE. It will be shown that the stability of the discretization is controlled by a ghost problem.

3.1 Classification of DAEs

Not all DAEs of the form (1.1) can be solved. Some of them may have no solution or infinitely many solutions. (Of course, it depends on the BC too). Actually it is not reasonable to try to directly solve those DAEs with either no solution or infinitely many solutions numerically. Therefore we need a definition to classify the solvability of DAEs. If a DAE is solvable, we further categorize it based on its structure. The latter is the notion of index. The following definition is due to Campbell, [14]
Definition 3.1 (Solvability) The DAE (1.1) is solvable on a subset $\Omega$ of $\mathbb{R}^{n+1}$ if there is an $r$-parameter family of solutions $x(t,c)$ (defined for $t \in \mathcal{R}$, $c \in \mathcal{R}^r$) such that (i) $(t,x(t,c)) \in \Omega$, and (ii) if $\hat{x}(t)$ is any solution whose graph lies in $\Omega$, then $\hat{x} = x(t,\hat{c})$ for some $\hat{c}$, and (iii) the graph $x(t,c)$ is an $(r+1)$-dimensional manifold.

We generally take $\Omega = [t_0,T] \times \mathbb{R}^n$. The definition guarantees that there are solutions, and that they are uniquely defined by their value at $t = t_0$.

Having defined solvability of a DAE, we could further categorize its structure with a quantity called its index. This notion of index has been defined in many ways, for a local index and a global index. We are primarily interested in the global index in this thesis and often we just write "index" for "global index". The definition given below is due to Gear, [8].

Definition 3.2 (Index) The global index, $m$, of a solvable DAE (1.1) with $F$ sufficiently smooth is defined as follows:

$m = 0$: if $J(t)$ (as defined in (1.3)) is nonsingular.

$m > 0$: otherwise, consider the system of equations

\[
F(x,x',t) = 0 \\
\frac{\partial F}{\partial t} = \frac{\partial F}{\partial x'} x^{(2)} + \frac{\partial F}{\partial x} x' + \frac{\partial F}{\partial t} = 0 \\
\frac{\partial^2 F}{\partial t^2} = \frac{\partial F}{\partial x'} x^{(3)} + \cdots = 0 \\
\cdots
\]
\[ \frac{\partial^s F}{\partial t^s} = \frac{\partial F}{\partial x'} x^{(s+1)} + \cdots = 0 \]

as a system of equations with dependent variables \( x', x^{(2)}, \ldots, x^{(s+1)} \) and with independent variables \( x \) and \( t \). The global index, \( m \), is defined as the smallest \( s \) such that it is possible to solve for \( x' \) in terms of \( x \) and \( t \).

It may suffice to give some simple examples in here.

**Example 3.1** Any explicit ODE is a DAE with index 0.

**Example 3.2** Example (1.1) is an index 3 DAE.

\[
\begin{align*}
    x_1 &= g \\ 
    x_1' &= x_2 \\ 
    x_2' &= x_3 \\
    x_1' &= g' \\ 
    x_2^{(2)} &= x_2' \\ 
    x_3^{(2)} &= x_3' \\
    x_2^{(2)} &= g^{(2)} \\ 
    x_2^{(3)} &= x_2^{(2)} \\
    x_3^{(3)} &= g^{(3)}
\end{align*}
\]

It implies that \([x_1', x_2', x_3'] = [x_2, x_3, g^{(3)}]\). Since three steps of differentiation are required, it is of index 3.

It should be mentioned that, in principle, it is possible to use analytical differentiation techniques to rewrite a DAE from a higher index to a lower index. Intuitively, it is that by differentiating the algebraic equations, we can solve for \( x' \) in terms of \( x \) and \( t \). However, each differentiation will introduce additional constants of integration. Gear [8] discusses
an interesting algorithm for reducing higher index problems to lower index ones without introducing any constants of integration. In practice, such a reduction may not be feasible. But it helps to understand the underlying analytical structure of the problem.

Sometimes it may be useful to look at some special forms of (1.1). A DAE is in semi-explicit form if

\[ x_1' = f(x_1, x_2, t) \]  \hspace{1cm} (3.1)

\[ 0 = g(x_1, x_2, t) \]  \hspace{1cm} (3.2)

It is in “triangular” form if

\[ x_1' = f(x_1, x_2, t) \]  \hspace{1cm} (3.3)

\[ 0 = g(x_1, t) \]  \hspace{1cm} (3.4)

It is in linear time-varying form if it can be written as (1.2). When both \( E(t) \) and \( A(t) \) of (1.2) are constant matrices, then we have a linear constant coefficient form.

\[ E x' = A x + f(t) \]  \hspace{1cm} (3.5)

### 3.2 Symmetric Schemes for DAEs in BVPs

In this section we study in detail the stability issues arising when using symmetric schemes for DAEs in BVPs. Ascher in [1] and [2] shows that there is a potential risk in using symmetric schemes on general linear index 1 DAEs (1.2). If the DAE is not in semi-explicit
form, then the stability of the discretization is controlled by a ghost problem. Therefore the stability constant of the discretization method depends on the conditioning constant of this ghost problem. For the numerical solution of index 1 IVPs, there is a better alternative, namely BDF methods. However symmetric schemes are more appropriate for BVPs, since a well-conditioned problem may have both fast nonincreasing and fast nondecreasing modes. Symmetric schemes preserve dichotomy in a weak form (see [3, Ch.3,10) and BDF methods do not.

Let us study a simple example to show the potential danger of using symmetric schemes.

**Example 3.3** Consider a general linear DAE of the form (1.2) where

\[
E(t) = \begin{bmatrix} 0 & 0 \\ -1 & t \end{bmatrix} \quad A(t) = \begin{bmatrix} -\beta & \beta t + 1 \\ 1 & -(t+1) \end{bmatrix} \quad q(t) = \begin{bmatrix} \cos(t) \\ 0 \end{bmatrix}
\]

where \(\beta\) is a parameter, \(0 < t < 1\), and \(x_1(0) = -1\). The exact solution is

\[
x_1 = -(1 + \beta t)e^{-t} - t\cos(t) \quad x_2 = -\beta e^{-t} - \cos(t)
\]

We apply the midpoint scheme

\[
E(t_{i+1/2}) \frac{x_{i+1} - x_i}{h_i} = A(t_{i+1/2}) \frac{x_{i+1} - x_i}{h_i} + q(t_{i+1/2}), \quad 1 \leq i \leq N
\]

on a mesh

\[
\pi : \ 0 = t_1 < t_2 < \cdots < t_N < t_{N+1} = 1
\]

\[
h_i := t_{i+1} - t_i, \quad h := \max_{1 \leq i \leq N} h_i, \quad t_{i+1/2} := t_i + \frac{1}{2} h_i.
\]
Here we choose a uniform mesh (\( h = 1/N \)). In order to use the above midpoint scheme, we need another BC. Consider the following two cases:

Case A:

\[
 x_2(0) = -\beta - 1
\]

Case B:

\[
 -\beta x_1(1) + (1 + \beta)x_2(1) = -\cos(1)
\]

Table 3.1 and Table 3.2 show the errors and the rate of convergence of \( x \) of Case A and Case B, respectively. \( \beta \) is chosen to be 10.0. The errors computed are the maximum absolute errors. One can clearly see that errors in Table 3.1 are very large, while the errors in Table 3.2 are relatively small. The large error is not due to roundoff error accumulation, as the computed rates of convergence indicate. One can see easily from these two tables that if the extra BC is not posed properly, large errors could occur.
3.2.1 Analysis

In this section we recall an analysis [2] of the stability issue of using symmetric schemes for a general index 1 linear DAE (1.2). Given a DAE (1.2), assume that $E(t)$ can be written as

$$E(t) = S(t) \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} T^{-1}(t)$$  \hspace{1cm} (3.9)

where both $S(t)$ and $T(t)$ are smooth for all $t$, $a \leq t \leq b$. With the following transformation

$$\begin{bmatrix} y \\ z \end{bmatrix} = \hat{x} = T^{-1}x$$  \hspace{1cm} (3.10)

(1.2) can be transformed to *semi-explicit* form

$$0 = U^{11}(t)y + U^{12}(t)z + g^1(t)$$  \hspace{1cm} (3.11)

$$z' = U^{21}(t)y + U^{22}(t)z + g^2(t)$$  \hspace{1cm} (3.12)

where

$$\begin{bmatrix} U^{11} & U^{12} \\ U^{21} & U^{22} \end{bmatrix} = U = S^{-1}AT - \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} T^{-1}T'$$  \hspace{1cm} (3.13)
In (3.11) there are \( n_y \) equations and in (3.12) there are \( n_z \) equations, \( n_y + n_z = n \). The general linear DAE (1.2) has index 1 if \( U^{11}(t) \) is nonsingular for all \( t \in [a, b] \). Using (3.11) we then have

\[
y(t) = -[U^{11}(t)]^{-1}[U^{12}(t)z(t) + g^1(t)]
\]  

(3.14)

By substituting (3.14) into (3.12), we obtain an explicit ODE with \( z \) only. Therefore we actually require only \( n_z \) BC’s for \( z \). However unless a transformation like (3.10) is used we must require another \( n_y \) BC’s when applying a numerical method to solve (1.2) so as to fully specify the BC’s required for \( x \).

If a midpoint scheme is used for (1.2) then we are actually approximating \( y(t) \) in a wrong space, in the sense that we force \( y(t) \) to be as smooth as \( z(t) \). A result of this is that the stability of the scheme depends on the stability of a ghost ODE

\[
\dot{y} = -M(t)\hat{y}
\]  

(3.15)

where

\[
M = [U^{11}]^{-1}(S^{-1}AT')^{11} \in \mathcal{R}^{n_y \times n_y}
\]  

(3.16)

(if \( U^{12} = 0 \) then simply \( M = (T^{-1}T')^{11} \)) (see [1] and [2] for details). The new function \( \hat{y} \) comes from the discretization of \( y \). Though the DAE may be well-conditioned (which we will assume), the stability constant of the discretization, which is controlled by the conditioning constant of the above ghost ODE subject to the BC used, may not be small.
CHAPTER 3. BACKGROUND MATERIAL FOR DAE

at all. If the BC’s for \( y \) are not properly posed with respect to the ghost problem, instability will arise. One simple observation is that it is better to transform the DAE (1.2) to semi-explicit form (3.11), (3.12) first because \( M = 0 \) in that case (as \( T' = 0 \)). However this explicit transformation may not be available or feasible in practice.

Using Example 3.3 one can show that with the following transformation

\[
S = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}, \quad T^{-1} = \begin{bmatrix}
t & -1 \\
1 & 0
\end{bmatrix}
\]

The ghost problem is \( \dot{y}' = \beta \dot{y} \), i.e. \( M = -\beta \). Obviously this symmetric scheme will have a large stability constant if \( \beta \) is large and positive, and the BC for \( y \) is put at the left hand side (see Table 3.1), but it has a moderate stability constant if the BC for \( y \) is put at the right hand side (see Table 3.2). In the next chapter we give an algorithm to arrange the BC’s for \( y \) automatically (in fact the BC used for Table 3.2 has been generated by the proposed algorithm).

Remarks:

The decomposition (3.9) is not unique. Given \( T \) and \( S \) satisfying (3.9), we can generally write

\[
E(t) = \dot{S}(t) \begin{bmatrix}
0 & 0 \\
0 & I
\end{bmatrix} \dot{T}^{-1}(t)
\]

\[
S(t) = S(t) \begin{bmatrix}
I \\
R_S(t)
\end{bmatrix} \begin{bmatrix}
F_S(t) & 0 \\
0 & G(t)
\end{bmatrix} \begin{bmatrix}
I & 0 \\
R_T(t) & I
\end{bmatrix} \begin{bmatrix}
F_T(t) & 0 \\
0 & G(t)
\end{bmatrix}
\]

Here \( F_S, F_T \in \mathcal{R}^n_y \times n_y \) and \( G(t) \in \mathcal{R}^n_x \times n_z \) are sufficiently smooth non-singular scaling matrices. The rectangular matrices \( R_T(t) \in \mathcal{R}^n_y \times n_z \) and \( R_S(t) \in \mathcal{R}^n_x \times n_y \) can be chosen
to obtain certain special structure of $U$ in (3.13) (see [1]). With the above transformation matrices $\hat{T}(x)$ and $\hat{S}(x)$ we can show that the ghost ODE is

$$\dot{y} = -\tilde{M}(t)y$$  \hspace{1cm} (3.19)$$

$$\tilde{M} = F_T^{-1} M F_T + F_T^{-1} F_T'$$  \hspace{1cm} (3.20)$$

where $M$ is defined as (3.16). The change from the old to the new ghost ODE only depends on $F_T$. If the fundamental solution of the original ghost ODE, (3.15), is $Y_{old}$, then the fundamental solution of the new ghost ODE, (3.19), is $Y_{new} = F_T Y_{old}$. Let us define a transformation $Q(t)$ such that

$$-M = QVQ^{-1} + Q'Q^{-1}$$  \hspace{1cm} (3.21)$$

where $V$ is upper-triangular and so its diagonal elements are the kinematic eigenvalues of the ghost ODE. Then we can show that

$$-\tilde{M} = \tilde{Q}V\tilde{Q}^{-1} + \tilde{Q}'\tilde{Q}^{-1}$$  \hspace{1cm} (3.22)$$

where

$$\mathcal{Q} = F_T^{-1} Q$$  \hspace{1cm} (3.23)$$

This means that the kinematic eigenvalues are the same no matter how $T$ is changed. It is very important from the perspective of whether scaling will affect the magnitude and the sign of kinematic eigenvalues, and the answer is negative. Of course, this conclusion
is based on the assumption that $F_T$ is smooth and nonsingular within the interval of integration.
Chapter 4

Algorithm (linear index 1 DAEs in BVPs)

In the last chapter we have seen the danger of using symmetric schemes in solving linear index 1 DAEs. The problem is that the algebraic part, \( y \), is approximated in the wrong space, and the ghost problem generated may not have a small conditioning constant. If the boundary conditions with respect to the algebraic part, \( y \), are not posed properly, large stability constants may occur. In this chapter we propose an algorithm to arrange the boundary conditions for the algebraic part so as to preserve the well-conditioning, so long as the ghost ODE has a dichotomy. The main skeleton of the algorithm is presented first, followed by details of each step.

Consider the DAE (1.2) subject to the consistent separated BC

\[
B_a x(a) = \beta_0 \quad B_b x(b) = \beta_1
\]  

(4.1)
CHAPTER 4. ALGORITHM (LINEAR INDEX 1 DAES IN BVPS)

ALGORITHM

Step 1: Find a $T$ and a $T'$ satisfying (3.9) at the two interval ends. Using $T$ we may now isolate the solution components $y$ and $z$, and form $U^{11}, U^{12}, g^1$ at each interval end.

Step 2: Using the relation (3.14), obtain from (4.1) a set of BC's for $z$ alone. Denote them by

$$
\mathcal{B}_a z(a) = \beta_0 \quad \mathcal{B}_b z(b) = \beta_1
$$

(4.2)

If there are more than $n_x$ BC's, then they may be projected appropriately. That is, substitute (3.14) into (3.12) (only at $t = a, b$), and obtain an explicit differential equation on $z$ alone, denote it as

$$
z' = H(t)z + h(t)
$$

(4.3)

Form the matrices $H(a)$ and $H(b)$, and analyze the eigenvalues of $H$ using the QR algorithm. Let

$$
V(t) = Q^T(t)H(t)Q(t) \quad \text{at } t = a, b
$$

where $Q$ is orthogonal and $V$ is upper triangular with the eigenvalues arranged in increasing order of real parts, from large negative to large positive. Let $n_a^-, n_b^+$ be the number of eigenvalues of $H(a)$ which have a large negative (positive) real part, and similarly define $n_a^+, n_b^-$ for $H(b)$.

If $n_a^- + n_b^+ > n_x$

(a) no projection is performed, and just pick arbitrary $n_a^-$ BC's for $t = a$, and $n_x - n_a^-$ BC's for $t = b$;

else

(b) write the first part of (4.2) as follows:

$$
\mathcal{B}_a Q^T z(a) = \beta_0 \\
\mathcal{B}_a := \mathcal{B}_a Q
$$

(4.4)

(c) set $k := \max(n_a^-, n_b^+)$;

(d) pick $k$ linearly independent BC's from $\mathcal{B}_a$, denote these BC's as $\mathcal{B}_a$

$$
\mathcal{B}_a Q^T z(a) = \beta_0 \\
\mathcal{B}_a := [\mathcal{B}_{a1}, \mathcal{B}_{a2}]
$$

(4.5)

where $\mathcal{B}_{a1} \in \mathbb{R}^{k \times k}$, and $\mathcal{B}_{a2} \in \mathbb{R}^{k \times (n_x - k)}$

(e) set the BC (4.5) at $t = a$;

(f) perform the same process for $\mathcal{B}_b$:

pick $n_x - k$ linearly independent BC's from $\mathcal{B}_b$, denote these BC's as $\mathcal{B}_b$

$$
\mathcal{B}_b Q^T z(b) = \beta_1 \\
\mathcal{B}_b := [\mathcal{B}_{b1}, \mathcal{B}_{b2}]
$$

(4.6)

where $\mathcal{B}_{b1} \in \mathbb{R}^{(n_x - k) \times k}$, and $\mathcal{B}_{b2} \in \mathbb{R}^{(n_x - k) \times (n_x - k)}$

(b) set the BC (4.6) at $t = b$;

End if
CHAPTER 4. ALGORITHM (LINEAR INDEX 1 DAES IN BVPS)

Step 3: Form the matrices $M(a)$ and $M(b)$ by (3.16), and analyze the eigenvalues using the QR algorithm. Let

\[ V(t) = Q^T(t)M(t)Q(t) \quad \text{at } t = a, b \]

where $Q$ is orthogonal and $V$ is upper triangular with the eigenvalues arranged in increasing order of real parts, from large negative to large positive. Let $n^-_a$ ($n^+_a$) be the number of eigenvalues of $M(a)$ which have a large negative (positive) real part, and similarly define $n^-_b$, $n^+_b$ for $M(b)$.

If $n^-_a + n^+_b > n_y$ or $n^-_b + n^+_a > n_y$ then

(a) another method (e.g. transforming explicitly to (3.11), (3.12) everywhere first) should be used;
(because there may not be a dichotomy for $y$)

else

(b) set $k := \max(n^-_a, n^-_b)$;
(c) set the last $n_y - k$ components of $Q^Ty$ at $t = a$ according to (3.14);
(d) set the first $k$ components of $Q^Ty$ at $t = b$ according to (3.14);
(e) transform back to obtain BC for $x(a)$ and $x(b)$.

end if

Step 4: Solve the discretized equations (3.7) with the obtained BC from Step 2 and Step 3.

This completes specification of the BC for the application of the midpoint scheme (3.7).

4.1 Details of Step 1

Step 1 is performed by the singular value decomposition (SVD). Let $E(a) = USV^T$, where $\Sigma$ is a diagonal matrix and its elements are the singular values of $E(a)$. Denote these singular values, which are above a threshold $t_0 > 0$, by $\sigma_1, \sigma_2, \ldots, \sigma_{n_z}$. In general, $n_z \neq n$ and $n_z$ is the dimension of the differential part of the DAE. Then set $S(a) = US$ and $T^{-1} = VT$, where $\hat{\Sigma}$ is the $n \times n$ diagonal matrix with diagonal elements $[\sigma_1, \sigma_2, \ldots, \sigma_{n_z}, 1, \ldots, 1]$. Likewise perform the same steps for $E(b)$.

This is a key step of the whole algorithm, and therefore has to be carried out correctly. We have remarked at the end of Chapter 3 that this decomposition is not unique (cf. (3.17)). Even though the non-uniqueness will not affect the kinematic eigenvalues,
it affects the eigenvalues in the actual implementation. We will discuss this in detail in Section 4.3. The derivative of \( T \) (i.e. \( T' \)) is approximated (e.g. at \( t = a \)) by

\[
T'(a) = \frac{T(a + h) - T(a)}{h}
\]

\( T(a + h) \) and \( T(a) \) are obtained by applying SVD to \( E(a + h) \) and \( E(a) \) respectively. Of course, \( h \) has to be small enough. In the actual implementation, it is chosen to be the square root of the machine epsilon. But we know that \( T \) is not unique, and therefore the two transformations computed at \( a \) and \( a + h \) may not belong to the same \( T(x) \). That is, performing SVD at \( E(a) \) obtains \( T_1(a) \) and performing SVD at \( E(a + h) \) obtains \( T_2(a + h) \), where \( T_1(x) \) may be different from \( T_2(x) \). It does not make sense to approximate \( T' \) using the above formula if \( T_1(x) \neq T_2(x) \). Even though \( h \) is chosen to be very small in the actual implementation, the \( T \) computed is still not unique up to the sign and permutation of the first \( n_y \) columns of \( T \). It is because when we look at (3.9),

\[
\begin{bmatrix}
0 & 0 \\
0 & I \\
\end{bmatrix} T^{-1} = \begin{bmatrix}
0 & 0 \\
0 & I \\
\end{bmatrix} \begin{bmatrix}
T_{11}^{inv} & T_{12}^{inv} \\
T_{21}^{inv} & T_{22}^{inv} \\
\end{bmatrix}
= \begin{bmatrix}
0 & T_{21}^{inv} \\
T_{11}^{inv} & T_{12}^{inv} \\
\end{bmatrix}
\]

One can easily see that \( E \) in (3.9) does not depend on the first \( n_y \) rows of \( T^{-1} \), but the last \( n - n_y \) rows of it are unique. It implies that the first \( n_y \) columns of \( T \) computed are not unique (since the \( T \) computed is orthogonal, by SVD).

This problem is resolved by comparing and rearranging the first \( n_y \) columns of \( T_1(a) \) and \( T_2(a + h) \) so that \( T_1(x) \) is the same as \( T_2(x) \). Given the \( i \)th column of \( T_1(a) \),
look for a column in the first $n_y$ columns of $T_2(a + h)$ such that these two columns are very close to each other. Swap this located column with the $i$th column of $T_2(a + h)$. By doing so, hopefully $T_1(x) = T_2(x)$. The following describes the algorithm in detail.

For $i = 1$ to $n_y$
- pick the $i$th column $T_1(a)$, call it $t_{1;i}$;
- find the maximum element (in absolute value) of $t_{1;i}$, say the $j$th element, denote it as $t_{1;j}^1$;
- compare $t_{1;j}^1$ (in absolute value) with the $j$th elements of column $i$ to column $n_y$ of $T_2(a + h)$: i.e
  If there exists a column $k$ such that
  (1) $|t_{1;j}^1| - |t_{2;j}^2| \leq \epsilon_1$ and
  (2) $t_{2;k} = sign(t_{2;k}^2) sign(t_{1;j}^1) t_{2;k}$
    : then swap the $k$th column of $T_2(a + h)$ with the $i$th column of $T_2(a + h)$;
  Else
    : stop the whole algorithm;
      (because fail to compute $T^i$ properly)
End if
End For

The two parameters $\epsilon_1$ and $\epsilon_2$ are two small positive numbers. Condition (2) guarantees that the square sum of the residual of $t_{1;i} - t_{2;j}$ is less than some small number. However, $t_{1;i}(t)$ may equal to $\pm t_{2;j}(t)$ and therefore we have to consider the sign of $t_{2;j}$.

Similarly, perform the same steps for $T$ at $t = b$. This simple algorithm has performed very well in practice, for all the test cases we have tried.

The theory behind this simple algorithm is as follows [6]: Denote the $i$th column of $T(t)$ as $T^{(i)}(t)$. Since we are using SVD, $T(t)$ is an orthogonal matrix, and $||T^{(i)}(a) - T^{(j)}(a)|| = 2$, $i \neq j$.

$$||T^{(i)}(a) - T^{(j)}(a)|| = ||T^{(i)}(a) - T^{(j)}(a + h) - T^{(j)}(a) + T^{(j)}(a + h)||$$
$$\leq ||T^{(i)}(a) - T^{(j)}(a + h)|| + ||T^{(j)}(a + h) - T^{(j)}(a)||$$
\[ ||T^{(i)}(a) - T^{(j)}(a)|| - ||T^{(j)}(a + h) - T^{(j)}(a)|| \leq ||T^{(i)}(a) - T^{(j)}(a + h)|| \]
\[ 2 - ||T^{(j)}(a + h) - T^{(j)}(a)|| \leq ||T^{(i)}(a) - T^{(j)}(a + h)|| \quad i \neq j \]

Since \( T(t) \) is continuous, \( ||T^{(j)}(a) - T^{(j)}(a + h)|| \to 0 \) as \( h \to 0 \). Therefore
\[ 1 \leq ||T^{(i)}(a) - T^{(j)}(a + h)|| \]

It implies that for all columns in \( T_2(a + h) \), there exists only one of them which is close to the \( i \)th column of \( T_1(a) \), \( 1 \leq i \leq n \). This explains why the algorithm performs so well in practice.

### 4.2 Details of Step 2

Equation (4.2) can be obtained in the following way. Suppose that there are \( n_a \) BC's at \( t = a \) and \( n_b \) BC’s at \( t = b \). Using (3.14), we have (e.g. at \( t = a \))

\[
B_a x(a) = B_a T(a) \begin{bmatrix} y \\ z(a) \end{bmatrix} = \beta_0
\]

\[
B_a T(a) \begin{bmatrix} -[U^{11}]^{-1} U^{12} z(a) \\ z(a) \end{bmatrix} + \begin{bmatrix} -[U^{11}]^{-1} g^1(a) \\ 0 \end{bmatrix} = \beta_0
\]

\[
B_a T \begin{bmatrix} -[U^{11}]^{-1} U^{12}(a) \\ I \end{bmatrix} z(a) = B_a T \begin{bmatrix} [U^{11}]^{-1} g^1(a) \\ 0 \end{bmatrix} + \beta_0
\]

Obviously, in (4.2)

\[
\hat{B}_a z(a) = B_a T \begin{bmatrix} -[U^{11}]^{-1} U^{12}(a) \\ I \end{bmatrix} \hat{\beta}_0 = B_a T \begin{bmatrix} [U^{11}]^{-1} g^1(a) \\ 0 \end{bmatrix} + \beta_0
\]

Likewise, the same process is performed on \( B_b \).
However sometimes the user may supply more than \( n_z \) BC’s (i.e. \( n_a + n_b > n_z \)), even though only \( n_z \) of them are required to specify the BC’s for the whole problem. In this case, the BC’s have to be projected properly. The technique used to perform projection in here has been explained in Section 2.1. \( \hat{B}_{a1} \) in (4.5) and \( \hat{B}_{b2} \) in (4.6) should be nonsingular. The BC’s at \( t = a \) have to control the decreasing modes of \( z \) and the ones at \( t = b \) have to control the increasing modes. By substituting (3.14) into (3.12) (for the endpoints only), we obtain a set of explicit differential equations on \( z \) alone,

\[
z' = \{U^{22} - U^{21}[U^{11}]^{-1}U^{12}\}z + \{g^2 - U^{21}[U^{11}]^{-1}g^1\}
\]

Therefore in (4.3),

\[
H = \{U^{22} - U^{21}[U^{11}]^{-1}U^{12}\} \quad h = \{g^2 - U^{21}[U^{11}]^{-1}g^1\}
\]

Step 2(d) and Step 2(e) can be performed in a very simple way. Suppose that \( w(a) = QTz(a) \), then (4.4) can be written as

\[
\hat{B}_aw(a) = \hat{\beta}_0
\]

Let us write \( \hat{B}_a \) as

\[
\hat{B}_a := [\hat{B}_{a1}, \hat{B}_{a2}]
\]

where \( \hat{B}_{a1} \in R^{n_a \times k} \) and \( \hat{B}_{a2} \in R^{n_a \times (n_a - k)} \). Assume that \( \hat{B}_{a1} \) has a row rank of at least \( k \), where \( k \) is as defined in Step 2(c). Compute the Gaussian Elimination for (4.7) with partial pivoting. Use an array, say \( PIVOT \), to keep track of which rows have been
swapped. For instance, row i and row j have been swapped. Then \( PIVOT(i) = j \) and \( PIVOT(j) = i \). (A QR decomposition is not used because it is more expensive). Let say, after the Gaussian Elimination, (4.7) becomes

\[
\dot{B}_a w(a) = \dot{\beta}_0
\]

where \( \dot{B}_{a11} \in \mathcal{R}^{k \times k}, \dot{B}_{a22} \in \mathcal{R}^{(n_a-k) \times (n_a-k)}, \dot{\beta}_{01} \in \mathcal{R}^k \) and \( \dot{\beta}_{02} \in \mathcal{R}^{(n_a-k)} \). Then \( \dot{B}_a \) and \( \dot{\beta}_0 \) in (4.5) can be chosen in the following way: Row i of \( \dot{B}_a \) is the same as row \( PIVOT(i) \) of \( \dot{B}_a \) in (4.7), and \( \dot{\beta}_0(i) = \dot{\beta}_0(PIVOT(i)), 1 \leq i \leq k \) (as defined in Step 2(c)). The rows of \( \dot{B}_a \) in (4.7) are used for \( \dot{B}_a \), instead of the rows in \( \dot{B}_a \), because Gaussian Elimination does not preserve conditioning. Using the notation of (4.5) we know that \( \dot{B}_{a1} \) is nonsingular because \( \dot{B}_{a11} \) is upper triangular. Step 2(f) and 2(h) can be performed in the same way.

### 4.3 Details of Step 3

This is another key step of the whole algorithm. By studying the ghost problem and rearranging the BC’s for \( y \), hopefully large stability constants may be prevented. However there is a major assumption in this step, and it is that the eigenvalue analysis does represent the kinematic eigenvalue analysis. When we look at the ghost problem (3.15), ideally we should find a transformation \( Q \) such that

\[
V = Q^{-1}(-M)Q - Q^{-1}Q'
\]
where $\mathcal{V}$ is upper-triangular and the diagonal elements are the \textit{kinematic eigenvalues}. This is the famous \textit{Lyapunov Equation} (see Definition 2.2). But in actual practice, we use the eigenvalue analysis (though it may be misleading as shown in Chapter 2) because it is expensive (or even infeasible) to find the kinematic eigenvalues. The eigenvalues are the same as the kinematic eigenvalues in general only when $M$ (hence $Q$) is a constant matrix, and they approximate the kinematic eigenvalues well only when the second term of Lyapunov Equation is relatively small (i.e. $\|Q^{-1}Q'\|$ is small when compared with the first term).

In other words, it is that when $M(t)$ has eigenvalues with large real part (the only case where this matters), its variation in $t$ is slow compared to the size of such eigenvalues. Then the eigenvalue analysis approximates Lyapunov's equation well. The decoupling can be done by the transformation $Q^Ty$.

In the last remark of Chapter 3, we show that the non-uniqueness of $T(t)$ and $S(t)$ does not affect the kinematic eigenvalues of the ghost ODE. But does the non-uniqueness affect the eigenvalues of $M$? The answer is positive. Rewriting (3.21) and (3.22) using (3.23), we obtain

$$\mathcal{V} = Q^{-1}(-M)Q - Q^{-1}Q'$$

$$\mathcal{V} = \tilde{Q}^{-1}(-\tilde{M})\tilde{Q} - \tilde{Q}^{-1}\tilde{Q}'$$

$$= \tilde{Q}^{-1}(-\tilde{M})\tilde{Q} - [Q^{-1}Q' - Q^{-1}F_TF_T^{-1}Q]$$

(4.9) (4.10)
CHAPTER 4. ALGORITHM (LINEAR INDEX 1 DAES IN BVPS)

Of course, if $F_T = Q$, then the second term of the above equation will disappear. Even if $||Q^{-1}Q'||$ is small, $||F_T^r F_T^{-1}||$ may not be. Similarly, $||Q^{-1}Q'||$ may be large, but $||Q^{-1}Q' - Q^{-1}F_T^r F_T^{-1}Q||$ may be small. One may think of trying to find an $F_T$ such that the second term is small. But it is hard to control this term in actual implementation.

There are two more practical issues:

1. How does one know that the eigenvalue analysis is accurate enough?

2. If the eigenvalue analysis is insufficient, what should one do?

We attempt to answer the first question, and discuss the second. We can approximate $Q$ and $Q'$ using the following formulas:

\[
V(t) = Q_t^{-1}M(t)Q_1 \\
V(t + h) = Q_2^{-1}M(t + h)Q_2 \\
Q \approx Q_1 \\
Q' \approx (Q_2 - Q_1)/h
\]

where $Q_1$, $Q_2$ are obtained by the QR algorithm. Substitute $Q$ and $Q'$ into the Lyapunov Equation to see if $V$ is close to upper-triangular. Of course, if $M$ is a constant matrix, this algorithm works well. In practice, however, this simple algorithm does not always work well. Even if the elements of the strictly lower triangular part of the approximated $V$ are small, it does not guarantee that the eigenvalue analysis is accurate enough. But
if they are large, it may be a good indicator that the eigenvalue analysis may not be 
sufficient. Another difficulty is the expense involved: in order to obtain $Q_1$, we require 
an eigenvalue analysis of $M(t)$, and in order to obtain $Q_2$, we require another eigenvalue 
analysis of $M(t + h)$. Performing an eigenvalue analysis could be very expensive. The 
second question is much harder to answer. As far as we know, eigenvalue analysis is the 
only tool we have to approximate the kinematic eigenvalues efficiently. One may think 
of using Riccati-like methods. However, these methods require the user to supply the 
number of nonincreasing modes and the number of nondecreasing modes, which the user 
usually does not have. We cannot use the result from the eigenvalue analysis because 
we do not know if it is correct or not. Another problem is that it is quite inefficient. 
On the other hand, if the eigenvalue analysis fails, the worst that could happen is large 
truncation errors. This could be detected by error estimation in a code. So one can 
switch to another method if the errors are too large.

This step also includes a check for dichotomy of the ghost problem ($n_0^- + n_1^+ > n_y$ 
and $n_0^+ + n_1^- > n_y$). If this condition is not satisfied then it simply means that a dichotomy 
does not exist. We know that a necessary condition for well-conditioning for BVP is the 
existence of dichotomy. The intuitive meaning of these two conditions is that the total 
number of rapidly increasing modes and rapidly decreasing modes should be no more 
than $n_y$. If it is more than that, it means that one of the modes changes from one type 
to the other.
The last part of this step is transforming back to obtain BC on \( x(a) \) and \( x(b) \). This can be done quite straightforwardly. We have \( n_z \) BC's from Step 2, and \( n_y \) BC's from this step. Then we have (e.g. at \( t = a \))

\[
B_a \begin{bmatrix} y(a) \\ z(a) \end{bmatrix} = \beta \\
B_a T^{-1}(a)x(a) = \beta
\]

### 4.4 Details of Step 4

The last step is just solving the problem with the BC's configured.
Chapter 5

Numerical Examples

The algorithm proposed in Chapter 4 has been implemented in a code, DAESOLVE. The code uses the symmetric midpoint scheme. DAESOLVE has been tested on numerous index 1 DAEs, and in this chapter we examine three representative examples. Each example illustrates certain aspect of the algorithm proposed in Chapter 4. By looking at these three examples, the performance of the algorithm will then be demonstrated. The first two examples illustrate the importance of arranging the boundary conditions for the ghost ODE and performing the projection. The third one illustrates how the non-uniqueness of the decomposition of $E(t)$ (see the remark at the end of Chapter 3) affects the eigenvalue analysis (as mentioned in Section 4.1 and Section 4.3), and as a result fails to arrange the boundary conditions for the algebraic part of the DAE properly.

For the following examples the code DAESOLVE uses a uniform mesh ($h = 1/N$) (see (3.7) and (3.8)). The errors computed are the maximum absolute errors. All computations were performed on a Sun 3 running a UNIX f77 compiler in double precision.
CHAPTER 5. NUMERICAL EXAMPLES

5.1 Example A

This example demonstrates that the algorithm works well in a case when the eigenvalue analysis of the ghost matrix $M$ (3.15) approximates the kinematic eigenvalues properly, even if $M$ is not a constant matrix. Using the notation of (1.2), the DAE is as follows:

$$
E = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
-1/t & 2 & -1/t
\end{bmatrix}
A = \begin{bmatrix}
t(1 + 2\beta) + (1 - \beta) & -(1 + 2\beta) \\
t(1 - \beta) - 2\beta & t(1 + 2\beta) \\
0 & -1
\end{bmatrix}
$$

$$
q^T = \begin{bmatrix}
(\beta - 1)e^{-t/\beta}, & t(1 + 2\beta) + (2\beta - (1 - \beta)t)e^{-t/\beta}, & t - 1/t
\end{bmatrix}
$$

$\beta$ is a parameter, $x_1(1) = e^{-t/\beta}$, and $1 \leq t \leq 1.1$. The exact solution is:

$$
x_1 = e^{-t/\beta}, \quad x_2 = te^{-t/\beta}, \quad x_3 = t^2e^{-t/\beta} + t
$$

The transformation matrices $S$ and $T$ can be chosen as follows: $S = I$ and

$$
T = \begin{bmatrix}
1 & -t & 0 \\
t & 1 - t^2 & 1 \\
t^2 & 2t - t^3 & t
\end{bmatrix}
T^{-1} = \begin{bmatrix}
1 & t & 1 \\
t & 0 & 1/t \\
-t & 2 & -1/t
\end{bmatrix}
det(T) = -t
$$

Then

$$
\begin{bmatrix}
y_1 \\
y_2 \\
y_3
\end{bmatrix} = \begin{bmatrix}
t + e^{-t/\beta} \\
1 \\
-1
\end{bmatrix}
M = \begin{bmatrix}
t & (t + 1)(1/\beta - t) + 1 \\
1/\beta - (t + 1) & 1
\end{bmatrix}
$$

With the following transformation we can find the kinematic eigenvalues of $M$, $\mathcal{V} = T^{-1}(MT - T')$:

$$
T = \begin{bmatrix}
t + 1 & 1 \\
1 & 0 \\
1 & 0
\end{bmatrix}
\mathcal{V} = \begin{bmatrix}
1/\beta & 1 \\
0 & -1
\end{bmatrix}
$$

Therefore the kinematic eigenvalues of $M$ are $1/\beta$ and $-1$. The stability constant is very large when $\beta$ is close to 0 and the BC is not properly posed. Actually the index of this
DAE is close to 2 when $\beta$ is close to 0. This can easily be seen when we look at $U^{11}$ of (3.13)

$$U^{11} = \begin{bmatrix} 1 - \beta & -t(1 - \beta) - (1 + 2\beta) \\ t(1 - \beta) - 2\beta & -t^2(1 - \beta) - t \end{bmatrix} \quad \text{det}(U^{11}) = -2\beta(1 + 2\beta)$$

If $\beta = 0$, then $\text{det}(U^{11}) = 0$. Note that $M(t)$ and $T(t)$ are not constant.

However, a simple computation shows that the eigenvalues of $M$ are

$$\begin{bmatrix} \frac{1}{2\beta}(1 - \beta + \sqrt{1 + 2\beta + 5\beta^2}) \\ \frac{1}{2\beta}(1 - \beta - \sqrt{1 + 2\beta + 5\beta^2}) \end{bmatrix} \approx \begin{bmatrix} \frac{1}{\beta} \\ -0.5 \end{bmatrix} \quad \text{when } \beta \to 0 \quad (5.1)$$

The large eigenvalue is very close to the large kinematic eigenvalue. This implies that the algorithm should perform well. The experimental results do show that this is true.

In order to use the numerical scheme, two other BC's are required. Suppose that these two BC's are as follows:

$$x_2(1) = e^{-1}/\beta, \quad x_3(1) = e^{-1}/\beta + 1$$

and $\beta$ is chosen to be $-0.01$. Table 5.1a shows the numerical results when the DAE is posed as an IVP using the BC's above (i.e. the algorithm proposed in Chapter 4 is not used). Table 5.1b shows the numerical results when the code DAESOLVE is used.
CHAPTER 5. NUMERICAL EXAMPLES

Table 5.1a

<table>
<thead>
<tr>
<th>h</th>
<th>error1</th>
<th>error2</th>
<th>error3</th>
<th>rate1</th>
<th>rate2</th>
<th>rate3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.200-01</td>
<td>0.204+04</td>
<td>0.433+04</td>
<td>0.704+04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.100-01</td>
<td>0.701+03</td>
<td>0.151+04</td>
<td>0.245+04</td>
<td>0.152+01</td>
<td>0.152+01</td>
<td>0.152+01</td>
</tr>
<tr>
<td>0.500-02</td>
<td>0.229+03</td>
<td>0.486+03</td>
<td>0.792+03</td>
<td>0.163+01</td>
<td>0.163+01</td>
<td>0.163+01</td>
</tr>
<tr>
<td>0.250-02</td>
<td>0.622+02</td>
<td>0.132+03</td>
<td>0.215+03</td>
<td>0.188+01</td>
<td>0.188+01</td>
<td>0.188+01</td>
</tr>
<tr>
<td>0.125-02</td>
<td>0.159+02</td>
<td>0.337+02</td>
<td>0.549+02</td>
<td>0.197+01</td>
<td>0.197+01</td>
<td>0.197+01</td>
</tr>
<tr>
<td>0.625-03</td>
<td>0.400+01</td>
<td>0.847+01</td>
<td>0.138+02</td>
<td>0.199+01</td>
<td>0.199+01</td>
<td>0.199+01</td>
</tr>
</tbody>
</table>

Table 5.1b

<table>
<thead>
<tr>
<th>h</th>
<th>error1</th>
<th>error2</th>
<th>error3</th>
<th>rate1</th>
<th>rate2</th>
<th>rate3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.200-01</td>
<td>0.135+00</td>
<td>0.275+00</td>
<td>0.444+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.100-01</td>
<td>0.412-01</td>
<td>0.838-01</td>
<td>0.135+00</td>
<td>0.171+01</td>
<td>0.171+01</td>
<td>0.172+01</td>
</tr>
<tr>
<td>0.500-02</td>
<td>0.111-01</td>
<td>0.226-01</td>
<td>0.360-01</td>
<td>0.189+01</td>
<td>0.189+01</td>
<td>0.190+01</td>
</tr>
<tr>
<td>0.250-02</td>
<td>0.284-02</td>
<td>0.578-02</td>
<td>0.917-02</td>
<td>0.197+01</td>
<td>0.197+01</td>
<td>0.197+01</td>
</tr>
<tr>
<td>0.125-02</td>
<td>0.714-03</td>
<td>0.145-02</td>
<td>0.230-02</td>
<td>0.199+01</td>
<td>0.199+01</td>
<td>0.199+01</td>
</tr>
<tr>
<td>0.625-03</td>
<td>0.179-03</td>
<td>0.364-03</td>
<td>0.576-03</td>
<td>0.200+01</td>
<td>0.200+01</td>
<td>0.200+01</td>
</tr>
</tbody>
</table>

Actually the computed eigenvalues for Table 5.1b are

\[-101.6 \text{ and } -0.39 \text{ at } t=1\]
\[-101.6 \text{ and } -0.42 \text{ at } t=1.1\]

These values are close to the ones in (5.1). For Table 5.1b, the algorithm allocates two BC's at $t = 1$ and one at $t = 1.1$.

5.2 Example B

This example is used to demonstrate the importance of projecting the boundary conditions for the differential part of the DAE (i.e. Step 2 of the proposed algorithm). Since sometimes the user provides more boundary conditions than the algorithm requires, the
projection algorithm in Section 4.2 will choose an appropriate combination from those available. Using the notation of (1.2), consider

\[ E = \begin{bmatrix} 0 & 0 \\ -(t+1)^{-1} & 1 \end{bmatrix}, \quad A = \begin{bmatrix} \beta & 1 - \beta(t+1) \\ 0 & 1 \end{bmatrix} \]

\[ q^T = \begin{bmatrix} 0, \quad (t+1)^{-1} - 2\beta - \beta t \end{bmatrix} \]

\( \beta > 0, \ 0 \leq t \leq 1 \) and

\[ x_1(1) = -2 + 4\beta \tag{5.2} \]

The exact solution is:

\[ x_1 = -(t+1) + \beta(t+1)^2, \quad x_2 = \beta(t+1) \]

However the user may think that one boundary condition is not enough, because the BVP has two variables, namely \( x_1 \) and \( x_2 \). Therefore the user may also specify one more BC. Let say, for instance, this BC is as follows:

\[ x_1(0) = -1 + \beta \tag{5.3} \]

and \( \beta \) is chosen to be 10.

When we look at the proposed algorithm in Chapter 4, we know that Step 3 (see Section 4.3) generates BC's for the algebraic part of the DAE without using any given BC's (4.1). However, the algorithm requires the BC for the differential part to be specified. Using the above example, the dimension of the differential part is one, it implies
that only one BC is required for the differential part. But there are two BC’s that the algorithm could use. Which one should be chosen?

We consider two cases. The first one is that no projection is performed (i.e. Step 2 of the proposed algorithm in Chapter 4 is omitted) and the algorithm randomly picks (5.3) as the BC for the differential part. Table 5.2a shows the experimental results of this case. The second case is that the projection step (See Section 4.2) is performed. Table 5.2b shows the experimental results of this case. One can easily see that the errors recorded in Table 5.2b are much better than those in Table 5.2a.

<table>
<thead>
<tr>
<th>h</th>
<th>error1</th>
<th>error2</th>
<th>rate1</th>
<th>rate2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.500-01</td>
<td>0.555+10</td>
<td>0.294+10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250-01</td>
<td>0.177+10</td>
<td>0.933+09</td>
<td>0.165+01</td>
<td>0.166+01</td>
</tr>
<tr>
<td>0.125-01</td>
<td>0.506+09</td>
<td>0.267+09</td>
<td>0.180+01</td>
<td>0.181+01</td>
</tr>
<tr>
<td>0.625-02</td>
<td>0.132+09</td>
<td>0.694+08</td>
<td>0.194+01</td>
<td>0.194+01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>h</th>
<th>error1</th>
<th>error2</th>
<th>rate1</th>
<th>rate2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.500-01</td>
<td>0.108+00</td>
<td>0.561-01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250-01</td>
<td>0.299-01</td>
<td>0.152-01</td>
<td>0.185+01</td>
<td>0.189+01</td>
</tr>
<tr>
<td>0.125-01</td>
<td>0.772-02</td>
<td>0.388-02</td>
<td>0.195+01</td>
<td>0.197+01</td>
</tr>
<tr>
<td>0.625-02</td>
<td>0.195-02</td>
<td>0.975-03</td>
<td>0.199+01</td>
<td>0.199+01</td>
</tr>
</tbody>
</table>

The reason can be seen as follows: Using the following transformation matrices

\[ S = I \]

\[ T = \begin{bmatrix} (t + 1) & 0 \\ 1 & 1 \end{bmatrix} \]

\[ T^{-1} = \begin{bmatrix} (t + 1)^{-1} & 0 \\ -(t + 1)^{-1} & 1 \end{bmatrix} \]

\[ \det(T) = t + 1 \]
the linear DAE can be reduced to a semi-explicit form

\[ y = [\beta(t + 1) - 1]z, \quad z' = [1 + (t + 1)^{-1}]y + z + q_2 \]

where \( q_2 = (t + 1)^{-1} - 2\beta - \beta t \). Substituting \( y \) into the ODE for \( z \),

\[ z' = (1 - z)q_2 \tag{5.4} \]

If \( \beta > 1/2 \), then (5.4) will have a growing mode, and therefore the boundary condition for the differential part should be specified at \( t = 1 \). This explains the poor results in Table 5.2a. The results in Table 5.2b are much better because the projection algorithm studies the sign and magnitude of \( q_2 \) in (5.4), and decides to put the boundary condition of the differential part at \( t = 1 \).

It should be noted that the phenomenon demonstrated in this example has little to do with the use of the mid-point scheme. Rather, it points out the general danger present when randomly picking boundary conditions in a consistent, overdetermined problem, regardless of the numerical discretization.

### 5.3 Example C

This example is used to demonstrate that the eigenvalue analysis for the ghost ODE could be sensitive to the choice of \( T \). Since \( T \) is not unique, a different \( T \) may generate a different ghost problem, and hence produce different eigenvalues. (See Section 4.3 for
CHAPTER 5. NUMERICAL EXAMPLES

details). Using the notation of (1.2), consider

\[
E = \begin{bmatrix}
-t & 2 & -1/t \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\quad
A = \begin{bmatrix}
\beta t & -\beta \\
2\beta t - 2 & (\alpha^2 + \beta^2)t - 2\beta t \\
-2\beta t & (\alpha^2 + \beta^2)t - 2\beta t
\end{bmatrix}
\]

\[
q^T = \begin{bmatrix}
-2\alpha, \quad -\alpha(\alpha^2 + \beta^2)t^2 - 2\alpha(\beta t - 1)e^{-t}, \quad 2\alpha\beta t e^{-t}
\end{bmatrix}
\]

\[\beta < 0, \quad \alpha > 0, \quad 1 \leq t \leq 2, \text{ and} \]

\[x_1(1) = \alpha e^{-1}\]

The exact solution is:

\[
x_1 = \alpha e^{-t}, \quad x_2 = \alpha t e^{-t}, \quad x_3 = \alpha t^2(1 + e^{-t})
\]

If we choose the following \(S\) and \(T\)

\[
S = \begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
\alpha & \beta & 0 \\
\alpha t & \beta t - 1 & 1 \\
\alpha t^2 & \beta t^2 - 2t & t
\end{bmatrix}, \quad T^{-1} = \begin{bmatrix}
1/\alpha & -\beta/\alpha & \beta/\alpha \\
0 & 1 & -1/t \\
-\beta t & 2 & -1/t
\end{bmatrix}, \quad \det(T) = \alpha t
\]

then

\[
\begin{bmatrix}
y_1 \\
y_2 \\
z
\end{bmatrix} = \begin{bmatrix}
\beta t + e^{-t} \\
-\alpha t \\
-\alpha t
\end{bmatrix}, \quad M = \begin{bmatrix}
\beta & \alpha \\
-\alpha & \beta
\end{bmatrix}
\]

Since \(M\) is a constant matrix, the kinematic eigenvalues are the same as the eigenvalues, namely \(\beta \pm \alpha i\). Because \(\beta\) is negative, the two boundary conditions for the algebraic part \(y\) should be put at \(t = 2\), and also because \(\alpha > 0\), the boundary condition for the differential part \(z\) should be put at \(t = 1\). That is, there should be one boundary condition at the left end point, and two boundary conditions at the right end point.
However the matrix function $M$ of the ghost problem could be quite different if another $T$ is chosen. For instance, using the above $S$ with

$$T_{\text{new}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t^2 & 2t & -t \end{bmatrix} \quad T_{\text{new}}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t & 2 & -1/t \end{bmatrix}$$

then

$$M_{\text{new}} = \begin{bmatrix} (\alpha^2 + \beta^2)t & -\frac{1}{2}t & -(\alpha^2 + \beta^2) \\ (\alpha^2 + \beta^2)t - 2\beta t & -\frac{1}{2}t & -(\alpha^2 + \beta^2)t + 2\beta \end{bmatrix}$$

and the eigenvalues of $M_{\text{new}}$ are 0 and $2\beta$. But, we know that the kinematic eigenvalues are $\beta \pm \alpha i$. Thus, the eigenvalues are very different from the kinematic eigenvalues. This implies that the algorithm only performs well when $T$ is properly chosen. The following will demonstrate this phenomenon.

In order to use the numerical scheme, two other BC's are required. Suppose that these two BC's are as follows:

$$x_2(1) = \alpha e^{-1}, \quad x_3(1) = \alpha (1 + e^{-1})$$

Table 5.3a shows the numerical results of running DAESOLVE with $\beta = -5$ and $\alpha = 15$.

The kinematic eigenvalues are then $-5 \pm 15i$. The eigenvalues computed are

$$\begin{align*}
-5.5 + 4.7i \quad \text{and} \quad -5.5 - 4.7i & \quad \text{at} \ t = 1 \\
-5.4 + 7.0i \quad \text{and} \quad -5.4 - 7.0i & \quad \text{at} \ t = 2
\end{align*}$$

The real parts of the eigenvalues are close to the real parts of the kinematic eigenvalues, which explains the good results of Table 5.3a.
Table 5.3b shows the numerical results of running DAESOLVE with $\beta = -10$ and $\alpha = 6$. The kinematic eigenvalues are then $-10 \pm 6i$. But the eigenvalues computed are

$$-19.4 \text{ and } -1.6 \text{ at } t = 1$$

$$-18.2 \text{ and } -2.5 \text{ at } t = 2$$

The eigenvalues are very different from the kinematic eigenvalues. Assuming that the eigenvalues $-1.6$ and $-2.5$ are not considered as large, the algorithm proposed in Chapter 4 will then set two boundary conditions at $t = 1$ and one at $t = 2$. However we know that there should be one boundary condition at $t = 1$ and two at $t = 2$. This explains the poor results of Table 5.3b.

<table>
<thead>
<tr>
<th>$h$</th>
<th>error1</th>
<th>error2</th>
<th>error3</th>
<th>rate1</th>
<th>rate2</th>
<th>rate3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.500-01</td>
<td>0.131+01</td>
<td>0.194+01</td>
<td>0.396+01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250-01</td>
<td>0.385+00</td>
<td>0.626+00</td>
<td>0.128+01</td>
<td>0.177+01</td>
<td>0.163+01</td>
<td>0.163+01</td>
</tr>
<tr>
<td>0.125-01</td>
<td>0.996-01</td>
<td>0.167+00</td>
<td>0.341+00</td>
<td>0.195+01</td>
<td>0.191+01</td>
<td>0.191+01</td>
</tr>
<tr>
<td>0.625-02</td>
<td>0.251-01</td>
<td>0.424-01</td>
<td>0.866-01</td>
<td>0.199+01</td>
<td>0.198+01</td>
<td>0.198+01</td>
</tr>
<tr>
<td>0.313-02</td>
<td>0.209-02</td>
<td>0.361-02</td>
<td>0.753-02</td>
<td>0.200+01</td>
<td>0.194+01</td>
<td>0.194+01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$h$</th>
<th>error1</th>
<th>error2</th>
<th>error3</th>
<th>rate1</th>
<th>rate2</th>
<th>rate3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.500-01</td>
<td>0.149+03</td>
<td>0.281+03</td>
<td>0.529+03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250-01</td>
<td>0.553+02</td>
<td>0.105+03</td>
<td>0.200+03</td>
<td>0.143+01</td>
<td>0.142+01</td>
<td>0.140+01</td>
</tr>
<tr>
<td>0.125-01</td>
<td>0.282+02</td>
<td>0.539+02</td>
<td>0.103+03</td>
<td>0.970+00</td>
<td>0.964+00</td>
<td>0.957+00</td>
</tr>
<tr>
<td>0.625-02</td>
<td>0.934+01</td>
<td>0.179+02</td>
<td>0.342+02</td>
<td>0.159+01</td>
<td>0.159+01</td>
<td>0.159+01</td>
</tr>
<tr>
<td>0.313-02</td>
<td>0.253+01</td>
<td>0.485+01</td>
<td>0.928+01</td>
<td>0.188+01</td>
<td>0.188+01</td>
<td>0.188+01</td>
</tr>
</tbody>
</table>

Though this example only shows that the eigenvalue analysis fails to approximate the kinematic eigenvalues of the ghost matrix $M$, a similar failure is also expected when using
the eigenvalues to approximate the kinematic eigenvalues in the process of projection (see Section 4.2).
Chapter 6

Conclusion and Future Research

In this thesis we have presented an algorithm to solve linear boundary value problems for DAEs of index 1 with separated boundary conditions. The algorithm arranges the boundary conditions for the algebraic part and rearranges the boundary conditions for the differential part of the DAE. By doing so, we hope that the conditioning constants of the ODEs (from the differential part and from the additional ghost ODE generated by the symmetric discretization of the algebraic part) will be of moderate size. These features have been implemented in a code, DAESOLVE.

DAESOLVE has been tested on numerous examples, and the results have been compared with a code which chooses the BC arbitrarily. The experimental results are promising. DAESOLVE outperforms the other code in most cases, and performs as well as the other one in other cases.

However, DAESOLVE has two weaknesses which may sometimes cause it to fail to perform better than with arbitrary BC. The first one is that its performance is critically
affected by the ability of using the eigenvalues to approximate the kinematic eigenvalues. It is well-known in BVP literature that this approximation may fail in some pathological cases. The second, which is more serious, comes from the non-uniqueness of the decomposition of $E(t)$. The non-uniqueness of $T(t)$ has created an important problem: It may affect the eigenvalue analysis (see Section 4.3).

In theory, the non-uniqueness of $T(t)$ could lead to an unpredictable effect on the eigenvalue analysis (cf. (4.10)). The magnitude of $||Q^{-1}Q' - Q^{-1}F_T F_T^{-1}Q||$ could be either large or small depending on $F_T$. However, the problem lies in the uncertainty of what could happen. It is very hard to control what could happen in practice.

Like most works, this thesis suggests more problems than it solves. The most obvious are the ones mentioned above. A simple extension is to allow DAESOLVE to handle non-separated boundary conditions. The whole algorithm proposed in Chapter 4, except for the step of projection, could be applied to the case of non-separated boundary conditions. It is a non-trivial question to ask how non-separated boundary conditions could be projected. Ascher in [1] has suggested one way to do so, but it cannot be used easily in practice. Last, but not the least, is to apply the algorithm proposed in this thesis for non-linear problems.
Bibliography


[14] B.J. Leimkuhler, “Approximation methods for the consistent initialization of
of Illinois at Urbana-Champaign, Dept. of Computer Science.


*Proceedings of IMACS World Congress, Montreal, Canada*, 1982.


“Solvability of large scale descriptor systems,” *Boeing Computer Services Company*,
report, BOE Contract No. ET-78-C-01-2876.