

# ON THE NUMERICAL SOLUTION OF DIFFERENTIAL-ALGEBRAIC EQUATIONS

Michael A. Aspoas

Degree awarded with distinction on 30 June 1994.

A thesis submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science.

### **Abstract**

We give an overview of the numerical solution of the initial value differential-algebraic equation (DAE) from the underlying theory, through both the development of numerical techniques and software and a survey of the major areas of application, to the implementation of available software codes in the solution of DAEs arising in applications. The experimental part serves to demonstrate the need for specific DAE, rather than simply ordinary differential equation (ODE), methods and the special considerations required for the successful numerical solution of DAEs, as well as verifying predictions made in the theory. It is hoped that this dissertation can be used as a reference by those working in areas of application, or at least as a pointer to the relevant literature.

# DECLARATION

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University, nor has it been prepared under the aegis or with the assistance of any other body or organization or person outside the University of the Witwatersrand, Johannesburg.

MA Aspöas

Michael A. Aspöas

The 7th day of March, 1994.

*To my parents, without whose support  
this would not have been possible*

### Acknowledgements

I would like to thank my supervisor, Professor Colin J. Wright, for his guidance, and both the Foundation for Research and Development and the University of the Witwatersrand for their financial support.

# Contents

|          |                                               |           |
|----------|-----------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                           | <b>1</b>  |
| 1.0.1    | Motivation . . . . .                          | 1         |
| 1.0.2    | Overview . . . . .                            | 3         |
| 1.1      | Applications . . . . .                        | 4         |
| 1.1.1    | Graph Theory . . . . .                        | 4         |
| 1.1.2    | Singular Perturbation . . . . .               | 9         |
| 1.1.3    | Control Problems . . . . .                    | 10        |
| 1.1.4    | Fluid Dynamics . . . . .                      | 12        |
| <b>2</b> | <b>Theory</b>                                 | <b>13</b> |
| 2.1      | Linear Constant-Coefficient Problem . . . . . | 13        |
| 2.2      | Linear Variable-Coefficient Problem . . . . . | 16        |
| 2.2.1    | Solvability . . . . .                         | 16        |
| 2.2.2    | Index . . . . .                               | 17        |
| 2.2.3    | Index Reduction . . . . .                     | 20        |
| 2.2.4    | Analytical Solutions . . . . .                | 22        |
| 2.3      | General Nonlinear DAE Systems . . . . .       | 23        |
| 2.3.1    | Index . . . . .                               | 23        |
| 2.3.2    | Structural Forms . . . . .                    | 25        |
| 2.3.3    | Index Reductions . . . . .                    | 26        |
| <b>3</b> | <b>Numerical Solution</b>                     | <b>30</b> |
| 3.1      | Introduction . . . . .                        | 30        |
| 3.1.1    | DAEs and Stiff ODEs . . . . .                 | 30        |
| 3.1.2    | The Backward Euler Method . . . . .           | 31        |
| 3.2      | Backward Differentiation Formulae . . . . .   | 35        |
| 3.2.1    | Linear Constant-Coefficient DAEs . . . . .    | 36        |
| 3.2.2    | Linear Variable-Coefficient DAEs . . . . .    | 36        |
| 3.2.3    | Nonlinear DAEs . . . . .                      | 37        |
| 3.3      | Runge-Kutta Methods . . . . .                 | 41        |
| 3.3.1    | Survey . . . . .                              | 42        |
| 3.3.2    | Linear Constant-Coefficient DAEs . . . . .    | 44        |
| 3.3.3    | Fully Implicit Index 1 Systems . . . . .      | 47        |
| 3.3.4    | Semi-Explicit Index 2 Systems . . . . .       | 48        |
| 3.4      | Other Methods . . . . .                       | 49        |
| 3.4.1    | Extrapolation . . . . .                       | 49        |

|                                                    |     |
|----------------------------------------------------|-----|
| 3.4.2 Taylor-Type Methods                          | 52  |
| 3.5 Consistent Initialization                      | 57  |
| 4 Experiments                                      | 62  |
| 4.1 Code Introduction                              | 63  |
| 4.2 Methodology                                    | 63  |
| 4.2.1 Framework for Comparison of Code Performance | 63  |
| 4.2.2 Higher Index                                 | 68  |
| 4.3 Two-Transistor Amplifier                       | 69  |
| 4.4 Two-Phase Plug Flow                            | 73  |
| 4.5 Discharge Pressure Control                     | 77  |
| 4.6 Plane Pendulum                                 | 81  |
| 4.7 Van der Pol Oscillator                         | 91  |
| 4.8 Ring Modulator                                 | 95  |
| 4.9 Cascade of Differential Amplifiers             | 99  |
| 4.10 MOL Transmission Line                         | 106 |
| 4.11 Small Contrived Problems                      | 111 |
| 4.11.1 Irregular Solvable DAE                      | 111 |
| 4.11.2 Variable Index DAE                          | 112 |
| 4.12 Conclusion                                    | 114 |
| 4.12.1 Report on Techniques                        | 114 |
| 4.12.2 Code Comparison                             | 114 |
| 5 Critique                                         | 116 |
| A Further Reading                                  | 118 |
| B Sample Fortran Driver                            | 119 |
| B.1 Amplifier Driver Code                          | 119 |
| B.2 Amplifier Output File                          | 122 |
| Bibliography                                       | 123 |

# List of Tables

|      |                                                                             |     |
|------|-----------------------------------------------------------------------------|-----|
| 3.1  | Coefficients and stability for k-step BDF methods . . . . .                 | 35  |
| 3.2  | Orders of convergence for some Runge-Kutta methods . . . . .                | 43  |
| 3.3  | Orders of convergence for linear constant-coefficient DAEs . . . . .        | 47  |
| 3.4  | Orders for nonlinear index 1 systems . . . . .                              | 48  |
| 3.5  | Predicted and observed orders for index 2 pendulum problem . . . . .        | 49  |
| 3.6  | Order pattern for extrapolation . . . . .                                   | 51  |
| 4.1  | Abbreviations . . . . .                                                     | 64  |
| 4.2  | Basic DAE code properties . . . . .                                         | 64  |
| 4.3  | User interface to DAE codes . . . . .                                       | 65  |
| 4.4  | RADAU5 solution statistics for two-transistor amplifier . . . . .           | 70  |
| 4.5  | DASSL solution statistics for two-transistor amplifier . . . . .            | 70  |
| 4.6  | DASSL solution to two-phase plug flow Systems I and II . . . . .            | 75  |
| 4.7  | LSODI solution to two-phase plug flow System II . . . . .                   | 75  |
| 4.8  | RADAU5 solution to two-phase plug flow System II . . . . .                  | 76  |
| 4.9  | RADAU5 solution statistics for discharge pressure control . . . . .         | 80  |
| 4.10 | RADAU5 solution statistics for plane pendulum . . . . .                     | 83  |
| 4.11 | DASSL and LSODI solution statistics for plane pendulum . . . . .            | 83  |
| 4.12 | Effective order estimates for plane pendulum . . . . .                      | 84  |
| 4.13 | Summary of results for plane pendulum . . . . .                             | 85  |
| 4.14 | RADAU5 solution to Van der Pol System I . . . . .                           | 92  |
| 4.15 | DASSL solution to Van der Pol System I . . . . .                            | 92  |
| 4.16 | LSODI solution to Van der Pol System I . . . . .                            | 93  |
| 4.17 | RADAU5 solution to cascade problem: index 1, 2, 3 scaling options . . . . . | 102 |
| 4.18 | RADAU5 solution statistics for cascade problem: $s = 1, 2, 3$ . . . . .     | 102 |
| 4.19 | DASSL solution statistics for cascade problem: $s = 1, 2, 3, 4$ . . . . .   | 104 |
| 4.20 | Sample problems for MOL transmission line . . . . .                         | 107 |
| 4.21 | Solution statistics for MOL transmission line . . . . .                     | 107 |
| 4.22 | Solution statistics for irregular solvable DAE . . . . .                    | 112 |
| 4.23 | Comparison of code performance . . . . .                                    | 115 |

# List of Figures

|      |                                                                         |     |
|------|-------------------------------------------------------------------------|-----|
| 4.1  | Circuit diagram for two-transistor amplifier . . . . .                  | 69  |
| 4.2  | RADAU5 solution to two-transistor amplifier . . . . .                   | 71  |
| 4.3  | LSODI solution to two-transistor amplifier . . . . .                    | 72  |
| 4.4  | Work-precision diagrams for two-transistor amplifier . . . . .          | 73  |
| 4.5  | Solution to two-phase plug flow problem . . . . .                       | 75  |
| 4.6  | Work-precision diagrams for two-phase plug flow problem. . . . .        | 76  |
| 4.7  | Schematic for discharge pressure control problem . . . . .              | 77  |
| 4.8  | RADAU5 solution to discharge pressure control . . . . .                 | 79  |
| 4.9  | RADAU5 work-precision diagrams for discharge pressure control . . . . . | 80  |
| 4.10 | Solution to plane pendulum . . . . .                                    | 82  |
| 4.11 | RADAU5 solution to plane pendulum: index 3 . . . . .                    | 86  |
| 4.12 | DASSL solution to plane pendulum: index 3 . . . . .                     | 87  |
| 4.13 | LSODI solution to plane pendulum: index 3 . . . . .                     | 88  |
| 4.14 | RADAU5 solution to plane pendulum ODE . . . . .                         | 89  |
| 4.15 | Work-precision diagrams for plane pendulum . . . . .                    | 90  |
| 4.16 | Work-precision diagrams for Van der Pol oscillator . . . . .            | 93  |
| 4.17 | RADAU5 solution to Van der Pol oscillator System I . . . . .            | 94  |
| 4.18 | RADAU5 solution to Van der Pol oscillator System IV . . . . .           | 95  |
| 4.19 | DASSL solution to ring modulator . . . . .                              | 98  |
| 4.20 | Work-precision diagram for ring modulator . . . . .                     | 98  |
| 4.21 | Cascade of differential amplifiers . . . . .                            | 99  |
| 4.22 | RADAU5 work-precision diagram for cascade problem . . . . .             | 103 |
| 4.23 | DASSL solution to MOL Problem 1 . . . . .                               | 108 |
| 4.24 | DASSL solution to MOL Problem 3 . . . . .                               | 109 |
| 4.25 | DASSL solution to MOL Problem 4 . . . . .                               | 109 |
| 4.26 | Global errors for MOL Problem 1 and work-precision diagrams . . . . .   | 110 |
| 4.27 | DASSL solution to variable index DAE . . . . .                          | 113 |
| 4.28 | DASSL solution statistics for variable index DAE . . . . .              | 113 |

# Chapter 1

## Introduction

### 1.0.1 Motivation

The term differential-algebraic equation (DAE) has been coined to describe differential equations with constraints (differential equations on manifolds) and singular implicit ordinary differential equations (ODEs). In the pure mathematics literature DAEs are sometimes referred to as algebraic-differential equations (ADEs). Such problems arise and have been solved in a variety of applications, including constrained mechanical systems, fluid dynamics, chemical reaction kinetics, the simulation of electrical circuits and semiconductor devices, control engineering, models of power systems and from the application of the steady-state approximation or perturbation methods. The study of DAEs also gives us insight into the behaviour of numerical methods for stiff ODEs, which has also attracted the interest of many engineers and numerical analysts.

The most general form of the problem we will be considering is the first order initial value differential equation:

$$F(x, x', t) = 0, \quad x(t_0) = x_0, \quad (1.1)$$

where  $F$ ,  $x(t)$  and  $x'(t)$  are  $n$ -dimensional vectors, primes denote differentiation,  $t$  is the independent variable (usually time) and (1.1) is required to be solved for  $x(t)$  on  $[t_0, T]$ . This can be seen as a generalization of the explicit ODE problem,

$$x' = f(x, t), \quad x(t_0) = x_0,$$

and can be solved numerically by a number of ODE methods, as we shall see in Chapter 3. (1.1) can be made formally autonomous by including  $t$  in  $x$  and  $t' - 1$  in  $F$ . Methods for solving the general problem (1.1) must be able to handle the case where the partial Jacobian  $\partial F / \partial x'$  is singular:

$$\det \left( \frac{\partial F}{\partial x'} \right) = 0. \quad (1.2)$$

DAEs are problems of the form (1.1) for which (1.2) holds. This can be seen more obviously in a semi-explicit form:

$$f(x, x', y, t) = 0, \quad \dim(f) = \dim(x) = \dim(x'), \quad (1.3)$$

$$g(x, y, t) = 0, \quad \dim(g) = \dim(y), \quad (1.4)$$

where the Jacobian  $\partial f / \partial x'$  is invertible in some neighbourhood of the solution. We assume throughout that the DAEs we study are correctly dimensioned and are not over or underdetermined. For overdetermined DAEs, see [39]. The differential equations (1.3) must be solved simultaneously with the algebraic equations (1.4). Here are two simple practical examples of DAEs.

1. Constrained mechanical systems [65], where (1.3) is the equation of motion, which is generally of second order and can be reduced to first order by including first derivatives (velocities) in the unknown vector  $x$  of (1.1), and (1.4) geometrically or kinematically constrains the system to lie on a manifold. The procedure of finding appropriate generalized coordinates leading to an ODE seems not to be realizable automatically, whereas systems of the form (1.3), (1.4) are the natural result of formal manipulations using, eg., Cartesian coordinates [76].
2. The linear control problem [17] with specified output:

$$\begin{aligned} x' &= Ax + Bu \\ y &= Cx + Du \end{aligned} \implies \begin{pmatrix} x' \\ 0 \end{pmatrix} = \begin{pmatrix} Ax + Bu \\ Cx + Du - y \end{pmatrix}.$$

Here,  $x$  is the state vector,  $u$  is the control and  $y$  is the output.

There are essentially two distinct classes of problem.

**Type I** have solutions which must be found along with consistent initial conditions on an interval on which the appropriate "structure" of the system, for example the rank, index, or core-rank of some appropriate matrix, remains constant.

**Type II** describe solutions near a point  $t^*$  where the system's structure is different from the structure on a deleted neighbourhood of  $t^*$ .

In the linear case,

$$A(t)x' + B(t)x = f(t),$$

if  $A(t)$  is invertible for  $t \neq t^*$  but  $A(t^*)$  is singular, we have a Type II problem that has been widely studied [22, 35]. In this case, the description of the solutions involves series type techniques, and the solutions exhibit turning point behaviour. Surprisingly enough, prior to 1980, Type I problems, which we shall concentrate on, had not been as well studied as some of the Type II problems. This is due to the importance of understanding jump behaviour and switching effects [89] in electrical circuits.

### 1.0.2 Overview

Historically, the first paper to refer explicitly to the DAE problem is [36] in 1971, and the early papers (1971-1980) were mainly applications-oriented. In the 1980's, the DAE problem was put on a more numerical analytic footing and since then it has become a popular area in the numerical solution of ODEs. As this is a fast-growing field, it is likely that the understanding of both the theory and the numerical solution of DAEs will grow in the near future, and the material presented here is by no means conclusive. The literature covers several areas in the field of DAEs, including the following.

1. Pure mathematical theory:

- (a) Differential-algebraic groups [18], properties of meromorphic solutions [44], and geometric characterizations [83], all of which we omit.
- (b) Existence, uniqueness, problem classification and analytical solutions, which we shall use in later chapters.

2. Presentation of ODE methods and packages applicable to DAEs, including the derivation of methods, order and convergence analysis, details of implementation and experimentation.

3. Applications, notably in electrical circuits, control theory and mechanics.

We hope to reflect the emphasis of the numerical literature, thus yielding a set of tools to be used on the wide range of applications in science and engineering.

- In the remainder of this chapter, some major application areas are introduced, thus providing the background for the example problems of Chapter 4.
- In Chapter 2, the theory of DAEs is developed, including solvability, analytical solutions and the classification of DAEs according to structural form. The concept of index of nilpotency is introduced and studied for three important DAE classes, and index reduction techniques are examined. Several examples are included for clarity.
- In Chapter 3, discretization methods for the numerical solution of DAEs are given. DAEs are compared to stiff ODEs, and popular stiff methods, namely backward differentiation formulae, implicit Runge-Kutta methods, extrapolation and Taylor-type methods, are extended to the solution of DAE's, yielding order results and method comparisons. Implementation problems specific to DAEs, such as consistent initialization, ill-conditioning of the iteration matrix and modification to error and convergence tests, are mentioned and, to some extent, resolved.
- In Chapter 4, the experimental part of the dissertation, currently available software packages developed for solving DAEs are applied to particular problems both to illustrate the behaviour of numerical solutions and to test theoretical predictions on method and code performance. Our results are also compared with other work in the field.

The bibliography, including both cited references and others (Appendix A), incorporates most definitive papers and standard texts and many papers on applications, published before 1990. We also refer to more recent publications.

## 1.1 Applications

### 1.1.1 Graph Theory

Our first major application is the integration of equations resulting from a graph-theoretic formulation of major scientific disciplines [11]. We look specifically at two major computational areas — electrical circuits and constrained multibody systems.

The modelling of systems in many areas of engineering and physics depends on the concept of a *network* of interconnected elements. These elements could be masses, springs and dashpots (in mechanical systems), or inductors, resistors and capacitors (in electrical circuits). Such networks necessarily include

1. A pair of functionally related conjugate variables characterizing each network element: *through* variables, such as current, force, fluid flow and heat flow, which correspond to physical quantities passing or acting through an element, and *across* variables, such as voltage, velocity, pressure difference and temperature difference, which act across the ends of the element, eg. current  $i$  and voltage  $v$ , such that  $v \propto i$  for a resistor,  $v \propto di/dt$  for an inductor and  $i \propto dv/dt$  for a capacitor; and
2. Topological algebraic constraints imposed on the conjugate variables by the pattern of interconnection of the elements.

An understanding of the fundamentals of the network problem provides a unified basis for analyzing networks of all kinds, whether electrical, mechanical, structural, molecular, acoustic, hydraulic or thermal. In (1) above, the properties of each element are described by constitutive relations involving the conjugate variables, and are mathematically similar in that they are linear or nonlinear algebraic or differential equations, ie. (1.1). The topological constraints imposed on the network by interconnecting the elements are:

1. The through variables add to zero at every point of interconnection, eg. Kirchhoff's current law (KCL).
2. The across variables add to zero around every loop (closed path) of the network, eg. Kirchhoff's voltage law (KVL).

The topological constraints thus give rise to strictly linear algebraic equations.

Directed linear graphs (digraphs) contain nodes and directed edges. The edges are either branches (if part of a tree containing all the nodes of the graph) or links (if not on the tree). A tree is part of the graph with no closed paths (or loops). Two of the topological matrices characterizing the connectivity properties of a digraph show how the

branches are connected to the nodes ( $A$  matrix) and included in the loops ( $C$  matrix):

$$\begin{Bmatrix} a_{ij} \\ c_{ij} \end{Bmatrix} = \begin{Bmatrix} +1, -1, 0 \end{Bmatrix} \text{ if the } i\text{-th} \begin{Bmatrix} \text{incident on} \\ \text{branch is [pos, neg,} \end{Bmatrix} \text{ the } j\text{-th} \begin{Bmatrix} \text{node} \\ \text{loop} \end{Bmatrix}.$$

Historically, network theory was developed primarily for the study of electrical circuits, which explains both why electrical circuits are a major application of DAEs and why graph-theoretic concepts are often mixed with electrical circuit concepts. Furthermore, electrical circuits are a very clean application of graph theory due to the obvious correlation between circuit diagrams and the circuits they represent. Network analysis includes:

1. Formulation:

- (a) The characterization of individual elements by constitutive equations, which is primarily discipline-dependent and requires an understanding of the variables involved. Often these are simple differential or algebraic equations embodying a fundamental physical law, such as Ohm's law or Newton's second law; and
- (b) The determination of all pertinent topological constraint equations, in the form of topological matrices.

2. The solution of the resulting differential-algebraic system, either analytically or numerically.

The line of demarcation between the formulation and solution phases of network analysis has become somewhat blurred. The dangers of this are apparent, for example, in the blind application of the Euler method for discretizing time derivatives, resulting in such dubious practices as the construction of companion models for circuits. Another example of this blurring is symbolic manipulation in order to minimize the number of equations to solve, an example of which we will presently see, manifest in the nodal approach vs. state equation debate of the 1970's. Together with the advent of labour- and space-saving sparse matrix methods, the DAE methods presented here allow for vast improvement, especially in terms of robustness, order of accuracy, stability and computational cost. Beyond the formulation and solution by DAE methods, we can further take advantage of the underlying structure of the network problem to improve the efficiency for highly repetitive network problems, such as those involving the numerical solution of partial differential equations (PDEs).

The applicability of the network approach to a wide variety of dynamic systems serves to unify the formulation in the modelling of such systems, and allows the automation of their solution by DAE solvers (network equations are also characteristically stiff), sparse matrix methods, etc. This also justifies further research into modifications of existing DAE methods or development of new methods, based on the underlying structure of the network formulation.

### Electrical Circuits

We now limit our discussion to the electrical case and adopt the composite branch convention, where a branch can contain an  $R$ ,  $G$ ,  $C$  or  $L$  element as well as a series voltage

source  $E$  and a parallel current source  $J$ . We distinguish between *element* voltage  $V$  and current  $I$ , and the *branch* voltage  $e$  and current  $j$ , giving the relations

$$V = E + e, \quad I = J + j.$$

Our problem is, given a network of known topology, known elements  $C$ ,  $G$ ,  $R$  and  $L$  and known sources  $E$  and  $J$ , to find all the response voltages and currents  $e$  and  $j$  simultaneously satisfying the constitutive laws and topological constraints. Now KVL can be stated equivalently as:

$$e = Ae_n, \quad \text{where } e \text{ contains branch voltages} \quad (1.5)$$

$$\text{or } C^T e = 0, \quad \begin{array}{l} \text{and } e_n \text{ contains node-to-ground voltages,} \\ \text{voltages around a loop sum to zero.} \end{array} \quad (1.6)$$

(The relation  $C^T A = 0$  is easily proved.) KCL can be stated equivalently as:

$$j = Cj_L, \quad j: \text{ branch currents, } j_L: \text{ link currents,} \quad (1.7)$$

$$\text{or } A^T j = 0. \quad (1.8)$$

The constitutive equations are given equivalently by

$$V = ZI \quad \text{— impedance form, } Z = \text{one of } R, G^{-1}, Ld/dt \text{ or } \int C^{-1} dt$$

$$I = YV \quad \text{— admittance form, } Y = \text{one of } R^{-1}, G, \int L dt \text{ or } Cd/dt$$

$$\text{or in hybrid form} \quad \begin{pmatrix} I_y \\ V_z \end{pmatrix} = \begin{pmatrix} Y_y & \\ & Z_z \end{pmatrix} \begin{pmatrix} V_y \\ I_z \end{pmatrix}. \quad (1.9)$$

Since we require a system of DAEs for the numerical solution phase of the problem, our choice of the formulation of the constitutive equations is limited to a hybrid form, partitioned so as to exclude integral terms  $Z = \int C^{-1} dt$  or  $Y = \int L dt$ , in favour of differential terms  $Z = Ld/dt$  and  $Y = Cd/dt$ . More specific applications might not require hybrid formulation. The simplest hybrid formulation is the hybrid nodal approach [51], which uses (1.9) for the constitutive law, and only the matrix  $A$  for topology (ie. (1.5) and (1.8)). This results in a linear system of equations, whose matrix is referred to as the *hybrid tableau*. Another hybrid combination is the *state variable* formulation, in which the objective is to derive a minimal set of d.e.'s of the form

$$x' = Bx + f(t), \quad (1.10)$$

describing the network. The state variable formulation uses the simple branch convention, the  $y$ - and  $z$ -type branches are divided into  $C$ - and  $G$ -types and  $R$ - and  $L$ -types respectively and branch types are placed in the order  $E, C, G, R, L, J$ , in order to identify and discard  $E$ -links and  $J$ -branches. The state variables  $x_1$  are the  $C$ -type branch voltages and the  $L$ -type link currents, since these characterize the energy storage in the circuit.

In addition, algebraic variables  $x_2$  comprise voltages across and currents through  $G$ - and  $R$ -type branches. The resulting system of equations is a DAE problem:

$$\begin{aligned} Px_1' + Q_{11}x_1 + Q_{12}x_2 &= f_1(t) \\ Q_{21}x_1 + Q_{22}x_2 &= f_2(t). \end{aligned}$$

In the literature, this differential-algebraic system is typically dealt with by reducing it to the ODE (1.10). We now look at two particular formulations.

- **Sparse tableau approach [45]**

We summarize the formulation of the appropriate tableau for the automatic design and analysis of electrical networks. This is useful for calculations of sensitivity to a set of design parameters, and for tolerance analysis, which is often the final step in circuit design. In general, we consider design objectives of the class

$$\min_p \Phi \equiv \int_0^T \phi(w, p, t) dt,$$

where  $\Phi$  is a generalized scalar objective functional and  $\phi$  is an objective function of the unknown vector  $w$ , the design parameter vector  $p$  and time  $t$ . Typically,  $\phi$  is a measure of the difference between some desired output and an obtainable output. The network equations are

$$F(\dots, t)(V, i, v, q, \Phi, p)^T = 0, \quad \begin{cases} V & \text{node voltages} \\ i & \text{branch currents} \\ v & \text{branch voltages} \\ q & \text{energy storage} \end{cases}$$

As can be expected, the tableau operator  $F$  is generally differential-algebraic. The tableau matrix  $F_t(w, t)$  is a linearization of the tableau operator around the operating point  $w$ :  $F_t(w, t) = \partial F(w, t) / \partial w$ . For general networks,  $F_t$  has the form below.

| $V$ | $i$   | $v$     | $q, \Phi, p$ |
|-----|-------|---------|--------------|
| 0   | $A^T$ | 0       | 0            |
| $A$ | 0     | $-I$    | 0            |
| 0   | $Z$   | $Y$     | $B$          |
| 0   | $E$   | $-d/dt$ |              |

The rows, as partitioned, embody KCL, KVL, constitutive relations and dynamic relations describing generalized energy storage elements, respectively. The matrices  $A$  (the topological matrix  $A$  as before),  $E$ ,  $B$ ,  $Y$  and  $Z$  are generally sparse.

- **RC networks [66]**

Sinacev et al. give the following equations for a network containing  $E$  and linear  $C$

and  $R$  elements:

$$\begin{pmatrix} CV'_E \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & -I & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & -R & 0 \\ 0 & -I & 0 & 0 & 0 & 0 & A_E \\ -I & 0 & 0 & 0 & 0 & 0 & A_C \\ 0 & 0 & -I & 0 & 0 & 0 & A_R \\ 0 & 0 & 0 & A_E^T & A_C^T & A_R^T & 0 \end{pmatrix} \begin{pmatrix} V_C \\ V_E \\ V_R \\ I_E \\ I_C \\ I_R \\ V_N \end{pmatrix} = \begin{pmatrix} 0 \\ E \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (1.11)$$

$V_i$  and  $I_i$  are branch voltages and currents ( $i = E, C, R$ ), and  $V_N$  is the node voltage vector. Note that equations (1.5), (1.7) and (1.8) have been used, and there is no need to use the hybrid formulation. Again,  $A_i$  are branch-node incidence matrices.

Most network specifications assume one of the following responses:

1. *DC response*: time domain response of a purely resistive network to constant sources.
2. *AC response*: frequency domain response of a network of linear elements to a sinusoidal signal input.
3. *Transient response*: time domain response of any network.

For some circuits, AC or DC analysis is sufficient, but a fully automated network analysis program must also include transient analysis, which can also handle

1. Nonlinear networks, where element values depend on circuit variables, response is generally not proportional to excitation, and the tableau approach is insufficient, eg. networks containing diodes or transistors; and
2. Time-varying networks, where element values vary with time, but are independent of excitation, eg. the control of circuits by switching and circuits using as input the nonlinear non-sinusoidal output from Van der Pol oscillators.

General network analysis therefore requires effective numerical DAE software.

### Constrained Multibody Systems

"Dynamics of multibody systems is of great importance in the fields of robotics, biomechanics, spacecraft control, road and rail vehicle design, and dynamics of machinery." (W. Schiehlen 1990)

The above quote justifies the substantial amount of work done on DAEs applied to mechanical systems, eg. [11, 13, 34, 43, 65, 97]. Much of the theory of index reduction and maintaining solution invariants of DAEs (in later chapters) was developed from the Euler-Lagrange formulation and the theory of Lagrange multipliers which, for the constrained mechanical system with position  $x$ , velocity  $u$ , kinetic energy  $T(x, u)$ , external

force  $f(x, u, t)$  and constraint  $\phi(x) = 0$ , yield the DAE

$$T_{uu}(x)u' = g(x, u) - G^T(x)\lambda \quad (1.12)$$

$$x' = u \quad (1.13)$$

$$0 = \phi(x), \quad (1.14)$$

with derivative array  $G = \partial\phi/\partial x$  and Lagrange multiplier  $\lambda$ . (Subscripts denote partial derivatives.) If  $T_{uu}(x)$  is invertible (it is typically positive definite), then this system can be posed and solved numerically in several different ways, as we shall see later.

A recent paper on cybernetics [7] presents a graph-theoretic Newton-Euler formulation of constrained multibody systems in order to *"combine the topological information of a constrained rigid body system with the ... equations of motion in a direct, systematic and consistent procedural fashion that lends itself to a straightforward computer implementation."* This graph-theoretic simulation is necessary, as *"classical kinematics and dynamics analysis lack general methods that can handle ... many different types of mechanisms, their components, configurations and arrangements."*

The dynamic simulation problem for mechanical systems is stated as follows:

Given a set of external forces or torques and an initial state of a mechanical system, calculate the positions, velocities and accelerations, and their angular counterparts, as well as internal reaction forces and internal moments at points of interest in the system at a later time.

The paper [7] represents the state-of-the-art of simulation of multibody systems, yet the authors realize that the power of graph-theoretic formulated multibody systems has not yet been fully exploited.

We note that the classic *descriptor* and *state-space* formulations of the equations of constrained mechanical motion both have severe limitations. Some proposed reformulations, with improved numerical properties, are discussed in a recent paper [34].

### 1.1.2 Singular Perturbation

The linear constant-coefficient version of the problem studied here is

$$A(\epsilon)x' = C(\epsilon)x + f, \quad (1.15)$$

which depends on a small positive parameter  $\epsilon$  in such a way that the full ODE, when  $\epsilon > 0$ , is of fuller rank than the reduced equation, obtained by setting  $\epsilon = 0$ . The reduced form is known as the *quasi steady state approximation* to the original equation. Singular perturbation techniques [72, 73, 94, etc.] can be used to solve the smaller, reduced problem, and then to build an approximate solution to the full problem. An analytical study of the system

$$\begin{aligned} x' &= A_1(\epsilon)x + A_2(\epsilon)y, \\ \epsilon y' &= B_1(\epsilon)x + B_2(\epsilon)y, \end{aligned}$$

and its reduced form can be found in [22]. We look more specifically at the problem [47]

$$\begin{aligned} y' &= f(y, z), \\ \epsilon z' &= g(y, z), \quad 0 < \epsilon \ll 1, \end{aligned} \quad (1.16)$$

where it is assumed that

$$\left\langle \frac{\partial g}{\partial z} v, v \right\rangle \leq -\|v\|^2, \quad \forall v \quad (1.17)$$

holds for some scalar product in a neighbourhood of the solution. Outside the initial transient phase, (1.16) is known to have an  $\epsilon$ -expansion of the form

$$\begin{aligned} y(x) &= y_0(x) + \epsilon y_1(x) + \dots + \epsilon^N y_N(x) + O(\epsilon^{N+1}), \\ z(x) &= z_0(x) + \epsilon z_1(x) + \dots + \epsilon^N z_N(x) + O(\epsilon^{N+1}), \end{aligned} \quad (1.18)$$

with smooth  $\epsilon$ -independent coefficients  $y_k, z_k$ . Inserting the expansion (1.18) into the problem (1.16) and comparing powers of  $\epsilon$  yields a sequence of differential-algebraic systems in the coefficients:

$$y'_0 = f(y_0, z_0) \quad (1.19)$$

$$0 = g(y_0, z_0), \quad (1.20)$$

$$y'_1 = f_y(y_0, z_0)y_1 + f_z(y_0, z_0)z_1 \quad (1.21)$$

$$z'_0 = g_y(y_0, z_0)y_1 + g_z(y_0, z_0)z_1, \quad (1.22)$$

and in general

$$y'_k = f_y(y_0, z_0)y_k + f_z(y_0, z_0)z_k + \phi_k(y_0, z_0, \dots, y_{k-1}, z_{k-1}) \quad (1.23)$$

$$z'_{k-1} = g_y(y_0, z_0)y_k + g_z(y_0, z_0)z_k + \psi_k(y_0, z_0, \dots, y_{k-1}, z_{k-1}). \quad (1.24)$$

### 1.1.3 Control Problems

We consider here two basic problems in control theory: optimal and constrained control [17, 53, 70, 71, etc.].

The aim of optimal control is to choose a control  $u$  so as to cause the state  $x$  of a system to exhibit some desired behaviour and minimize a cost functional  $J(x, u)$ . We consider the autonomous control process

$$x' = Ax + Bu \quad (1.25)$$

with quadratic cost functional

$$J = \frac{1}{2} \int_{t_0}^T \langle Hx, x \rangle + \langle Qu, u \rangle dt, \quad (1.26)$$

where  $H$  and  $Q$  are positive definite, and  $Q$  is generally not expected to be invertible. (1.25) and (1.26) are referred to as the basic quadratic cost problem. Using Lagrange multipliers  $\lambda$ ,

$$\begin{pmatrix} I & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \lambda' \\ x' \\ u' \end{pmatrix} + \begin{pmatrix} A^* & H & 0 \\ 0 & -A & -B \\ B^* & 0 & Q \end{pmatrix} \begin{pmatrix} \lambda \\ x \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

is the necessary condition for optimization. Optimal control problems with singular matrices in the quadratic cost functional have received much attention, and they occur naturally as a first order approximation to more general optimal control problems.

A different type of control problem is one where the output is specified. The control equations

$$\begin{aligned} x' &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

result in the DAE system

$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x' \\ u' \end{pmatrix} + \begin{pmatrix} -A & -B \\ C & D \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ y \end{pmatrix}.$$

Another problem, related to the above, is the general continuous linear programming problem, which is to minimize

$$\int_{t_0}^T c^*(t)x(t) dt$$

subject to

$$B(t)x(t) + \int_{t_0}^t K(t,s)x(s) ds = b(t), \quad t \in [t_0, T], \quad (1.27)$$

where each component of  $x$  is positive ( $x > 0$ ). A special case of this is the linear optimal control problem, which is to minimize

$$\int_{t_0}^T c_1^*(t)x(t) + c_2^*(t)u(t) dt$$

subject to

$$\begin{aligned} x' &= Ax + Bu + b_1 \\ 0 &= Cx + Du + b_2 \\ x &> 0, \quad u > 0, \quad x(t_0) = x_0. \end{aligned} \quad (1.28)$$

Note that (1.27) is an integrated form of (1.28).

An example of the use of numerical DAE codes is in the numerical algorithm DAREC [54] for the solution of Riccati differential equations:

$$\begin{aligned} -E^* \dot{X}(t)E &= C^*QC + E^*X(t)A + A^*X(t)E \\ &\quad - (B^*X(t)E + S^*C)^* R^{-1} (B^*X(t)E + S^*C). \end{aligned} \quad (1.29)$$

DAREC reduces (1.29) to a linearly implicit DAE via singular value decomposition of  $E$ , and then calls a numerical linear DAE code to solve it.

### 1.1.4 Fluid Dynamics

The Navier-Stokes equations for incompressible viscous flow [22] are given by

$$\begin{aligned}\partial u / \partial t + (u \cdot \nabla) u &= -\nabla p + \gamma \nabla^2 u \\ \nabla \cdot u &= 0,\end{aligned}\tag{1.30}$$

where  $u$  is the velocity,  $p$  the pressure and  $\gamma$  the kinematic viscosity. After spatial discretization with a finite difference or finite element method the vectors  $U(t)$  and  $P(t)$ , approximating  $u(x, t)$  and  $p(x, t)$ , satisfy the DAE system

$$\begin{aligned}MU' + (K + N(U))U + CP &= f(U, P) \\ C^T U &= 0,\end{aligned}\tag{1.31}$$

where  $M$  is the mass matrix,  $C$  discretizes  $\nabla$  and the forcing function  $f$  emanates from the boundary conditions. The system of PDEs (1.30) has been transformed by the method-of-lines (MOL) into the DAE system (1.31), which has the form

$$\begin{aligned}f_1(x, x', y, t) &= 0 \\ f_2(x) &= 0.\end{aligned}\tag{1.32}$$

An alternative formulation is

$$\begin{aligned}MU' + (K + N(U))U + CP &= f(U, P) \\ AP &= C^T M^{-1}(f - (K + N(U))U),\end{aligned}\tag{1.33}$$

where  $A = C^T M^{-1}C$  discretizes  $\nabla^2$ . An advantage of (1.33) over (1.32) is that (1.33) can be solved using an explicit method. A disadvantage is that the matrix  $A$  may not be very sparse if the finite element approach is taken.

The Euler equations describe a compressible inviscid isentropic medium:

$$\begin{aligned}\partial \rho / \partial t + (u \cdot \nabla) \rho + \rho \nabla \cdot u &= 0 \\ \partial u / \partial t + (u \cdot \nabla) u + \nabla p / \rho &= 0 \\ p - f(\rho) &= 0, \quad \rho = \text{scalar density}.\end{aligned}$$

A finite difference spatial discretization, where  $D$ ,  $U$  and  $P$  approximate  $\rho$ ,  $u$  and  $p$ , respectively, yields the DAE system

$$\begin{aligned}D' + C_1(U)D + C_2(D)U &= 0 \\ U' + C_3(U)U + C_4(D)P &= 0 \\ P - F(D) &= 0,\end{aligned}$$

which has the form

$$\begin{aligned}f_1(x, x', y, t) &= 0 \\ f_2(x, y, t) &= 0.\end{aligned}$$

This can easily be reduced to an ODE system by replacing  $P$  with  $F(D)$ .

## Chapter 2

# Theory

A function  $x(t)$  is a solution of the general nonlinear DAE,

$$F(x, x', t) = 0, \quad (2.1)$$

on an interval  $\mathcal{I}$  if  $x$  is continuously differentiable on  $\mathcal{I}$  and satisfies (2.1) for all  $t \in \mathcal{I}$ . From this point onwards, we are concerned only with the case where solutions to (2.1) exist and are uniquely defined on the interval of interest.

**Definition 2.0.1 [14]** Let  $\mathcal{I}$  be an open subinterval of  $\mathcal{R}$ ,  $\Omega$  a connected open subset of  $\mathcal{R}^{2m+1}$ , and  $F$  a differentiable function from  $\Omega$  to  $\mathcal{R}^n$ . Then the DAE (2.1) is solvable on  $\mathcal{I}$  in  $\Omega$  if there is an  $r$ -dimensional family of solutions  $\phi(c, t)$  defined on a connected open set  $\tilde{\Omega} \times \mathcal{I}$ ,  $\tilde{\Omega} \subset \mathcal{R}^r$ , such that

1.  $\phi(c, t)$  is defined on all of  $\mathcal{I}$  for each  $c \in \tilde{\Omega}$ ;
2.  $(\phi(c, t), \phi'(c, t), t) \in \Omega$  for  $(c, t) \in \tilde{\Omega} \times \mathcal{I}$ ;
3. If  $\psi(t)$  is any other solution with  $(\psi(t), \psi'(t), t) \in \Omega$ , then  $\psi(t) = \phi(c, t)$  for some  $c \in \tilde{\Omega}$ ; and
4. The graph of  $\phi$  as a function of  $(c, t)$  is an  $r + 1$ -dimensional manifold.

This definition says that locally there is an  $r$ -dimensional family of solutions, the solutions are continuous functions of the initial conditions and no bifurcations exist (thereby excluding Type II). We develop our understanding of DAEs by considering them under various structural assumptions, since this approach has proven useful and is now traditional in deriving new algorithms and results.

### 2.1 Linear Constant-Coefficient Problem

In some sense, the simplest or at least the best understood class of truly differential-algebraic problems is that which is linearly implicit with constant coefficients:

$$Ax'(t) + Bx(t) = f(t), \quad x(t_0) = x_0, \quad (2.2)$$

where  $A$  and  $B$  are  $n \times n$  constant matrices. Here and throughout the text, all functions are assumed to be sufficiently continuous and differentiable, unless otherwise stated. We see below that solvability of the DAE problem (2.2) is equivalent to *regularity* of the matrix pencil  $(\lambda A + B)$ , i.e. the invertibility of  $\lambda A + B$  for some  $\lambda$ . Without becoming deeply involved in the theory of matrix nilpotency, we merely state the necessary standard results.

**Definition 2.1.1** [3] For  $A, B \in \mathbb{C}^{n \times n}$  and  $t_0 \in \mathbb{R}$ , the vector  $x_0 \in \mathbb{C}^n$  is said to be a consistent initial condition (c.i.c.) associated with  $t_0$  for (2.2) if (2.2) possesses at least one solution. Equation (2.2) is said to be tractable at the point  $t_0$  if the IVP (2.2) has a unique solution for each consistent initial vector  $x_0$  associated with  $t_0$ .

**Theorem 2.1.2** [22] For  $A, B \in \mathbb{C}^{n \times n}$ , the homogeneous differential equation

$$Ax'(t) + Bx(t) = 0 \quad (2.3)$$

is tractable ("solvable") if and only if there is a  $\lambda \in \mathbb{C}$  such that  $(\lambda A + B)^{-1}$  exists.

The proof is given in [22], where it is also noted that the choice of  $\lambda$  is unimportant, due to scaling. In fact, the development of analytic solutions to (2.3) given in [22] is independent of  $\lambda$ . It is also noted that (2.2) is tractable only if (2.3) is tractable.

When  $\lambda A + B$  is singular for all values of  $\lambda$ , either no solutions exist or infinitely many solutions exist. It is not even reasonable to try to solve these systems numerically. Fortunately, numerical ODE methods reject these problems automatically because, after discretization, they are required to invert the matrix  $A + h\beta B$ , where  $h$  is the steplength and  $\beta$  depends only on the method and recent steplength history. If the tractability condition in Theorem 2.1.2 is not satisfied, then this matrix is singular for all values of  $h$ . The following theorem leads to the solution of (2.2).

**Theorem 2.1.3** [35] If  $(\lambda A + B)^{-1}$  does exist, then there exist nonsingular constant matrices  $P, Q \in \mathbb{C}^{n \times n}$  such that

$$PAQ = \begin{pmatrix} I & 0 \\ & N \end{pmatrix} \quad \text{and} \quad PBQ = \begin{pmatrix} C & \\ & I \end{pmatrix}, \quad (2.4)$$

where  $I$  is the unit matrix, blank matrix entries are zero,  $C$  and  $N$  are constant,  $C$  is invertible and  $N$  is nilpotent of degree  $\nu \leq \dim(N)$ , i.e.  $N^\nu = 0$ ,  $N^{\nu-i} \neq 0$ ,  $i = 1, \dots, \nu$ ,  $\nu \in \mathbb{Z}^+$ . The ODE case, when  $N$  is absent, corresponds to  $\nu = 0$ .

By premultiplying system (2.2) by  $P$ , and introducing new variables  $y$  such that  $x = Qy$ , we obtain an equivalent system (i.e. one that has the same solution):

$$PAQy'(t) + PBQy(t) = Pf(t), \quad y(t_0) = Px_0. \quad (2.5)$$

Using (2.4), (2.5) reduces to two smaller systems;

$$(D) \quad y_1'(t) + Cy_1(t) = f_1(t), \quad (2.6)$$

$$(A) \quad Ny_2'(t) + y_2(t) = f_2(t), \quad (2.7)$$

$$\text{where } y = (y_1^T, y_2^T)^T \text{ and } Pf = (f_1^T, f_2^T)^T. \quad (2.8)$$

Subsystems (2.6) and (2.7) are called the differential (D) and algebraic (A) parts, respectively, and together they are known as the *Kronecker (or Jordan) canonical form (KCF)* of (2.2). Now (D) is a standard system of ODE's, and all the singularity is contained in (A). It is easily verified (by substitution) that the solution of (2.6), (2.7) can be expressed analytically as

$$(D) \quad y_1(t) = e^{-C(t-t_0)} y_1(t_0) + \int_{t_0}^t e^{(s-t)C} f_1(s) ds, \quad (2.9)$$

$$(A) \quad y_2(t) = \sum_{i=0}^{\nu-1} (-N)^i f_2^{(i)}(t). \quad (2.10)$$

Clearly,  $y_1(t_0)$  can be chosen arbitrarily, while  $y_2(t_0)$  must satisfy (2.10) in order to be consistent. This will be dealt with more generally in §3.5. Note the presence of the  $(\nu - 1)$ -th derivative of the driving function  $f$ , and thus its required smoothness. In this context, the degree of nilpotency of  $N$  is called the (nilpotency) index of the DAE system (2.2), denoted by  $\text{Ind}(N)$  or  $\text{Ind}(A, B)$ . In *standard canonical form (SCF)*,  $N$  is strictly triangular. In KCF,

$$N = \text{blockdiag}(N_1, \dots, N_k), \text{ where } N_i = \begin{pmatrix} 0 & 1 & & \\ & 0 & 1 & \\ & & \ddots & \ddots \\ & & & 1 \\ & & & & 0 \end{pmatrix}_{m_i \times m_i} \text{ and } \nu = \max_i \{m_i\},$$

so that (2.7) decouples to  $k$  subsystems of the following form (new subscripts denote components within the  $i$ -th subsystem):

$$N_i y_{2,i}'(t) + y_{2,i}(t) = f_{2,i}(t),$$

$$\text{ie. } y_{2,2}'(t) = -y_{2,1}(t) + f_{2,1}(t), \quad (2.11)$$

$$y_{2,3}'(t) = -y_{2,2}(t) + f_{2,2}(t), \quad (2.12)$$

$$\vdots$$

$$y_{2,m}'(t) = -y_{2,(m-1)}(t) + f_{2,(m-1)}(t), \quad (2.13)$$

$$0 = y_{2,m}(t) + f_{2,m}(t). \quad (2.14)$$

The above system can be solved for  $y_2(t)$  by performing the following in each subsystem:

$$\begin{aligned} \text{Solve (2.14) for } y_{2,m}(t) : & \quad y_{2,m}(t) = -f_{2,m}(t) \\ \text{Substitute } y_{2,m}(t) \text{ into (2.13) :} & \quad y_{2,(m-1)}(t) = f'_{2,m}(t) + f_{2,(m-1)}(t) \\ & \quad \vdots \\ \text{etc.} & \quad \vdots \\ \text{until :} & \quad y_{2,1}(t) = \pm f_{2,m}^{(m-1)}(t) + \dots \end{aligned}$$

Note that this routine requires  $N$  in SCF, not necessarily in KCF. The above shows that higher-index problems require more continuity in  $f_2$ . In fact, an index  $\nu$  problem requires  $f_2$  to be  $C^{\nu-1}$  in some of its components. Later we will see that these higher order derivatives drastically affect the truncation error of ODE methods, and this is the root of the problem with using ODE methods. In general, index 1 problems are easy to solve, whereas higher-index problems are not easy.

One might hope that the study of (2.2) could be used as a guide for understanding more complicated systems. In general, this fails to be true. The structure of the local constant-coefficient linearized version of (2.1) may not describe the behaviour of solutions to the DAE, for nonlinear or even linear variable-coefficient systems where  $\nu > 1$ . Numerical methods which work for (2.2) break down when the matrices are time-dependent or where  $\nu > 1$ .

## 2.2 Linear Variable-Coefficient Problem

The solution of the linear variable-coefficient DAE system,

$$A(t)x'(t) + B(t)x(t) = f(t), \quad x(t_0) = x_0, \quad (2.15)$$

is a much more intricate problem. For example, there is no equivalence between the solvability of (2.15) and the regularity of the matrix pencil  $(\lambda(t)A(t) + B(t))$  for all  $t$ . There are examples where an irregular DAE can be transformed into a regular solvable DAE [22]. Also, a regular DAE need not be solvable [41].

Although the properties of (2.15) are not as yet fully known, unlike the constant-coefficient problem, we study them because they arise naturally in applications and they also give us insights into the general nonlinear problem.

### 2.2.1 Solvability

Consistent with Definition 2.0.1, a singular system (2.15) is solvable on  $I = [t_0, T]$  if, for any sufficiently smooth  $f(t)$ , solutions to (2.15) exist and are uniquely determined by their values at  $t_0$ . As for the constant-coefficient case, a system (2.15) in the form

$$(D) \quad y_1'(t) + C(t)y_1(t) = f_1(t) \quad (2.16)$$

$$(A) \quad N(t)y_2'(t) + y_2(t) = f_2(t), \quad (2.17)$$

where  $N(t)$  is nilpotent,  $s \times s$  and strictly upper (or lower) triangular, is in SCF. If  $N$  is constant (ie.  $N(t) = N$ ), the system is said to be in *strong SCF (SSCF)* or *semi-canonical*

form. The relationship between solvability of (2.15) and the existence of the SCF is given below.

**Theorem 2.2.1** [9] *If  $A(t)$  and  $B(t)$  are real analytic on an interval  $I$ , then the singular system (2.2) is solvable iff there are transformations of the form  $x = Q(t)y$  and left-multiplications  $P(t)$  (where  $P$  and  $Q$  are analytic and nonsingular), which transform (2.2) to SCF everywhere on  $I$ . (For generally non-real  $A$  and  $B$ , see [14, p.28].)*

The proof of Theorem 2.2.1 yields an algorithm for computing the SCF. If (2.3) is transformable to SCF by  $(P, Q)$ , then we have

$$\begin{aligned} P(t)A(t)Q(t) &= \begin{pmatrix} I & \\ & N(t) \end{pmatrix}, \\ P(t)A(t)Q'(t) + P(t)B(t)Q(t) &= \begin{pmatrix} C(t) & \\ & I \end{pmatrix}. \end{aligned} \quad (2.18)$$

It is the time-varying coordinate change,  $x = Q(t)y$ , which causes the derivative  $Q'(t)$  in (2.18). If  $Q$  can be chosen constant, such that  $Q' \equiv 0$ , the transformations are called *safe* and the solution takes the same form as (2.9), (2.10). We now give an example of a Type II problem which fails to be analytically solvable.

**Example 2.1** *The problem*

$$\begin{pmatrix} t-1 & -1 \\ 0 & \end{pmatrix} y' + y = \begin{pmatrix} 0 \\ e^t \end{pmatrix}$$

has the solution  $y = \left( \frac{e^t + c}{t-1}, e^t \right)^T$ ,  $c$  constant, at all points except the turning point,  $t^* = 1$ . Note that the SCF for this example changes structure at  $t^*$ .

## 2.2.2 Index

The definition of index given in §2.1 can be extended to the variable-coefficient case, and there are at least three possible ways to define the index. The *local index* at a time  $\hat{t}$  is defined as  $\nu_1(\hat{t}) = \text{Ind}(A(\hat{t}), B(\hat{t}))$ , the index of the constant-coefficient problem at  $\hat{t}$ . So there exist nonsingular matrices  $\hat{P}$  and  $\hat{Q}$  such that

$$\hat{P}A(\hat{t})\hat{Q} = \begin{pmatrix} I & \\ & \hat{N} \end{pmatrix}, \quad \hat{P}B(\hat{t})\hat{Q} = \begin{pmatrix} \hat{C} & \\ & I \end{pmatrix},$$

and  $\nu_1(\hat{t}) = \text{Ind}(\hat{N})$ . If (2.15) is transformable to SCF (2.16), (2.17) by a safe transformation  $(P, Q)$ , then the index,  $\nu(t)$ , is defined as the nilpotency of  $N(t)$  in (2.17). (2.18) is called a *fixed index  $\nu$  system* if the local index is the same for all  $t$ . If a system (2.15) is transformable to SSCF, then  $\nu(t) = \nu = \text{Ind}(N)$  is called the *global index*.

**Example 2.2** If a system (2.2) is transformable to SCF with

$$N(t) = \begin{pmatrix} 0 & t & 0 & 0 \\ 0 & 0 & 1-t & 0 \\ 0 & 0 & 0 & 1-t \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

then

$$\nu_1(t) = \begin{cases} 4 & \text{for } t \neq 0, 1 \\ 3 & \text{for } t = 0 \\ 2 & \text{for } t = 1 \end{cases}$$

and we have a variable-index higher-index system.

It is the global index that determines the behaviour of the solution. If the global index is a constant  $m$ , we know that  $\dim(D)$  independent initial values can be chosen, and that the driving term  $f$  can be subject to differentiation  $m-1$  times.

The local index in some sense governs the behaviour of the numerical method. For example, if the matrix pencil is singular at a point  $t^*$ , numerical ODE methods cannot solve the problem at  $t^*$ . In understanding why these methods break down, we need to know how global and local indices are related.

In [41] it is shown that if the local index  $\leq 1$ , then it is not changed by a smooth transformation  $(P, Q)$ . Otherwise a smooth, nonconstant transformation of variables in (2.15) will yield a system whose local index is 2 unless additional constraints are satisfied by the transformation. A restricted set of transformations will cause the index to be greater than 2, or the pencil to be singular. When the transformation to SSCF is used, this shows the relationship between the local and global indices.

**Example 2.3** Consider the system

$$\begin{pmatrix} 0 & 0 \\ 1 & \eta t \end{pmatrix} y' + \begin{pmatrix} 1 & \eta t \\ 0 & 1+\eta \end{pmatrix} y = \begin{pmatrix} g(t) \\ 0 \end{pmatrix}. \quad (2.19)$$

This was obtained from

$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} x' + x = \begin{pmatrix} g(t) \\ 0 \end{pmatrix}$$

by a change of variables,  $x = \begin{pmatrix} 1 & \eta t \\ 0 & 1 \end{pmatrix} y$ , so the global index is 2. If  $\eta = -1$ , the pencil in (2.19) is singular. Otherwise it is easy to verify that its local index is also 2, using the transformation

$$P = I, \quad Q = \begin{pmatrix} 1 & \frac{-\eta t}{1+\eta} \\ 0 & \frac{1}{1+\eta} \end{pmatrix}.$$

**Example 2.4** Consider the system

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & \eta t \\ 0 & 1 & 0 \end{pmatrix} x' + \begin{pmatrix} 1 & 0 & \eta t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} g(t) \\ 0 \\ 0 \end{pmatrix}. \quad (2.20)$$

This can be seen to have local index 3 by the transformation

$$P(t) = I, \quad Q(t) = \begin{pmatrix} 1 & 0 & -\eta t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (2.21)$$

and global index 2 by the transformation

$$P(t) = I, \quad Q(t) = \begin{pmatrix} 1 & t & -t \\ 0 & 0 & 1 \\ 0 & -1/\eta & 1/\eta \end{pmatrix}. \quad (2.22)$$

The definitions for the local and global indices are useful for studying the underlying structure of complicated DAE systems, but they fail to provide us with any kind of simple procedure for finding the index of a given DAE system, for which Algorithm 2.2.5 below is useful. (See also [14, p.30].)

We first mention two structural forms arising in applications whose solvability we do know about.

**Definition 2.2.2** [14] The DAE (2.15) is in Hessenberg form of size  $r$  if it can be written as

$$\begin{pmatrix} I & 0 & \cdot & \cdot & 0 \\ 0 & I & \cdot & \cdot & \cdot \\ \cdot & \cdot & I & \cdot & \cdot \\ \cdot & \cdot & \cdot & I & \cdot \\ 0 & \cdot & \cdot & \cdot & 0 \end{pmatrix} \begin{pmatrix} x'_1 \\ \cdot \\ \cdot \\ \cdot \\ x'_r \end{pmatrix} + \begin{pmatrix} B_{11} & * & * & B_{1,r-1} & B_{1r} \\ B_{21} & * & * & B_{2,r-1} & 0 \\ 0 & * & * & * & \cdot \\ \cdot & \cdot & * & * & \cdot \\ 0 & \cdot & 0 & B_{r,r-1} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_r \end{pmatrix} = \begin{pmatrix} f_1 \\ \cdot \\ \cdot \\ \cdot \\ f_r \end{pmatrix}$$

where  $x_i$  are vectors,  $B_{ij}$  (also represented by  $*$ ) are matrices, and the Hessenberg condition, that the product  $B_{r,r-1}B_{r-1,r-2}\dots B_{1r}$  is nonsingular, is satisfied.

In [14] it is proposed that a DAE in Hessenberg form of size  $r$  is solvable and has global and local index  $r$ . Often in applications, systems appear only as a combination of the solvable forms previously mentioned.

**Definition 2.2.3** [14] The DAE (2.15) is an  $r$ -th order triangular chain if it is in the form

$$\begin{pmatrix} A_{11} & 0 & \cdot & 0 \\ A_{21} & A_{22} & \cdot & \cdot \\ * & * & * & 0 \\ A_{r1} & * & * & A_{rr} \end{pmatrix} \begin{pmatrix} x'_1 \\ \cdot \\ \cdot \\ x'_r \end{pmatrix} + \begin{pmatrix} B_{11} & 0 & \cdot & 0 \\ B_{21} & B_{22} & \cdot & \cdot \\ * & * & * & 0 \\ B_{r1} & * & * & B_{rr} \end{pmatrix} \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ x_r \end{pmatrix} = \begin{pmatrix} f_1 \\ \cdot \\ \cdot \\ f_r \end{pmatrix}$$

and each subsystem  $A_{ii}x_i' + B_{ii}x_i = f_i - \sum_{j=1}^{i-1} (A_{ij}x_j' + B_{ij}x_j)$  is either global index one or zero, in Hessenberg form, or in SCF.

Any system which is analytically equivalent to a triangular chain is solvable. In Theorems 3.4.1 and 3.4.2, characterizations of solvability and consistent initial conditions of other higher-index problems are obtained using Taylor-type methods.

### 2.2.3 Index Reduction

It is sometimes possible to use analytical techniques to rewrite the system in a form with lower index, which can then be solved numerically. We discuss two reduction techniques, the first of which is useful for determining the index of a system, and reducing it, while the second is the idea behind the technique for constructing the transformation matrices  $P(t)$  and  $Q(t)$  of Theorem 2.2.1.

#### Algorithm 2.2.4 [42]

1. If  $A(t)$  in (2.15) is nonsingular for all  $t$ , then we are done.

2. Otherwise, premultiply (2.15) by a nonsingular matrix  $P(t)$  to zero out a maximal number of rows of  $A$  and permute them to the bottom, to obtain

$$\begin{pmatrix} A_{11} \\ 0 \end{pmatrix} x' + \begin{pmatrix} B_{11} \\ B_{12} \end{pmatrix} x = g(t).$$

3. Differentiate the bottom half of the system, to obtain the new system

$$\begin{pmatrix} A_{11} \\ B_{12} \end{pmatrix} x' + \begin{pmatrix} B_{11} \\ B_{12}' \end{pmatrix} x = \hat{g}(t).$$

4. Now apply steps 1-3 to the new system.

The idea behind this algorithm is that each differentiation of the algebraic constraints yields an equivalent system of lower index. For a solvable linear system (2.15) with no turning points, Algorithm 2.2.4 terminates (resulting in an ODE system) in  $m$  iterations iff the global index is  $m$ , and does not terminate at all for systems which are not solvable. Note that we have introduced a number of constants of integration, which means that we must determine the correct initial conditions. This can be done by satisfying the initial system and each of its differentiated forms at the initial point (cf. §3.5). Algorithm 2.2.4 can be used to find the local index by considering matrices  $A$  and  $B$  at some  $t = \bar{t}$  and applying the algorithm to the resulting system, in which case the algorithm terminates in  $m$  steps iff the local index is  $m$ . Since the local index is equal to the global index if the index is 1, the algorithm terminates after one iteration iff the index is 1.

Another approach to simplifying the system (2.15) is based on using the algebraic constraints to solve for some variables in terms of the remaining variables, thus reducing

the size of the system. This approach, embodied in the algorithm below, can be used to reduce solvable systems (2.15) to semi-canonical form. When it cannot be completed, the system is not solvable.

**Algorithm 2.2.5 [42]**

1. If  $A(t)$  in (2.15) is nonsingular for all  $t$ , then we are done.
2. Otherwise, premultiply (2.15) by a nonsingular matrix  $P(t)$  to zero out a maximal number of rows of  $A$ , and permute them to the bottom, to obtain

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} + \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} g_1(t) \\ g_2(t) \end{pmatrix}.$$

3. Permute the columns of  $A$  and  $B$  so that  $B_{22}$  is nonsingular. (It can be shown that for solvable systems  $(B_{21}, B_{22})$  has full rank.)
4. Solve for  $x_2 = B_{22}^{-1}(g_2 - B_{21}x_1)$  and differentiate to solve for  $x_2'$  in terms of  $x_1$  and  $x_1'$ .
5. Substitute the expressions for  $x_2$  and  $x_2'$  into the top half of the system, and rewrite to obtain a system of form (2.15) for  $x_1$ . This system is always smaller than the original one.
6. Go back to step 1.

**Example 2.5** For the RC network equation (1.11), it can be shown that if the columns of  $(A_E A_C A_R)^T$  and the rows of  $(A_E A_C)^T$  are linearly independent, then the index is 1. The first condition is equivalent to requiring that the network is connected, and the second that there are no loops containing  $E$  and  $C$  elements only. If the second condition is satisfied, (1.11) can be reduced to state variable form

$$\begin{pmatrix} A_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} y' + \begin{pmatrix} 0 & A_2 & 0 \\ A_2^T & 0 & 0 \\ B_1 & B_2 & I \end{pmatrix} y = g(t), \quad (2.23)$$

where  $A_1$  is invertible. Making use of Algorithm 2.2.4, we differentiate rows 2 and 3 and substitute for the upper part of  $y'$  from row 1:

$$\begin{pmatrix} A_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & B_2 & I \end{pmatrix} y' + \begin{pmatrix} 0 & A_2 & 0 \\ A_2^T & -A_2^T A_1^{-1} A_2 & 0 \\ B_1 & B_2 - B_1 A_1^{-1} A_2 & 0 \end{pmatrix} y = \hat{g}(t).$$

Swapping rows 2 and 3, the index is 1 if  $(A_2^T A_1^{-1} A_2)^{-1}$  exists, i.e. iff  $A_2$  has linearly independent columns, and it follows that the global index of (2.23) is 2. That the local index is also 2 can be seen using a similar argument applied to the local linearization of (2.23).

### 2.2.4 Analytical Solutions

For completeness we consider the analytical solution [9] of system (2.15) in its SCF (2.16), (2.17).

#### D-part

The D-part (2.16) is a standard ODE system, and it is well-known that if  $C(t)$  is Lipschitz continuous, the corresponding homogeneous system,  $y_1'(t) + C(t)y_1(t) = 0$ , has a continuously differentiable fundamental matrix solution  $\Phi(t)$ , such that  $\Phi' + C(t)\Phi = 0$ . Using  $\Phi$ , the solution of (2.16) can be written as

$$y_1(t) = \Phi(t)\Phi^{-1}(t_0) y_1(t_0) + \Phi(t) \int_{t_0}^t \Phi^{-1}(s) f_1(s) ds, \quad (2.24)$$

where  $y_1(t_0)$  is an arbitrarily chosen initial condition. Note that for the constant-coefficient case  $C(t) \equiv C$  and  $\Phi(t) = e^{-tC}$ , so (2.24) is equivalent to (2.9). If  $C$  commutes with its derivative then

$$\begin{aligned} \Phi(t)\Phi^{-1}(t_0) &= \exp\left(\int_{t_0}^t -C(s) ds\right), \\ \text{otherwise } \Phi(t)\Phi^{-1}(t_0) &= T(t) \exp\left(\int_{t_0}^t \Lambda(s) ds\right) T^{-1}(t_0), \end{aligned}$$

where  $\Lambda$  is diagonal. These exponential representations are of interest in asymptotic estimates of  $y_1(t)$ , i.e. in finding estimates of  $\|y_1(t)\|$  as well as  $\|\Phi(t)\Phi^{-1}(t_0)\|$ .

#### A-part

**Theorem 2.2.6** *The unique solution of (2.17) can be written as*

$$y_2(t) = \sum_{i=0}^{s-1} (-N(t)\mathcal{D})^i f_2(t),$$

where  $\mathcal{D}$  is the differentiation operator ( $\mathcal{D}y = y'(t)$ ).

This is proved by:

1. Substitution into (2.17);
2.  $(-N(t)\mathcal{D})^i f(t) = 0, i \leq s$  (proved by induction on  $i$  using the triangularity of  $N$ );
3. Uniqueness follows by *reductio ad absurdum*.

At the component level,  $y_2$  can be computed as

for  $i := s$  downto 1,

$$y_{2,i}(t) := f_{2,i}(t) - \sum_{j=i+1}^s n_{ij}(t) y'_{2,j}(t), \quad \text{where } N = [n_{ij}].$$

(This is easily proved by substituting components in reverse order into (2.17).)

## 2.3 General Nonlinear DAE Systems

A general DAE takes the fully implicit form

$$F(x, x', t) = 0, \quad (2.25)$$

where  $x(t) \in \mathcal{R} \rightarrow \mathcal{R}^n$  and  $F \in \mathcal{R}^n \times \mathcal{R}^n \times \mathcal{R} \rightarrow \mathcal{R}^n$  is a function for which we will assume suitable differentiability. In this section, we generalize the concepts of index and index transformation and reduction, to the above problem.

### 2.3.1 Index

At least two forms of index for the general problem appear in the literature. The first, used by a number of authors, including Gear, Lötstedt and Petzold, in [38, 42, 66, etc.], is the number of times the DAE must be differentiated in order to generate an ODE satisfied by the solution. This can be seen to be equivalent to the global index defined for linear systems, by referring to Algorithms 2.2.4 and 2.2.5 and we call it the *differential index*  $d_i$ . More recently, Hairer, Lubich and Roche [47] define the index by considering the effect of perturbations of the equations on the solutions. This index, which we call the *perturbation index*  $p_i$ , is one more than the highest order derivative of the perturbation appearing in the estimate of the bound of the change in the solution. We discuss both indices and the relationships between them.

#### Differential Index $d_i$

1. If the Jacobian  $\partial F / \partial x'$  is nonsingular, the index  $d_i = 0$ . Then (2.25) is an implicit ODE which can, in principal, be inverted, yielding  $x' = f(x, t)$ .
2. Otherwise ( $d_i > 0$ ) consider

$$\begin{aligned} F(x, x', t) &= 0 \\ dF/dt &= (\partial F / \partial x') x'' + (\partial F / \partial x) x' + (\partial F / \partial t) = 0 \\ d^2 F / dt^2 &= (\partial F / \partial x') x''' + \dots = 0 \\ &\vdots \\ d^s F / dt^s &= (\partial F / \partial x') x^{(s+1)} + \dots = 0 \end{aligned} \quad (2.26)$$

as a system of equations in the *separate* dependent variables  $x', x'', \dots, x^{(s+1)}$ , and solve for these variables as functions of  $x$  and  $t$  considered as independent variables. Since  $\partial F / \partial x'$  is singular, it will not be possible to solve for  $x^{(s+1)}$ , or possibly even for  $x^{(q)}$  for smaller  $q$ . If it is possible to solve for  $x'$  for some finite  $s$ , then the index  $d_i$  is defined as the smallest  $s$  for which (2.26) can be solved for  $x'(x, t)$ .

The relationship between  $d_i$  and the index as defined earlier, can easily be seen for linear DAEs in semi-canonical form. For the differential subsystem

$$Ny' + y = f(t), \quad (2.27)$$

$$\begin{aligned} \text{ie. } y_2' + y_1 &= f_1(t) \\ &\vdots \\ y_m' + y_{m-1} &= f_{m-1}(t) \\ y_m &= f_m(t), \end{aligned}$$

we differentiate the  $i$ -th equation  $i$  times ( $i = 1, 2, \dots, m$ ) to obtain

$$\begin{aligned} y_2'' + y_1' &= f_1'(t) \\ &\vdots \\ y_m^{(m)} + y_{m-1}^{(m-1)} &= f_{m-1}^{(m-1)}(t) \\ y_m^{(m)} &= f_m^{(m)}(t). \end{aligned}$$

Adding up these equations with alternating signs produces  $y_1'$ . This, together with the first  $m-1$  equations of (2.27), gives an explicit ODE, so that  $d_i = \text{Ind}(N)$ .

The differential index determines some major characteristics of the underlying structure, for example the number of manifolds on which the solution is constrained to lie, which is the number of sets of algebraic relations developed in the index reduction process described below.

### Perturbation Index $p_i$

The perturbation index  $p_i$  is defined as the smallest integer  $m$  such that the difference between the solution of (2.25) and the solution of the perturbed equation

$$F(y, y', t) = \varepsilon(t)$$

can be bounded by an expression of the form

$$\begin{aligned} \max_{t \in [t_0, T]} \|y(t) - x(t)\| &\leq k \left[ \|x(t_0) - y(t_0)\| + \max_{t \in [t_0, T]} (\|\varepsilon(t)\|) \right. \\ &\quad \left. + \max_{t \in [t_0, T]} (\|\varepsilon'(t)\|) + \dots + \max_{t \in [t_0, T]} (\|\varepsilon^{(m-1)}(t)\|) \right], \quad (2.28) \end{aligned}$$

where  $k$  is an arbitrary constant and  $\|\cdot\|$  denotes a suitable vector norm. To allow for  $p_i$  to be zero, we include within the square brackets the term

$$\max_{t \in [t_0, T]} \left( \|\epsilon^{(-1)}(t)\| \right); \quad \text{where} \quad \epsilon^{(-1)}(t) = \int_{t_0}^t \epsilon(\tau) d\tau.$$

For (2.27),  $p_i = \max \{m_i\} = \text{Ind}(N)$ , since  $y_1$  depends on the  $(m-1)$ -th derivative of  $f_m$ .

The perturbation index is an important factor in the numerical solution of a DAE due to its major rôle in determining the impact of roundoff errors. If the method performs numerical differentiation on roundoff errors of  $O(\epsilon)$ , it is clear that we can expect resulting errors of  $O(\epsilon/h^{p_i-1})$ .

### Relationship between $d_i$ and $p_i$

We restrict ourselves to problems for which both indices are defined. It is always possible to restate a problem in a form for which the formal definitions do not apply. For example, the problem  $x^2 = 0$  has no perturbation index, since the bound (2.28) does not exist. (One differentiation yields  $xx' = 0$ , which has two solutions,  $x' = 0$  with  $d_i \neq 1$ , and  $x = 0$  with  $d_i = 2$ .) In [40], the following relations are given.

1.  $d_i \leq p_i \leq d_i + 1$ .
2.  $p_i = d_i$  if the DAE has integral form, ie. if  $F(x, x', t) \equiv a_x(x, t)x' + b(x, t) = 0$ . In particular, linear DAEs in semi-canonical form have  $p_i = d_i = \text{Ind}(A)$ , as shown earlier.
3.  $p_i = d_i + 1$  for general systems, unless the DAE satisfies some special conditions which are difficult to describe in terms of the original DAE.
4. An explicit or implicit ODE system has  $d_i = 0$ ,  $p_i = 1$ .

### 2.3.2 Structural Forms

Nonlinear versions of all the forms in the previous section appear in the literature.

**Definition 2.3.1** [14] *The DAE (2.25) is in Hessenberg form of size  $r$  if it has the form*

$$\begin{aligned} x_1' &= F_1(x_1, x_2, \dots, x_r, t) \\ x_2' &= F_2(x_1, x_2, \dots, x_{r-1}, t) \\ &\vdots \\ x_i' &= F_i(x_{i-1}, x_i, \dots, x_{r-1}, t), \quad 3 \leq i \leq r-1, \\ &\vdots \\ 0 &= F_r(x_{r-1}, t) \end{aligned}$$

and  $(\partial F_r / \partial x_{r-1})(\partial F_{r-1} / \partial x_{r-2}) \dots (\partial F_2 / \partial x_1)(\partial F_1 / \partial x_r)$  is nonsingular (Hessenberg condition).

Remember that the  $x_i$  are not components but vectors such that  $x = (x_1^T, \dots, x_r^T)^T$ . Assuming the  $F_i$  to be sufficiently differentiable, the Hessenberg form of size  $r$  is solvable and has local, differential and perturbation indices  $r$  [14]. (The Hessenberg form is of integrator form.)

**Definition 2.3.2** [14] *The DAE (2.25) is a triangular chain of order  $r$  if it can be written in the form*

$$\begin{aligned} F_1(x'_1, x_1, t) &= 0 \\ F_2(x'_1, x'_2, x_1, x_2, t) &= 0 \\ &\vdots \\ F_r(x'_1, \dots, x'_r, x_1, \dots, x_r, t) &= 0 \end{aligned}$$

and the system defined by each function  $F_i$  is an explicit or implicit ODE, index one, a Hessenberg form, or in SCF in the variable  $(x_1^T, \dots, x_i^T)^T$ .  $\parallel$

Triangular chains are always solvable.

### 2.3.3 Index Reductions

From this point onwards, the term 'index' (or  $\nu$ ) refers to the global or differentiation index  $d_i$ . Firstly, we mention that Algorithm 2.2.4 applies directly to problems of the form (2.25), when  $F$  is linear in  $x'$ . We now consider the following special form of DAE:

$$\begin{aligned} u' &= f(u, v, t) \\ 0 &= g(u, v, t). \end{aligned} \quad (2.29)$$

It is pointed out in [89, 96] (and shown above for the linear case) that these equations may not have continuous solutions. Here, we show an equivalence between semi-explicit equations (2.29) and fully implicit DAEs (2.25) of index one lower. We also give transformations between (2.29) and (2.25), and show how a problem can be reduced, without introducing additional constants of integration, to an index 1 semi-explicit form, which can then be reduced to an implicit ODE whose solution yields the solution to the original problem by a differentiation and a linear transformation.

1. We can transform (2.25) into (2.29) by replacing  $x'$  with  $v$  and  $x$  with  $u$ , to get

$$\begin{cases} u' = v \\ 0 = F(u, v, t). \end{cases}$$

2. We can also transform (2.29) into (2.25) by replacing  $u$  with  $x_1$  and  $v$  with  $x'_2$  to get

$$\begin{aligned} x'_1 - f(x_1, x'_2, t) &= 0 \\ g(x_1, x'_2, t) &= 0. \end{aligned} \quad (2.30)$$

Gear <sup>(2)</sup> conjectures that conversion (1) (or (2)) above always raises (or lowers) the index by one unless (2.29) is an ODE, i.e. there is no  $g$ , in which case there is no index change.

Note that although (2) reduces the index, we still have to differentiate  $x_2$  to recover the original variable  $v$ . So the effective index has stayed the same, as far as the numerical method is concerned. Also, restating (2.29) in the form (2.25) and then using (2) to transform back to (2.29),

$$\begin{aligned} x_1' - f(x_1, x_2, t) &= 0 \\ g(x_1, x_2, t) &= 0 \end{aligned} \quad (2.31)$$

$$\begin{aligned} \Rightarrow \quad u_1' &= v_1 \\ u_2' &= v_2 \\ v_1 - f(u_1, u_2) &= 0 \\ g(u_1, u_2) &= 0, \end{aligned}$$

formally increases the index by one, but only because it requires the differentiation of additional variables  $v_1$  and  $v_2$  to compute  $u''$  and  $v''$  unnecessarily.

### Reduction Algorithm

(Note similarity with Algorithms 2.2.4 and 2.2.5.) The index of the semi-explicit system (2.29) can be reduced by 1 by replacing the algebraic equation in (2.29) with its derivative

$$g_u f(u, v, t) + g_v v' + g_t = 0. \quad (2.32)$$

To remove constants of integration, we can consider (2.29) and (2.32) simultaneously, resulting in an overdetermined system. This problem is dealt with in [37] by introducing Lagrange multipliers whose solutions are zero. The procedure formally differentiates (2.29) to get an index 0 system and then reintroduces the algebraic constraints. This may not always be practical, however.

Let  $r_1$  be the rank of  $F_y$  in (2.25). If  $r_1 = n$ , the index is already 0. Otherwise, there is a nonsingular  $r_1 \times r_1$  submatrix of  $F_y$ , assumed to be the upper principal minor. By the implicit function theorem the first  $r_1$  equations in (2.25) can be solved for the first  $r_1$  components of  $x'$ . Now break  $x$  into  $(x_1^T, v_1^T)^T$ ,  $\dim(x_1) = r_1$ , so that

$$x_1' = e_1(x_1, v_1, v_1', t) = 0. \quad (2.33)$$

Substitute this into the last  $n - r_1$  equations in (2.25) to get the implicit relationship

$$g_1(x_1, v_1, t) = 0. \quad (2.34)$$

Differentiating this gives

$$g_{1,x_1} e_1(x_1, v_1, v_1', t) + g_{1,v_1} v_1' + g_{1,t} = 0. \quad (2.35)$$

Now (2.33), (2.34) is equivalent to (2.25), while (2.33), (2.35) is of index one lower.

We then repeat this process, which terminates after  $m$  steps if the index is  $m$ , yielding the set of ODE's

$$\dot{x}_i = f_i(x, t). \quad (2.36)$$

The solutions of (2.36) include the solutions of (2.25) but also additional solutions due to constants of integration introduced by differentiation. Algebraic equations of the form (2.34), generally written as

$$g(x, t) = 0, \quad g = \{g_i\}, \quad (2.37)$$

need to be appended to (2.36), yielding an overdetermined system. So we then introduce an additional vector variable  $\mu$ ,  $\dim(\mu) = \dim(g)$ , and add the term  $g_{x_i}^T \mu$  to the RHS of (2.36), yielding

$$\dot{x}_i - f_i(x, t) - g_{x_i}^T \mu = 0. \quad (2.38)$$

**Theorem 2.3.3** (Proved in [38].) Any system of equations (2.38) for  $j = 1 \dots m$  and (2.37) has  $\mu = 0$  and  $x$  as a solution of (2.25). Conversely, any solution of (2.25) is a solution to (2.38) and (2.37) with  $\mu = 0$ .

The above process has reduced (2.25) to the index 2 semi-explicit form (2.37), (2.38). The index can be reduced to 1 by defining the minimal form, ie. by replacing  $\mu$  with  $\nu'$ , to get

$$\begin{aligned} \dot{x}' &= f(x, t) + G^T \nu', & G &= g_x \\ g(x, t) &= 0. \end{aligned}$$

$G$  is referred to as the *derivative array*.

**Example 2.6** We illustrate by using the above procedure on the constrained Euler-Lagrange equations (cf. §1.1.1). Such systems always have index 3 and can easily be written in semi-explicit form.

• **Index 3 formulation (position level, descriptor form)**

The original system

$$\dot{u}' = T_{uu}^{-1} h(x, u') - T_{uv}^{-1} h \quad (2.39)$$

$$\dot{x}' = u \quad (2.40)$$

$$0 = \phi(x), \quad (2.41)$$

with  $H = \partial\phi/\partial x$  (note the change in notation from §1.1.1 and the assumed invertibility of  $T_{uu}$ ) is Hessenberg of size 3 if

$$HT_{uu}^{-1}H^T \text{ is invertible,} \quad (2.42)$$

ie. if  $H$  has linearly independent rows.

- **Index 2 formulation (velocity level)**

Differentiate (2.41) (with respect to  $t$ ), yielding

$$0 = H(x)u, \quad (2.43)$$

and replace (2.41) with (2.43). This system is Hessenberg of size 2 if (2.42) is satisfied.

- **Index 1 formulation (acceleration level)**

Differentiate (2.41) again, yielding

$$0 = H_x(x)(u, u) + H(x)T_{uu}^{-1}(x)(h(x, u) - H^T(x)\lambda) \quad (2.44)$$

Replacing (2.41) with (2.44), the index 1 assumption is (2.42).

- **Index 0 formulation**

Differentiate (2.41) again, yielding

$$\lambda' = \lambda'(x, u). \quad (2.45)$$

The resulting explicit ODE system, (2.39), (2.40), (2.45), has no constraint stabilization.

- **F&L formulation [34]**

Append the constraints (2.41), (2.43), (2.44) (collectively  $0 = g(x)$ ) to the differential system (2.39), (2.40), (2.45) shifted by  $G^T \mu$ , where  $G = \partial g / \partial x$ . This system has index 2 by the index reduction technique if (2.42) holds. Defining  $\mu = \nu'$ , however, does not seem to allow the explicit solution for  $\nu$  in a symbolic form.

- **GGL formulation [43]**

An incomplete form of the above, satisfying only position and velocity level constraints, is obtained from the index 3 formulation by shifting only the velocity equation (2.40) by  $H^T \beta$  and including (2.41). This system satisfies the index 2 assumption, again provided that (2.42) is satisfied.

**Example 2.7** The Navier-Stokes equations of §1.1.4,

$$\begin{aligned} MU' + (K + N(U))U + CP &= f(U, P) \\ C^T U &= 0, \end{aligned} \quad (2.46)$$

have both local and global index 2 if either  $\partial f / \partial P = 0$  and  $C$  has full column rank, or  $C^T M^{-1}(C - \partial f / \partial P)^{-1}$  exists (usually true). The alternative formulation

$$\begin{aligned} MU' + (K + N(U))U + CP &= f(U, P) \\ AP &= C^T M^{-1}(f - (K + N(U))U), \end{aligned} \quad (2.47)$$

with  $A = C^T M^{-1}C$ , can be obtained from (2.46) by one implementation of the index reduction algorithm, and thus has index 1.

## Chapter 3

# Numerical Solution

### 3.1 Introduction

As there is no closed-form solution to the general DAE problem,

$$F(x, x', t) = 0, \quad (3.1)$$

we turn to numerical approximation techniques for the solution of DAEs. In fact, characterizations of solvability and index for linear and more general DAEs can be derived using the Taylor-type numerical methods of §3.4.2. We have looked at techniques for changing higher-index systems to lower-index ones so that, in theory at least, DAEs can be reduced to ODEs and solved by conventional numerical ODE methods [58]. In this chapter, we look at numerical methods for solving DAEs as given, for the following very important reasons.

- Reductions of index might be costly, and we might not even know how to effect them, either analytically or numerically.
- Reductions generally destroy any sparseness present in the original formulation. This is of extreme importance, since the major areas of application require solution of very large, very sparse systems. For example, in the transient network analysis of integrated circuits, one chip alone can comprise of the order of  $10^8$  components, each modelled by several equations. In such cases, the only feasible method of solution is a numerical DAE solver used in conjunction with the latest in sparse matrix techniques. Also, some of these very large problems are known to have an index of 2, 3 or higher.
- A fully automated package for the numerical solution of general DAE systems of unknown index requires problems posed without prior manipulation as input.

#### 3.1.1 DAEs and Stiff ODEs

DAEs are, in fact, very similar to stiff ODEs [1]. For example, if  $0 < \epsilon \ll 1$  then

$$(A + \epsilon B)y'(t) = -By(t) + g(t) \quad (3.2)$$

is a stiff system, with stiffness  $O(\epsilon^{-1})$ , near to the linear DAE

$$Ay'(t) + By(t) = g(t), \quad (3.3)$$

and is known as the *regularization* of the DAE. Here,  $A$  is rank-deficient but  $A + \epsilon B$  is not. Restating (3.2) as

$$\begin{aligned} y' &= -(A + \epsilon B)^{-1}By + f(t) \\ &= Jy + f(t), \end{aligned}$$

the ratio of largest to smallest negative eigenvalues of  $J = -(A + \epsilon B)^{-1}B$ , or the ratio of time constants of the exponentially decreasing eigenfunction solution, is  $O(\epsilon^{-1})$ . Thus we expect that if the underlying DAE ( $\epsilon = 0$ ) has a high index, many of the problems associated with DAEs such as (3.3) should occur in problems like (3.2).

Since DAEs can be seen as an extension of stiff ODEs (with infinite stiffness), it is not unreasonable that many of the methods we use are based on stiff ODE solvers, which are necessarily A-stable or stiffly stable, such as the backward differentiation formulae (BDFs) and implicit Runge-Kutta methods (IRKs). A study of the test equation

$$y'(t) = \lambda y(t) + g'(t) - \lambda g(t), \quad \lambda \ll 0, \quad (3.4)$$

using IRKs and letting stiffness  $\lambda \rightarrow -\infty$ , yields the predicted DAE order loss (cf. §3.3). The recent research on order reduction for IRK methods applied to stiff ODEs [33], obtaining sharper and sharper error bounds, can thus be used to make order predictions on the DAE obtained by taking the stiffness to its limit. Likewise, it is possible to derive order results for certain classes of stiff problems from DAE order results [46, 95]. However, stiff regularizations of DAEs generally exhibit worse order behaviour than their limiting DAEs. So stiff problems are, in a sense, harder to solve than their infinitely stiff (DAE) counterparts.

### 3.1.2 The Backward Euler Method

The idea of using ODE methods for solving DAE systems directly was introduced by Gear in [36] and is best illustrated by considering the simplest possible algorithm, based on the backward Euler method (BEM), which is the simplest A-stable ODE method. Here, we approximate  $x'(t_{n+1})$  by a backward difference of  $x(t)$ , and the resulting system of (generally) nonlinear equations,

$$F\left(x_{n+1}, \frac{x_{n+1} - x_n}{t_{n+1} - t_n}, t_{n+1}\right) = 0,$$

is solved for  $x_{n+1}$  using a modified Newton's method, thus advancing the solution from time  $t_n$  to  $t_{n+1}$ . Higher order methods such as BDFs, IRKs and extrapolation methods are generalizations of this simple idea.

In this section we apply the BEM to three simple index 2 and 3 linear DAEs [89] in order to investigate some problems characteristic to the numerical solution of DAEs. Later, we discuss BDFs applied to general nonlinear systems, also covering implementation details common to all DAE methods. We then look specifically at IRKs and some of the more obscure methods.

**Example 3.1** *The simplest index 3 problem, constant-coefficient and in SCF, is*

$$\begin{aligned}x_1 &= g(t) \\x_1' - x_2 &= 0 \\x_2' - x_3 &= 0.\end{aligned}\tag{3.5}$$

*Its solution is  $x_1 = g(t)$ ,  $x_2 = g'(t)$ ,  $x_3 = g''(t)$ .*

*Note the following:*

1. *If initial values are specified for the  $x_i$ , the solution has a jump discontinuity at the initial point, unless these initial values are consistent.*
2. *If the driving term  $g(t)$  is not twice differentiable everywhere, the solution will not exist everywhere, eg.*

$$g(t) = H(t - t_0) \quad \Rightarrow \quad \begin{aligned}x_2 &= \delta(t - t_0) \\x_3 &= \delta'(t - t_0).\end{aligned}$$

*Here,  $H$  is the Heaviside function and  $\delta$  is the Dirac-delta function.*

*Applying the BEM to (3.5) yields*

$$\begin{aligned}x_{1,n} &= g_n \\x_{2,n} &= (x_{1,n} - x_{1,n-1})/h, \quad (h = t_n - t_{n-1}) \\x_{3,n} &= (x_{2,n} - x_{2,n-1})/h.\end{aligned}$$

*The values of  $x_1$  will be correct at all steps, ignoring roundoff error, although the initial value  $x_{1,0}$  may be inconsistent. However, inconsistent  $x_{1,0}$  yield incorrect values for  $x_{2,1}$  and  $x_{3,1}$ , which diverge as  $h \rightarrow 0$ . After two steps  $x_{2,2}$  is obtained by a divided difference of  $g(t)$  and is thus  $O(h)$  correct, and after the third step  $x_{3,3}$  is also  $O(h)$  correct. Clearly, in this case, the accuracy expected from solving ODEs is obtainable for DAEs, but only after a few steps. Also, we can already see that a change in steplength will reduce the accuracy for a few steps.*

**Example 3.2** *Consider Example 2.3, which is a linear variable-coefficient index 2 problem for  $\eta \neq -1$ . The BEM yields the solution*

$$\begin{aligned}x_{2,n} &= \left( \frac{\eta}{1+\eta} \right) x_{2,n-1} - \frac{g_n - g_{n-1}}{h(1+\eta)} \\x_{1,n} &= g_n - \eta t_n x_{2,n}.\end{aligned}$$

The true solution is  $x_2(t_n) = -g'_n$ ,  $x_1(t_n) = g_n + \eta t_n g'_n$ . The numerical solution is clearly unstable if  $\eta < -\frac{1}{2}$ . If  $\eta > -\frac{1}{2}$ , it converges to the true solution. From numerical ODE theory we also know that this stability problem would not be solved by increasing the order of the method.

**Example 3.3** Consider Example 2.4, another linear variable-coefficient problem. If we examine the canonical form after transforming from  $x$  to  $y$ , and write  $y = (u, v, w)^T$ , we have the equations

$$\begin{aligned} u &= g(t) \\ v &= -u' \\ w' &= (v - w)/\eta \end{aligned}$$

for the underlying index 2 system. The ODE component  $w$  is stable if  $\eta > 0$ . Applying the BEM to the original system, we obtain the recurrence relations for  $x = (p, q, r)^T$ ,

$$\begin{aligned} r_n &= -\eta(r_{n-1} - r_{n-2})/h + (g_n - \eta g_{n-1} + g_{n-2})/h^2 \\ q_n &= \eta r_n - (g_n - g_{n-1})/h \\ p_n &= g_n - \eta t_n r_n, \end{aligned}$$

which are unstable if  $h < 2\eta$  or  $h < -\eta$ . Consequently, the method cannot converge as  $h \rightarrow 0$ .

The delayed effect of the driving function  $g$  on the higher-index components, mentioned in Example 3.1, has another curious effect [80]. Suppose that  $g$  is insufficiently continuous, such as  $g(t) = \delta t H(t)$  ( $H$  is the Heaviside function), which is  $C^0$  but not  $C^1$  at  $t = 0$ . If we were solving such a problem numerically, we would hope that the code would find the discontinuity and pass over it, or at least fail at the discontinuity. Instead, a code based on the BEM with the usual error control mechanism — with an error estimate proportional to the difference between predicted and corrected values — is likely to fail on the step after the step which spans the discontinuity. This leaves us with little indication that the discontinuity was the source of the difficulty, and is especially surprising if we think of one-step methods as having no memory.

To gain an understanding of the source of the problem, we compare the local error at each step with the error estimates. One step of the BEM for the linear constant-coefficient system,

$$Ax'(t) + Bx(t) = g(t), \quad (3.6)$$

yields

$$A \left( \frac{x_{n+1} - x_n}{h_{n+1}} \right) + Bx_{n+1} \approx g(t_{n+1}). \quad (3.7)$$

If  $x(t_{n+1})$  is the true solution to (3.6) at time  $t_{n+1}$ , then we have

$$x(t_{n+1}) = x(t_n) + h_{n+1}x'(t_{n+1}) + \frac{h_{n+1}^2}{2}x''(\zeta), \quad \zeta \in [t_n, t_{n+1}]. \quad (3.8)$$

Substitute  $x'(t_{n+1})$  from (3.8) into (3.6) at  $t_{n+1}$  to get

$$A \left( \frac{x(t_{n+1}) - x(t_n) - \frac{h_{n+1}^2}{2} x''(\zeta)}{h_{n+1}} \right) + Bx(t_{n+1}) = g(t_{n+1}). \quad (3.9)$$

Subtract (3.9) from (3.7) and define the global error  $e_{n+1} = x_{n+1} - x(t_{n+1})$ . Then

$$A \left( \frac{e_{n+1} - e_n + \frac{h_{n+1}^2}{2} x''(\zeta)}{h_{n+1}} \right) + Be_{n+1} = 0,$$

$$\text{or } e_{n+1} = (A + h_{n+1}B)^{-1} A e_n - (A + h_{n+1}B)^{-1} A \left( \frac{h_{n+1}^2}{2} \right) x''(\zeta).$$

For the local truncation error, set  $e_0 = 0$  and then  $e_1$  is the local error,

$$e_1 = -(A + h_{n+1}B)^{-1} A \left( \frac{h_{n+1}^2}{2} \right) x''(\zeta). \quad (3.10)$$

For the index 2 version of Example 3.1 above,

$$x''(\zeta) = \begin{pmatrix} g''(\zeta) \\ g'''(\zeta) \end{pmatrix} \quad \text{and} \quad (A + h_{n+1}B)^{-1} A = \begin{pmatrix} 0 & 0 \\ -1/h_{n+1} & 0 \end{pmatrix},$$

so that the local error is  $(0 \ h_{n+1}g'''(\zeta)/2)^T$ . The local error for  $x_2(t)$  is  $O(h_{n+1})$ , not  $O(h_{n+1}^2)$ , and depends on  $x_1''(t)$  and not  $x_1'(t)$ , so that the usual error estimate, whose  $i$ -th component is based on  $(h_{n+1}^2/2)x_i''$ , is asymptotically a gross underestimate.

There are more severe problems for index  $\geq 3$  systems. For the index 3 problem (3.5),

$$(A + h_{n+1}B)^{-1} A = \begin{pmatrix} 0 & 0 & 0 \\ -1/h_{n+1} & 0 & 0 \\ 1/h_{n+1}^2 & -1/h_{n+1} & 0 \end{pmatrix}$$

$$\frac{h_{n+1}^2}{2} x''(\zeta) = \frac{h_{n+1}^2}{2} \begin{pmatrix} g''(\zeta) \\ g'''(\zeta) \\ g^{(iv)}(\zeta) \end{pmatrix}$$

$$\text{so that } e_1 = \begin{pmatrix} 0 \\ \frac{h_{n+1}^2}{2} g'''(\zeta) \\ -\frac{g''(\zeta)}{2} + \frac{h_{n+1}}{2} g'''(\zeta) \end{pmatrix},$$

and the local error can not be made small by choosing any small  $h_{n+1}$ .

### 3.2 Backward Differentiation Formulae

The class of multistep methods given by

$$\sum_{j=0}^k \alpha_j x_{n-j} = h \beta_0 x'_n$$

and based on the  $k$ -th order interpolating polynomial on  $\{x_{n-j}\}_{j=0}^k$ , is known as the backward differentiation formulae (BDF). BDFs can be restated as  $x'_n = \rho x_n / h$ , where  $\rho$  is the difference operator defined by

$$\rho w_n = \frac{1}{\beta_0} \sum_{j=0}^k \alpha_j w_{n-j}. \quad (3.11)$$

The  $k$ -step BDF,  $k = 1, 2, \dots, 6$ , has order of accuracy  $k$  (ie.  $\rho w(t_n) = h w'(t_n) + O(h^k)$ ), and an infinite region of absolute stability  $S$  when applied to ODEs. BDFs are:

1. Stiffly stable (ie.  $S \supset \{\mu; \operatorname{Re}(\mu) < -D; D > 0\}$  with the method "accurate" in rectangle  $-D \leq \operatorname{Re}(\mu) \leq a$ ,  $-\theta \leq \operatorname{Im}(\mu) \leq \theta$ ,  $a > 0$ ,  $\theta \approx \pi/5$ ).
2.  $A(\alpha)$ -stable (ie.  $S \supset \{\mu; |\arg(-\mu)| < \alpha, \mu \neq 0\}$ ).
3. A-stable (ie.  $S \supset C^-$ ) for  $k=1, 2$ .

Here,  $\mu = h\lambda$  and  $\lambda$  is the Lipschitz constant. Coefficients and stability properties are given below. We also mention the  $k$ -step generalized BDF methods (GBDFs) [53], for which

| $k$ | $\beta_0$ | $\alpha_0$ | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | $\alpha_4$ | $\alpha_5$ | $\alpha_6$ | $\alpha$   | $D$   |
|-----|-----------|------------|------------|------------|------------|------------|------------|------------|------------|-------|
| 1   | 1         | 1          | -1         |            |            |            |            |            | $90^\circ$ | 0     |
| 2   | 2/3       | 1          | -4/3       | 1/3        |            |            |            |            | $90^\circ$ | 0     |
| 3   | 6/11      | 1          | -18/11     | 9/11       | -2/11      |            |            |            | $80^\circ$ | 0.083 |
| 4   | 12/25     | 1          | -48/25     | 36/25      | -16/25     | 3/25       |            |            | $73^\circ$ | 0.367 |
| 5   | 60/137    | 1          | -300/137   | 300/137    | -200/137   | 75/137     | -12/137    |            | $51^\circ$ | 2.327 |
| 6   | 60/147    | 1          | -360/147   | 450/147    | -400/147   | 225/147    | -72/147    | 10/147     | $18^\circ$ | 6.075 |

Table 3.1: Coefficients and stability for  $k$ -step BDF methods

(3.11) is replaced by the divided difference operator

$$\varrho w_n = \frac{1}{h} \sum_{j=0}^k \alpha_{nj} w_{n-j},$$

which more closely resembles differentiation than does  $\rho$ . GBDFs can be stable beyond order 6, where the order  $p$  (generally  $p < k$ ) is given by  $\varrho w_n = w'(t_n) + O(h^p)$ .

### 3.2.1 Linear Constant-Coefficient DAEs

The behaviour of the general BDF method is very similar to that of the BEM for fixed stepsize.

**Theorem 3.2.1** [93] *If the  $k$ -step constant-stepsizes BDF is applied to the constant-coefficient linear problem*

$$Ax'(t) + Bx(t) = y(t) \quad (3.12)$$

*of index  $\nu$  with  $k < 7$ , the solution is  $O(h^k)$  accurate globally after a maximum of  $(\nu-1)k+1$  steps. For variable stepsize, where the ratio of adjacent stepsizes is bounded, the global error is  $O(h_{\max}^q)$ , where  $q = \min(k, k-\nu+2)$ . (The constant-stepsizes case is easily proved using Theorem 2.1.3 and the analytical solution (2.9), (2.10).)*

The difficulty can be seen by considering the previous index 3 problem (3.5) for variable stepsizes and  $k = 1$ :

$$\begin{aligned} x_{1,n} &= g_n \\ x_{2,n} &= (x_{1,n} - x_{1,n-1})/h_n \\ x_{3,n} &= (x_{2,n} - x_{2,n-1})/h_n. \end{aligned}$$

Ignoring initial errors,

$$x_{3,n} = \left( \frac{g_n - g_{n-1}}{h_n} - \frac{g_{n-1} - g_{n-2}}{h_{n-1}} \right) / h_n,$$

and the error is

$$\frac{1}{2} \left( \frac{h_{n-1}}{h_n} - 1 \right) g''(\zeta) + O(h_n),$$

which is  $O(1)$  and does not converge as  $h \rightarrow 0$  if  $h_n = O(h_{n-1})$  but  $h_n \neq h_{n-1}$ . For constant stepsize the error is  $O(h)$  which converges as  $h \rightarrow 0$ . This suggests that variable-step BDFs are not suitable for solving constant-coefficient linear DAEs.

### 3.2.2 Linear Variable-Coefficient DAEs

We apply the  $k$ -step BDF to the time-varying subsystems

$$x_1'(t) + C(t)x_1'(t) = f_1(t) \quad (3.13)$$

$$N(t)x_2'(t) + x_2(t) = f_2(t) \quad (3.14)$$

in SCF. As seen earlier, the D-part (3.13) is an ODE system, and it is well-known that the BDF method is stable and converges globally for ODEs. For the A-part (3.14), the convergence properties are stated in the following theorem.

**Theorem 3.2.2** [9] *If the  $k$ -step constant-stepsizes BDF method is applied to the  $s \times s$  linear variable-coefficient implicit system (3.14) with  $k < 7$ , the solution is  $O(h^k)$  accurate globally after a maximum of  $(s-1)k+1$  steps.*

Proof

Defining backward difference operators  $\nabla$ :

$$\begin{aligned}\nabla x(t) &= x(t) - x(t-h) \\ \nabla^i x(t) &= \nabla^{i-1} x(t) - \nabla^{i-1} x(t-h)\end{aligned}$$

and  $D_k = 1/h \sum_{i=1}^k \nabla^i / i$ , we have for the differentiation operator [9]

$$\mathcal{D} = \frac{1}{h} \sum_{i=1}^{\infty} \frac{\nabla^i}{i} = D_k + O(h^k). \quad (3.15)$$

The BDF method applied to (3.14) at  $t_n$  yields (dropping the subscript 2)

$$\begin{aligned}(N_n D_k + I)x_n &= f_n \\ \Rightarrow x_n &= (N_n D_k + I)^{-1} f_n \\ &= \sum_{i=0}^{\infty} (-N_n D_k)^i f_n \\ &= \sum_{i=0}^{s-1} (-N_n D_k)^i f_n \quad (\text{Theorem 2.2.6}) \\ &= \sum_{i=0}^{s-1} (-N_n \mathcal{D})^i f_n + O(h^k) \quad (\text{using (3.15) and induction on } i) \\ &= x(t_n) + O(h^k). \quad (\text{Theorem 2.2.6})\end{aligned}$$

The computation of  $x_n$  involves only  $N_{n-l}$ ,  $f_{n-l}$ ,  $l = 0, 1, \dots, (s-1)k$ , so that  $x_n$  is independent of the initial values after at most  $(s-1)k + 1$  steps.  $\square$

### 3.2.3 Nonlinear DAEs

We state convergence results for BDFs applied to index  $\leq 3$  generally nonlinear implicit DAE systems, and then consider practical implementation, much of which applies equally to other DAE methods.

In [66] it is shown that for constant-stepsize BDFs applied to a problem in differential-algebraic form

$$\begin{aligned}0 &= F_1(x, x', y, t), & x(t_0) &= x_0 \\ 0 &= F_2(x, y, t), & y(t_0) &= y_0\end{aligned} \quad (3.16)$$

with nonsingular Jacobian  $\partial F_1 / \partial x'$ , provided the system is solvable and has a global index of 1, 2 or 3, the global error of a zero-stable  $k$ -step constant-stepsize BDF method is  $O(h^k)$  if the Newton iteration is solved to  $O(h^{k+1})$ . Proofs for the fully implicit *uniform* index 1 and semi-implicit index 2 and 3 Hessenberg forms are given in [14]. In [15] it is shown that starting values which are numerically consistent to  $O(h^{k+1})$  are sufficient to ensure

the above-stated accuracy. In the index 3 case, this is true only after  $k+1$  steps. GBDFs have been shown to converge for Hessenberg DAEs of index 3, 4 and possibly 5 [53]. Convergence is assured by  $p \geq 2$  for index 3 and  $p > 3$  for index 4. Other, less general, results can be found in the papers mentioned above.

### Practical Implications

We discuss here some of the practical considerations of implementing BDF methods within an automatic code. First we note that for all the implicit methods studied here (BDF, IRK, etc.), a Newton iteration of some sort is typically used to solve the implicit algebraic system resulting from discretization, taking as initial guess the solution at the previous step. The following applies specifically for DAEs in differential-algebraic form (3.16), but the basic ideas hold for the fully general DAE.

#### (a) Conditioning of matrices [82]

Conditioning is a problem for DAE systems, especially higher-index ones, because the condition number of the Newton iteration matrix for an index  $m$  problem is  $O(h^{-m})$ . We describe schemes for scaling DAEs so that iteration matrices are no longer singular as  $h \rightarrow 0$ , and show how this scaling can easily be implemented into existing DAE software. The  $j$ -th Newton iteration for the discretized version of (3.1),

$$F(x_n, \rho x_n/h, t_n) = 0,$$

reads

$$\left( \frac{\partial F}{\partial x_n} \right)^{[j]} (x_n^{[j+1]} - x_n^{[j]}) = F(x_n^{[j]}, \rho x_n^{[j]}/h, t_n),$$

which is a system of linear equations of the form  $Jz = b$ . Using Gaussian elimination with partial pivoting, the computed solution  $z + \Delta z$  satisfies

$$(J + \Delta J)(z + \Delta z) = b, \quad (3.17)$$

$$\text{where } \|\Delta J\|_{\infty} \leq ru \|J\|_{\infty}, \quad (3.18)$$

$r$  is a moderate number and  $u$  is the machine unit. The accuracy of this computation is improved if we introduce a row scaling of  $J$  by premultiplying by a diagonal matrix  $D$ :

$$\begin{aligned} |\Delta z_i| &\leq |(J^{-1} D^{-1} D \Delta J (z + \Delta z))_i| \\ &\leq \sum_j |(J^{-1} D^{-1})_{ij}| |(D \Delta J (z + \Delta z))_j| \\ &\leq \sum_j |(J^{-1} D^{-1})_{ij}| \|D \Delta J\|_{\infty} \|z + \Delta z\|_{\infty} \\ &\leq ru \sum_j |(J^{-1} D^{-1})_{ij}| \|D J\|_{\infty} \|z + \Delta z\|_{\infty}. \end{aligned} \quad (3.19)$$

We consider the explicit index 1 or 2 problem

$$\begin{aligned} x' &= f(x, y, t) \\ 0 &= g(x, y, t), \end{aligned} \quad (3.20)$$

although these ideas extend easily to more general problems. The iteration matrix is written as

$$hJ_n = \begin{pmatrix} \alpha_0 I - hf_x & -hf_y \\ hg_x & hg_y \end{pmatrix}.$$

1. For the index 1 case,  $g_y$  is nonsingular, so  $hJ_n$  is nonsingular as  $h \rightarrow 0$  if we scale the rows corresponding to the algebraic constraint by  $1/h$ , the effect of which should be to improve the accuracy of the solution for all variables.
2. For index 2,  $g_y \equiv 0$ . Explicit computation of  $(hJ_n)^{-1}$  gives

$$(hJ_n)^{-1} = \begin{pmatrix} O(1) & O(1/h) \\ O(1/h) & O(1/h^2) \end{pmatrix}.$$

Again, scaling the algebraic rows of  $hJ_n$  by  $1/h$ , the scaled matrix can be written as

$$h\hat{J}_n = \begin{pmatrix} \alpha_0 I - hf_x & -hf_y \\ g_x & 0 \end{pmatrix}.$$

It follows from (3.19) that roundoff errors of  $O(u/h)$  and  $O(u/h^2)$  are introduced in  $x$  and  $y$  respectively, for the unscaled system, but  $O(u)$  and  $O(u/h)$  for the scaled system. As  $h \rightarrow 0$  the error in  $y$  can begin to dominate. We will discuss this later. Since the state variables  $x$ , whose solution we are most interested in finding, can be found accurately and errors in the nonstate (algebraic) variables  $y$  do not affect  $x$  at later stages, this scaling is a considerable improvement over the original one.

3. For index 3 problems of the form (1.12)–(1.14), such as in Example 2.6, it is suggested to scale the row corresponding to (1.14) by  $1/h$  or (1.13) by  $h$ .

What this analysis shows is that the errors introduced by ill-conditioning are concentrated in the algebraic variables, so we must be careful about using algebraic variables in other error tests in the code.

#### (b) Tests for terminating the corrector iteration [82]

Even if we are prepared to tolerate large errors in the nonstate variables, we still need some criterion for deciding when to terminate the corrector iteration, so these errors are still a source of difficulty. Here we show that it is sufficient to terminate the Newton iteration based on the errors in the scaled variables, as described above.

We will examine the propagation of errors due to interrupting the Newton iterations. Let  $(\tilde{x}_n, \tilde{y}_n)$  be the computed solution and  $(x_n, y_n)$  be the true solution to the difference

equation. Then  $x_n = \hat{x}_n + \delta_n^x$  and  $y_n = \hat{y}_n + \delta_n^y$ . Let  $e_n^x = \hat{x}_n - x(t_n)$  and  $e_n^y = \hat{y}_n - y(t_n)$  be the global errors and  $\tau_n$  the local truncation error. Then

$$\begin{aligned}
 0 &= F(x_n, \rho x_n/h, y_n, t_n) \\
 &= F(\hat{x}_n + \delta_n^x, \rho(\hat{x}_n + \delta_n^x)/h, \hat{y}_n + \delta_n^y, t_n) \\
 &= F(x(t_n) + e_n^x + \delta_n^x, \rho(x(t_n) + e_n^x + \delta_n^x)/h, y(t_n) + e_n^y + \delta_n^y, t_n) \\
 &= F(x(t_n), x'(t_n), y(t_n), t_n) + \frac{\partial F}{\partial x}(e_n^x + \delta_n^x) \\
 &\quad + \frac{\partial F}{\partial x'}(\rho(e_n^x + \delta_n^x)/h + \tau_n) + \frac{\partial F}{\partial y}(e_n^y + \delta_n^y) + \text{higher order terms.} \quad (3.21)
 \end{aligned}$$

Introducing

$$\begin{aligned}
 F_{1x} &= \partial F_1 / \partial x, \quad F_{1x'} = \partial F_1 / \partial x', \quad A_1 = \alpha_0 F_{1x}, \quad A_2 = \partial F_1 / \partial y, \\
 A_3 &= \partial F_2 / \partial x \quad \text{and} \quad A_4 = \partial F_2 / \partial y,
 \end{aligned}$$

we have

$$\begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix} \begin{pmatrix} e_n^x + \delta_n^x \\ e_n^y + \delta_n^y \end{pmatrix} = \begin{pmatrix} -F_{1x'}(c_n + h\tau_n) \\ 0 \end{pmatrix} + \text{h.o.t.'s}, \quad (3.22)$$

$$\text{where} \quad c_n = \sum_{i=1}^k \alpha_i (e_{n-i}^x + \delta_{n-i}^x).$$

From this we find that

$$\begin{pmatrix} e_n^x \\ e_n^y \end{pmatrix} = \begin{pmatrix} -\delta_n^x \\ -\delta_n^y \end{pmatrix} + \begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix}^{-1} \left[ \begin{pmatrix} \zeta \\ 0 \end{pmatrix} + \text{h.o.t.'s} \right],$$

where  $\zeta$  is the RHS of the top equation of (3.22). It follows from (3.21) that

$$0 = hF(x_n, (\alpha_0 \hat{x}_n + c_n)/h, y_n, t_n) + \begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix} \begin{pmatrix} \delta_n^x \\ \delta_n^y \end{pmatrix} + \text{h.o.t.'s}.$$

Hence

$$\begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix} \begin{pmatrix} \delta_n^x \\ \delta_n^y \end{pmatrix} \approx h\eta = -h \begin{pmatrix} F_{1n} \\ F_{2n} \end{pmatrix}. \quad (3.23)$$

In the index 1 case,  $\eta$  must be of  $O(h^k)$  [66], so the iteration should be terminated when  $\delta_n^x$  and  $\delta_n^y$  are of  $O(h^{k+1})$  in (3.23). Let  $\eta^T = (\eta_1^T, \dots, \eta_r^T)$ . If  $A_4 \equiv 0$  and the index is 2, then  $\eta_1$  and  $\eta_2$  must be of  $O(h^k)$  and  $O(h^{k+1})$  respectively, which is achieved by letting  $\delta_n^x$  and  $h\delta_n^y$  be proportional to  $h^{k+1}$  in (3.23).

In general one should never totally exclude variables from the corrector iteration convergence test. This is true even if a variable occurs linearly since we often use codes that

make use of Jacobians calculated at previous steps, possibly based on different stepsizes, so the Newton iteration will not necessarily converge in one step.

(c) Error test [82]

Here we see why some of the variables should be excluded from the error test in an automatic code. It follows from (3.21) and (3.22) that the previous errors,  $e_i^y, i < n$ , in the algebraic variable  $y$ , do not directly influence any of the other variables at  $t_n$ , for systems of index 1 or 2 (when  $\partial F_2/\partial y = 0$ ), since they do not appear in (3.21). So we can consider ignoring these variables in the error estimate. If we base the error test on  $y$ , which could have a large error and still pass the Newton iteration convergence test, this could cause a problem, because the new  $y$  value does not approach the old incorrect value, thus causing the error test to fail.

Another possible error estimate is based on the truncation error  $h\tau_n$  in (3.22):

$$e_{\tau_n} = \begin{pmatrix} e_{\tau_n^x} \\ e_{\tau_n^y} \end{pmatrix} = (-hJ_n)^{-1} \begin{pmatrix} F_{1x} h\tau_n \\ 0 \end{pmatrix} + \text{h.o.t.'s}. \quad (3.24)$$

This is the contribution to the global error that is directly influenced by a change in stepsize. An estimate of  $e_{\tau_n}$  can be obtained by multiplying the usual error estimate by the matrix

$$(hJ_n)^{-1} \begin{pmatrix} F_{1x} & 0 \\ 0 & 0 \end{pmatrix}.$$

An approximate factorization of  $hJ_n$  is available from the Newton iteration. The stepsize is chosen so that the estimate  $e_{\tau_n}$  in (3.24) is within a user-specified tolerance.

### 3.3 Runge-Kutta Methods

It is well-known that explicit Runge-Kutta methods are inadequate for solving any but the mildest stiff ODE systems, so we cannot use them for DAEs. On the other hand, it is easy to find A-stable and even L-stable implicit Runge-Kutta's, associated with various types of quadrature rules. However, the solution of the resulting system of  $s \cdot n$  (where  $s$  is the number of stages and  $n$  is the dimension of the system) simultaneous nonlinear equations by Newton iteration poses a formidable computational task. Recently, much work has been done using IRKs to solve stiff systems, and the latest methods and techniques are also being applied to DAEs. We shall first give a brief review of the literature in this field, and then discuss in depth the solution of the linear constant-coefficient problem and the nonlinear index 1 and 2 cases, revising IRK concepts as we go.

The IRKs comprise a wide, and growing, family of methods, derived in different ways and for different reasons. Their application to a variety of DAE problems in the literature has resulted in a huge body of specific results. As yet, no real attempt has been made to generalize either IRK methods or results, so no general conclusions can be drawn

at this stage. Consequently, we merely mention some of the work that has been done, and will not attempt to summarize the literature.

### 3.3.1 Survey

1. Petzold [81] obtained the first convergence results for RK methods applied to the general DAE,  $F(y, y', t) = 0$ , of index 1, by formally applying an  $s$ -stage IRK method, to obtain the system of difference equations

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

$$y(t_n) \approx y_{n-1} + h \sum_{i=1}^s b_i Y_i', \quad (3.25)$$

$$\text{where } c_i = \sum_{j=1}^s a_{ij} \quad \text{and} \quad h = t_n - t_{n-1}.$$

The stage derivatives  $Y_i'$  are estimates for  $y'(t_{n-1} + c_i h)$ . An IRK method is uniquely defined by  $(c, A, b^T)$ , where  $c = (c_1, \dots, c_s)^T$  is the vector of quadrature nodes,  $A = [a_{ij}]$  is the coefficient matrix and  $b^T = (b_1, \dots, b_s)$  is the vector of quadrature weights. In terms of the Butcher identities,

$$\begin{aligned} A_1(k) &: b^T A^{-1} c^j = 1, \quad j = 1, \dots, k \\ B(k) &: b^T c^{j-1} = 1/j, \quad j = 1, \dots, k \\ C(k) &: A c^{j-1} = c_j/j, \quad j = 1, \dots, k \\ D(k) &: \sum_{i=1}^s b_i c_i^{l-1} a_{ij} = b_j/l(1 - c_j^l), \quad l = 1, \dots, k, \end{aligned} \quad (3.26)$$

the internal order  $k_I$  of the RK formula is defined to be the largest integer  $k$  for which  $B(k)$  and  $C(k)$  hold. Condition  $B(k)$  (or  $C(k)$ ) means that the quadrature formula with weights  $b_1, \dots, b_s$  (or  $a_{i1}, \dots, a_{is}$ ) and nodes  $c_1, \dots, c_s$  integrates polynomials up to degree  $k-1$  exactly on  $[0, 1]$  (or  $[0, c_i]$ ). If  $k_d$  is the order of the RK formula as an ODE method, Petzold showed that the order of convergence for index 1 constant-coefficient linear implicit DAEs is at least  $\min(k_d, k_I + 1)$ , under the assumptions that

- (a)  $A$  is nonsingular; and
- (b)  $|r| < 1$ , where the stability constant  $r$  is given by  $r = b^T A 1$ , and  $1$  is the vector of ones. (This is the strict stability condition.)

The reduced order of convergence,  $k_I + 1 < k_d$ , found experimentally, is most striking for diagonally and singly diagonally implicit formulae (DIRKs and SDIRKs), which

are computationally the most efficient IRKs. (For DIRKs, the matrix  $A$  is lower triangular, the first stage is a BEM step and  $k_1 = 1$ . SDIRKs have, in addition, equal diagonal elements in  $A$ .)

The above result has been extended to the case  $r = 1$  in [19]. Brenan and Petzold [16] determined the local discretization error for linear constant-coefficient systems of arbitrary index and obtained order of convergence  $\min(k_d, k_1)$  for nonlinear semi-explicit index 2 systems, which again exhibit reduced order. These two cases are studied in more depth below. In [2], order reduction for DIRKs is overcome for arbitrary index constant-coefficient linear DAEs by allowing  $A$  to be singular.

2. Hairer, Lubich and Roche [47] use methods based on Gauss, Radau and Lobatto quadrature, whose coefficients are uniquely determined by

$$\begin{aligned} \text{Gauss} & : B(2s), \quad C(s), \quad D(s). \\ \text{Radau IA} & : B(2s-1), \quad C(s-1), \quad D(s), \quad c_0 = 0. \\ \text{Radau IIA} & : B(2s-1), \quad C(s), \quad D(s-1), \quad c_s = 1, \quad b_i = a_{si}. \\ \text{Lobatto IIIA} & : B(2s-2), \quad C(s), \quad D(s-2), \quad c_1 = 0, \quad c_s = 1, \quad b_i = a_{si}. \\ \text{Lobatto IIIC} & : B(2s-2), \quad C(s-1), \quad D(s+1), \quad c_1 = 0, \quad c_s = 1, \quad b_i = a_{si}. \end{aligned}$$

Applying these methods to explicit DAEs of index 1 (of form (3.20)), 2 or 3 (of Hessenberg form), they study the influence of perturbations, local error and convergence, asymptotic expansions, convergence of simplified Newton iterations for the arising nonlinear systems, and the problems of local error estimation and inconsistent starting values. Orders of convergence are given in Table 3.2. They also developed the package RADAU5, using the 5th order Radau IIA method, to solve the constant-coefficient linear implicit problem  $By' = f(y, t)$  for index 1, 2 and 3.

| Method       | No. of stages | Index 1 |        | Index 2 |       | Index 3 |       |       |
|--------------|---------------|---------|--------|---------|-------|---------|-------|-------|
|              |               | $x$     | $y$    | $x_1$   | $x_2$ | $x_2$   | $x_1$ | $x_3$ |
| Gauss        | $s$ odd       | $2s$    | $s+1$  | $s+1$   | $s-1$ | —       | —     | —     |
|              | $s$ even      | $2s$    | $s$    | $s$     | $s-2$ | —       | —     | —     |
| Radau IA     | $s=3$         | $2s-1$  | $s$    | $s$     | $s-1$ | $s-1$   | $s-1$ | $s-2$ |
|              | $s>3$         | $2s-1$  | $s$    | $s$     | $s-1$ | $s$     | $s-1$ | $s-2$ |
| Radau IIA    | $s=2,3$       | $2s-1$  | $2s-1$ | $2s-1$  | $s$   | $2s-2$  | $s$   | $s-1$ |
|              | $s>3$         | $2s-1$  | $2s-1$ | $2s-1$  | $s$   | $s+2$   | $s$   | $s-1$ |
| Lobatto IIIA | $s$ odd       | $2s-2$  | $2s-2$ | $2s-2$  | $s-1$ | —       | —     | —     |
|              | $s$ even      | $2s-2$  | $2s-2$ | $2s-2$  | $s$   | —       | —     | —     |
| Lobatto IIIC | $s=3,4$       | $2s-2$  | $2s-2$ | $2s-2$  | $s-1$ | $2s-4$  | $s-1$ | $s-2$ |
|              | $s>4$         | $2s-2$  | $2s-2$ | $2s-2$  | $s-1$ | $s+1$   | $s-1$ | $s-2$ |

Table 3.2: Orders of convergence for some Runge-Kutta methods

3. In an early paper, Liniger [63] used one-leg (OL) methods, which for the ODE

$y' - f(y, t) = 0$  are given by

$$1/h_n \left( \sum_{j=0}^k \alpha_{j,n} y_{n-k+j} \right) - f \left( \sum_{j=0}^k \beta_{j,n} y_{n-k+j}, \sum_{j=0}^k \beta_{j,n} t_{n-k+j} \right) = 0.$$

Note that back values of  $y$  and  $t$  are used in  $f$ , as with other Runge-Kutta methods, while a multistep backward differentiation is used for  $y'$ . The interest in OL methods is that they have stronger stability and/or contractivity properties than the multistep methods, without the computational effort required for other IRKs. See also [75].

4. Rosenbrock (and Rosenbrock-type, Rosenbrock-Wanner or ROW, methods, where the Jacobian, estimated in the Newton iteration, is explicitly included in the method, have been applied to explicit index-1 DAEs [69, 84, 87, 90]. Rosenbrock methods do not require the solution of nonlinear systems of equations but only one evaluation of the Jacobian and one LU decomposition per step.
5. In [48], order conditions and stability properties are discussed for a variety of IRK methods, resulting in the development of the following codes (based on RADAU5) for solving constant-coefficient index 1 DAEs of the form  $By(t) = f(y, t)$ : SDIRK4 (4th order 5-step SDIRK method), ROS4 (4th order Rosenbrock method), RODAS (Rosenbrock method), SEULEX (extrapolation based on Euler method) and SODEX (extrapolation based on mid-point method). Experimental comparisons of these codes are also given in [48].
6. Collocation methods and projected Runge-Kutta's (PIRKs) [5, 64, 68] have recently been applied to DAEs of index 1 and 2, and have proved popular for solving boundary value DAEs (DABVPs), which we omit.
7. Error behaviour and order reduction of IRKs for stiff and DAE systems have been discussed in [19, 46, 95]. In [46], it is shown that the coefficients in the expansion of the global error in powers of the stiffness parameter are the global errors of the RK method applied to a DAE. A study of these errors and of the remainder of the expansion yields sharp error bounds for the stiff problem.

### 3.3.2 Linear Constant-Coefficient DAEs

After [16], we state necessary and sufficient conditions to ensure that the local error of an IRK method attains a given order when applied to the canonical subsystem

$$Ny'(t) + y(t) = g(t) \tag{3.27}$$

of arbitrary index  $\nu$ . We then study error propagation and obtain an expression for the global error. Applying the IRK method to (3.27), we have

$$\begin{aligned} NY'_i + y_{n-1} + \left( h \sum_{j=1}^s a_{ij} Y'_j \right) &= g(t_{n-1} + c_i h), \quad i = 1, \dots, s. \\ y_n &= y_{n-1} + h \sum_{i=1}^s b_i Y'_i. \end{aligned} \quad (3.28)$$

Let  $Y'_i = (Y'_{i,1}, Y'_{i,2}, \dots, Y'_{i,\nu})^T$ , where  $Y'_{i,j}$  is the  $i$ -th stage derivative corresponding to the  $j$ -th component of the solution vector. Because of the structure of  $N$ , the difference equations (3.28) can be solved componentwise, first for the stage derivatives  $Y'_{1,1}, Y'_{2,1}, \dots, Y'_{s,1}$  corresponding to  $y_1(t)$ , then for  $Y'_{1,2}, \dots, Y'_{s,2}$  corresponding to  $y_2(t)$ , and so on. So (3.28) can be solved uniquely for  $Y' = (Y'_1, \dots, Y'_s)^T$  if  $A$  is nonsingular. In general, for each component of the solution  $y_j(t)$ ,  $j = 1, \dots, \nu$ , we solve a subset of  $s$  equations from (3.28) for the corresponding stage derivatives:

$$\begin{aligned} (Y'_{1,j}, \dots, Y'_{s,j})^T &= (1/h) A^{-1} ((g_j(t_{n-1} + c_1 h), \dots, g_j(t_{n-1} + c_s h))^T \\ &\quad - 1 y_{n-1,j} - (Y'_{1,j-1}, \dots, Y'_{s,j-1})^T). \end{aligned} \quad (3.29)$$

Note that the stage derivative of the  $j$ -th component depends only on the numerical solution  $y_{n+1,j}$  and on the derivative of the  $(j-1)$ -th component. If we define

$$\begin{aligned} G_i &= \begin{pmatrix} (g_i(t_{n-1} + c_1 h) - g_i(t_{n-1}))/h \\ \vdots \\ (g_i(t_{n-1} + c_s h) - g_i(t_{n-1}))/h \end{pmatrix} \\ c^i &= (c_1^i, \dots, c_s^i)^T, \end{aligned}$$

use the local error assumption,  $y_{n-1} = y(t_{n-1})$ , and substitute for the  $(s-1)$ -th stage derivatives in (3.29), we obtain the following expressions for the local stage derivative,

$$\begin{aligned} (Y'_{1,1}, \dots, Y'_{s,1})^T &= A^{-1} G_1, \\ (Y'_{1,2}, \dots, Y'_{s,2})^T &= A^{-1} G_2 - (1/h) A^{-2} G_1 + (1/h) A^{-1} 1 g'_1(t_{n-1}), \\ (Y'_{1,3}, \dots, Y'_{s,3})^T &= A^{-1} G_3 + (1/h) A^{-1} 1 [g'_2(t_{n-1}) - \bar{g}'_1(t_{n-1})] \\ &\quad - (1/h) A^{-2} G_2 + (1/h^2) A^{-3} G_1 - (1/h^2) A^{-2} 1 g'_1(t_{n-1}), \\ &\text{etc.} \end{aligned} \quad (3.30)$$

We define the local error by

$$d_n = y(t_{n-1}) + h \sum_{i=1}^s b_i Y'_i - y(t_n), \quad (3.31)$$

where  $d_n = (d_{n,1}, \dots, d_{n,\nu})^T$ , and  $Y_i'$  are given by (3.30). Expanding (3.31) in a Taylor series about  $t_{n-1}$  and equating like powers of  $h$ , it is easy to see that the local error in the first component is  $d_{n,1} = O(h^{k_{a,1}+1})$ , when  $b^T A^{-1} c^i = 1$ , for  $i = 1, \dots, k_{a,1}$ .  $k_{a,1}$  is called the *algebraic order* of the IRK method applied to index 1 constant-coefficient systems. In a similar fashion, we obtain local error estimates for the other components, and finally

$$d_{n,\nu} = O(h^{k_{a,\nu}-\nu+2}) + O(h^{k_{a,\nu-1}-\nu+3}) + \dots + O(h^{k_{a,1}+1}),$$

where  $k_{a,\nu}$  is the largest integer satisfying

$$\begin{aligned} b^T A^{-i} \mathbf{1} &= b^T A^{-\nu} c^{\nu-i} / (\nu-i)!, & i = 1, 2, \dots, \nu-1 \\ b^T A^{-\nu} c^i &= i(i-1) \cdots (i-\nu+1), & i = \nu, \nu+1, \dots, k_{a,\nu}. \end{aligned}$$

Clearly, the higher the index, the more difficult it is to obtain IRK methods which are convergent in all components, restricting our choice of coefficients.

To examine the error propagation we solve (3.27) by the perturbed RK method:

$$\begin{aligned} NZ_i' + \left( z_{n-1} + h \sum_{j=1}^s a_{ij} Z_j' - \delta_n^{(i)} \right) &= g(t_{n-1} + c_i h), & i = 1, \dots, s, \\ z_n &= z_{n-1} + h \sum_{i=1}^s b_i Z_i' - \delta_n^{(s+1)}, \end{aligned} \quad (3.32)$$

where the perturbations,  $\delta_n^{(i)} = (\delta_{n,1}^{(i)}, \dots, \delta_{n,\nu}^{(i)})$ , have a bounded norm. These perturbations could be due to roundoff error, or errors in solving the linear systems at each stage, or they could be interpreted as truncation errors at each stage. Subtracting (3.32) from (3.28), and defining  $e_n = y_n - z_n$  and  $E_i' = Y_i' - Z_i'$ , we obtain

$$NE_i' + \left( e_{n-1} + h \sum_{j=1}^s a_{ij} E_j' + \delta_n^{(i)} \right) = 0 \quad (3.33)$$

$$e_n = e_{n-1} + h \sum_{i=1}^s E_i' + \delta_n^{(s+1)}. \quad (3.34)$$

Solving (3.33) for  $E_i'$  and substituting into (3.34) yields a relation describing the error propagation. Petzold did this for  $\nu = 1$ :

$$e_{n,1} = (1 - b^T A^{-1} \mathbf{1}) e_{n-1,1} - (b^T A^{-1} \delta_{n,1} - \delta_{n,1}^{(s+1)}), \quad \delta_{n,j} = (\delta_{n,j}^{(1)}, \dots, \delta_{n,j}^{(s)}).$$

This recurrence relation is unstable unless the strict stability condition  $|1 - b^T A^{-1} \mathbf{1}| < 1$  is satisfied. Continuing until the  $\nu$ -th component, we have

$$e_{n,\nu} = (1 - b^T A^{-1} \mathbf{1}) e_{n-1,\nu} - (b^T A^{-1} \delta_{n,\nu} - \delta_{n,\nu}^{(s+1)}) - \sum_{i=1}^{\nu-1} \frac{(-1)^i}{h^i} b^T A^{-i-1} (\delta_{n,\nu-i} + 1 e_{n-1,\nu-i}).$$

Note that:

1. The strict stability condition is no longer sufficient for stability if  $\nu > 1$ .
2. For small stepsizes, roundoff error can be significant in higher-index systems.
3. Sensitivity to roundoff errors is confined to higher-index components and does not propagate back into the earlier components.

These error propagation relations are required for extending the index 1 result of §3.3.1 to higher-index DAEs.

**Theorem 3.3.1** [16] *Suppose the IRK method (3.25) satisfies the strict stability condition. Then the constant-coefficient order  $k_{c,\nu}$  of the global error of this method is*

$$k_{c,\nu} = \min_{1 \leq i \leq \nu} (k_d, k_{a,i} - \nu + 2).$$

Results are given in Table 3.3 (from [16]).

| L-stable methods    | $k_d$ | $k_{a,1}$ | $k_{a,2}$ |
|---------------------|-------|-----------|-----------|
| Two-stage SIRK      | 2     | $\infty$  | 2         |
| Five-stage DIRK     | 4     | $\infty$  | 1         |
| Three-stage SIRK    | 3     | 3         | 2         |
| Seven-stage Extrap. | 3     | $\infty$  | 3         |

Table 3.3: Orders of convergence for linear constant-coefficient DAEs

### 3.3.3 Fully Implicit Index 1 Systems

**Theorem 3.3.2** [14] *For the fully implicit DAE (3.1) suppose that*

1. *It has uniform index 1, ie. its linearization is well-behaved and everywhere index 1 in a neighbourhood of the solution;*
2. *The IRK method satisfies the stability condition  $|r| \leq 1$  (cf. p.42);*
3. *The errors in the initial conditions are  $O(h^G)$ ; and*
4. *The errors in terminating the Newton iterations are  $O(h^{G+\delta})$ , where  $\delta = 1$  if  $|r| = 1$  and  $\delta = 0$  otherwise, and  $G > 1$ .*

*Then the global errors are  $O(h^G)$ , where*

$$G = \begin{cases} q, & \text{if } C(q) \text{ and } B(q) \\ q+1, & \text{if } C(q), B(q+1) \text{ and } -1 \leq r \leq 1 \\ q+1, & \text{if } C(q), B(q+1), A_1(q+1) \text{ and } |r| = 1. \end{cases}$$

These lower bounds for the order are supported by experimentation, which has also shown that certain methods exhibit higher order for some DAEs. In the light of the above theorem, SIRK methods, whose  $A$  matrices have a single  $s$ -fold real eigenvalue, appear promising, because high stage orders are possible and because they can be implemented efficiently. In general,  $|r| = 1$  schemes are not recommended for solving problems of the form considered here. Results of experiments comparing the observed global order  $k_g$  with the orders predicted above are given for some likely candidates for DAEs in Table 3.4 (references in [14]).

| Method              | $\tau$ | $k_d$ | $k_{d+1}$ | $k_I$ | $k_G$ | $k_g$ |
|---------------------|--------|-------|-----------|-------|-------|-------|
| Two-stage SIRK      | 0      | 2     | $\infty$  | 2     | 2     | 2     |
| Five-stage DIRK     | 0      | 4     | $\infty$  | 1     | 2     | 2     |
| Three-stage SIRK    | 0      | 3     | 3         | 2     | 3     | 3     |
| Seven-stage Extrap. | 0      | 3     | $\infty$  | 1     | 2     | 3     |
| Houbak & Thomsen    | 0      | 2     | 1         | 1     | 2     | 2     |
| Lobatto IIIC        | 0      | 4     | $\infty$  | 2     | 3     | 4     |
| Midpoint            | -1     | 2     | 1         | 1     | 2     | 2     |
| Gauss-Legendre      | 1      | 4     | 2         | 2     | 2     | 2     |

Table 3.4: Orders for nonlinear index 1 systems

### 3.3.4 Semi-Explicit Index 2 Systems

In this section we solve nonlinear semi-explicit index 2 systems of the form

$$\begin{aligned} f(x, x', y, t) &= 0 \\ g(x, y, t) &= 0 \end{aligned} \quad (3.35)$$

where  $f_x^{-1}$  exists and is bounded in a neighbourhood of the solution, and  $g_y$  has constant rank.

**Theorem 3.3.3** (Proof in [16].) *Given the nonlinear semi-explicit index 2 system (3.35), to be solved by the  $s$ -stage IRK method (3.28), suppose*

1. *The IRK method has internal order  $k_I$ ;*
2. *The IRK method satisfies the strict stability condition; and*
3. *The initial conditions satisfy  $\|(I - H)e_0^\pi\| = O(h^{k_G})$ , where  $(I - H)$  is a projection operator defined by the problem,  $e_0^\pi = x_0 - x(t_0)$ , and  $k_G = \min(k_a, k_I + 1)$ .*

*Then the global errors in  $x_n$  and  $y_n$  are  $O(h^{k_G})$  and  $O(h^{k_I})$  respectively.*

In [16], SIRKs, DIRKs and extrapolation methods are applied to a variety of index 2 problems arising in applications, such as the index 2 pendulum formulation (cf. §4.6).

| Method                | $k_d$ | $k_{u,1}$ | $k_{u,2}$ | $k_I$ | $k_G^x$ | $k_g^x$ | $k_G^y$ | $k_g^y$ |
|-----------------------|-------|-----------|-----------|-------|---------|---------|---------|---------|
| Two-stage SIRK        | 2     | $\infty$  | 2         | 2     | 2       | 2       | 2       | 2       |
| Five-stage DIRK       | 4     | $\infty$  | 1         | 1     | 2       | 2       | 1       | 1       |
| Three-stage SIRK      | 3     | 3         | 2         | 2     | 3       | 3       | 2       | 2       |
| Seven-stage Extrapol. | 3     | $\infty$  | 3         |       | 2       | 3       | 2       | 3       |

Table 3.5: Predicted and observed orders for index 2 pendulum problem

It is found that the observed convergence rates can sometimes be as slow as the error bounds derived. Some formulae, especially extrapolation methods, exhibit higher order of accuracy. The SIRKs, with order  $\geq m - 1$ , are expected to perform very well on index 2 problems. However, the development of an efficient IRK code for DAEs with  $\nu > 1$  remains a challenge because of difficulties in developing appropriate error control strategies for all of the variables.

### 3.4 Other Methods

We now consider some of the other numerical methods that have been applied to DAE problems, studying in depth both the extrapolation methods, mentioned previously, and the Taylor methods derived by Campbell in [23].

#### 3.4.1 Extrapolation

Extrapolation methods have been used for solving linearly implicit index 1 and 3 DAEs [30, 41, 48, 67, 78]. For the sake of brevity, we consider only the index 1 case.

$$(a) \quad Bx'(t) = f(x, t) \quad [30]$$

The DAE can be stated in an autonomous form by including the equation  $t' = 1$  in the system and  $t$  in  $x$ . The resulting system,  $Bx' = f(x)$ , reduced to

$$y' = f(y, z), \quad y(0) = y_0 \quad (3.36)$$

$$0 = g(y, z), \quad z(0) = z_0, \quad (3.37)$$

inspires the study of the following class of one-step methods:

$$\begin{aligned} y_{n+1} &= y_n + h\Phi(y_n, z_n, h), \\ z_{n+1} &= \Psi(y_n, z_n, h). \end{aligned} \quad (3.38)$$

If the method (3.38) satisfies the consistency conditions

$$\begin{aligned} y(t+h) - y(t) - h\Phi(y(t), z(t), h) &= d_{p+1}(t+h)h^{p+1} + d_{p+2}(t+h)h^{p+2} + \dots \\ z(t+h) - \Psi(y(t), z(t), h) &= e_p(t+h)h^{p+1} + e_{p+1}(t+h)h^{p+1} + \dots \end{aligned}$$

and the contractivity equation

$$\left\| \frac{\partial \Phi(y, z, 0)}{\partial z} \right\| \leq \alpha < 1$$

in a neighbourhood of the solution, then the global error has a perturbed asymptotic expansion of the form

$$\begin{aligned} y_n - y(t_n) &= b_p(t_n)h^p + (b_{p+1}(t_n) + \beta_n^{p+1})h^{p+1} + \cdots + (b_N(t_n) + \beta_n^N)h^N + B(n, h)h^{N+1} \\ z_n - z(t_n) &= (c_p(t_n) + \gamma_n^p)h^p + \cdots + (c_N(t_n) + \gamma_n^N)h^N + C(n, h)h^{N+1}, \end{aligned}$$

where  $t_n = nh$ , all coefficients vanish for  $n = 0$ ,  $\beta_n^j, \gamma_n^j = O(\bar{\alpha}^n)$  for some  $\bar{\alpha}$ ,  $\alpha < \bar{\alpha} < 1$ , and  $B, C$  are uniformly bounded for  $nh \leq t$ . For the semi-implicit Euler method, which solves the differential equation  $y' = f(y)$  reads

$$(I - hf'(y_0))(y_{n+1} - y_n) = hf(y_n)$$

and corresponds to the BEM followed by one Newton iteration, the following results hold.

- $\beta_n^2 = 0, \beta_n^3 = 0, \beta_n^4 = 0, \gamma_n^1 = 0$ , for  $n \geq 0$ .
- $\gamma_n^2 = 0, \gamma_n^3 = 0$ , for  $n \geq 1$ .
- $\beta_n^{j+1} = 0, \gamma_n^j = 0$ , for  $n \geq j-2$  and  $j \leq 4$ .

If  $Y_{n,1}$  and  $Z_{n,1}$  (for  $n = 1, 2, 3, \dots$ ) denote the numerical solution for  $y$  and  $z$  respectively in (3.36), (3.37) at  $H = nh_n$ , obtained by applying (3.38)  $n$  times with stepsize  $h_n$ , the approximations obtained by repeated Richardson extrapolation are given by

$$Y_{i,k+1} = Y_{i,k} + \frac{Y_{i,k} - Y_{i-1,k}}{(n_i/n_{i-k}) - 1}.$$

One choice for the numbers  $n_i$  is  $n_i = i$ . A similar relation holds for  $Z_{i,k}$ . The above results can be used (see [30] for details) to explain the order reduction observed in some entries of the extrapolation tableau (Table 3.6). The global order of accuracy of  $Y_{i,j}$  (or  $Z_{i,j}$ ) is  $O(H^q)$ , where  $q$  is the  $(i, j)$ -th entry of the tableau. This method has been incorporated into the code LIMEX (cf. §3.3.1), which also includes the following implementation considerations.

### Order and stepsize control

Separation of  $y$  and  $z$  components involves impractical rank-decision devices. However, it was found that using Table 3.6(a) for all components results in treating the  $z$ -parts more cautiously than necessary, which is acceptable in a robust algorithm. Otherwise, an ODE order and stepsize control policy is used.

|   |   |   |   |   |   |   |   |   |   |  |
|---|---|---|---|---|---|---|---|---|---|--|
| 1 |   |   |   |   |   |   |   |   |   |  |
| 1 | 2 |   |   |   |   |   |   |   |   |  |
| 1 | 2 | 3 |   |   |   |   |   |   |   |  |
| 1 | 2 | 3 | 4 |   |   |   |   |   |   |  |
| 1 | 2 | 3 | 4 | 4 |   |   |   |   |   |  |
| 1 | 2 | 3 | 4 | 4 | 4 |   |   |   |   |  |
| 1 | 2 | 3 | 4 | 4 | 5 | 4 |   |   |   |  |
| 1 | 2 | 3 | 4 | 4 | 5 | 5 | 4 |   |   |  |
| 1 | 2 | 3 | 4 | 4 | 5 | 6 | 5 | 4 |   |  |
| 1 | 2 | 3 | 4 | 4 | 5 | 6 | 6 | 5 | 4 |  |

|   |   |   |   |   |   |   |   |   |   |  |
|---|---|---|---|---|---|---|---|---|---|--|
| 2 |   |   |   |   |   |   |   |   |   |  |
| 2 | 2 |   |   |   |   |   |   |   |   |  |
| 2 | 2 | 3 |   |   |   |   |   |   |   |  |
| 2 | 2 | 3 | 4 |   |   |   |   |   |   |  |
| 2 | 2 | 3 | 4 | 4 |   |   |   |   |   |  |
| 2 | 2 | 3 | 4 | 5 | 4 |   |   |   |   |  |
| 2 | 2 | 3 | 4 | 5 | 5 | 4 |   |   |   |  |
| 2 | 2 | 3 | 4 | 5 | 6 | 5 | 4 |   |   |  |
| 2 | 2 | 3 | 4 | 5 | 6 | 6 | 5 | 4 |   |  |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 6 | 5 | 4 |  |

Table 3.6: Order pattern for extrapolated (a)  $y$ -components, (b)  $z$ -components

## Index monitor

For the BEM we have:

- $y_1 - y_0 = c_2 h^{2-\nu}$  for  $\nu > 0$ , given consistent initial values.
- $y_1 - y_0 = c_3 h^{1-\nu}$  for  $\nu > 0$ , for inconsistent initial values.

This can easily be monitored in the context of extrapolation. A rough estimate of the exponent  $p$  can be found from

$$\frac{\|y_1(h_i) - y_0\|}{\|y_1(h_{i-1})\|} = \rho \left( \frac{h_i}{h_{i-1}} \right)^p, \quad (3.39)$$

on the basis of which

$$\nu = \begin{cases} 1, & \nu = 0 \\ 1, & \nu = 1 \text{ with consistent } y_0 \\ 0, & \nu = 1 \text{ with inconsistent } y_0 \\ 0, & \nu = 2 \text{ with consistent } y_0 \\ < 0, & \nu > 2 \end{cases}$$

and LIMEX terminates when  $p \leq 0$  is detected. We note that LIMEX has been found to be comparable in performance to the widely-used BDF code DASSL (see later), for those problems for which LIMEX is applicable [30].

$$(b) \quad B(x, t)x' = f(x, t) \quad [67]$$

As for the constant-coefficient case, perturbed asymptotic expansions can be derived for the semi-explicit Euler method applied to the embedded system

$$\begin{aligned} y' &= z \\ 0 &= f(y) - B(y)z. \end{aligned}$$

## 3.4.2 Taylor-Type Methods

 $(i, j)$ -Methods

We discuss advantages, disadvantages and implementation of a family of Taylor-type methods [23] for solving almost all known classes of solvable linear DAEs,

$$A(t)x'(t) + B(t)x(t) = f(t), \quad (3.40)$$

of size  $n$ . These methods circumvent the ill-conditioning of the iteration matrix as  $h \rightarrow 0$ , and can attain arbitrarily high order.

First we develop the methods. Suppose (3.40) is solvable, and we require  $x(t)$ , knowing  $x(\hat{t})$ . The Euler method consists of finding  $x'(\hat{t})$  and then estimating  $x(\hat{t} + h)$  by

$$x(\hat{t} + h) \approx \hat{x}(\hat{t} + h) = x(\hat{t}) + hx'(\hat{t}).$$

First we try to solve (3.40) for  $x'(\hat{t})$ . Let

$$x(t) = \sum_{i=0}^{\infty} \delta^i x_i, \quad A(t) = \sum_{i=0}^{\infty} \delta^i A_i, \quad B(t) = \sum_{i=0}^{\infty} \delta^i B_i, \quad f(t) = \sum_{i=0}^{\infty} \delta^i f_i,$$

where  $\delta = t - \hat{t}$  and  $c_i = c^{(i)}(\hat{t})/i!$  for  $c = x, A, B, f$ . Substituting these Taylor expansions into (3.40) yields the system

$$\begin{pmatrix} A_0 & 0 & 0 & 0 & \cdots \\ A_1 + B_0 & 2A_0 & 0 & 0 & \cdots \\ A_2 + B_1 & 2A_1 + B_0 & 3A_0 & 0 & \cdots \\ A_3 + B_2 & 2A_2 + B_1 & 3A_1 + B_0 & 4A_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{pmatrix} = \begin{pmatrix} f_0 - B_0 x_0 \\ f_1 - B_1 x_0 \\ f_2 - B_2 x_0 \\ f_3 - B_3 x_0 \\ \vdots \end{pmatrix}$$

or  $A\mathcal{X} = \mathcal{F}$ . (3.41)

Any solution of (3.41) must satisfy

$$\begin{pmatrix} A_0 & 0 & 0 & \cdots \\ A_1 + B_0 & 2A_0 & 0 & \cdots \\ \vdots & \ddots & \ddots & \vdots \\ A_{j-1} + B_{j-2} & \cdots & \cdots & jA_0 \end{pmatrix}_{nj \times nj} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \end{pmatrix}_{nj \times 1} = \begin{pmatrix} f_0 - B_0 x_0 \\ f_1 - B_1 x_0 \\ \vdots \\ f_{j-1} - B_{j-1} x_0 \end{pmatrix}_{nj \times 1}$$

ie.  $A_j \mathcal{X}_j = \mathcal{F}_j, \quad j \geq 0. \quad (3.42)$

Since  $A$  is singular, (3.42) does not determine  $\mathcal{X}_j$  uniquely. We call  $A_j$   $r$ -full if there is a matrix  $R$  such that

$$RA_j = \begin{pmatrix} I_{s \times s} & 0 \\ 0 & H \end{pmatrix}, \quad z = rn. \quad (3.43)$$

Thus (3.42) uniquely determines  $x_1, \dots, x_s$  iff  $A_j$  is  $r$ -full with  $r \geq s$ . Since solutions to (3.42) are nonunique, we are free to choose to solve it in the least squares sense, so that

$\mathcal{X}_j = A_j^\dagger \mathcal{F}_j$ , where  $A_j^\dagger$  is the Moore-Penrose (generalized or minimal norm) inverse of  $A_j$ .  $A_j^\dagger$  is uniquely given by  $AA^\dagger A = A$  and  $A^\dagger AA^\dagger = A^\dagger$ . We can now define the  $i$ -th order  $j$ -block method, or  $(i, j)$ -method, with stepsize  $h$  as follows:

Given  $x_0 = x(\hat{t})$ , form  $A_j \mathcal{X}_j = \mathcal{F}_j$  and solve for  $\tilde{\mathcal{X}}_j = A_j^\dagger \mathcal{F}_j$ ,  $\tilde{\mathcal{X}}_j = (\tilde{x}_1^T, \dots, \tilde{x}_i^T)^T$ . Then estimate  $x(\hat{t} + h)$  by

$$\tilde{x}(\hat{t} + h) = x_0 + h\tilde{x}_1 + h^2\tilde{x}_2 + \dots + h^i\tilde{x}_i. \quad (3.44)$$

If  $A_j$  is  $i$ -full, (3.44) will be  $O(h^{i+1})$  accurate if  $x(\hat{t})$  is  $O(h^i)$  accurate. There are obvious advantages over implicit methods:

1. The method should result in a globally  $O(h^i)$  approximation.
2. The conditioning of (3.42) is  $h$ -independent.
3. The stepsize can easily be varied.

Note that for  $A = I$  the method reduces to a high order Taylor ODE method.

### Applicability

The applicability of  $(i, j)$ -methods depends on the generality of the fullness assumption on  $A_j$ . The following theory is stated and proved in [23].

1. Suppose  $A_j$  is  $s$ -full on  $[a, b]$  and there is a continuous invertible  $R(t)$  such that

$$R(t)A_j(t) = \begin{pmatrix} I_{ns \times ns} & 0 \\ 0 & H(t) \end{pmatrix}. \quad (3.45)$$

Then the  $(s, j)$ -method is  $O(h^s)$  accurate on  $[a, b]$ .

### Proof

Let  $R(t) = \begin{pmatrix} R_1(t) & R_2(t) \\ R_3(t) & R_4(t) \end{pmatrix}$ . Then (3.42) yields

$$\begin{pmatrix} x_1 \\ \vdots \\ x_s \end{pmatrix} = \begin{pmatrix} R_1(t) \\ R_2(t) \end{pmatrix} \mathcal{F}_j(t). \quad (3.46)$$

Since  $R_1, R_2$  are continuous and hence uniformly bounded on  $[a, b]$ , we are done.  $\square$

If (3.45) holds, with invertible and  $l$ -times differentiable  $R(t)$ , we call  $A_j$   $l$ -times smoothly  $s$ -full.

2. If  $A_j$  is 1-full on  $[a, b]$ , there is a continuous invertible  $R(t)$  such that

$$R(t)A_j(t) = \begin{pmatrix} I_{n \times n} & 0 \\ 0 & H(t) \end{pmatrix}, \quad R(t) = \begin{pmatrix} R_{11} & \dots & R_{1j} \\ R_{21} & \dots & R_{2j} \end{pmatrix},$$

and each  $R_{1r}$  is  $n \times n$ ,  $r = 1, \dots, j$ , then every solution of (3.40) is a solution of

$$x' = Q(t)x + \bar{f}(t), \quad (3.47)$$

$$\text{with } Q(t) = - \sum_{i=0}^{j-1} R_{1,i+1}(t)B_i(t) \quad \text{and} \quad \bar{f}(t) = \sum_{i=0}^{j-1} R_{1,i+1}(t)f_i(t).$$

Note that the  $(i, j)$ -method is really solving (3.47) by an explicit Euler method and that the above procedure has embedded the solutions of the DAE (3.40) into those of the ODE (3.47). The continuity in the first  $ni$  rows of  $A$  is crucial.

3. If  $A_j$  is  $s$ -full on  $[a, b]$ , and  $A_j$  has constant rank on  $[a, b]$ , then the first  $ns$  rows of  $A_j^\dagger(t)$ , denoted  $(\bar{R}_1(t), \bar{R}_2(t))$  may replace  $(R_1, R_2)$  in (3.46) and  $\bar{R}_1(t), \bar{R}_2(t)$  are as smooth as  $A(t), B(t)$  are. Note that this does not assume that  $A(t)$  has constant rank, but  $A(t)$  must have.
4. Suppose (3.40) is solvable with constant-coefficients  $A, B$  and index  $\nu$ . Then  $A_{\nu+1}$  is 1-full so that the  $(1, \nu+1)$ -method gives an  $O(h)$  approximation on  $[a, b]$ .
5. For the above problem, if  $f \in C^{\nu+i}$  then  $A_{\nu+i}$  is  $i$ -full and the  $(i, \nu+i)$ -method is  $O(h^i)$  accurate on  $[a, b]$  if the starting value is  $O(h^i)$  accurate.
6. For variable-coefficient systems of the form

$$A(t)x' + x = f(t), \quad (3.48)$$

suppose that  $A(t)$  has index  $\leq 2$  on  $[a, b]$  and is  $C^1$  and  $A^2(t)$  has constant rank. Define  $P, Q, N, C$  by  $P = A^D A$ ,  $Q = I - P$ ,  $C = AA^D A$ ,  $N = (I - A^D A)A$ , where  $A^D$  is the Drazin inverse of  $A$ , uniquely specified by  $AA^D = A^D A$  and  $A^D AA^D = A^D$ . If  $I - N'(t)$  is invertible, then (3.48) is solvable and we can find its solutions using transformations  $(P, Q)$  to obtain SCF  $(N, C)$ . If, in addition,  $I \pm 2N'(t)$  is invertible and  $\text{rank}(Q(I - N')Q) = \text{rank}(Q(-2N')Q) = \text{rank}(Q)$ , then  $A_3$  has constant rank and is smooth  $s$ -full.

Suppose we are trying to solve (3.40) by using differentiations and coordinate changes (cf. reduction techniques in Chapter 2). It is shown in [23] that when  $(i, j)$ -methods work with  $j$ -differentiations, then  $A_{j+1}$  is 1-full. Since the  $(i, j)$ -methods work directly with derivatives of  $A(t), B(t)$ , instead of differentiating time-varying coordinate changes applied to differentiated blocks, this method is probably easier to apply to most higher-index problems.

It is shown in [23] that the corresponding coordinate changes on  $A, B$ , such that  $A_j \rightarrow \bar{A}_j$ , do not alter the rank or fullness. Thus, to show that  $A_j$  is, say, 1-full and of constant rank, we need only show this for  $\bar{A}_j$ .

Additional comments

1. Error tolerance: If  $A_j$  is  $s$ -full,  $\hat{A}_j = A_j + O(\epsilon)$ , and  $\text{rank}(\hat{A}_j) = \text{rank}(A_j)$ , then  $\hat{A}_j^\dagger = A_j^\dagger + O(\epsilon)$ . Thus we need  $\epsilon = h^s$ , to maintain local  $O(h^s)$  accuracy.
2. It is possible to limit the amount of smoothness of the solution required for (3.42).
3.  $(i, j)$ -methods easily handle many higher-index solvable systems with nonconstant ranks.
4. Although Taylor-type methods are less implementable than other ODE methods, they provide a tool for understanding linear and more general DAEs. In particular, we can now give characterizations of solvability and index for linear systems.

**Theorem 3.4.1** [14] Suppose that  $A(t)$  and  $B(t)$  are  $2m$ -times continuously differentiable and (3.40) is solvable on  $I$ . Then

1.  $A_j$  has constant rank on  $I$  for  $j = m+1$ ;
2.  $A_j$  is smoothly 1-full on  $I$  for  $j = m+1$ ; and
3.  $\mathcal{R}(A_j) + \mathcal{R}(B_j) = \mathcal{R}^{mj} \forall t \in I, 1 \leq j \leq m+1$ . Here  $B_j = (B_0 \dots B_{j-1})^T$ .

If  $A(t)$  and  $B(t)$  are  $3m$ -times differentiable, then (1)–(3) above imply that (3.40) is solvable.

It is proposed in [14] that the index is the smallest integer  $\nu$  such that  $A_{\nu+1}$  is 1-full and has constant rank.

**Theorem 3.4.2** [14] Suppose  $A(t)$  and  $B(t)$  are  $2m$ -times continuously differentiable, (3.40) is solvable, and  $A_j$  is 1-full and has constant rank. Then the linear manifold of consistent initial conditions is the set of all  $x(t_0)$  such that (3.42) is consistent, and can be found by the following algorithm:

1. Compute  $W$  such that  $A_j W = 0$  and  $\text{rank}(W) = \text{nullity}(A_j)$ .
2. Solve  $W^T F_j = 0$  for  $x$  at  $t_0$ .

**Least Squares Methods**

Recent improvements of  $(i, j)$ -methods have resulted in the *constrained least squares* (CLS) methods [8], and, more generally, *least squares completions* [24]. For the same amount of work, CLS methods have the same order of accuracy and better stability properties, and have been found experimentally to be more accurate, than  $(i, j)$ -methods. The main difference lies in their implicit structure, allowing exact integration of the algebraic part

of the DAE. CLS methods use  $j$  differentiations of the DAE, and different CLS methods use different additional conditions to determine the solution uniquely.

### Taylor-type (CLS- $T_{j,k}$ )

Using the Taylor-expansion

$$x(t_n) - x'(t_n)h + x''(t_n)h^2/2 \dots (-1)^k x^{(k)}(t_n) h^k/k! \approx x(t_{n-1}),$$

$\mathcal{X}_j = (x_n, x_n^{(1)}, \dots, x_n^{(j)})^T$ ,  $j \geq k$ , (note the difference from the  $\mathcal{X}$  of (3.41)) is determined from the least squares problem

$$G_2 \mathcal{X}_j \approx x_{n-1} \quad (3.49)$$

$$\text{subject to } G_1 \mathcal{X}_j = b. \quad (3.50)$$

Here,  $G_1$  and  $b$  are derived from the first  $j$  derivatives of the DAE, and the underdetermined system (3.49) is derived from the Taylor series, ie.

$$G_2 = (I, -hI, h^2 I/2, \dots, (-1)^k h^k I/k!, 0, \dots, 0).$$

### Padé-type (CLS- $P_{j,k}$ )

A higher order of accuracy is obtained using derivatives at both  $t_n$  and  $t_{n-1}$ . In the above, (3.49) is replaced by

$$H_2 \mathcal{X}_j \approx c,$$

$$\text{with } H_2 = (I, \mu_1 h I, \mu_2 h^2 I, \dots, \mu_k h^k I, 0, \dots, 0),$$

$$\text{and } c = x_{n-1} + \omega_1 h x_{n-1}^{(1)} + \dots + \omega_l h^l x_{n-1}^{(l)}.$$

The parameters  $\mu_i$ ,  $i = 1 \dots k$ , and  $\omega_i$ ,  $i = 1 \dots l$ , are chosen from

$$\frac{1 + \sum_{i=1}^l \omega_i h^i}{1 + \sum_{i=1}^k \mu_i h^i} \approx e^h.$$

Order results and experiments for linear time-varying DAEs can be found in [8]. CLS methods may be preferable to BDFs and IRKs for higher index systems, but are time-consuming for index 1 systems. Their high accuracy (up to  $O(h^{2k+1})$ ) is not surprising since analytical computation of derivatives of the DAE should give better results than methods using only function evaluations.

In [24], general results for least squares completions using the derivative array (see page 28) for general DAEs are obtained. However, theoretical foundations, generality and limitations of least squares methods remain to be determined.

### The Strangeness Index

We summarize three recent papers by Kunkel & Mehrmann on linear DAEs, generalizing Taylor-type methods:

- [57] New existence and uniqueness results are obtained for linear DAEs, based on the KCF.
- [56] A new set of local characterizing quantities are found, leading to new global invariants generalizing the differential index,  $d_i$ , to include undetermined solution components. The global invariants are the dimensions of the differential and algebraic variables (after reduction to canonical form) and the strangeness index,  $s_i$ . When both  $d_i$  and  $s_i$  are well-defined, either  $d_i = s_i = 0$  or  $d_i = s_i + 1$ . Nonvanishing strangeness corresponds to higher-index.
- [55] A class of discretization methods for linear DAEs is developed. Computation of the local invariants leads to the numerical evaluation of global invariants, especially  $s_i$ , and a strangeness-free system, solved by any index 1 method, is extracted from the original system and its derivatives. Experiments show the success of these methods for arbitrarily high-index DAEs.

### 3.5 Consistent Initialization

For the DAE

$$F(x, x', t) = 0, \quad x(t_0) = x_0, \quad \text{with singular } \partial F / \partial x', \quad (3.51)$$

there is an underlying algebraic subsystem which imposes a consistency requirement on the solution, so that there is not a solution through every initial value. When the ODE methods described above are used to solve a DAE, small inconsistencies in the initial values cause the method to fail or to become extremely inefficient. Poor approximations to the initial derivatives may lead to a sequence of costly failures, as the stepsize is reduced to pass the relevant error test because of the poor initial prediction. Also, the first iteration at the first step will either not converge, converge very slowly, or converge to an undesired solution.

Although consistent initial conditions (c.i.c.'s) can sometimes be determined from physical considerations, their calculation more often requires a greater understanding of the underlying relationships between variables than is needed to state the problem. As an example consider the index 3 plane pendulum (of §4.6). The initial position can be any point on the unit circle, satisfying the algebraic constraint at  $t_0$ , but the initial velocity must be consistent with the first derivative of the constraint at  $t_0$ . The tension  $T$  must be derived either from physical considerations or by a second differentiation of the constraint, and its derivative  $T'$  by yet another. The point is that we need information about the state of the system which is hard to find even though that information is contained in the system itself. So we need to address the *consistent initialization problem*:

Given specified information about the initial state of the system, sufficient to specify a unique solution to the DAE, determine the complete initial vector  $(x(t_0), x'(t_0))$  corresponding to this unique solution.

We do this by developing a system of equations involving derivatives of (3.51) that an initial value  $x(t_0)$  and derivative  $x'(t_0)$  must satisfy in order to lie on a smooth solution of the DAE. Note that this is essentially what we did for the Taylor-type methods considered earlier. Since the derivatives of the general nonlinear DAE are often difficult to obtain analytically, numerical methods are devised.

We first discuss work that has been done on the consistent initialization problem.

1. In the code DASSL [14], Petzold determines the initial derivative  $x'(t_0)$  by taking a small backward Euler step and relying on a damped Newton iteration to determine the correct value. However, the initial value  $x(t_0)$  must be known.
2. Campbell [23] suggests a Taylor-series expansion approach for determining c.i.c.'s for the linear time-varying problem which, however, requires a knowledge of the derivatives of the matrices. This has recently been extended to nonlinear problems but it is unlikely to be competitive for problems whose partial Jacobian matrices are either unknown or difficult to compute.
3. Pantelides [79] uses a graph-theoretic algorithm to determine the minimal set of equations to be differentiated to solve for c.i.c.'s. This algorithm has low computational complexity and is also useful for transforming DAEs to a form amenable to solution by existing codes with minimal distortion of the structure of the original system. Unfortunately, it is possible for an equation subset that ought to be differentiated to escape detection by the algorithm.
4. Mrziglod (cf. [79]) determines consistent initial values using a thorough decomposition of the system structure. This approach isolates the algebraic constraints by index reduction and requires knowledge of analytical expressions for various derivatives of the problem.
5. Leimkuhler *et al.* [59] solve the consistency equations for certain important classes of DAEs using a family of one-sided finite difference approximations to the original equations.

We now discuss implementation of the methods of (5) above. We first introduce the *pseudo-derivative* operator, in order to restrict our attention to the initial point. The pseudo-derivative  $DG$  of a function  $G = G(\zeta_0, \dots, \zeta_p, t)$  is given by

$$DG(\zeta_0, \dots, \zeta_{p+1}, t) = \left( \frac{\partial G}{\partial \zeta_0} \right) \zeta_1 + \left( \frac{\partial G}{\partial \zeta_1} \right) \zeta_2 + \dots + \left( \frac{\partial G}{\partial \zeta_p} \right) \zeta_{p+1} + \left( \frac{\partial G}{\partial t} \right),$$

where the partials are evaluated at  $(\zeta_0, \dots, \zeta_p, t)$ . (Here,  $\zeta_i$  is  $x^{(i)}(t)$ .) We call (3.51) *smoothly solvable* on  $[t_0, T]$  if  $F$  has  $m$  continuous derivatives and the nonlinear system of

equations

$$\begin{aligned} F(\zeta_0, \zeta_1, t) &= 0 \\ DF(\zeta_0, \zeta_1, \zeta_2, t) &= 0 \\ &\vdots \\ D^m F(\zeta_0, \zeta_1, \zeta_2, \dots, \zeta_{m+1}, t) &= 0 \end{aligned}$$

can be solved for  $\zeta_1 = \phi(\zeta_0, t)$  on  $[t_0, T]$ , where  $\phi$  is Lipschitz continuous. Clearly,  $m$  is the differential index. Note that the  $\zeta_i$ ,  $i = 2, \dots, m+1$ , are not uniquely determined by the system of derivatives. The equation  $x' = \phi(x, t)$ , referred to as the *underlying ODE*, is assumed to have a very smooth ( $C^{m+1}$ ) solution in a neighbourhood of  $t_0$ .

### The Consistency Equations

The user may have information about the initial state, in the form of algebraic equations

$$B(x_0, x'_0) = 0, \quad (3.52)$$

in addition to which the problem itself must be satisfied at  $t_0$ :

$$F(x_0, x'_0, t_0) = 0. \quad (3.53)$$

Also, depending on the index,  $(x_0, x'_0)$  must satisfy the  $m$  derivative equations:

$$\begin{aligned} DF(x_0, x'_0, x''_0, t_0) &= \left( \frac{\partial F}{\partial x} \right) x'_0 + \left( \frac{\partial F}{\partial x'} \right) x''_0 + \left( \frac{\partial F}{\partial t} \right) \\ &\vdots \\ D^m F(x_0, x'_0, x''_0, \dots, x_0^{(m+1)}, t_0) &= 0 \end{aligned} \quad (3.54)$$

for some choice of  $(x''_0, \dots, x_0^{(m+1)})$ , where all partial derivatives of  $F$  are evaluated at  $(x_0, x'_0, t_0)$ . The  $m+2$  equations (3.52)–(3.54) are called the *consistency equations*. If the problem

$$\begin{aligned} F(x(t), x'(t), t) &= 0 \\ B(x(t_0), x'(t_0)) &= 0 \end{aligned}$$

has precisely one continuous solution  $\bar{x}(t)$  and it is smooth ( $\bar{x}(t) \in C^{m+1}[t_0, T]$ ), then the consistency equations have a solution  $(x_0, x'_0, \dots, x_0^{(m+1)})$ , and  $x_0 = \bar{x}_0$  and  $x'_0 = \bar{x}'(t_0)$  are uniquely defined. So, assuming that enough information is known about the initial state, we can in principle differentiate the constraints a few times and manipulate the resulting equations to find the c.i.c.'s. But this strategy is inadequate as it stands because the system of consistency equations requires the formation of pseudo-derivatives which are expensive to compute, and because (3.52)–(3.54) is hard to solve, in general.

## Compatible Approximations

We now show how to approximate the pseudo-derivatives using finite differences. Firstly, we consider a particular approximation to the first derivative in the index 1 case. Define  $z(t) = (x(t), x'(t), t)$ . We seek an approximation to the derivative

$$\left. \frac{dF(z(t))}{dt} \right|_{t=t_0} = J_F(z(t_0))z'(t_0),$$

where  $J_F$  is the Jacobian  $dF/dz$ . An obvious choice is the simple one-sided difference,

$$\frac{F(z_0 + hz'_0) - F(z_0)}{h}, \quad (3.55)$$

where  $z_0 = z(t_0)$  and  $z'_0 = z'(t_0)$ . The argument  $z_0 + hz'_0$  is a truncated Taylor series expansion for  $z(t_0 + h)$ . Since we assume  $F$  to be  $C^2$ , the truncation error is  $O(h)$ . For the general case, we note that:

1. Since  $F$  is defined only for  $t \geq 0$ , we are limited to one-sided differences.
2. The displacement  $h$  need not be related to the stepsize of integration.
3. Since numerical differentiation is ill-conditioned, roundoff errors made in approximating the  $k$ -th derivative of  $F$  will be magnified by  $O(h^{-k})$ .

The idea is to choose points  $z_1, \dots, z_s$  as truncated series expansions of  $z(t_0 + c_1 h), \dots, z(t_0 + c_s h)$ , and use the values of  $F$  at those points, combined with coefficients  $\alpha_1 \dots \alpha_s$ , to annihilate terms in the Taylor expansion of the approximation. This yields the class of approximation methods described by

$$D_h^k F = \frac{k!}{h^k} \left[ \sum_{i=1}^s \alpha_i F \left( z_0 + c_i h z'_0 + \dots + \frac{(c_i h)^r}{r!} z_0^{(r)} \right) - \left( \sum_{i=1}^s \alpha_i \right) F(z_0) \right], \quad r \leq k. \quad (3.56)$$

In [59] it is proved that the truncation error of (3.56), as an approximation to the  $k$ -th pseudo-derivative, is  $O(h^l)$  provided that  $z$  is  $C^{r+1}$  near  $t_0$ ,  $f$  is  $C^{p+1}$  near  $z(t_0)$  for some  $p \geq r$ ,  $p \geq k + l - 1$ , and

$$\sum_{i=1}^s \alpha_i c_i^j = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}, \quad j = 1, \dots, k + l - 1.$$

Also,  $\min(p, s) - k + 1$  is the maximum attainable order of the  $k$ -th derivative approximation (3.56).

Now, if we substitute derivative approximations into the consistency equations the resulting system will probably not have a solution. This is because we generally do not know which of the variables are determined by the constraints and which are arbitrary.

The natural approach, then, is to solve the consistency system in the least squares sense. However, the application of this method is complicated by the known rank deficiency of  $J_F$ . For certain classes of problems it is possible to manipulate the exact consistency equations, and this applies as well to the approximate equations, resulting in a system of nonlinear equations having full column-rank Jacobians. These classes include the index 1 constant-coefficient problem and DAEs in the Hessenberg form.

**Example 3.4** For a semi-explicit system with  $k = 2$ , the consistency equations are

$$\begin{aligned} u'_0 &= f(u_0, v_0) \\ 0 &= g(u_0) \\ u''_0 &= (\partial f / \partial u) u'_0 + (\partial f / \partial v) v'_0 \\ 0 &= (\partial g / \partial u) u'_0 \\ u'''_0 &= \psi(u_0, v_0, u'_0, v'_0, u''_0, v''_0) \\ 0 &= (\partial^2 g / \partial u^2) \langle u'_0, u'_0 \rangle + (\partial g / \partial u) u''_0, \end{aligned} \quad (3.57)$$

where  $\psi$  includes first and second derivatives of  $f$  and  $\langle \cdot, \cdot \rangle$  denotes the inner product. (3.57) can easily be reduced since we only want  $(u_0, v_0, u'_0, v'_0)$ . This yields

$$\begin{aligned} u'_0 &= f(u_0, v_0) \\ 0 &= g(u_0) \\ 0 &= (\partial g / \partial u) f(u_0, v_0) \\ 0 &= (\partial^2 g / \partial u^2) \langle f(u_0, v_0), f(u_0, v_0) \rangle + (\partial g / \partial u) [(\partial f / \partial u) f(u_0, v_0) + (\partial f / \partial v) v'_0]. \end{aligned}$$

This reduction only requires knowledge of the separation according to the Hessenberg form and can equally be applied to the approximate consistency equations.

In [59], it is shown that solutions of approximations to the resulting regular least-squares problems are convergent under mild conditions. The solution of the consistency equations is also investigated, including existence and uniqueness for linear constant-coefficient and index 1 linearly implicit DAEs. The above methods, embodied in the code DAIS [59], are also applied successfully to the nontrivial initialization of the trajectory control problem of Petzold.

## Chapter 4

# Experiments

"Differential-algebraic equations have received extensive attention in the last decade and the theory is well developed. But we still do not have methods for high-index problems, there are no general codes for problems of index greater than one, and the determination of initial values remains a difficulty." (C. Gear 1988 [10])

In this chapter, we attempt to solve a variety of small problems arising in the many areas of application, using some of the currently available DAE codes.

- We start with a brief description and comparison of the codes used from a user interface point-of-view.
- Next, we outline our experimental methodology, including a higher-index strategy.
- For the major part, we take an in-depth look at each problem, including:
  - **Formulation:** description, background, motivation, manipulation and theoretical predictions; and
  - **Solution:** results, graphical and tabular output, comparison of code performance with respect to applicability, robustness, computational cost and accuracy obtained, comparison of results with predictions and with the work of others in the field, and conclusions.
- Finally, we draw general conclusions regarding code performance and the experiments undertaken.

The experiments draw extensively on the theoretical and practical concepts presented in Chapters 2 and 3, especially as regards solvability, index and index reduction, canonical and Hessenberg structural forms, consistent initialization, the relation between stiff ODEs and DAEs, scaling for higher-index problems and order predictions.

## 4.1 Code Introduction

We make use of three of the most widely used general-purpose codes available for the numerical solution of DAEs:

- **LSODI** [49], presented by Hindmarsh and Painter in 1980, is based on the BDF stiff ODE solver GEAR1B and complies with the ODEPACK interface standard. LSODI was designed for linearly implicit ODEs with an extension to DAEs, for which it is not expected to be robust. We include it in our comparison as a first attempt at a general purpose DAE code and for direct comparison with DASSL, another BDF code.
- **DASSL** [14], presented by Petzold in 1982, uses a different implementation of the BDFs to LSODI [14] and was designed to solve index 0 or 1 fully implicit DAEs. DASSL won the first Wilkinson Prize for Numerical Software in 1991 and is probably the most widely used DAE code at present. DASSL has also been extended to more specific tasks, specifically ODASSL for overdetermined DAEs, DASSAC for sensitivity analysis, and DASSLRT, which finds roots.
- **RADAU5** [47], presented by Hairer and Wanner in 1988 (cf. §3.3.1), is included as a representative (and flagship) of the IRK codes of [48], for a BDF vs. IRK comparison and for directly solving higher-index problems.

A more exhaustive DAE code comparison would have included the RADAU5-type Rosenbrock, SDIRK and extrapolation codes of [48] (cf. §3.3.1) and the extrapolation code LIMEX [30] (cf. §3.4.1), but this would not have contributed much to the observations made in this chapter, and the use of these codes is omitted for the sake of brevity. For each of these codes, the theory has as yet only been developed for the small class of linearly implicit index 1 DAEs.

We consider the following two forms of DAEs:

$$\begin{aligned} F(y, y', t) &= 0, & \text{fully implicit} \\ M(t, y)y' &= f(t, y), & \text{linearly implicit.} \end{aligned}$$

The abbreviations we use are defined in Table 4.1, while Table 4.2 shows basic code properties and Table 4.3 is a summary of user control parameters and available integration information provided by each code. A more detailed description and specific default values of sophisticated user parameters can be found in the relevant code documentation.

## 4.2 Methodology

### 4.2.1 Framework for Comparison of Code Performance

The framework we use for comparing code performance is built on the approach taken by other authors, notably [14], [32], [47] and [48], and finds its roots in the numerical solution of stiff ODEs. We start with some quotes and notes taken from these references.

|                 |                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------------------------------|
| $t, T$          | independent variable, endpoint of integration                                                               |
| $y, y(0)$       | dependent variable, initial value                                                                           |
| $y'(0)$         | initial derivative                                                                                          |
| c.i.c.          | consistent initial condition                                                                                |
| $h$             | discretization parameter, steplength                                                                        |
| RES             | residual vector $F$ or $My' - f$                                                                            |
| JAC             | Jacobian matrix ( $\partial F/\partial y$ or $\partial f/\partial y$ )<br>or number of Jacobian evaluations |
| MASS            | mass matrix $M$                                                                                             |
| p.d.            | partial derivative ( $\partial F/\partial y'$ )                                                             |
| f, func.        | function $F$ or $f$                                                                                         |
| FCN             | number of function evaluations                                                                              |
| ETF's, CTF's    | (number of) error test failures, convergence<br>test failures causing stepsize reduction                    |
| STEPS           | number of integration steps taken so far                                                                    |
| ORDER           | order of accuracy of method                                                                                 |
| $\nu$ , index   | differential index ( $d_i$ )                                                                                |
| ABS, REL        | absolute, relative                                                                                          |
| TOL, ATOL, RTOL | error tolerance, absolute and relative, resp.                                                               |
| NEWT            | number of modified Newton iterations                                                                        |
| ETIME           | computation time                                                                                            |
| $\Delta$ time   | output interval                                                                                             |

Table 4.1: Abbreviations

| Code              | RADAU5                          | LSODI                           | DA5SL             |
|-------------------|---------------------------------|---------------------------------|-------------------|
| Problem           | linear implicit<br>(constant M) | linear implicit<br>(variable M) | fully implicit    |
| $\nu$             | 0,1,2,3                         | 0,1                             | 0,1               |
| Method            | 5th order IRK                   | 1st-5th order BDF               | 1st-5th order BDF |
| Single/multi call | single                          | single/multi                    | single/multi      |

Table 4.2: Basic DAE code properties

| Code                                                        | RADAU5 | LSODI              | DASSL |
|-------------------------------------------------------------|--------|--------------------|-------|
| <b>Input options</b>                                        |        |                    |       |
| Output routine for single call                              | ✓      | x                  | x     |
| Abs/rel scalar/vector TOL                                   | ✓      | ✓                  | ✓     |
| Automatic/user-supplied JAC, $y'(0)$                        | ✓      | ✓                  | ✓     |
| $y'(0)$ consistency flag                                    | —      | x                  | ✓     |
| Automatic/user-supplied p.d.                                | —      | —                  | ✓     |
| Full/banded JAC storage                                     | ✓      | both or<br>neither | ✓     |
| Full/banded MASS/p.d. storage                               | ✓      |                    | ✓     |
| <b>Method control</b>                                       |        |                    |       |
| Max. STEPS per call                                         | ✓      | ✓                  | ✓     |
| Max. NEWT per step                                          | ✓      | x                  | x     |
| Starting value for Newton calculated                        | ✓      | x                  | ✓     |
| $h$ prediction safety and limits                            | ✓      | x                  | x     |
| JAC recomputation control                                   | ✓      | x                  | x     |
| Newton stopping criterion                                   | ✓      | x                  | x     |
| Initial $h$ ( $h_0$ ), max. $h$                             | ✓      | ✓                  | ✓     |
| Min. $h$                                                    | x      | ✓                  | x     |
| Optional TOL norm/weight function                           | x      | ✓                  | ✓     |
| Max. ORDER                                                  | —      | ✓                  | ✓     |
| Max. number of error messages                               | x      | ✓                  | x     |
| Provision for user parameters in<br>RES, JAC, etc. routines | x      | x                  | ✓     |
| <b>Output flags</b>                                         |        |                    |       |
| Success                                                     | ✓      | ✓                  | ✓     |
| Excessive work done                                         | x      | ✓                  | ✓     |
| Excessive accuracy required                                 | x      | ✓                  | ✓     |
| Illogical/incorrect input                                   | x      | ✓                  | ✓     |
| $y'(0)$ computation failure                                 | x      | ✓                  | ✓     |
| Repeated ETF's                                              | ✓      | ✓                  | ✓     |
| Repeated CTF's                                              | —      | ✓                  | ✓     |
| Component failing integration test                          | x      | ✓                  | ✓     |
| <b>General information</b>                                  |        |                    |       |
| Current $t$                                                 | x      | ✓                  | ✓     |
| Old $h$ , new $h$                                           | ✓ x    | ✓                  | ✓     |
| Old ORDER, new ORDER                                        | —      | ✓                  | ✓     |
| STEPS, JAC evals, p.d. calls                                | ✓      | ✓                  | ✓     |
| ETF's, CTF's                                                | x      | ✓                  | ✓     |

Table 4.3: User interface to DAE codes

• [32]: On the comparison of stiff ODE solvers:

- A stiff method is viewed as consisting of a strategy for choosing the next stepsize and order, a formula for computing the next approximation, a formula for estimating the local truncation error in the approximation and a rule for deciding whether to accept the approximation.
- "The user would like a bound on the global error that he/she is prepared to tolerate, but methods are designed to control some measure of the local error, and the user is required to specify a local error tolerance." (The three codes we use all attempt to keep the local error of  $y_i$  below  $RTOL_i \times ABS(y_i) + ATOL_i$ .)
- Suggested reliability criterion based on TOL:

$$\|y(t_{i+1}) - y_{i+1}\| < 6 \cdot TOL \cdot \frac{t_{i+1} - t_i}{t_{i+1} - t_0}, \quad TOL = 10^{-2}, 10^{-4}, 10^{-6}, \dots$$

- In the comparison of cost and reliability statistics, tests are restricted to relatively routine integration steps over a range of accuracy requirements, avoiding difficulties due to discontinuities, etc.
- Cost of JAC  $\approx$  cost of FCN in most cases.

• [14]: On the implementation of DAE codes:

- "There are issues other than the order of convergence of the methods which are important in successfully implementing and using the methods. These issues are more complex for DAEs than for ODEs. For example, the initial conditions for a DAE must be chosen to be consistent, the linear system which must be solved at each timestep is ill-conditioned for small stepsizes, error estimates used in the selection of stepsizes are sensitive to inconsistencies in the initial conditions and sharp changes in the solution, and the solution methods are dependent on a more accurate approximation to the iteration matrix than is generally needed for ODEs."
- "In general, the diagnostics in DAE codes are not as well developed as those in nonstiff ODE codes."
- Suggested use of TOL: for component  $y_i$ , set  $RTOL_i = 10^{-(m+1)}$ , where  $m$  is the number of desired significant digits, and set  $ATOL_i$  to a value at which  $|y_i|$  is essentially insignificant.

- As a reasonable measure of efficiency, [48] makes use of 'work-precision diagrams', in which global error is compared to computational effort over a range of integral, half-integral, etc., exponent tolerances. Typically, scalar tolerances and  $ATOL=RTOL=TOL$  is used, and the error is the maximum taken over all components and all output points, using a very strict TOL solution as a truth model if the true solution is unknown. Work is measured in terms of ETIME or FCN. The obtained data are displayed

as a polygonal line ('work-precision curve') in double logarithmic scales. The more this line is to the left, the higher was the obtained precision, and the lower the line is, the more economical was the code. The "slope" of the curve expresses the *effective order* of the code:  $\text{effective order} = -1/(\text{slope of work-precision curve})$ , so that  $m$  orders of magnitude of precision are obtained per order of magnitude of work if the effective order is  $m$ . The relationship between convergence order for a method and effective order, which is more appropriate for code comparison, is a complex one, depending on code and implementation particulars, and the two are only equal for constant work done per unit step. (The relation between work, steps and integration interval depicted in Figure 4.28 is typical for a routine integration.) Code comparisons can also be derived from work/precision ratios at fixed work or precision.

To the above we add the observation that effective code comparison is difficult because:

1. Different codes use different tuning parameters and defaults, and are suitable for different types of problems;
2. There is no standard set of example DAEs for code comparison (as there is for stiff ODEs), only several popular problems; and
3. Comparisons in the literature are few, typically for methods rather than codes, based on fixed-steplength order rather than user-defined parameters and used on idealized and artificial problems.

In the light of the above comments, we now outline our experimental technique. For each example, we first attempt to solve the problem as would a user, with particular values for TOL and  $T$ , and default parameters, requiring graphical and tabular solution output for components of interest as well as available integration statistics and error messages. We then attempt a more general comparison of code performance on a wide TOL range and a relatively routine integration interval, using safe parameter settings. We make particular use of work-precision diagrams (with context-dependent 'work' and 'precision'), from which we obtain values for effective order and work/precision ratios, using linear regression on some or all of the data.

Unless otherwise stated, user-defined parameters keep the default values provided by the source code, or are given values which are found not to be critical for the solution. For example, Jacobian matrices are calculated numerically by the code (default) unless results are found to be significantly different using an analytical user-defined Jacobian.

ETIME, when stated, is only used for relative comparison. One reason for this is its machine-dependence. Another is that intermediate output, usually to a data file, tends to dominate the computation time for small systems, especially if all solution components are required. For this reason, comparisons assume that identical amounts of intermediate output are requested. ETIME values are stated for either no intermediate output (no) or for output to hard disk (disk). The requested output interval ( $\Delta\text{time}$ ) is also stated. For RADAU5, which is called only once per implementation, intermediate output is provided

by an external routine, while DASSL and LSODI require multiple calls for intermediate output. In general, intermediate output is requested, enough to reflect graphically the required accuracy and solution behaviour.

All computations were performed on the same PC (33MHz 80486 DX chip). All source code, both DAE solvers (RADAU5, LSODI and DASSL) and the linear algebra libraries used by the solvers (DECSOL, supplied with RADAU5, and LINPACK, used by LSODI and DASSL), as well as all driver programs (written by the author) calling the solvers, were compiled and run in Microsoft FORTRAN 5.10 using double precision and time optimization. All output data processing, for error analysis, visualization, etc., was performed by MATLAB 4.

#### 4.2.2 Higher Index

Although all three codes were designed to solve DAEs of index 0 or 1, we do attempt to solve higher-index problems. The reason for this is that general purpose codes for solving higher-index DAEs are as yet unavailable. RADAU5 includes an option for scaling the error estimate used for stepsize selection for DAEs in Hessenberg form of size 2 and 3. For this, the Hessenberg form must be known and the unknown vector is ordered as follows: for Hessenberg of size 2,  $y = (x_1, x_2)^T$ , where  $x_1$  has length RWORK(5) and  $x_2$  ('index 2 components') has length RWORK(6); for Hessenberg of size 3,  $y = (x_2, x_1, x_3)^T$ ,  $x_2$  has length RWORK(5),  $x_1$  ('index 2') has length RWORK(6), and  $x_3$  ('index 3') has length RWORK(7). RADAU5 scales contributions to the error estimate used for stepsize selection by  $h$  for 'index 2' components and  $h^2$  for 'index 3' components.

The convention of ascribing the property of index to individual components, closely related to the concept of dynamic order in nonlinear control theory [14], has not found general acceptance in the literature, which caters almost exclusively for  $\nu = 0, 1$  DAEs. The reason we use it is that some DAEs do not appear to have a structural form to which order results, etc., in the theory can be applied. For example, (4.27) was not found to be reducible to Hessenberg form. Both differential and perturbation indices can be redefined intuitively on component level: the differential index of a component  $x_i$  of  $x$  is the least number of differentiations of the original DAE required to determine  $x'_i = x'_i(x, t)$ , and the perturbation index of  $x_i$  is defined by considering the  $i$ -th component of (2.28). This is consistent with DAE index in that some components will have an index as large as  $\nu$ , but none larger. Note that the Hessenberg form separates variables of different index, and Hessenberg size and index corresponds to maximal component index. The canonical form (cf. §2.1 and §2.2) also suggests that this component-level definition of index is consistent.

Unless otherwise stated, for RADAU5 we make use of the above scaling option, and use no other higher-index adjustment. For DASSL and LSODI, we attempt both matrix scaling (by scaling algebraic rows of RES by  $h^{-1}$ ) and variable scaling (by scaling TOL) (cf. §3.2.3) if integration is unsuccessful. Strategies requiring source code modification, such as omitting some variables from error tests, are not used here.

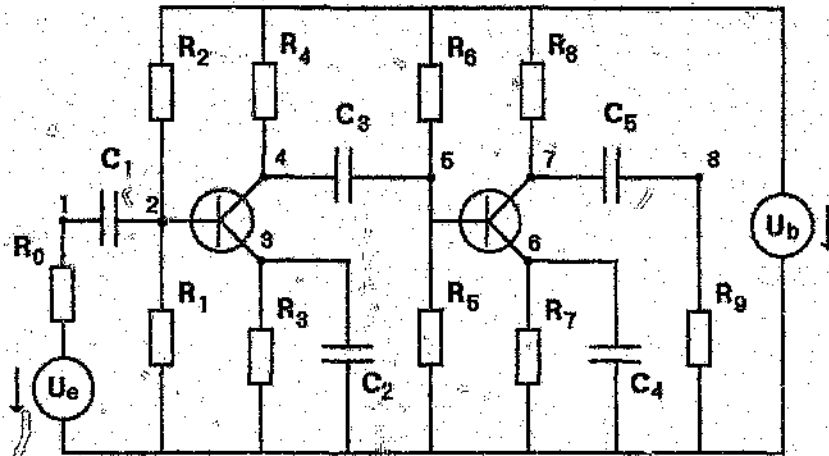


Figure 4.1: Circuit diagram for two-transistor amplifier

### 4.3 Two-Transistor Amplifier

#### Formulation

We model the circuit for a two-transistor amplifier [84] (see Figure 4.1). In terms of the unknown node voltages,  $y = (y_1, \dots, y_8)^T$  (cf. §1.1.1), and using (1.8) for Kirchhoff's current law and admittance form constitutive equations, the nodal equations  $Ay' = g(y)$  and capacity matrix  $A$  are given by

$$A = \begin{pmatrix} -1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & -1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & -1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{pmatrix}, g = \begin{pmatrix} \frac{1}{R_0 C_1} y_1 - \frac{1}{R_0 C_1} U_e(t) \\ \frac{1}{C_1} \left( \frac{1}{R_1} + \frac{1}{R_2} \right) y_2 - \left( \frac{\alpha-1}{C_1} \right) f(y_2 - y_3) - \frac{U_b}{C_1 R_2} \\ \frac{1}{C_2} f(y_2 - y_3) - \frac{1}{R_3 C_2} y_3 \\ \frac{U_b}{R_4 C_5} - \frac{1}{R_4 C_5} y_4 - \frac{\alpha}{C_5} f(y_2 - y_3) \\ \frac{1}{C_5} \left( \frac{1}{R_5} + \frac{1}{R_6} + \frac{1}{R_7} \right) y_5 - \left( \frac{\alpha-1}{C_5} \right) f(y_5 - y_6) \\ \frac{1}{C_4} f(y_5 - y_6) - \frac{1}{C_4 R_7} y_6 \\ \frac{U_b}{R_8 C_5} - \frac{1}{R_8 C_5} y_7 - \frac{\alpha}{C_5} f(y_5 - y_6) \\ \frac{C_4}{R_9} y_8 \end{pmatrix}$$

Dots denote zero entries. The characteristics of the transistors are modelled by

$$f(x) = \beta(e^{x/U_f} - 1).$$

Parameters and a set of c.i.c.s, derived from physical considerations, are

$$U_b = 6, \quad \alpha = 0.99, \quad \beta = 10^{-6}, \quad R_0 = 1000, \quad R_k = 9000, \quad k = 1 \dots 9,$$

$$C_k = k \cdot 10^{-6}, \quad k = 1 \dots 5, \quad U_f = 0.026, \quad U_e(t) = 0.1 \sin(200\pi t),$$

$$y(0) = \left( 0, \frac{U_b}{R_1 + 1}, \frac{U_b}{R_1 + 1}, U_b, \frac{U_b}{R_5 + 1}, \frac{U_b}{R_5 + 1}, U_b, 0 \right)^T = (0, 3, 3, 6, 3, 3, 6, 0)^T.$$

Of interest is the amplified exit voltage,  $y_3$ , for a given input signal  $U_e$ .

Perform a SVD on  $A$ :  $A = USV^T$  with  $U, V$  unitary and  $S$  diagonal.

Then  $Av' = g(v) \Rightarrow U^T Av' = U^T g(v) \Rightarrow SV^T v' = U^T g(v)$ ,

and  $V^T v = \begin{pmatrix} y \\ z \end{pmatrix}$ ,  $\begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = U^T g \Rightarrow \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y' \\ z' \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$ ,

with diagonal matrix  $D$  containing the nonzero elements of  $S$ , i.e. the singular values of  $A$ . This semi-explicit DAE has  $\nu = 1$  if  $\partial f_2 / \partial z$  has a bounded inverse, which in this case can easily be verified.

### Solution

The interval of integration was set to  $[0, 0.2]$  and  $\text{TOL} = 10^{-5}$  was requested. The DAE is banded, and we made full use of the bandedness. Full (rather than banded) storage increased ETIME by  $\approx 5$  seconds (for DASSL).  $\Delta\text{time} = 2 \cdot 10^{-4}$  was used. The initial derivative, required for DASSL and LSODI, was obtained by solving the DAE at the initial point, both analytically, with  $y'(0) = \{0, 0, -y_3(0)/R_3 C_2, 0, 0, -y_6(0)/R_7 C_4, 0, 0\}^T$ , and numerically, with  $y'(0) = 0$ . For DASSL, a nonzero INFO(11) indicates an inconsistent  $y'(0)$ , for which DASSL takes a small implicit Euler step using  $y(0)$ , and the inconsistent  $y'(0)$  as a starting guess, to find a numerically consistent  $y'(0)$ .

Refer to Table 4.4 and Figure 4.2 for the successful RADAU5 results, and Table 4.5 for the successful DASSL solution statistics. The 'error' is the maximum absolute global error over all components and all 1000 output points on  $[0, 0.2]$ , using the  $\text{TOL} = 10^{-12}$  RADAU5 solution as 'true', and ETIME values are for output to disk. The DASSL driver code and output are given in Appendix B. LSODI was partially successful, integrating to

| Jacobian   | STEPS | FCN   | JAC | ETF's | Error               | ETIME(no) | ETIME(disk) |
|------------|-------|-------|-----|-------|---------------------|-----------|-------------|
| analytical | 1194  | 10098 | 975 | 79    | $2.1 \cdot 10^{-3}$ | 10.4s     | 15.4s       |
| numerical  | 1196  | 10080 | 976 | 75    | $2.1 \cdot 10^{-3}$ | 10.9s     | 15.5s       |

Table 4.4: RADAU5 solution statistics for two-transistor amplifier

| Jacobian   | $y'(0)$    | INFO(11) | STEPS | FCN   | JAC  | ETF's | Error               | ETIME |
|------------|------------|----------|-------|-------|------|-------|---------------------|-------|
| analytical | analytical | 0        | 5638  | 13513 | 3986 | 2066  | $6.7 \cdot 10^{-3}$ | 27.4s |
|            |            | 1        | 8099  | 23032 | 8628 | 4389  | $6.5 \cdot 10^{-3}$ | 42.5s |
|            | numerical  | 0        | 5994  | 14484 | 4392 | 2304  | $6.6 \cdot 10^{-3}$ | 29.2s |
|            |            | 1        | 7775  | 21565 | 7930 | 4032  | $6.6 \cdot 10^{-3}$ | 40.3s |
| numerical  | analytical | 0        | 6820  | 17170 | 5580 | 2854  | $6.4 \cdot 10^{-3}$ | 39.7s |
|            | numerical  | 1        | 5469  | 13015 | 3701 | 1941  | $6.7 \cdot 10^{-3}$ | 30.4s |

Table 4.5: DASSL solution statistics for two-transistor amplifier

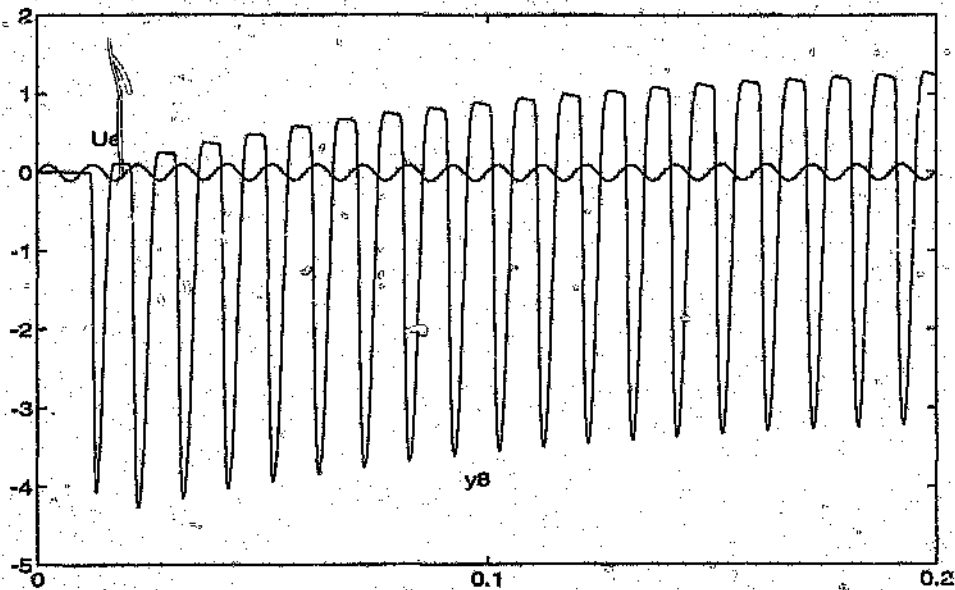


Figure 4.2: RADAU5 solution to two-transistor amplifier: Amplified exit voltage  $y_8(t)$  and input signal  $U_e(t)$ .

$t_{\text{lim}}$ , when repeated error test failures halted integration at a sharp gradient change.  $t_{\text{lim}}$  was found to vary erratically, depending on both number of LSODI calls ( $= T/\Delta\text{time}$ ) and the initial derivative. See Figure 4.3.

- Figure 4.2 appears to be identical to Figure 9.4 of [47], so that our results agree with [47] for the RADAU5 solution on  $t \in [0, 0.2]$  with  $\text{TOL} = 10^{-5}$ .
- For both RADAU5 and LSODI, solution statistics differed marginally when numerical Jacobians were used instead of analytical ones.
- For DASSL, the cost of obtaining a numerically consistent i.c. from one given as inconsistent was a significant portion of the total cost, even if the supplied i.c. is consistent (eg.  $\text{ETIME} = 40\text{s}$  compared to  $30\text{s}$ ). Solution statistics are marginally different for numerical or analytical  $y'(0)$ .
- The obtained minimal global accuracy is 2 to 3 orders of magnitude worse than  $\text{TOL}$ . Although no direct comparison should be made, 2 to 3 significant digits were obtained, compared to the heuristic request of 4 (cf. p. 66).
- For  $\text{TOL} = 10^{-5}$ , DASSL required at least  $\approx 5$  times as many steps,  $\approx 4$  times as many Jacobians,  $\approx 30\%$  more function evaluations and  $\approx 80\%$  more time, obtaining half an order of magnitude less accuracy, than RADAU5.
- The numerical difficulty experienced by DASSL (and which also caused LSODI to fail) is evidenced by the number of failed error tests causing step reduction ( $\approx 30$  times

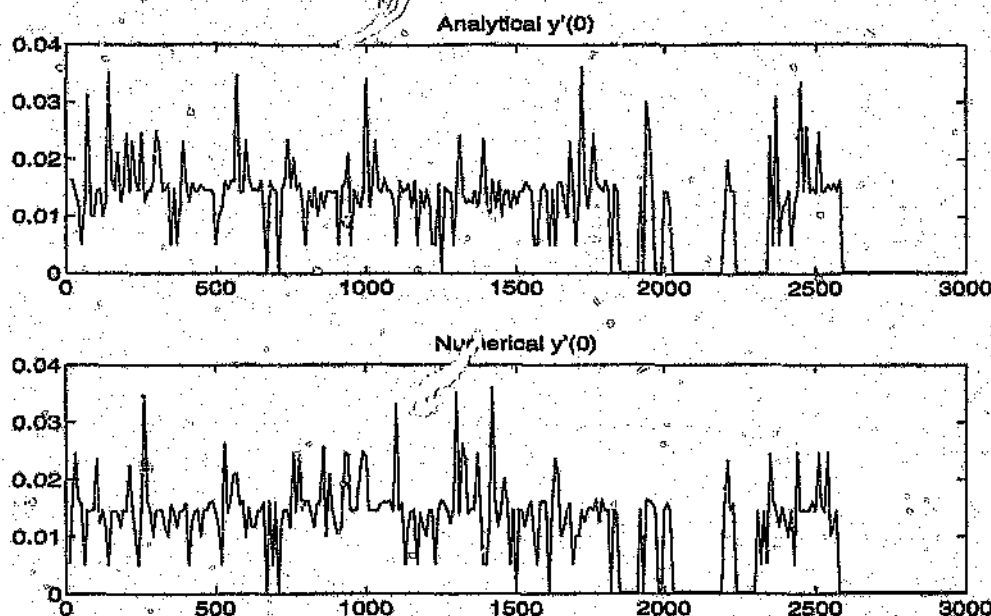


Figure 4.3: LSODI solution to two-transistor amplifier:  $t_{lim}$  vs. no. of calls.

as many for DASSL as for RADAU5).

The work-precision diagrams of Figure 4.4 give a more general code performance comparison. Precision was taken as the maximum (over all components) of the maximum/mean global absolute error, using the same truth model as before and analytical Jacobian and  $y'(0)$ .  $TOL = 10^{-m}$ ,  $m = 2, \dots, 9$  (including half-integral values) was used, with integral values of  $m$  marked.

- TOL was limited to  $m \geq 5$  for RADAU5 due to an exponential overflow error (as warned in [47]), and to  $m \leq 6$  for DASSL due to repeated ETF's.
- The work-precision graphs for RADAU5 are fairly linear and have well-spaced points, so that TOL is a good control of work or precision, but this is less so for DASSL.
- RADAU5 showed a higher effective order (5, 5, 6 and 6) than did DASSL (4, 4, 4 and 5), using linear regression on all data.
- As for the single  $TOL = 10^{-5}$  case, RADAU5 obtained higher accuracy for less work than DASSL over the TOL range used, requiring roughly half an order of magnitude less work for the same precision.
- The above comments hold for both measures of both work and precision.

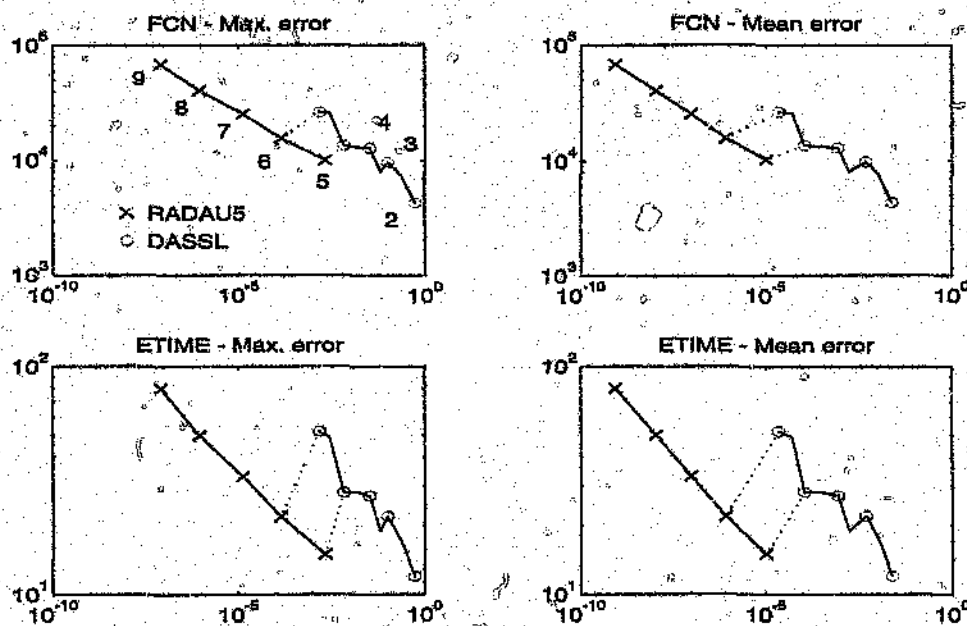


Figure 4.4: Work-precision diagrams for two-transistor amplifier.

## 4.4 Two-Phase Plug Flow

### Formulation

A pipeline problem, described in [20], modelling the pressure  $P(x)$  and the thickness of the annular phase  $y(x)$  in an oil pipeline, is to be solved on  $x \in [0, L]$ .

System I:

$$\pi \sqrt{\frac{R}{2\rho}} (R-y)^2 \sqrt{-P'} \left( 2.5 \ln \left[ \sqrt{\frac{\rho R}{2}} \frac{y}{\mu} \sqrt{-P'} - 5 \right] + 10.5 \right) - bQ_{co} - \frac{P_0}{P} Q_{co}(1-b) = 0, \quad (4.1)$$

$$2\pi \sqrt{\frac{R}{2\rho}} \sqrt{-P'} \left( (2.5Ry - 1.25y^2) \ln \left[ \sqrt{\frac{\rho R}{2}} \frac{y}{\mu} \sqrt{-P'} - 5 \right] + 3Ry - 2.125y^2 - 13.6R\mu \sqrt{\frac{2}{\rho R}} \frac{1}{\sqrt{-P'}} \right) - Q_a = 0. \quad (4.2)$$

By introducing  $u = \sqrt{-P'}$ , this implicit system becomes the semi-explicit System II:

$$P' = -u^2 \quad (4.3)$$

$$0 = f(u, y, P) \quad (4.4)$$

$$0 = g(u, y). \quad (4.5)$$

On differentiating the constraints once,

$$0 = f_u u' + f_y y' - u^2 f_P \quad (4.6)$$

$$0 = g_u u' + g_y y', \quad (4.7)$$

and  $u', y'$  can be found explicitly in terms of  $u, y, P$  (ie.  $\nu = 1$ ) iff the matrix

$$\begin{pmatrix} f_u & f_y \\ g_u & g_y \end{pmatrix}$$

is invertible along the solution, which is the case. Consistent i.c.'s are obtained by  $P(0) = P_0$  and solving (4.4) and (4.5) at  $x = 0$ . There are two qualitatively different cases of interest, resulting in normal and choked flow. A set of parameters is given for each case.

#### Normal flow

$$\begin{aligned} R &= 45.72, & \rho &= 0.814, & \mu &= 0.098, & b &= 0.606, \\ Q_{co} &= 1.1531 \cdot 10^6, & Q_a &= 2.035 \cdot 10^5, & P_0 &= 1.457 \cdot 10^8, & L &= 8.047 \cdot 10^6, \\ u(0) &= 1.651469869370661357709407, & y(0) &= 4.877400707738074248312839. \end{aligned}$$

#### Choked flow

$$\begin{aligned} R &= 45.72, & \rho &= 0.814, & \mu &= 0.098, & b &= 0.345, \\ Q_{co} &= 1.7153 \cdot 10^6, & Q_a &= 3.027 \cdot 10^5, & P_0 &= 1.378 \cdot 10^8, & L &= 3.2188 \cdot 10^7, \\ u(0) &= 2.8665271547403188758740152, & y(0) &= 4.8146652453167371525906527. \end{aligned}$$

#### Solution

All three solvers are suitable for this index 1 problem. The original implicit form I is appropriate for DASSL, but the semi-explicit form II is required for RADAU5 and LSODI, for which MASS is banded while JAC is full. We initially requested accuracy to within  $TOL = 10^{-6}$ . The initial derivative is unspecified but can be found for System II from (4.6), (4.7), and by differentiating (4.2) for System I. For LSODI, an approximate 'true'  $y'(0)$  was estimated using the RADAU5 solution after one step, while this 'true' value was given to DASSL as inconsistent. Refer to Figure 4.5 and Tables 4.6–4.8 for results.

- The results of interest, ie.  $P(x)$  and  $y(x)$  for normal flow and the choking point for choked flow, (in bold) agree globally to more than the 5 significant digits expected. The choking behaviour, with  $P$  and  $y$  decreasing rapidly to zero at a choking point, was observed in all cases for 'choked flow'.
- The inclusion of the scaling factor  $h_0 = 10^5$  saved RADAU5 much effort for normal flow, cost DASSL in effort, and otherwise had no great effect.
- For choked flow, repeated error test failures halted all three codes near the choking point at  $x \approx 1.0958 \cdot 10^7$ , agreeing with the literature [20, 47].

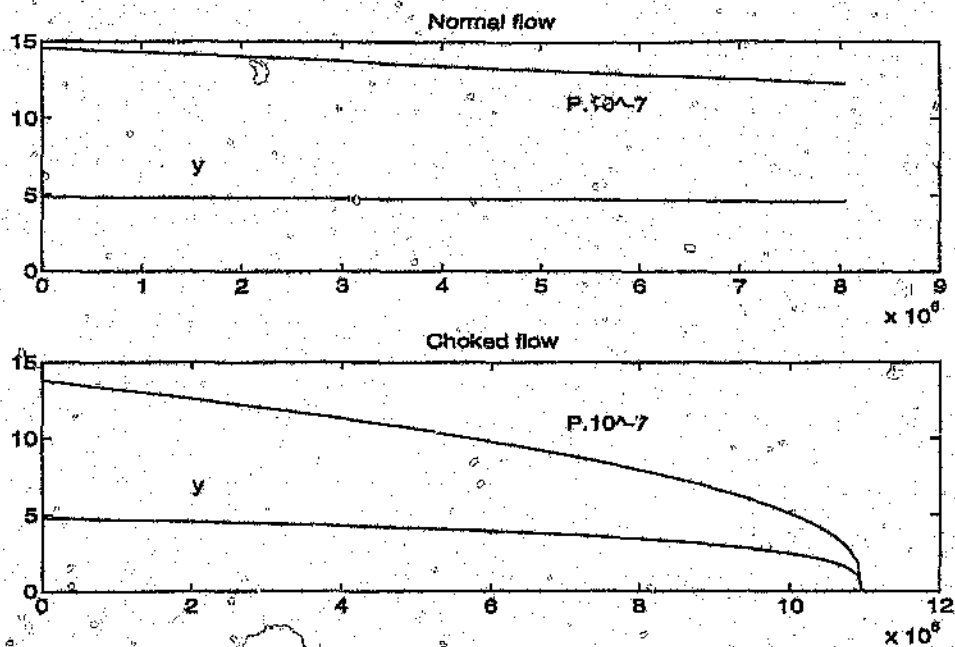


Figure 4.5: Solution to two-phase plug flow problem

| Flow   | Sys. | $h_0$  | FCN  | JAC | STEPS | ETF's | $x$                   | $P(x) \cdot 10^{-8}$ | $y(x)$    |
|--------|------|--------|------|-----|-------|-------|-----------------------|----------------------|-----------|
| normal | I    | —      | 67   | 13  | 47    | 0     | $L$                   | 1.2251804            | 4.5877494 |
|        |      | $10^5$ | 91   | 18  | 55    | 0     | $L$                   | 1.2251811            | 4.5877523 |
|        | II   | —      | 44   | 18  | 30    | 0     | $L$                   | 1.2251800            | 4.5877492 |
|        |      | $10^5$ | 53   | 17  | 31    | 0     | $L$                   | 1.2251806            | 4.5877500 |
| choked | I    | —      | 1331 | 112 | 748   | 83    | $1.095787 \cdot 10^7$ |                      |           |
|        |      | $10^5$ | 1407 | 115 | 748   | 89    | $1.095785 \cdot 10^7$ |                      |           |
|        | II   | —      | 941  | 67  | 449   | 63    | $1.095792 \cdot 10^7$ |                      |           |
|        |      | $10^5$ | 972  | 70  | 469   | 64    | $1.095792 \cdot 10^7$ |                      |           |

Table 4.6: DASSL solution to two-phase plug flow Systems I and II

| Flow   | $h_0$  | $y'(0)$ | FCN | JAC | STEPS | $x$                    | $P(x) \cdot 10^{-8}$ | $y(x)$    |
|--------|--------|---------|-----|-----|-------|------------------------|----------------------|-----------|
| normal | —      | true    | 47  | 5   | 23    | $L$                    | 1.2251791            | 4.5877468 |
|        | —      | 0       | 79  | 10  | 27    | $L$                    | 1.2251764            | 4.5877429 |
|        | $10^5$ | true    | 53  | 6   | 21    | $L$                    | 1.2251755            | 4.5877381 |
| choked | —      | true    |     |     |       | $1.0957855 \cdot 10^7$ |                      |           |
|        | —      | 0       |     |     |       | $1.0957813 \cdot 10^7$ |                      |           |
|        | $10^5$ | true    |     |     |       | $1.0957854 \cdot 10^7$ |                      |           |

Table 4.7: LSODI solution to two-phase plug flow System II

| Flow   | $h_0$  | FCN  | JAC | STEPS | ETF's | $x$                    | $P(x) \cdot 10^{-8}$ | $y(x)$               |
|--------|--------|------|-----|-------|-------|------------------------|----------------------|----------------------|
| normal | —      | 80   | 3   | 17    | 0     | $L$                    | 1.2251828            | 4.5877528            |
|        | $10^5$ | 32   | 2   | 5     | 0     | $L$                    | 1.2251829            | 4.5877528            |
| choked | —      | 2348 | 327 | 341   | 0     | $1.0958155 \cdot 10^7$ | $1.5 \cdot 10^{-8}$  | $9.24 \cdot 10^{-3}$ |
|        | $10^5$ | 2300 | 327 | 329   | 0     | $1.0958155 \cdot 10^7$ | $1.5 \cdot 10^{-8}$  | $9.21 \cdot 10^{-3}$ |

Table 4.8: RADAU5 solution of two-phase plug flow System II

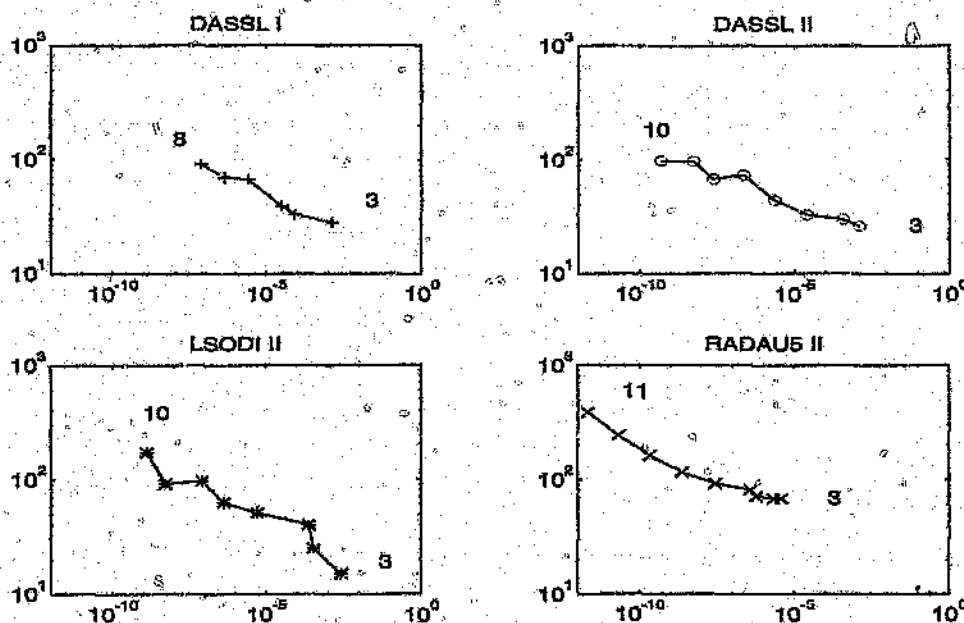


Figure 4.6: Work-precision diagrams for two-phase plug flow problem.

- Inconsistent  $y'(0)$  for LSODI cost  $\approx 70\%$  more evaluations but did not otherwise affect integration (within TOL).
- Figure 4.5 and Table 4.4 agree with Figures 9.1 and 9.2 (and comments) of [47] for the RADAU5 solution of System II.
- The successful solution of System I by DASSL and II by LSODI agrees with [20], in which no direct comparison is made.

Figure 4.6 shows work-precision diagrams for normal flow using FCN for work, maximum relative error (over all components and all 100 points on  $[0, L]$ ) with a RADAU5 TOL =  $10^{-15}$  truth model for precision, default parameters (esp.  $h_0$ ), a RADAU5  $y'(0)$  estimate and TOL =  $10^{-m}$ ,  $m = 3, 4, \dots$

- For DASSL and LSODI, accuracy was limited by either repeated ETF's at  $x = 0$  or an inability to calculate  $y'(0)$ .

- Effective orders (linear regression on  $m = 3, \dots, 8$ ) are approximately 8 for DASSL on System I, and 10 for DASSL, 7 for LSODI and 13 for RADAU5 on System II.
- Again, RADAU5 has a regular work-precision graph, DASSL less so, and LSODI even worse. For all three codes, obtained accuracy is within an order of magnitude of TOL, and work done is comparable at  $10^{-6}$  accuracy.

## 4.5 Discharge Pressure Control

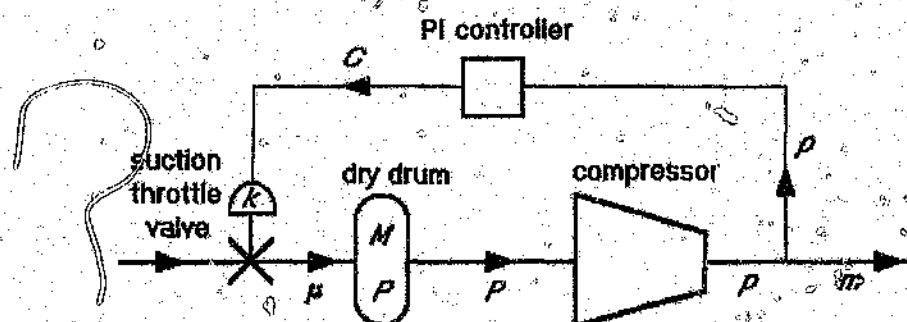


Figure 4.7: Schematic for discharge pressure control problem

### Formulation

System I below models a simple dynamic simulation problem in petrochemical engineering [47]. It consists of a compressor with a suction-throttle valve to control discharge pressure. Of interest is a comparison between the outlet and inlet mass flows,  $m(t)$  and  $\mu(t)$  (see Figure 4.7). We study this problem to show the failure of index 1 integrators on higher-index DAEs. System I:

$$k' = (c - k)/20 \quad (4.8)$$

$$(c + p/15)' = -(p - 99.1)/75 \quad (4.9)$$

$$M' = \mu - m \quad (4.10)$$

$$p/P = 3.35 - 0.075m + 0.001m^2 \quad (4.11)$$

$$P^2 = 49.58^2 - (\mu/1.2k)^2 \quad (4.12)$$

$$M = 20P \quad (4.13)$$

$$\mu = 15 + 5 \tanh(t - 10). \quad (4.14)$$

To determine the index, we first transform to a semi-explicit form:

Define  $s = c + p/15$ .

Replace (4.9) with  $s' = -(p - 99.1)/75$ .

Substitute  $\mu = \mu(t)$  from (4.14) into (4.12) and (4.10).

Substitute  $P = M/20$  from (4.13) into (4.12) and (4.11).

Substitute  $c = s - p/15$  from above into (4.8).

The problem now becomes System II:

$$k' = (s - p/15 - k)/20 \quad (4.15)$$

$$s' = -(p - 99.1)/75 \quad (4.16)$$

$$M' = \mu(t) - m \quad (4.17)$$

$$0 = 20p/M - (3.35 - 0.075m + 0.001m^2) \quad (4.18)$$

$$0 = (M/20)^2 - 49.58^2 + (\mu(t)/1.2k)^2 \quad (4.19)$$

The index is the number of times this must be differentiated to obtain  $(k', s', M', p', m')$  explicitly in terms of  $(k, s, M, p, m)$ . We already have  $k', s', M'$ .

$$\text{Diff. (4.18)} \Rightarrow p' = p'(m, M, p, M', m') \quad (4.20)$$

$$\text{Subst. (4.17)} \Rightarrow = p'(p, m, M, m')$$

$$\text{Diff. (4.19)} \Rightarrow 0 = f_1(k, M, k', M') \quad (4.21)$$

$$\text{Subst. (4.15), (4.17)} \Rightarrow = f_2(k, s, M, p, m)$$

$$\text{Diff. (4.21)} \Rightarrow 0 = f_3(k, s, M, p, m, k', s', M', p', m') \quad (4.22)$$

$$\text{Subst. (4.15) - (4.20)} \Rightarrow = f_4(k, s, M, p, m, m')$$

$$\text{Implicit function theorem} \Rightarrow m' = m'(k, s, M, p, m) \quad (4.23)$$

$$\text{Subst. into (4.20)} \Rightarrow p' = p'(k, s, M, p, m) \quad (4.24)$$

We had to differentiate (4.19) twice, so  $\nu = 2$ .

The above procedure gives us a set of c.i.c.'s for System I. We are free to choose  $k(0)$  and  $p(0)$ , and the above yields  $\mu(0), P(0), M(0), m(0), c(0)$ . (An alternative formulation has  $m$  specified and  $\mu$  unknown. Using a similar procedure to the one above, it is easy to show that this problem has  $\nu = 1$ . Also, we are then free to choose  $c(0)$ . Suppose physical considerations demand  $k'(0) = 0$ . Then (4.8) yields  $c(0) = k(0)$ .)

As input to our DAE solvers for System I, we thus have:

$$Ay' = f(y), \quad y = (k, c, p, M, P, m, \mu)^T,$$

$$A = \begin{pmatrix} 1 & . & . & . & . & . & . \\ . & 1 & \frac{1}{15} & . & . & . & . \\ . & . & . & 1 & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{pmatrix}, \quad f = \begin{pmatrix} (c - k)/20 \\ -(p - 99.1)/75 \\ \mu - m \\ 3.35 - 0.075m + 0.001m^2 - p/P \\ (49.58)^2 - [\mu/(1.2k)]^2 - p^2 \\ 20P - M \\ 15 + 5 \tanh(t - 10) - \mu \end{pmatrix}$$

$$\text{and } y(0) = \begin{pmatrix} 0.25 \\ 0.2500145529515559 \\ 99.1 \\ 734.0477598381585 \\ 36.70238799190793 \\ 9.998240239254898 \\ 15 + 5 \tanh(-10) \end{pmatrix}$$

Note that:

1.  $y'(0)$  is not specified, but can be found from the above manipulation.
2. The mass matrix  $A$  is banded, but the Jacobian is full.
3. We use System I, derived directly from the model, as it requires no manipulation.
4. Manipulation was, however, required to find the c.i.c.

### Solution

#### 1. DASSL, LSODI

Without high-index adjustment, both DASSL and LSODI halted execution due to repeated ETF's (for  $TOL > 10^{-3}$ ) or CTF's (for  $TOL < 10^{-3}$ ) at  $t \approx 6$ . This behaviour is expected for DASSL and LSODI, when attempting to solve a higher-index problem.

#### 2. RADAU5

RADAU5 successfully solved the problem, for both Systems I and II, on  $t \in [0, 40]$  (see Fig-

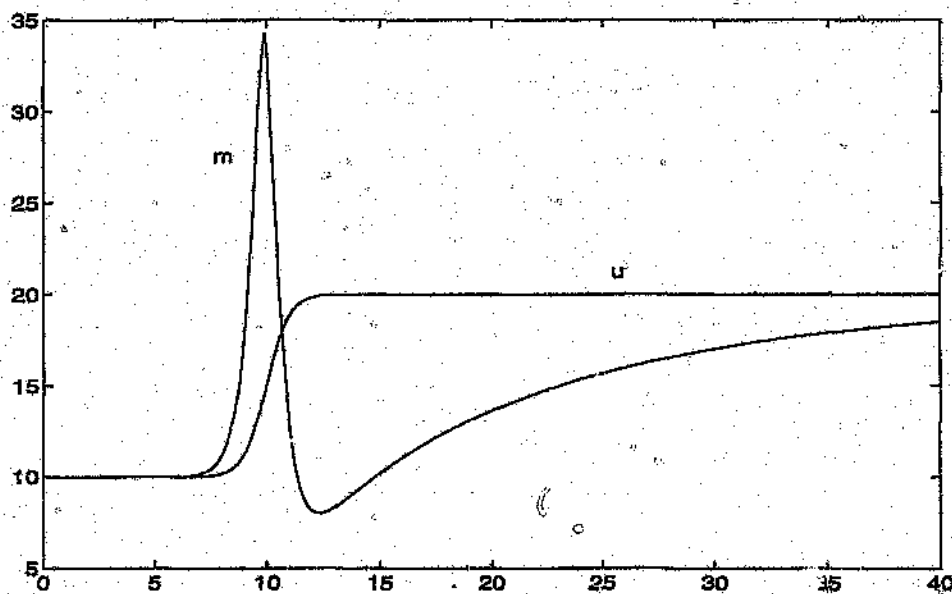


Figure 4.8: RADAU5 solution to discharge pressure control: Outlet mass flow  $m(t)$  and inlet mass flow  $u(t)$ .

ure 4.8). Although neither formulation is in Hessenberg form, the high-index scaling of §3.2.3 is applicable to System II. Scaling the algebraic variables as 'index 2' in both systems yielded successful solutions. Curiously, the mass bandwidth was required by RADAU5 to

be no less than the Jacobian bandwidth, so that A was stored in full. Table 4.9 shows the solution statistics for  $TOL = 10^{-4}$ . The 'Error in  $m$ ' is the maximum relative error in the index 2 component of interest,  $m$ , on 1000 points on  $[0, 40]$ .

| System | Jacobian  | FCN | JAG | STEPS | ETIME (no) | Error in $m$        |
|--------|-----------|-----|-----|-------|------------|---------------------|
| I      | numerical | 493 | 44  | 68    | 0.66s      | $4.6 \cdot 10^{-4}$ |
| II     | numerical | 441 | 40  | 58    | 0.44s      | $3.3 \cdot 10^{-4}$ |

Table 4.9: RADAU5 solution statistics for discharge pressure control

- Figure 4.8 agrees with [47, Figure 9.8] for the solution of System I by RADAU5.
- For  $TOL = 10^{-4}$ , less work was done and more accuracy obtained for the smaller semi-explicit System II than for the original linearly implicit System I.
- ETIME figures for disk output, at  $\Delta t_{ime} = 0.001$  rise to  $\approx 5.22s$  for System I and  $\approx 4.39s$  for System II.

Figure 4.9 shows work-precision diagrams for RADAU5, using FCN for work, the error in  $m$  for accuracy, and the  $TOL = 10^{-12}$  solution to System II as 'true'. A range of  $TOL = 10^{-m}$ ,  $m = 0, \dots, 7$  for System I and  $m = 2, \dots, 9$  for System II was used.

- Accuracy for System I was limited by singular iteration failure.
- Solutions are globally accurate to within an order of magnitude of  $TOL$ .
- Solutions to both systems have effective order  $\approx 6$  and roughly equal work/precision ratios, requiring  $O(10^3)$  FCN evaluations for a minimum global relative accuracy of  $10^5$ .

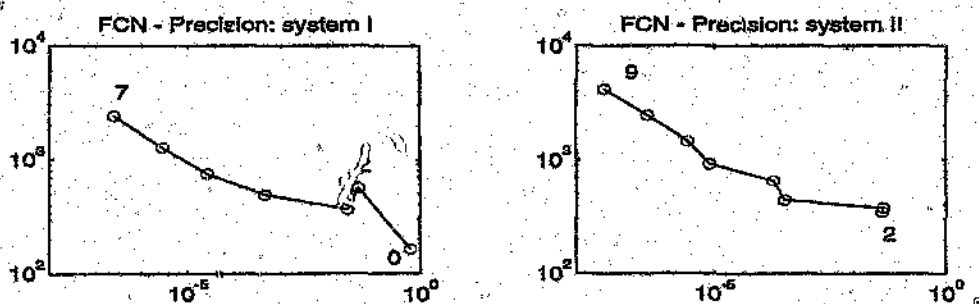


Figure 4.9: RADAU5 work-precision diagrams for discharge pressure control

## 4.6 Plane Pendulum

### Formulation

The Newton-Euler equations (cf. p.8) modelling a simple plane pendulum, after normalization, are given by System 1 below. Here,  $(x, y)$  is the position of the bob,  $(u, v)$  is its velocity, and  $T$  is the tension in the massless rod. We attempt to solve the problem in the six formulations discussed in Example 1.1.1: the original, naturally-posed index 3 System 1, three index-reduced forms, Systems 2, 3 and 4, whose constraints are obtained by successive differentiation of the constraint of System 1, and two index 2 reformulations with constraint stabilization, the GGL formulation System 5 and the F&L formulation System 6.

#### System 1 ( $\nu = 3$ )

$$\begin{aligned}x' &= u \\y' &= v \\u' &= -Tx \\v' &= -Ty - 1 \\0 &= x^2 + y^2 - 1 \\y &= \overbrace{(x, y, u, v, T)}^{x_1, x_2, x_3}{}^T \\y(0) &= (1, 0, 0, 0, 0)^T \\y'(0) &= (0, 0, 0, -1, 0)^T\end{aligned}$$

#### System 2 ( $\nu = 2$ )

$$\begin{aligned}x' &= u \\y' &= v \\u' &= -Tx \\v' &= -Ty - 1 \\0 &= xu + yv \\y &= \overbrace{(x, y, u, v)}^{x_1, x_2}{}^T \\y(0) &= (1, 0, 0, 0, 0)^T \\y'(0) &= (0, 0, 0, -1, 0)^T\end{aligned}$$

#### System 3 ( $\nu = 1$ )

$$\begin{aligned}x' &= u \\y' &= v \\u' &= -Tx \\v' &= -Ty - 1 \\0 &= u^2 + v^2 - T - y \\y &= \overbrace{(x, y, u, v, T)}^{x_1, x_2, x_3}{}^T \\y(0) &= (1, 0, 0, 0, 0)^T \\y'(0) &= (0, 0, 0, -1, 0)^T\end{aligned}$$

#### System 4 ( $\nu = 0$ )

$$\begin{aligned}x' &= u \\y' &= v \\u' &= -Tx \\v' &= -Ty - 1 \\T' &= -3v \\y &= \overbrace{(x, y, u, v, T)}^{x_1, x_2, x_3}{}^T \\y(0) &= (1, 0, 0, 0, 0)^T \\y'(0) &= (0, 0, 0, -1, 0)^T\end{aligned}$$

#### System 5 ( $\nu = 2$ )

$$\begin{aligned}x' &= u - x\lambda \\y' &= v - y\lambda \\u' &= -Tx \\v' &= -Ty - 1 \\0 &= x^2 + y^2 - 1 \\0 &= xu + yv \\y &= \overbrace{(x, y, u, v, T, \lambda)}^{x_1, x_2}{}^T \\y(0) &= (1, 0, 0, 0, 0, 0)^T \\y'(0) &= (0, 0, 0, -1, 0, 0)^T\end{aligned}$$

#### System 6 ( $\nu = 2$ )

$$\begin{aligned}x' &= u + 2x\lambda_1 + u\lambda_2 \\y' &= v + 2y\lambda_1 + v\lambda_2 - \lambda_3 \\u' &= -Tx + x\lambda_2 + 2u\lambda_3 \\v' &= -Ty - 1 + y\lambda_2 + 2v\lambda_3 \\T' &= -3v - \lambda_3 \\0 &= xu + yv \\0 &= u^2 + v^2 - T - y \\0 &= x^2 + y^2 - 1 \\y &= \overbrace{(x, y, u, v, T, \lambda_1, \lambda_2, \lambda_3)}^{x_1, x_2}{}^T \\y(0) &= (1, 0, 0, 0, 0, 0, 0, 0)^T \\y'(0) &= (0, 0, 0, -1, 0, 0, 0, 0)^T\end{aligned}$$

In each case, the index can be verified either by differentiating the constraints and substituting, or by noting that the higher-index systems are in Hessenberg form (cf. Definition 2.3.1). The Hessenberg conditions are easily found to hold, using the algebraic

constraints. Consistent i.c.'s are obtained from physical considerations, with the pendulum starting at rest in a horizontal position. We require solution on  $t \in [0, 100]$  and request accuracy to within  $TOL = 10^{-4}$ .

### Solution

Due to the semi-explicit form of all Systems 1-6, banded matrix storage was used, for all three codes.

Without high-index scaling, RADAU5 failed to solve the higher-index Systems 1, 2, 5 and 6, but was successful after appropriate scaling (see Table 4.10). Likewise, DASSL and LSODI solved Systems 3 and 4 without scaling, but failed to solve Systems 1 and 2, due to repeated ETF's. LSODI identified the 'algebraic' variable  $T$  as the trouble component. Matrix scaling, ie. scaling the residual of the algebraic equation, for Systems 1 and 2, by  $1/h$  for DASSL and  $1/100$  for LSODI, proved unsuccessful due to corrector convergence failure.

Suitable variable scaling yielded successful integration, although not always satisfactorily accurate solutions. Some successful integration statistics are given in Table 4.11. In each case, a looser unscaled TOL (by a factor of 10) or a looser scaled TOL (by a factor of  $h$ ) resulted in failure.  $h_0$  was set at 0.1 for scaling, although it was left to the code to choose  $h_0$  for integration. This value was found to be more successful than  $h_0 = 1$  or 0.01.

Graphically Figure 4.10 shows the solution over one period. Figures 4.11-4.13 show

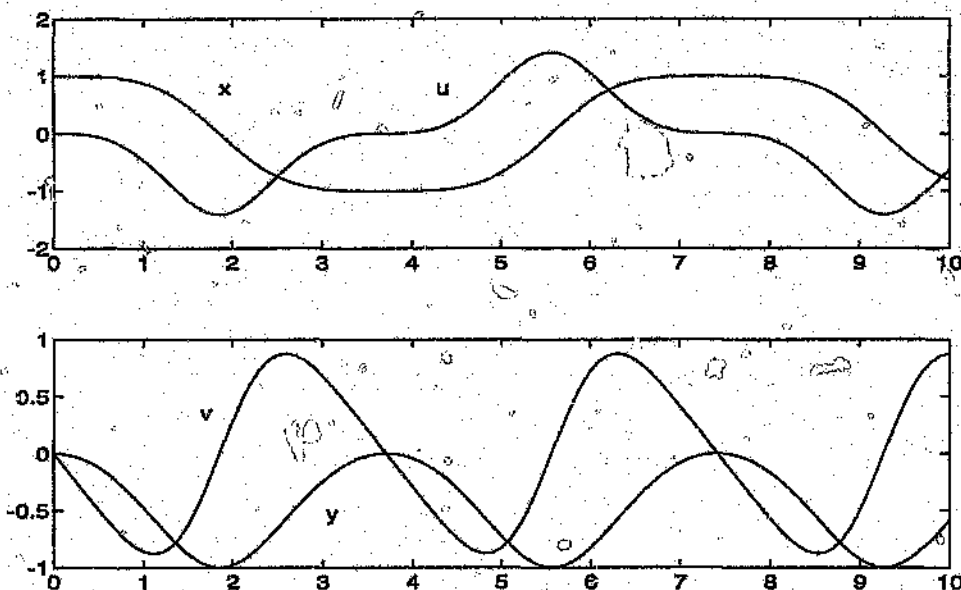


Figure 4.10: Solution to plane pendulum on  $t \in [0, 10]$

the state-space and phase-space solution, and constraint satisfaction on  $t \in [0, 100]$  for the original index 3 System 1, and Figure 4.14 shows the RADAU5 solution to the ODE System 4. Constraints 1, 2 and 3 refer to the right hand sides of the algebraic equation in

Systems 1, 2, and 3, ie.  $x^2 + y^2 = 1$  and its first two derivatives, respectively, and should be zero for all systems for all time. In Table 4.13, "Error" is the order of the maximum absolute value of constraints 1, 2 and 3 on  $[0, 100]$ , "+" signifies a continued growth in error and "Quality" is an accuracy evaluation based on the appearance of the state-space and phase-plane plots.

| System | $\nu$ | STEPS | FCN   | JAC | ETF's |
|--------|-------|-------|-------|-----|-------|
| 1      | 3     | 1009  | 10699 | 853 | 12    |
| 2      | 2     | 1156  | 11594 | 863 | 2     |
| 3      | 1     | 913   | 8179  | 675 | 2     |
| 4      | 0     | 511   | 3775  | 467 | 27    |
| 5      | 2     | 1054  | 11201 | 761 | 0     |
| 6      | 2     | 1190  | 13614 | 829 | 0     |

Table 4.10: RADAU5 solution statistics for plane pendulum: Systems 1-6 for  $TOL = 10^{-4}$ .

| System | $\nu$ | Code  | $TOL(x, y)$ | $TOL(u, v)$   | $TOL(T)$      | STEPS | FCN  | JAC |
|--------|-------|-------|-------------|---------------|---------------|-------|------|-----|
| 1      | 3     | DASSL | $10^{-6}$   | $10^{-6}/h$   | $10^{-6}/h^2$ | 4344  | 7985 | 543 |
|        |       | LSODI | $10^{-4}$   | $10^{-4}/h^3$ | $10^{-4}/h^6$ | 1193  | 5226 | 351 |
| 2      | 2     | DASSL | $10^{-4}$   | $10^{-4}$     | $10^{-4}/h$   | 2978  | 5686 | 266 |
|        |       | LSODI | $10^{-4}$   | $10^{-4}$     | $10^{-4}/h^6$ | 1812  | 5953 | 399 |
| 3      | 1     | DASSL | $10^{-4}$   | $10^{-4}$     | $10^{-4}$     | 1714  | 2589 | 90  |
|        |       | LSODI | $10^{-5}$   | $10^{-5}$     | $10^{-5}$     | 2817  | 6906 | 430 |
|        |       | LSODI | $10^{-4}$   | $10^{-4}$     | $10^{-4}/h$   | 1339  | 2821 | 148 |
| 4      | 0     | DASSL | $10^{-4}$   | $10^{-4}$     | $10^{-4}$     | 1266  | 1275 | 9   |
|        |       | LSODI | $10^{-4}$   | $10^{-4}$     | $10^{-4}$     | 946   | 1551 | 71  |

Table 4.11: DASSL and LSODI solution statistics for plane pendulum: Systems 1-4.

Work-precision diagrams, for RADAU5 on Systems 1-6 and DASSL and LSODI on the index 1 System 3, are displayed in Figure 4.15. FCN was used for work, and the maximum absolute value of constraints 1, 2 and 3 on all 1000 points on  $[0, 100]$  for precision. Safe integration parameters and a range of  $TOL = 10^{-m}$ ,  $m = 0, \dots, 9$  were used ( $m = 2, 3, 4$  failed for LSODI on System 3 due to failure). Effective orders, using linear regression on  $m = 5, \dots, 9$ , are given in Table 4.12.

The following can be deduced from the figures and tables.

- With regard to constraint satisfaction (using RADAU5 solutions only):
  - Solutions to System 1 show constraint growth (and the resulting drift from  $TOL$  at higher precision), lower effective order and higher work/precision for constraints 2 and 3 (more for 3);

| Code   | System | Effective order |     |     |
|--------|--------|-----------------|-----|-----|
|        |        | 1               | 2   | 3   |
| RADAU5 | 1      | 5.3             | 3.7 | 2.4 |
|        | 2      | 4.8             | 5.0 | 3.6 |
|        | 3      | 5.1             | 5.1 | 3.6 |
|        | 4      | 5.1             | 5.1 | 5.1 |
|        | 5      | 5.6             | 4.7 | 4.7 |
|        | 6      | 5.5             | 5.4 | 4.2 |
| DASSL  | 3      | 5.4             | 5.3 | 7.4 |
| LSODI  | 3      | 5.8             | 5.7 | 7.0 |

Table 4.12: Effective order estimates for plane pendulum

- Solutions to System 2 show constraint growth and higher work/precision for constraints 1 and 3 (more for 3) and loss of effective order for 3;
- Solutions to System 3 show constraint growth and higher work/precision for constraints 1 and 2, and constraint 3 loses effective order;
- Solutions to System 4 show constraint growth and higher work/precision for all constraints 1, 2 and 3; and
- For the index 2 reformulations, System 5 (GGL) clearly stabilized constraint 1 (but not 3) and System 6 (F&L) clearly stabilized both constraints 1 and 3, regaining both effective order and work/precision for stabilized constraints.

All this confirms the theory.

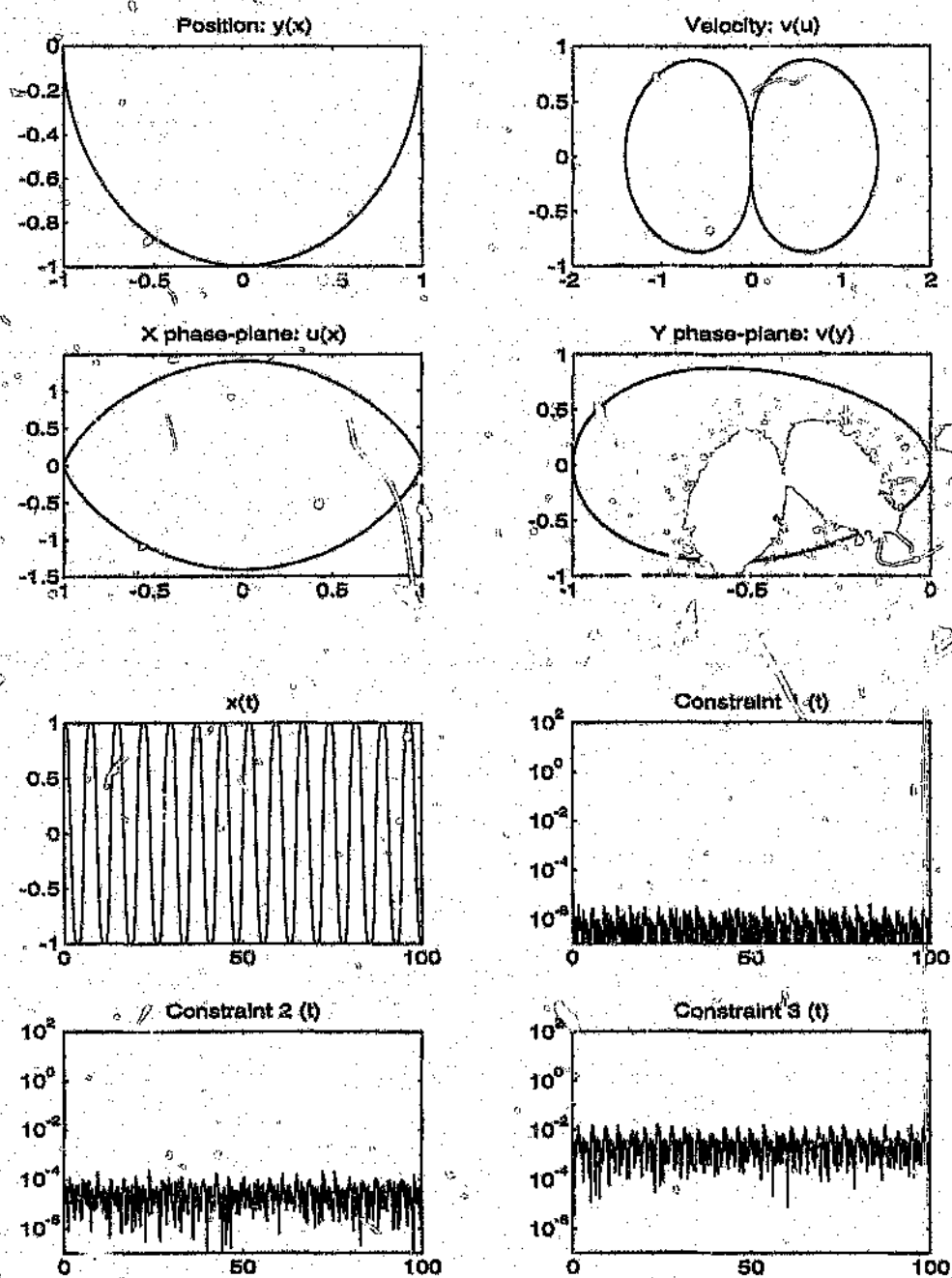
- Index-reduced Systems 3 and 4 shifted work/precision curves to the right, with precision falling orders of magnitude short of TOL. They are thus dangerous, due to the false confidence they inspire.
- In some cases, some components were unaffected by inaccuracy and instability in others (see Figures 4.12 and 4.13). This is due to two effects:  $(x, u)$  and  $(y, v)$  are mutually independent in the differential part, and higher-index instability does not affect lower index components, as mentioned in the theory and found in the literature.
- Comparing code performance for the index 1 System 3, we observe:
  - In decreasing order of precision/work (at  $10^{-5}$  accuracy): DASSL, then LSODI, then RADAU5.
  - In decreasing order of effective order: LSODI, then DASSL, then RADAU5.
  - DASSL and LSODI (RADAU5) have higher (lower) effective order at higher precision.
  - It appears that DASSL performed best, then RADAU5 and LSODI over this precision range for this problem.

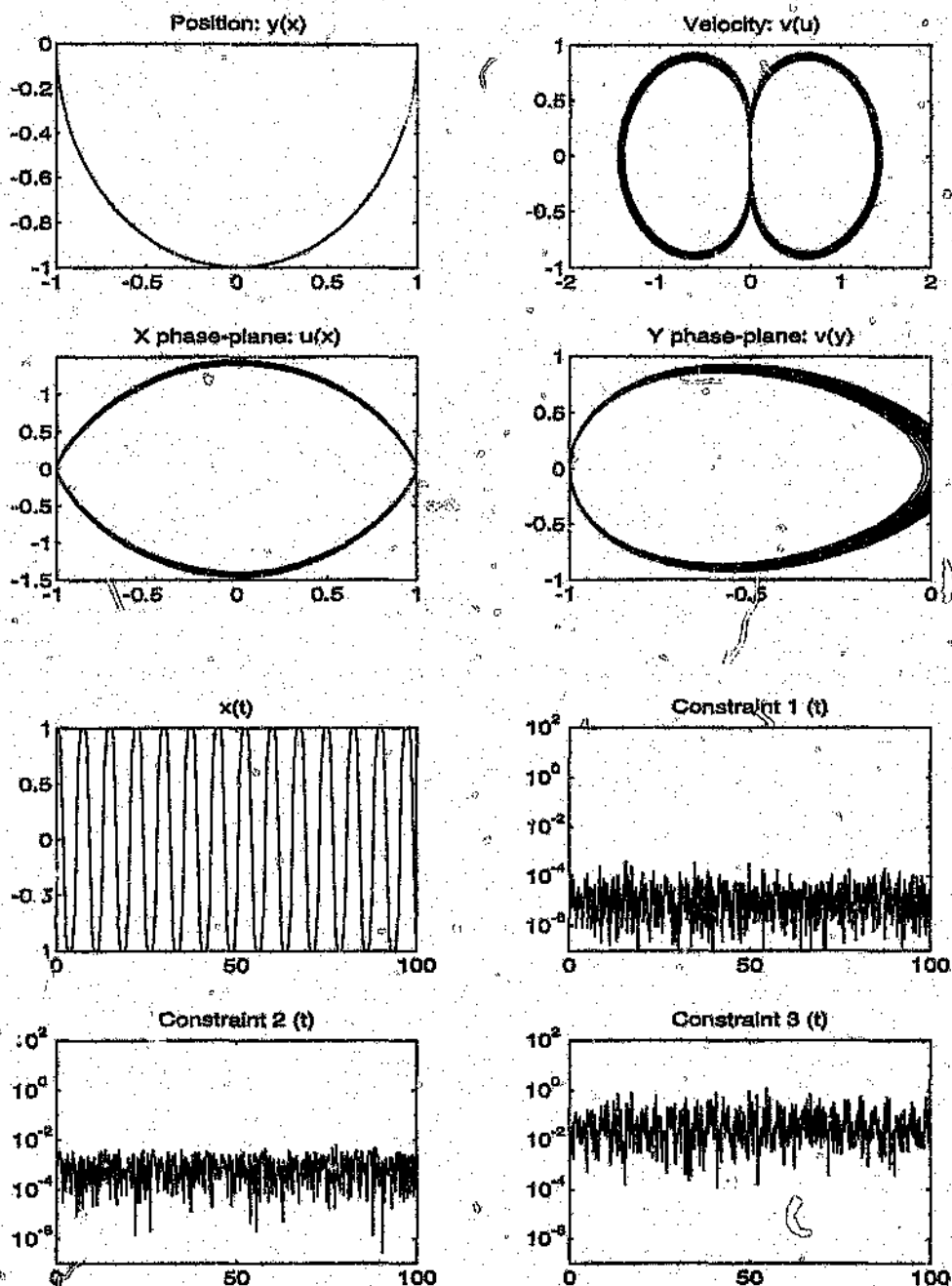
| System | $\nu$ | Code   | Error       |             |             | Quality |
|--------|-------|--------|-------------|-------------|-------------|---------|
|        |       |        | 1           | 2           | 3           |         |
| 1      | 3     | RADAU5 | $10^{-6}$   | $10^{-4}$   | $10^{-2}$   | ✓✓      |
|        |       | DASSL  | $10^{-4}$   | $10^{-3}$   | $10^{-0}$   | ✓       |
|        |       | LSODI  | $10^{-3}$   | $10^{-1}$   | $10^{+1}$   | x       |
| 2      | 2     | RADAU5 | $10^{-4}$ + | $10^{-5}$   | $10^{-3}$   | ✓✓      |
|        |       | DASSL  | $10^{-2}$ + | $10^{-4}$   | $10^{-2}$   | ✓       |
|        |       | LSODI  | $10^{-3}$ + | $10^{-3}$   | $10^{-1}$   | ✓       |
| 3      | 1     | RADAU5 | $10^{+2}$ + | $10^{+2}$ + | $10^{-4}$   | x       |
|        |       | DASSL  | $10^{-2}$   | $10^{-2}$   | $10^{-3}$   | x       |
|        |       | LSODI  | $10^{-2}$   | $10^{-3}$   | $10^{-4}$   | x       |
|        |       | LSODI  | $10^{-2}$   | $10^{-2}$   | $10^{-2}$   | x       |
| 4      | 0     | RADAU5 | $10^{-1}$ + | $10^{-1}$ + | $10^{-0}$ + | x       |
|        |       | DASSL  | $10^{+0}$ + | $10^{+0}$ + | $10^{+0}$ + | x       |
|        |       | LSODI  | $10^{+4}$ + | $10^{+4}$ + | $10^{+4}$ + | x       |
| 5      | 2     | RADAU5 | $10^{-5}$   | $10^{-5}$   | $10^{-3}$   | ✓✓      |
| 6      | 2     | RADAU5 | $10^{-5}$   | $10^{-5}$   | $10^{-5}$   | ✓✓      |

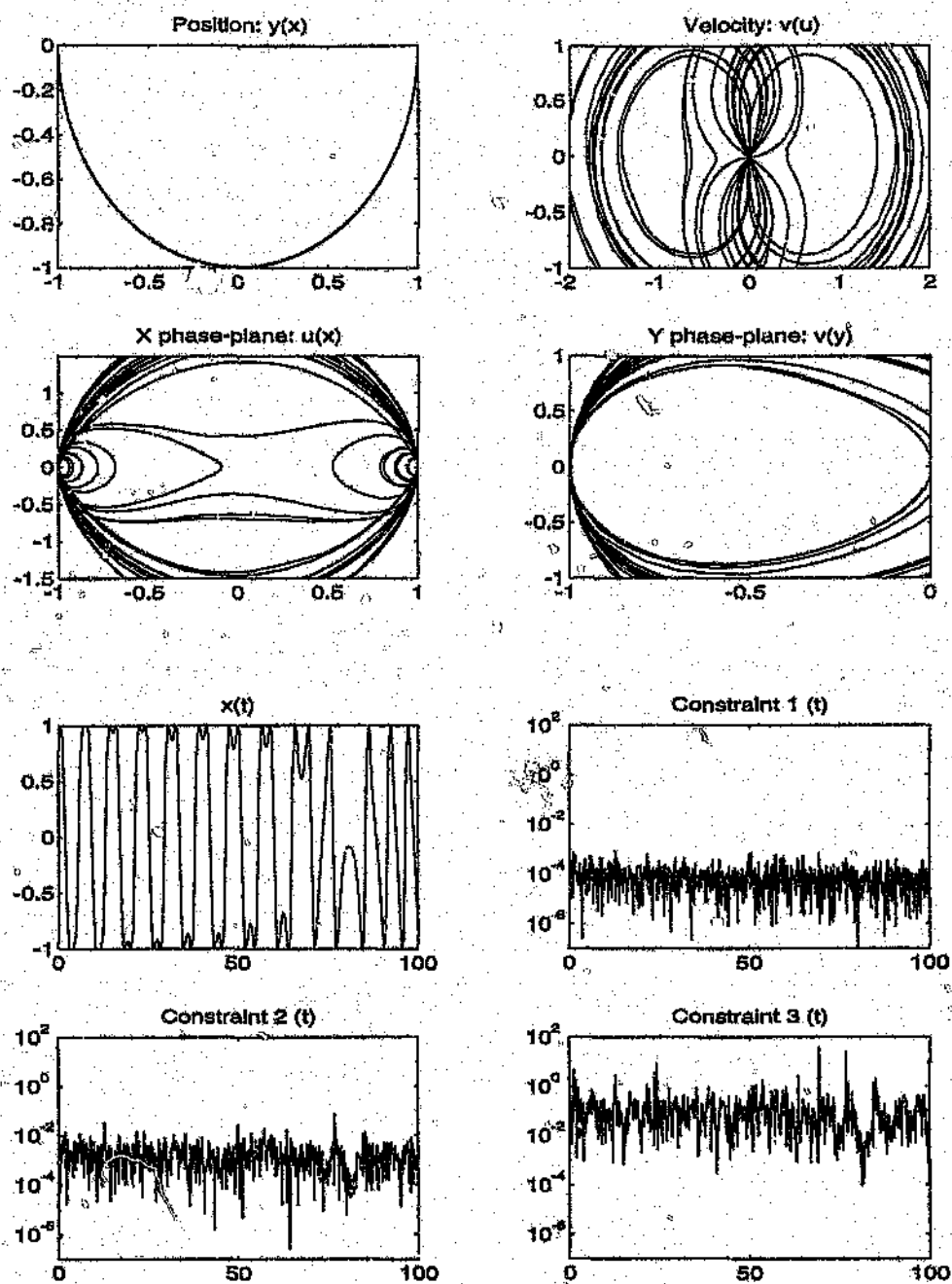
Table 4.13: Summary of results for plane pendulum

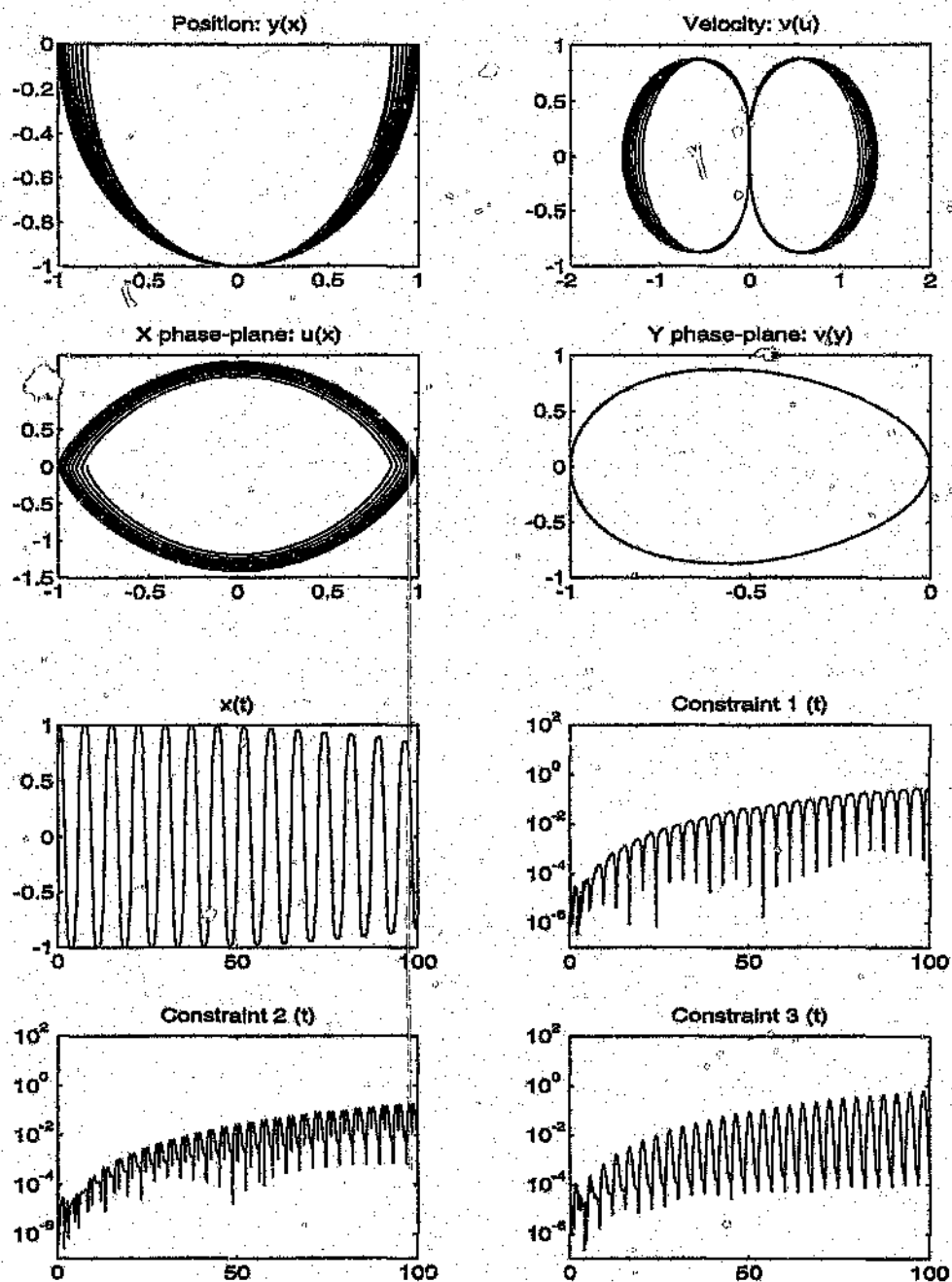
We now mention work done by other authors on this popular example problem.

1. In [47, p. 118], Systems 1, 2, 3 and 5 are solved successfully by RADAU5 on  $[0, 10]$ , and it is found that System 1 requires more work than the others (with which we agree), while System 2 is solved the fastest and System 3 requires less time but obtains less accuracy than the others: (This result depends on the safety parameters used.)
2. In [14, pp.155-157], Systems 3 and 5 are solved by DASSL on  $[0, 1]$ . For System 3, low accuracy is obtained for constraints 1 and 2, whereas constraint 3 keeps the same order of magnitude as TOL. (We agree.) After modifying the code to exclude 'algebraic' variables from the error test (but not convergence test), all three constraints for System 5 are satisfied to within TOL, and do not drift. (It is uncertain whether this would be the case on the extended interval  $[0, 100]$ , on which constraint 3 for our RADAU5 solution does drift.)
3. In [48, p. 539], for a related index 3 Euler-Lagrange system, solutions to Systems 2, 3 and 5 by DASSL and 1, 2, 3 and 5 by RADAU5 are obtained, and it is found that DASSL performs better than RADAU5 on Systems 2 and 3, but behaves irregularly for 5, and fails on System 1 (without scaling). This correlates with our results for the pendulum, except that we found DASSL to fail on Systems 2 and 5.

Figure 4.11: RADAU5 solution to plane pendulum: index 3,  $t \in [0, 100]$

Figure 4.12: DASSL solution to plane pendulum: index 3,  $t \in [0, 100]$

Figure 4.13: LSODI solution to plane pendulum: index 3,  $t \in [0, 100]$

Figure 4.14: RADAU5 solution to plane pendulum ODE,  $t \in [0, 100]$

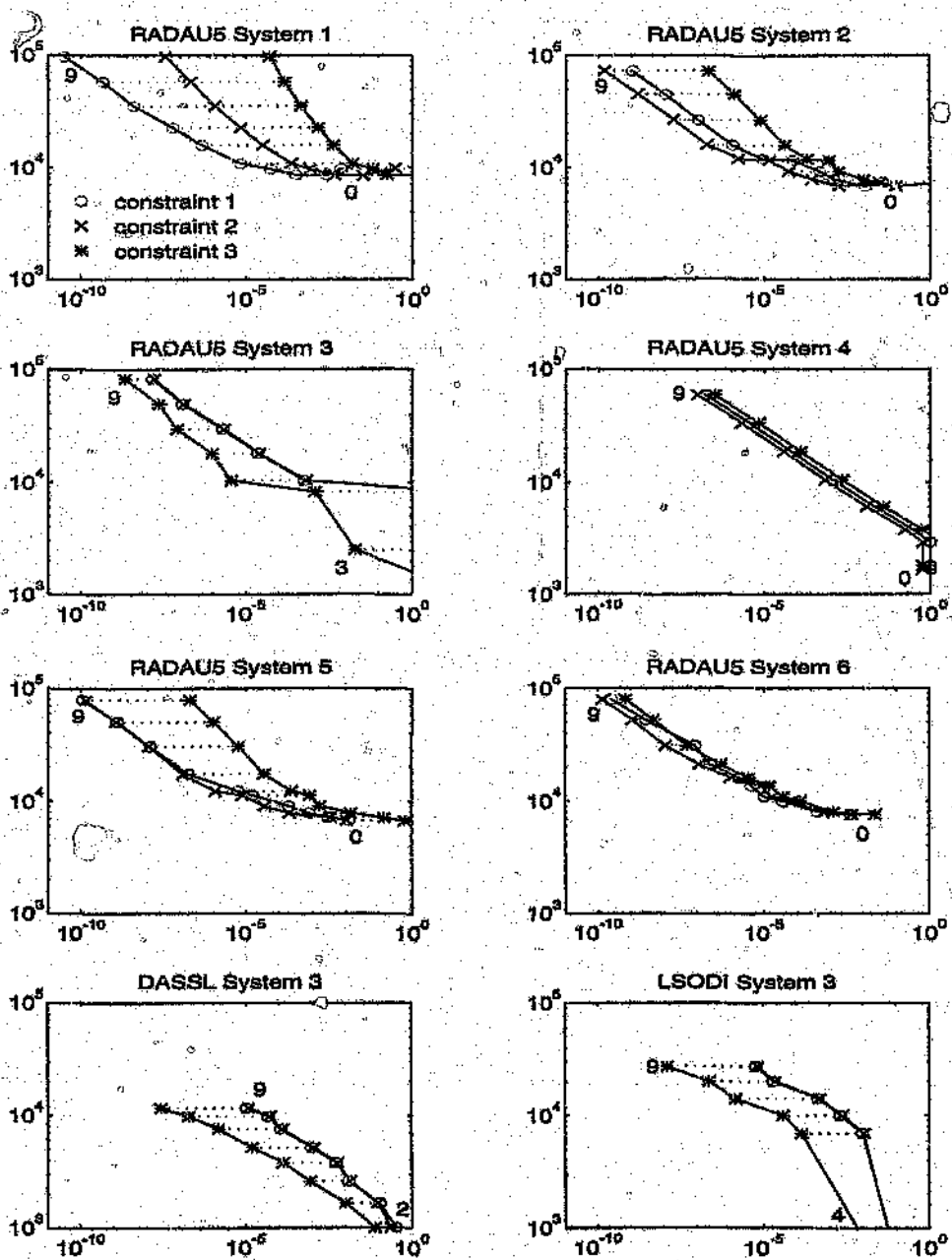


Figure 4.15: Work-precision diagrams for plane pendulum

## 4.7 Van der Pol Oscillator

### Formulation

We consider the Van der Pol oscillator, studied extensively in [48] for the implementation of IRKs in the solution of stiff ODEs. System I:

$$y' = z, \quad (4.25)$$

$$\varepsilon z' = (1 - y^2)z - y, \quad (4.26)$$

$$y(0) = 2, \quad z(0) = -2/3.$$

We examine the stiff case,  $0 \leq \varepsilon \ll 1$ , which has stiffness  $O(\varepsilon^{-1})$ . For nonzero  $\varepsilon$ , System I is a stiff ODE system ( $\nu = 0$ ), while  $\varepsilon = 0$  yields a semi-explicit DAE with  $\nu = 1$  iff  $y \neq \pm 1$ . If  $y = \pm 1$  occurs during integration of the DAE, the system becomes unsolvable since no choice of  $z$  can satisfy (4.26), which is then  $0 = \mp 1$ . We use this example to show how DAE methods solve stiff problems naturally, without limiting the choice of (stiffness) parameters. The solution to (4.25), (4.26) with  $\varepsilon \neq 0$  and  $y(0) = 2$  is smooth iff  $z(0) = -\frac{2}{3} + \frac{10}{81}\varepsilon - \frac{292}{2187}\varepsilon^2 - \frac{1814}{19683}\varepsilon^3 + O(\varepsilon^4)$  (cf. [48, p. 435]). The perturbation caused by our  $O(\varepsilon^0)$  choice of  $z(0)$  is not expected to cause numerical difficulties, and is in most cases significantly smaller (cf. [48]) than the tolerances we use.

System I can be studied using singular perturbations, as in §1.1.2. Expanding  $y(t)$  and  $z(t)$  in powers of  $\varepsilon$  and stopping after three terms, we obtain

$$\begin{aligned} y(t) &= y_0(t) + \varepsilon y_1(t) + \varepsilon^2 y_2(t) \\ z(t) &= z_0(t) + \varepsilon z_1(t) + \varepsilon^2 z_2(t) \quad \text{and} \end{aligned}$$

$$\left. \begin{aligned} y'_0 &= z_0 \\ 0 &= z_0(1 - y_0^2) - y_0 \end{aligned} \right\} \text{II} \quad \left. \begin{aligned} y'_1 &= z_1 \\ z'_0 &= z_1(1 - y_0^2) - 2y_0y_1z_0 - y_1 \end{aligned} \right\} \text{III} \quad \left. \begin{aligned} y'_2 &= z_2 \\ z'_1 &= z_2(1 - y_0^2) - 2y_0y_1z_1 - z_0(y_1^2 + 2y_0y_2) - y_2 \end{aligned} \right\} \text{IV}$$

after equating terms in like powers of  $\varepsilon$ . On differentiating the constraint (or noting Hessenberg form), we easily find that:

1. System II is an index 1 DAE in  $y = (y_0, z_0)^T$ , provided that  $y_0 \neq \pm 1$  (otherwise unsolvable).
2. System III is an index 2 DAE in  $y = (\overbrace{y_0, z_0}^{x_1}, \overbrace{y_1, z_1}^{x_2})^T$ . It is Hessenberg of size 2 iff  $y_0 \neq \pm 1$ .
3. System IV is an index 3 DAE in  $y = (\overbrace{y_0, z_0, y_1, z_1}^{x_2}, \overbrace{y_2, z_2}^{x_3})^T$ . It is Hessenberg of size 3 iff  $y_0 \neq \pm 1$ .

### Solution

System I has  $\nu = 0$  or 1 and is appropriate for all three solvers for all  $\epsilon$ . We requested accuracy to within  $\text{TOL} \approx 10^{-4}$  and attempted to solve on  $0 \leq t \leq 2$ . Because of the small size of the problem, an analytical Jacobian was used in order to reduce computational effort. For DASSL and LSODI, the initial derivative was chosen as  $y'(0) = -2/3$  (true to  $O(\epsilon^0)$ ) and  $z'(0) = 0$  (arbitrarily), and it was left to the codes to make this consistent. Strangely, both failed to do this when given an analytical Jacobian, and so the Jacobian was calculated numerically. See Tables 4.14–4.16 for solution statistics.

| RADAU5     |                                         |      |     |       |           |            |
|------------|-----------------------------------------|------|-----|-------|-----------|------------|
| $\epsilon$ | STEPS                                   | FCN  | JAC | ETF's | $y(2)$    | $z(2)$     |
| $10^{-3}$  | 194                                     | 1437 | 120 | 22    | 1.7628607 | -.83593853 |
| $10^{-6}$  | 324                                     | 2526 | 182 | 27    | 1.7061645 | -.89280868 |
| $10^{-9}$  | 449                                     | 3765 | 247 | 41    | 1.7055574 | -.89345261 |
| $10^{-12}$ | 574                                     | 4906 | 315 | 54    | 1.7055438 | -.89347145 |
| $10^{-15}$ | 701                                     | 6108 | 387 | 68    | 1.7055431 | -.89347040 |
| 0          | Exit at $t = 0.8068530$ , $y \approx 1$ |      |     |       |           |            |

Table 4.14: RADAU5 solution to Van der Pol System I

| DASSL      |                                                   |      |     |       |           |            |
|------------|---------------------------------------------------|------|-----|-------|-----------|------------|
| $\epsilon$ | STEPS                                             | FCN  | JAC | ETF's | $y(2)$    | $z(2)$     |
| $10^{-3}$  | 405                                               | 712  | 54  | 12    | 1.7611896 | -.83760529 |
| $10^{-6}$  | 727                                               | 1240 | 100 | 18    | 1.7043661 | -.89465151 |
| $10^{-9}$  | 925                                               | 1579 | 144 | 47    | 1.7038787 | -.89523551 |
| $10^{-12}$ | More than 500 calls on a step                     |      |     |       |           |            |
| $10^{-15}$ |                                                   |      |     |       |           |            |
| 0          | Repeated ETF's at $t = 0.8059089$ , $y \approx 1$ |      |     |       |           |            |

Table 4.15: DASSL solution to Van der Pol System I

In order to include the DAE case in our comparisons, we then halted integration at  $t = 0.8$  to obtain the work-precision diagrams (Figure 4.16) for the  $\epsilon = 0$  (DAE) and  $\epsilon = 10^{-3}$  (stiff ODE) cases. We used FCN for work, the maximum relative error over both components over all 100 points on  $[0, 0.8]$  for precision (using the  $\text{TOL} = 10^{-15}$  RADAU5 solutions as 'true'), and the range of  $\text{TOL} = 10^{-m}$ ,  $m = 3, \dots, 12$ .

For the singular perturbation problem, System IV was solved by RADAU5, with  $\epsilon = 10^{-3}$ ,  $\text{TOL} = 10^{-4}$  and  $y(0) = (2, -2/3, 0, 0, 0)^T$ . Integration failed at the singularity at  $t \approx 0.8$  due to unsolvability of the DAE subsystem II when  $y = \pm 1$  (as was the case for System I with  $\epsilon = 0$ , which is identical). This clearly shows the unsuitability of the singular perturbation solution within a boundary layer (see also Figure 4.18).

| LSODI         |                                                                           |      |     |           |             |
|---------------|---------------------------------------------------------------------------|------|-----|-----------|-------------|
| $\varepsilon$ | STEPS                                                                     | FCN  | JAC | $y(2)$    | $z(2)$      |
| $10^{-3}$     | 449                                                                       | 911  | 74  | 1.7617150 | -0.83709714 |
| $10^{-6}$     | 669                                                                       | 1497 | 115 | 1.7040915 | -0.89499189 |
| $10^{-9}$     | 994                                                                       | 2201 | 161 | 1.7040655 | -0.89512384 |
| $10^{-12}$    | More than 500 STEPS on one call<br>at $t \approx 0.8061$ , $y \approx -2$ |      |     |           |             |
| $10^{-15}$    |                                                                           |      |     |           |             |
| 0             | Repeated ETF's at $t = 0.8060991$ , $y \approx 1$                         |      |     |           |             |

Table 4.16: LSODI solution to Van der Pol System I

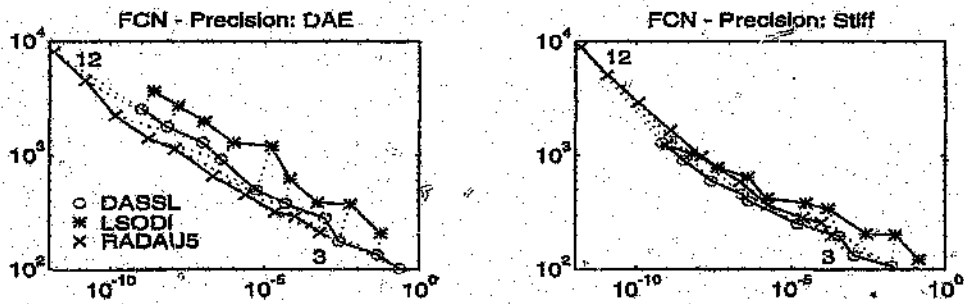
Figure 4.16: Work-precision diagrams for (a) DAE ( $\varepsilon = 0$ ) and (b) stiff ( $\varepsilon = 10^{-3}$ ) Van der Pol oscillator

Figure 4.17 shows the state space and phase space RADAU5 solution to System I for  $\varepsilon = 10^{-3}$ , and the number of steps taken during integration, for high-accuracy ( $\text{TOL} = 10^{-12}$ ) and then comparing Systems I and IV for  $\text{TOL} = 10^{-4}$ . Figure 4.18 shows the first three terms (up to System IV) of the singular perturbation solution up to  $t = 0.8$ , i.e. outside the boundary layer, and then the global absolute error in the second order solution, using the above truth model.

• For System I<sub>0</sub>

- Only RADAU5 yielded solutions globally accurate to within order of magnitude of  $\text{TOL}$ . LSODI and DASSL were two orders behind for  $\text{TOL} = 10^{-4}$  and worse for stricter  $\text{TOL}$ . (Figure 4.16)
- Solutions were insensitive to changes in  $\varepsilon$ ,  $0 \leq \varepsilon \ll 1$ , but numerical cost and difficulty increased on  $[0, 2]$  for smaller values of  $\varepsilon$ , with failure to integrate past the singularity (Tables 4.14–4.16). On  $[0, 0.8]$ , RADAU5 performance was found to be unaffected by the choice of  $\varepsilon = 10^{-3}$  or 0. (Figure 4.16)
- Numerical difficulty, cost and inaccuracy were concentrated at the singularities. (Figure 4.17)

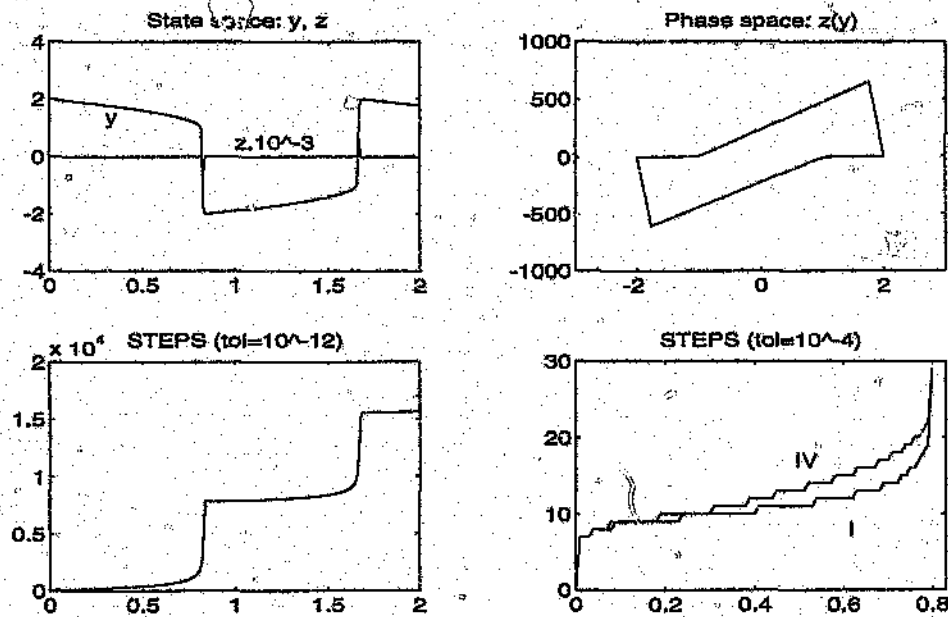


Figure 4.17: RADAU5 solution to Van der Pol oscillator: System I ( $\varepsilon = 10^{-3}$ ).

- On  $[0, 0.8]$  for  $\varepsilon = 10^{-3}$ , the three codes had almost equal work/precision ratios. Effective orders are  $\approx 6$  for DASSL,  $\approx 5$  for RADAU5 and  $\approx 8$  for LSODI (countered by its slightly higher work/precision at loose TOLs). (Figure 4.16(b))
- On  $[0, 0.8]$  for  $\varepsilon = 0$ , all three codes had effective order  $\approx 6$ . DASSL required more work than RADAU5 ( $\approx 60\%$  more at  $10^{-6}$  accuracy) and LSODI more work than DASSL ( $\approx 60\%$  more at  $10^{-6}$  accuracy). (Figure 4.16(a))
- As predicted, the DAE became unsolvable when  $y = \pm 1$ .
- RADAU5 approached the half-period singularity at  $t \approx 0.80685238$  [48] more closely than either DASSL or LSODI.
- For System IV, Figure 4.18 clearly shows (as expected) the importance of higher order terms in the expansion as the singularity is approached. Note also the instability near the singularity, evidenced by the cancelling effect of the first and second order terms, their exponential growth, and the growth in the error, which is stable outside the boundary layer.
- In [48, pp.165–171], RADAU5 and other codes are applied to the stiff problem, but no direct comparisons can be made with their work. Theoretical predictions are made on the global errors of some IRK methods applied to (4.25), (4.26) (for correct  $y(0), z(0)$ ). For RADAU5,

$$y_n - y(t_n) = O(h^5) + O(\varepsilon^2 h^3)$$

$$z_n - z(t_n) = O(h^5) + O(\varepsilon h^3),$$

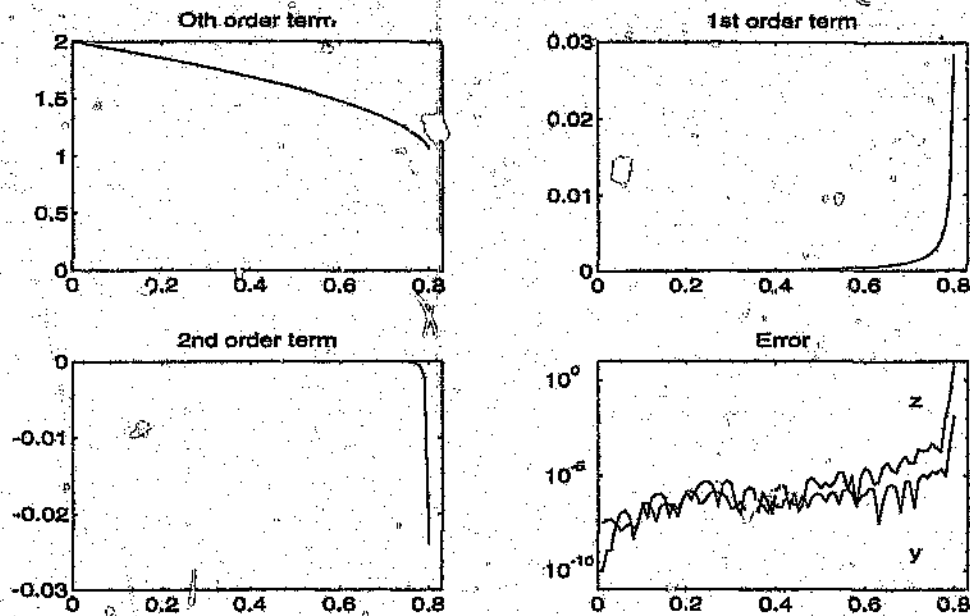


Figure 4.18: RADAU5 solution to Van der Pol oscillator; System IV ( $\epsilon = 10^{-3}$ ).

which agrees favourably with our value of 5 for the effective order of RADAU5 (unless  $\epsilon > O(h)$ ).

## 4.8 Ring Modulator

### Formulation

The ring modulator [47] mixes a low frequency signal  $e_1(t)$  with a high frequency signal  $e_2(t)$ , and the modulated signal is used as an input for an amplifier. The differential system modelling the ring modulator, and comprising seven admittance form constitutive equations combined with Kirchhoff's current law and eight impedance form constitutive equations combined with Kirchhoff's voltage law (p. 5), is

$$\begin{aligned}
 CU_1' &= I_1 - I_3/2 + I_4/2 + I_7 - U_1/R \\
 CU_2' &= I_2 - I_5/2 + I_6/2 + I_8 - U_2/R \\
 C_S U_3' &= I_3 - G(U_{D_1}) + G(U_{D_4}) \\
 C_S U_4' &= -I_4 + G(U_{D_2}) - G(U_{D_3}) \\
 C_S U_5' &= I_5 + G(U_{D_1}) - G(U_{D_3}) \\
 C_S U_6' &= -I_6 - G(U_{D_2}) + G(U_{D_4}) \\
 C_P U_7' &= -U_7/R_1 + G(U_{D_1}) + G(U_{D_2}) - G(U_{D_3}) - G(U_{D_4})
 \end{aligned}$$

$$\begin{aligned}
L_h I_1' &= -U_1 \\
L_h I_2' &= -U_2 \\
L_{S_2} I_3' &= U_1/2 - U - R_{g_2} I_3 \\
L_{S_3} I_4' &= -U_1/2 + U_4 - R_{g_3} I_4 \\
L_{S_2} I_5' &= U_2/2 - U_5 - R_{g_2} I_5 \\
L_{S_3} I_6' &= U_2/2 + U_6 - R_{g_3} I_6 \\
L_{S_1} I_7' &= -U_1 + e_1(t) - (R_0 + R_{g_1}) I_7 \\
L_{S_1} I_8' &= -U_2 - (R_a + R_{g_1}) I_8.
\end{aligned}$$

The diodes are modelled by

$$G(v) = I_s(e^{\lambda v} - 1)$$

and the diode voltages are

$$\begin{aligned}
U_{D_1} &= U_3 - U_5 - U_7 - e_2(t) \\
U_{D_2} &= -U_4 + U_6 - U_7 - e_2(t) \\
U_{D_3} &= U_4 + U_5 + U_7 + e_2(t) \\
U_{D_4} &= -U_3 - U_6 + U_7 + e_2(t).
\end{aligned}$$

Technical parameters are given by

$$\begin{aligned}
R_{g_1} &= 36.3, & R_{g_2} &= R_{g_3} = 17.3, & R_0 &= R_i = 50, \\
R_a &= 600, & R &= 25900, \\
C &= 16 \cdot 10^{-9}, & C_P &= 10 \cdot 10^{-9}, \\
L_h &= 4.45, & L_{S_1} &= 0.002, & L_{S_2} &= L_{S_3} = 0.0005, \\
I_s &= 40.67286402 \cdot 10^{-9}, & \lambda &= 17.7493332, \\
e_1(t) &= 0.5 \sin(2\pi 10^3 t), & e_2(t) &= 2 \sin(2\pi 10^4 t).
\end{aligned}$$

Following [47], we consider 3 values for the artificial capacity  $C_S$ :

$$C_S = 10^{-9}, \quad C_S = 2 \cdot 10^{-12}, \quad C_S = 0,$$

and a set of consistent initial values for all  $C_S$  is

$$\begin{aligned}
U_i(0) &= 0, \quad i = 1, \dots, 7, \\
I_i(0) &= 0, \quad i = 1, \dots, 8.
\end{aligned}$$

The above system,

$$\begin{aligned}
Ay' &= f(y, t), \quad y = (U_1, \dots, U_7, I_1, \dots, I_8)^T, \\
y(0) &= 0,
\end{aligned}$$

is stiff differential for  $C_S \neq 0$  and differential-algebraic for  $C_S = 0$  (since  $A$  is singular). The DAE can be transformed to a Hessenberg system of size 2, with  $\nu = 2$ , and we reorder variables in  $y$ :

$$y = (\overbrace{I_1, \dots, I_8}^{x_1}, \overbrace{U_1, U_2, U_7, U_3, \dots, U_6}^{x_2})^T.$$

A consistent initial derivative for the DAE is  $y'(0) = 0$ , since

$$Ay'(0) = f(y(0), 0) = f(0, 0) = 0 \iff y'(0) = 0.$$

We include this example as a higher-index linear constant-coefficient DAE which can be solved by index 1 integrators.

### Solution

We first attempted to solve the system with  $TOL = 10^{-6}$  over two periods of the slow input signal  $e_1(t)$ , ie.  $T = 2 \cdot 10^{-3}$ . RADAU5 (with the usual scaling) and DASSL (without scaling) were successful for all 3 values of  $C_S$ , but LSODI (without scaling) failed to solve the very stiff  $C_S = 2 \cdot 10^{-12}$  and differential-algebraic  $C_S = 0$  cases. For LSODI, the integration failure point was found to be further for shorter output intervals.

In order to compare code performance, we then integrated over a half-period using  $TOL = 10^{-m}$ ,  $m = 4, \dots, 10$ , and  $\Delta time = 10^{-6}$ , and obtained the work-precision diagram, Figure 4.20. We used FCN for work and the maximum (over all components over all 500 points on  $[0, 5 \cdot 10^{-2}]$ ) absolute global error for precision, using a RADAU5 solution with  $TOL = 10^{-15}$  as a truth model.

- From the work-precision diagram:

- LSODI, which failed at looser tolerances, took  $\approx 2.5$  times as much work as DASSL and RADAU5, while DASSL was more economical than RADAU5 at loose tolerances.
- Only RADAU5 solutions were accurate to within an order of magnitude of  $TOL$ , with DASSL and LSODI an order behind.
- The effective order is  $\approx 6$  for all three codes (within the range of comparison).

- The numerical difficulties reported in [47], especially for  $C_S = 2 \cdot 10^{-12}$ , were not observed, and the graph for the component  $U_3(t)$  in [47] is appreciably different (and dependent on the value of  $C_S$ ) from that obtained here, which appears identical for the three values of  $C_S$  used. The output components  $I_8(t)$  do appear to agree, however.
- Of significance is that no higher-index scaling was necessary here for successful solution by DASSL.

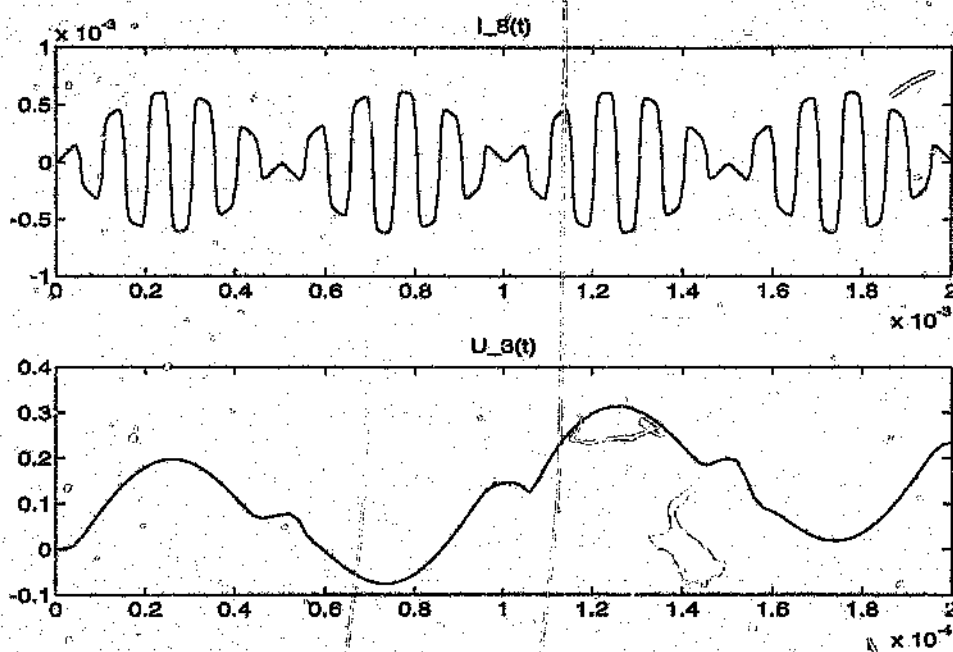


Figure 4.19: DASSL solution to ring modulator: components  $I_8(t)$  and  $U_3(t)$ , for  $C_S = 10^{-9}$ .

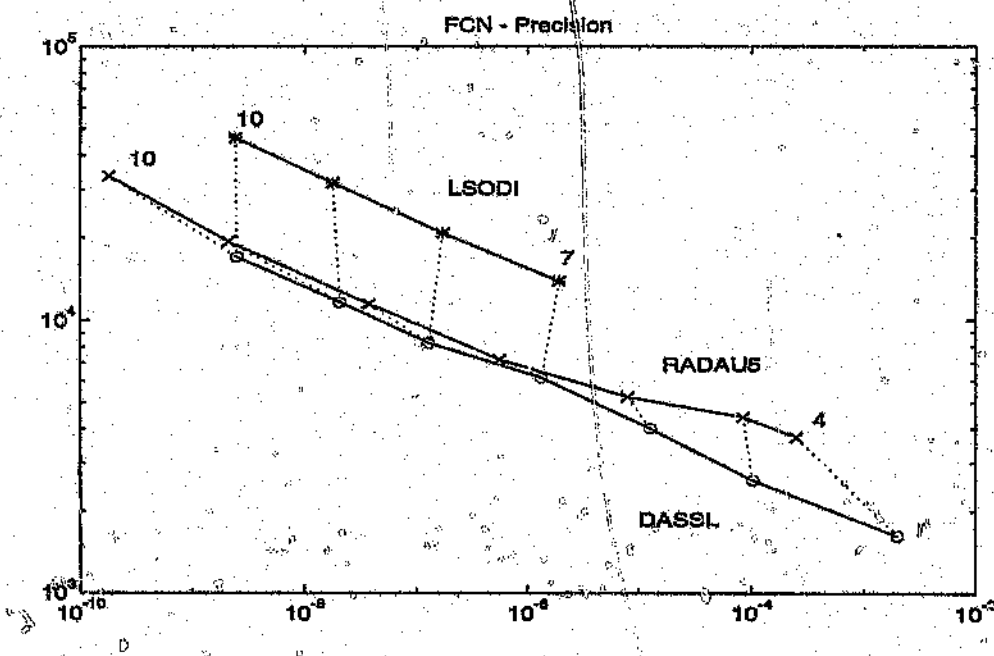


Figure 4.20: Work-precision diagram for ring modulator

## 4.9 Cascade of Differential Amplifiers

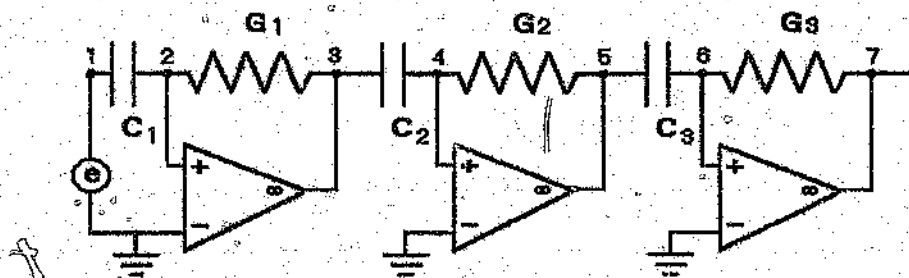


Figure 4.21: Cascade of differential amplifiers

## Formulation

Arbitrarily high-index DAEs can arise in circuits containing differential amplifiers [14]. We consider a simple cascade of  $s$  differential amplifiers, which we study in depth for  $s = 3$  (see Figure 4.21). Kirchhoff's laws (p. 5) yield the DAE system (where  $v_i$  is the potential at node  $i$ , and the resistors (of conductance  $g$ ) and capacitors (of capacitance  $c$ ) are scalar):

$$\begin{aligned}
 c_1(v_1' - v_2') &= 0 \\
 c_1(v_2' - v_1') + g_1(v_2 - v_3) &= 0 \\
 c_2(v_3' - v_4') + g_1(v_3 - v_2) + i_1 &= 0 \\
 c_2(v_4' - v_3') + g_2(v_4 - v_5) &= 0 \\
 c_3(v_5' - v_6') + g_2(v_5 - v_4) + i_2 &= 0 \\
 c_3(v_6' - v_5') + g_3(v_6 - v_7) &= 0 \\
 g_3(v_7 - v_6) + i_3 &= 0 \\
 v_1 &= e \\
 v_2 &= 0 \\
 v_4 &= 0 \\
 v_6 &= 0.
 \end{aligned} \tag{4.27}$$

This system, which is easily generalized to  $s \neq 3$ , is linear constant-coefficient in  $y = (v_1, \dots, v_7, i_e, i_1, \dots, i_3)^T$ . It seems natural to start with a grounded system at rest, so we set  $y(0) = 0$ . Index reduction, differentiating the constraints four times, reduces the system to the following ODE, from which we know that  $\nu = 4$ :

$$v_3' = \begin{pmatrix} v_1' \\ v_2' \\ v_3' \\ v_4' \\ v_5' \\ v_6' \\ v_7' \\ i_e' \\ i_1' \\ i_2' \\ i_3' \end{pmatrix} = \begin{pmatrix} e' \\ 0 \\ -\frac{c_1}{g_1} e'' \\ 0 \\ -\frac{c_1 c_2}{g_1 g_2} e''' \\ 0 \\ \frac{c_1 c_2 c_3}{g_1 g_2 g_3} e'''' \\ -c_1 e'' \\ c_1 e'' - \frac{g_1 c_2}{c_1} e''' \\ \frac{g_1 c_2}{c_1} e''' + \frac{c_1 c_2}{g_1 g_2} c_3 e'''' \\ -\frac{c_1 c_2}{g_1 g_2} c_3 e'''' \end{pmatrix}$$

In this case, the underlying ODE is easily integrated. The  $s = 3$  solution (easily generalized) is

$$y_3(t) = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ i_e \\ i_1 \\ i_2 \\ i_3 \end{pmatrix} = \begin{pmatrix} e(t) \\ 0 \\ -\frac{c_1}{g_1} e'(t) \\ 0 \\ -\frac{c_1 c_2}{g_1 g_2} e''(t) \\ 0 \\ \frac{c_1 c_2 c_3}{g_1 g_2 g_3} e'''(t) \\ -c_1 e'(t) \\ c_1 e'(t) - \frac{g_1 c_2}{c_1} e''(t) \\ \frac{g_1 c_2}{c_1} e''(t) + \frac{c_1 c_2}{g_1 g_2} c_3 e'''(t) \\ -\frac{c_1 c_2}{g_1 g_2} c_3 e'''(t) \end{pmatrix} \quad (4.28)$$

Note that:

1. Constants of integration are absent from (4.28) as they were found to be zero during index reduction.
2. Our choice of  $y(0) = 0$  is inconsistent (unless the voltage source  $e$  and its first  $s$  derivatives are initially zero) and, without index reduction,  $y'(0)$  is unknown, so that successful solution of this problem relies on the ability of the codes to recover from inconsistent initial values.

In this problem, which is not reducible to Hessenberg form, the very natural extension of index to component level can be seen. In the case of  $v_5$  above, for which the original system had to be differentiated three times to obtain  $v_5' = -\frac{c_1 c_2}{g_1 g_2} e'''$ , and a perturbation in the equation  $v_1 = e$  in (4.27) arrives differentiated twice in the solution  $v_5 = -\frac{c_1 c_2}{g_1 g_2} e''$ , we deduce that  $v_5$  has (both differentiation and perturbation) index 3. We can then reorder

$$y = (\underbrace{v_1, v_2, v_4, v_6}_{\nu=1}, \underbrace{i_e, v_3}_{\nu=2}, \underbrace{i_1, v_5}_{\nu=3}, \underbrace{i_2, v_7, i_3}_{\nu=4})^T.$$

This component classification was found to be useful for effective variable scaling during numerical integration. The above procedure can be generalized to a cascade of size  $s$  (and index  $\nu = s + 1$ ):

$$\begin{aligned} y_1 &= (\overbrace{v_1, v_2, i_e, v_3, i_1}^T)^T, \\ y_2 &= (\overbrace{v_1, v_2, v_4, i_e, v_3, i_1, v_5, i_2}^T)^T, \\ y_3 &= (\overbrace{v_1, v_2, v_4, v_6, i_e, v_3, i_1, v_5, i_2, v_7, i_3}^T)^T, \\ &\vdots \\ y_s &= (\overbrace{v_1, v_2, \dots, v_{2s}, i_e, v_3, \dots, i_{s-2}, v_{2s-1}, i_{s-1}, v_{2s+1}, i_s}^T)^T. \end{aligned}$$

### Solution

We attempt to solve the DAE for  $s = 1, 2, 3, 4$  on  $t \in [0, 10]$  with  $\text{TOL} = 10^{-4}$  and the parameters

$$c_i = i \cdot 10^{-3} \quad \forall i, \quad g_i = i^2 \cdot 10^{-3} \quad \forall i, \quad e(t) = \sin(t).$$

We start with the somewhat pessimistic predictions from the theory, the first being that the  $k$ -step variable-stepsizes BDF method (ie. DASSL and LSODI) can only guarantee accuracy to order  $O(h_{\max}^q)$ ,  $q = \min(k, k - \nu + 2)$  in some components (cf. Theorem 3.2.1). Since DASSL and LSODI both start with  $k = 1$ , we can expect errors of order  $O(h_0^{2-s})$  at the start of integration, becoming more likely to fail the error test as  $s$  increases. Note that the strategy of excluding high-index components from error tests is inappropriate for this problem, since it is the highest-index component (output voltage) that we are interested in. For RADAU5, we hope to solve at least the  $s = 1$  (index 2) and  $s = 2$  (index 3) problems, using the high-index scaling option provided for Hessenberg form DAEs.

### RADAU5

Table 4.17 shows the results of applying RADAU5 to the cascade problem of size 1, 2 and 3 under a variety of scaling options. Remember that RADAU5 requires the dimensions of the index 1, 2 and 3 variables, stored in that order in the unknown vector. In an attempt to scale the index 4 (for  $s = 3$ ) variables effectively, in order to include the steplength in the scaling it was necessary to convert to a multicall mode for RADAU5. 100 RADAU5 calls on  $t \in [0, 10]$ , using the last successful steplength on each call as the current  $h$ , using the RADAU5 index 3 scaling option for the index 3 and 4 variables, and then scaling the tolerance for the index 4 variables by  $1/h$ , yielded successful results. The same approach applied to the index 5 ( $s = 4$ ) system, ie. scaling index 4 tolerances by  $1/h$  and index 5 tolerances by  $1/h^2$ , was not successful, however.

Table 4.18 shows solution statistics for the successful solution of the  $s = 1, 2, 3$  problem by RADAU5. Using the true solution, we can define

$$\text{accuracy} = \text{round}(-\log_{10}(\text{mean}(\text{relative error}))),$$

| $s$ | $\nu$ | $\dim(\nu = 1)$ | $\dim(\nu = 2)$ | $\dim(\nu = 3)$ | Result     |
|-----|-------|-----------------|-----------------|-----------------|------------|
| 1   | 2     | 5               | 0               | 0               | Success    |
|     |       | 2               | 3               | 0               | Success    |
| 2   | 3     | 8               | 0               | 0               | Failure    |
|     |       | 5               | 3               | 0               | Failure    |
|     |       | 3               | 5               | 0               | Failure    |
|     |       | 3               | 2               | 3               | Success    |
| 3   | 4     | 11              | 0               | 0               | Failure    |
|     |       | 8               | 3               | 0               | Failure    |
|     |       | 6               | 2               | 3               | Failure    |
|     |       | 4               | 4               | 3               | Failure    |
|     |       | 4               | 2               | 5               | Failure    |
|     |       | 4               | 0               | 7               | Failure    |
|     |       | 0               | 4               | 7               | Disallowed |

Table 4.17: RADAU5 solution to cascade problem: index 1, 2, 3 scaling options

so that a component with a global relative error of  $TOL (= 10^{-4})$  has an accuracy of 4.

| $s$ | $\nu$ | STEPS | FCN | JAC | ETF's | Accuracy                    |
|-----|-------|-------|-----|-----|-------|-----------------------------|
| 1   | 2     | 54    | 241 | 12  | 11    | (5,32,3,3,3)                |
|     |       | 42    | 184 | 9   | 8     | (5,35,3,3,3)                |
| 2   | 3     | 59    | 259 | 10  | 12    | (5,29,25,3,3,2,2,2)         |
| 3   | 4     | 106   | 727 | 106 | 0     | (11,84,54,52,2,2,2,2,1,1,1) |

Table 4.18: RADAU5 solution statistics for cascade problem:  $s = 1, 2, 3$ 

- Numerical solution required prior manipulation, almost to the point of analytical solution, in order to learn enough about the high-index DAE for its solution.
- RADAU5's high-index scaling option appears to be consistent with the concept of component-level differential index, since the index 2 problem was solved more efficiently with correct scaling than without scaling, and the index 3 problem could be solved with correct scaling but not with other likely choices.
- RADAU5 was unsuccessful for index 4 (and higher) without modification of the calling routine to include high-index TOL scaling, only partially successful for index 4 (due to unacceptable inaccuracy in the required solution component) and unsuccessful for higher index.
- RADAU5 recovered successfully from the inconsistent i.c.'s. Replacing  $y(0) = 0$  with the true initial value  $y_3(0)$  from (4.28) did not affect Table 4.17 at all, but did improve the solution accuracy near  $t = 0$ , so that, for example, the  $s = 3$  accuracy became (11, 77, 54, 51, 4, 4, 3, 3, 1, 1, 1).

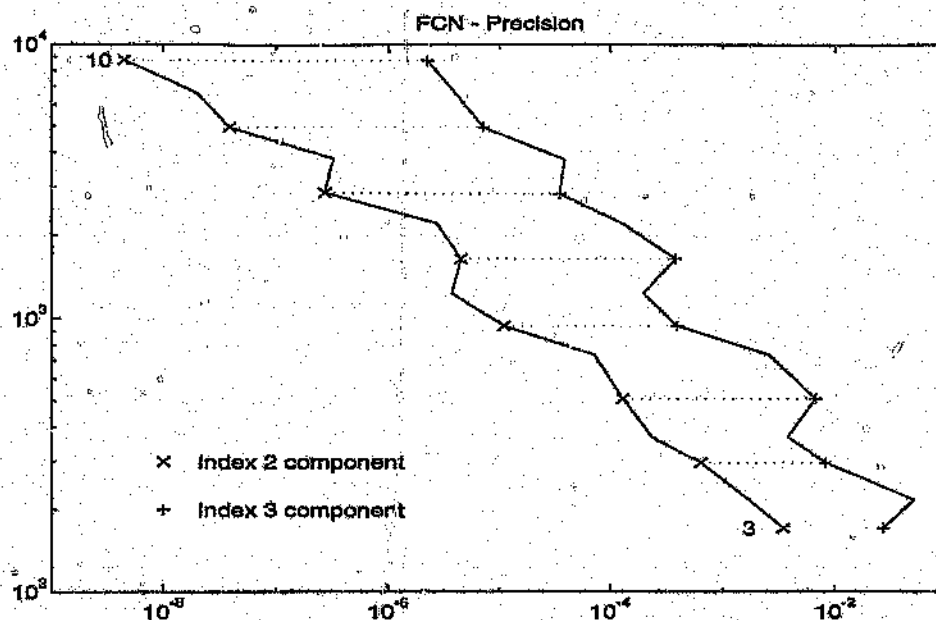


Figure 4.22: RADAU5 work-precision diagram for cascade problem:  $s = 2$

- The observed accuracy in Table 4.18 is a good reflection of the TOL scaling, with higher-index variables enjoying less accuracy in each case.

Figure 4.22 is the work-precision diagram for the index 3  $s = 2$  system, using FCN for work and the maximum absolute true error on all 1000 points of  $[0, 10]$  for accuracy, for both the index 2 component  $v_3$  and the index 3 component  $v_5$ . To ensure safe integration over a range of  $\text{TOL} = 10^{-m}$ ,  $2 \cdot m = 6, \dots, 20$ , a safe stopping criterion for the Newton iteration was set, the Jacobian was evaluated after each step and the NEWT per step allowance was increased. Rough estimates for the effective order, using linear regression on all data, are 4 for the index 2 component and 3 for the index 3 component. Also, the index 2 accuracy is an order behind TOL, while the index 3 accuracy fell further behind TOL as precision increased. This confirms the expected IRK order loss for higher index. Note the unusually irregular data distribution for RADAU5 on this problem.

### LSODI

The inconsistent initial values  $y(0) = y'(0) = 0$  halted integration by LSODI at the initial step, even for  $s = 1$ . On using true initial values, integration failed (due to repeated ETF's) before  $T = 10$  was reached. For  $s = 1$ , the partial solution was accurate until failure. For example, accuracy = (4,  $\infty$ , 3, 3, 3) on  $t \in [0, 4.1]$  using high-index variable scaling. Without scaling, integration failed earlier. No solution could be obtained for  $s = 2$  or higher.

## DASSL

Curiously, DASSL failed to compute an approximate  $y'(0)$  when  $y'(0) = 0$  was given as inconsistent, but was found to recover quickly from the inconsistent  $y'(0) = 0$ , given as consistent. Without appropriate tolerance scaling, integration failed due to repeated ETF's before  $T = 10$ , as with LSODI. The inconsistent  $y(0) = 0$  posed no great problem, with only an initial drop in solution accuracy, as with RADAU5.

Some successful integration statistics are given for  $s = 1, 2, 3, 4$  in Table 4.19. As with the pendulum experiment, weaker or stronger TOL, by a factor of 10 (unscaled) or  $h$  (scaled), resulted in less accuracy or outright failure characterized by repeated ETF's and/or CTF's, not always near  $t = 0$ . 'Accuracy' here corresponds to the accuracy of the output voltage and can be compared to the second last accuracy component in Table 4.18. In all cases,  $h_0 = 10^{-6}$  was used, with other choices being less successful. As for RADAU5, we experienced more difficulty and obtained less accuracy for higher-index components. (An option for setting  $h_{\min}$  in DASSL might have proved helpful for limiting high-index scaled tolerances.)

An attempt to demand an accuracy of  $\text{TOL} = 10^{-4}$  for the highest-index components, adjusting lower-index tolerances appropriately, generally failed, probably due to changes in  $h$  allowing inappropriately loose lower-index component tolerances.

The general approach of scaling index  $m$  components by  $h^{1-m}$  appears to be unsatisfactory, and this problem clearly verifies the numerical difficulty encountered at higher index.

| $s$ | TOL       |               |               |               |               | STEPS | FCN | JAC | ETF's | Accuracy |
|-----|-----------|---------------|---------------|---------------|---------------|-------|-----|-----|-------|----------|
|     | $\nu = 1$ | $\nu = 2$     | $\nu = 3$     | $\nu = 4$     | $\nu = 5$     |       |     |     |       |          |
| 1   | $10^{-4}$ | $10^{-4}/h$   |               |               |               | 117   | 165 | 23  | 2     | 3        |
|     | $h$       | 1             |               |               |               | 36    | 55  | 21  | 0     | 0        |
| 2   | $10^{-8}$ | $10^{-8}/h^2$ | $10^{-8}/h^4$ |               |               | 900   | 462 | 34  | 1     | 2        |
|     | $10^{-6}$ | $10^{-6}/h^2$ | $10^{-6}/h^3$ |               |               | 181   | 285 | 31  | 10    | 2        |
|     | $10^{-8}$ | $10^{-6}/h$   | $10^{-8}/h^3$ |               |               | 238   | 330 | 25  | 4     | 2        |
| 3   | $10^{-4}$ | $10^{-4}/h$   | $10^{-4}/h^2$ | $10^{-4}/h^3$ |               | 150   | 231 | 30  | 2     |          |
| 4   | $10^{-4}$ | $10^{-4}/h$   | $10^{-4}/h^2$ | $10^{-6}/h^3$ | $10^{-4}/h^5$ | 168   | 292 | 38  | 10    | 0        |

Table 4.19: DASSL solution statistics for cascade problem:  $s = 1, 2, 3, 4$

## Conclusion

This (arbitrarily) higher-index problem is clearly difficult to solve using these codes. The difficulty in obtaining DASSL solutions on a suitable range of TOL prevented the comparison of effective order, and we only conclude that RADAU5 and DASSL appear to obtain comparable accuracy for comparable work on successful integration runs. Due to effective higher-index scaling, RADAU5 proved successful where the others failed.

## 4.10 MOL Transmission Line

### Formulation

We consider a method-of-lines (MOL) discretization of the following nonlinear hyperbolic PDE, which has been used to describe transmission lines, cell movement and branching random walks [31].

$$\left. \begin{aligned} \epsilon^2 v_{tt} + (1 + g(v))v_t &= k^2 v_{xx} + f(v), \\ x &\in [0, L], \quad t \in [0, T], \\ v(x, 0) &= v_0(x), \quad v_t(x, 0) = 0, \\ g &\in C^2[0, 1], \quad 1 + g(u) > 0, \\ f &\in C^2[0, 1], \quad f(s) > 0 \quad \forall s \in [0, 1], \\ f(0) &= f(1) = 0, \quad f'(0)f'(1) < 0. \end{aligned} \right\} I$$

From theory [31], we know that System I has smooth travelling wave solutions

$$u(x, t) = U(x - ct) \quad \forall c \in \left[ \sqrt{\frac{4k^2 f'(0)}{1 + 4\epsilon^2 f'(0)}}, \frac{k}{\epsilon} \right),$$

and that for nonnegative  $v_0(x)$  with bounded support on the right,

$$\begin{aligned} v_0(x) &\geq 0, \quad x \in [0, x_r], \quad v_0(x) = 0, \quad x \in [x_r, L], \\ v(x_r + \bar{c}t, t) &\rightarrow 0 \quad \text{as } t \rightarrow \infty \quad \forall \bar{c} \in \left( \sqrt{\frac{4k^2 f'(0)}{1 + 4\epsilon^2 f'(0)}}, \frac{k}{\epsilon} \right). \end{aligned}$$

Similarly, on the left, the solution tends pointwise to 1, due to the growth term  $f(v)$ . This means that the bulk of the nonzero part of the solution asymptotically lies to the left of the line  $x = x_r + c_{\min}t$ , where the minimal wave speed  $c_{\min}$  is given by

$$c_{\min} = \sqrt{\frac{4k^2 f'(0)}{1 + 4\epsilon^2 f'(0)}}.$$

We now examine a particular application with  $0 \leq \epsilon \ll 1$  and consider differential-algebraic regularization of System I. Rephrasing I as the stiff ODE system in  $t$ ,

$$\left. \begin{aligned} v_t &= w \\ \epsilon^2 w_t &= k^2 v_{xx} + f(v) - (1 + g(v))w, \end{aligned} \right\} II$$

and setting  $\epsilon^2 = 0$ , this DAE system has  $\nu = 1$  iff  $(1 + g(v)) \neq 0$ . So the hyperbolic condition  $1 + g(v) > 0$  ensures  $\nu = 1$ . Note that the parabolic case, with  $1 + g(v) = 0$ , has undefined index, and DAE solvers are unsuitable for solving System I in this case.

Parameters are obtained from a generalization of the Hodgkin-Huxley and Fitzhugh-Nagumo models of the nerve axon to include a small induction along the length of the axon [91]. The equivalent circuit (where  $L$  is now induction) is modelled by

$$LC \frac{\partial^2 v}{\partial x^2} + \frac{\partial}{\partial t} (LG(v) + RCv) = \frac{\partial^2 v}{\partial x^2} - RG(v),$$

which, with  $\epsilon^2 = L/R \ll 1$ ,  $g(v) = LG'(v)/RC$ ,  $k^2 = 1/RC$ , and  $f(v) = -G(v)/C$ , yields the stiff system I. Specifically, we consider System I with  $\epsilon$  and  $k$  as free parameters,  $L = 1$ ,  $f(v) = \alpha(v - v^3)$ ,  $g(v) = -\epsilon^2 \alpha(1 - 3v^2)$ ,  $\alpha > 0$  and  $v_0(x) = 1 - H(x)$  ( $H$  Heaviside function), which comply with the conditions.

MOL discretization of System II replaces  $v$  and  $w$  with vectors representing numerical approximations to their values at equally spaced nodes on  $[0, L]$ , ie.

$$v_i(t) \approx v(x_i, t), \quad x_i = ih, \quad i = 0, \dots, n, \quad n = L/h = 1/h,$$

and  $v_{xx}$  is discretized by a second order central divided difference approximation:

$$v_{xx} = \frac{1}{h^2} Av, \quad A = \begin{pmatrix} -2 & +2 & & & \\ +1 & -2 & +1 & & \\ & \ddots & \ddots & \ddots & \\ & & +1 & -2 & +1 \\ & & & +1 & -2 \end{pmatrix}.$$

Note that  $h$  here is the space (not time) discretization parameter. The first and last rows of  $A$  include boundary conditions corresponding to  $v(x > 1, t) = 0$  and  $v_x(0, t) = 0$ , but these conditions are not otherwise enforced. Finally, the semi-explicit stiff/DA system we wish to solve (the largest system we consider) is

$$\begin{aligned} v'_i(t) &= w_i(t) \\ \epsilon^2 w'_i(t) &= \frac{k^2}{h^2} \sum_{j=1}^{n+1} A_{ij} v_j(t) + f_i(t) - (1 + g_i(t)) w_i(t), \end{aligned}$$

for  $i = 1, \dots, n+1 = 1/h$ , with  $\epsilon^2 \ll 1$ ,  $f_i = \alpha(v_i - v_i^3)$ ,  $g_i = -\epsilon^2 \alpha(1 - 3v_i^2)$ ,  $\alpha > 0$ , and initial conditions  $v(0) = (1, 0, \dots, 0)^T$ ,  $w(0) = v'(0) = 0$ .

### Solution

We attempted to solve System II over the time taken for the slowest predicted travelling wave to traverse the axon,  $T = 1/c_{\min}$ . By this time the potential applied at  $x = 0$  from  $t = 0$  is expected to have reached  $L = 1$ . Different choices for  $\epsilon^2$ ,  $k^2$  and  $\alpha$  give us 6 sample problems (Table 4.20). Required accuracy was set at  $\text{TOL} = 10^{-6}$ , and a 'true' solution was obtained using  $\text{TOL} = 10^{-9}$  (by RADAU5 in Problem 6 and by DASSL otherwise).

Due to an inability to calculate the initial derivative (for  $w$ ), LSODI failed at  $t = 0$  even for  $\epsilon^2 = 1$ . DASSL and RADAU5 showed varying degrees of success, clearly showing a travelling wave solution at a fixed speed. This and other properties, such as diffusion, can be seen in Figures 4.23–4.25. Of quantitative interest is the observed wavespeed  $c_*$ , derived from the output, and 'Error', the maximum absolute error over all components of  $v$  and  $w$  over the entire interval of integration, based on the 'true' solution. An output interval of  $\Delta t = T/200$  was used. Solution statistics are given in Table 4.21, and Figure 4.26 shows work-precision diagrams based on both ETIME and FCN.

| #  | $n$ | $\epsilon^2$ | $k^2$     | $\alpha$ | $c_{\min}$           | $c_{\max}$ | $T$   | Symbol   |
|----|-----|--------------|-----------|----------|----------------------|------------|-------|----------|
| 1  | 20  | $10^{-3}$    | $10^{-3}$ | 1        | $6.3 \cdot 10^{-2}$  | 1          | 15.8  | $\circ$  |
| 2a | 40  | $10^{-3}$    | $10^{-3}$ | 1        | $6.3 \cdot 10^{-2}$  | 1          | 15.8  |          |
| 2b | 80  | $10^{-3}$    | $10^{-3}$ | 1        | $6.3 \cdot 10^{-2}$  | 1          | 15.8  |          |
| 3  | 20  | $10^{-3}$    | 1         | 1        | 2.0                  | 31.6       | 0.501 | $\times$ |
| 4  | 20  | $10^{-3}$    | $10^{-3}$ | 10       | 0.20                 | 1          | 5.10  | $+$      |
| 5a | 20  | $10^{-6}$    | $10^{-6}$ | 1        | $2.00 \cdot 10^{-3}$ | 1          | 500   |          |
| 5b | 20  | $10^{-6}$    | $10^{-6}$ | 1        | $2.00 \cdot 10^{-3}$ | 1          | 100   |          |
| 6  | 20  | 0            | $10^{-6}$ | 1        | $2.00 \cdot 10^{-3}$ | $\infty$   | 500   |          |

Table 4.20: Sample problems for MOL transmission line

| #  | Code   | STEPS                                    | FCN  | JAC | ETF's | ETIME (disk) | Error               | $c_*$               |
|----|--------|------------------------------------------|------|-----|-------|--------------|---------------------|---------------------|
| 1  | DASSL  | 370                                      | 480  | 29  | 5     | 13.8s        | $1.3 \cdot 10^{-6}$ | $6.0 \cdot 10^{-2}$ |
|    | RADAU5 | 127                                      | 937  | 100 | 0     | 29.2s        | $2.3 \cdot 10^{-6}$ |                     |
| 2a | DASSL  | 352                                      | 488  | 37  | 7     | 35.4s        |                     |                     |
|    | RADAU5 | Exit at $t = 0$ due to singular matrix   |      |     |       |              |                     |                     |
| 2b | DASSL  | Underflow error at $t = 0$               |      |     |       |              |                     |                     |
|    | RADAU5 | Exit at $t = 0$ due to singular matrix   |      |     |       |              |                     |                     |
| 3  | DASSL  | 430                                      | 537  | 41  | 0     | 15.8s        | $4.4 \cdot 10^{-4}$ | 14.3                |
|    | RADAU5 | 203                                      | 957  | 7   | 5     | 20.8s        | $3.4 \cdot 10^{-6}$ |                     |
| 4  | DASSL  | 809                                      | 928  | 40  | 3     | 20.5s        | $1.8 \cdot 10^{-4}$ | 0.26                |
|    | RADAU5 | 344                                      | 2623 | 304 | 1     | 73.2s        | $1.8 \cdot 10^{-5}$ |                     |
| 5a | DASSL  | 1315                                     | 1849 | 100 | 40    | 35.3s        | $3.8 \cdot 10^{-3}$ | $3.3 \cdot 10^{-3}$ |
| 5b |        | 1231                                     | 1760 | 88  | 45    | 33.9s        |                     |                     |
| 5a | RADAU5 | 560                                      | 4353 | 481 | 56    | 116.8s       | $5.7 \cdot 10^{-5}$ |                     |
| 5b |        | 506                                      | 3949 | 438 | 52    | 106.0s       |                     |                     |
| 6  | DASSL  | $y'(0)$ could not be computed at $t = 0$ |      |     |       |              |                     |                     |
|    | RADAU5 | 556                                      | 4338 | 481 | 56    | 116.2s       | $6.5 \cdot 10^{-5}$ | $9.3 \cdot 10^{-3}$ |

Table 4.21: Solution statistics for MOL transmission line

From these results we can observe the following.

- Problem 1, taken as the basic form, results in a wavespeed slightly slower than the predicted  $c_{\min}$ . (See Figure 4.23.) Figure 4.26 shows the global absolute errors of all

components. Clearly, RADAU5 succeeded in controlling the inaccuracy while DASL exhibits a growing error.

- Problem 2, using a finer mesh with more interior nodes, results in a more ill-conditioned iteration matrix, which halted integration when numerically singular. DASL appears more successful in this regard, but this is possibly at the expense of accuracy. No 'true' solution could be obtained for sufficiently strict TOL.
- Problem 3, with large diffusion constant  $k$ , shows a wavespeed well within the predicted range. (See Figure 4.24.) Our value for  $c_*$  may be inaccurate due to rapid diffusion of the initial disturbance.
- Problem 4, with a proportionally larger shunt conductance in the equivalent circuit, shows a wavespeed within the predicted range, slightly higher than  $c_{\min}$ . (See Figure 4.25.)
- Problems 5 and 6, in which inductance is reduced, even to zero, show wavespeeds within the predicted range and, in DASL, an inability to handle the initial condition for the DAE Problem 6.
- Throughout this problem, DASL took less effort, but obtained less accuracy than RADAU5, significantly worse than TOL in some cases, whereas RADAU5 kept the accuracy close to TOL. Also, increasing stiffness resulted in both more work and less accuracy. (See Figure 4.26.)

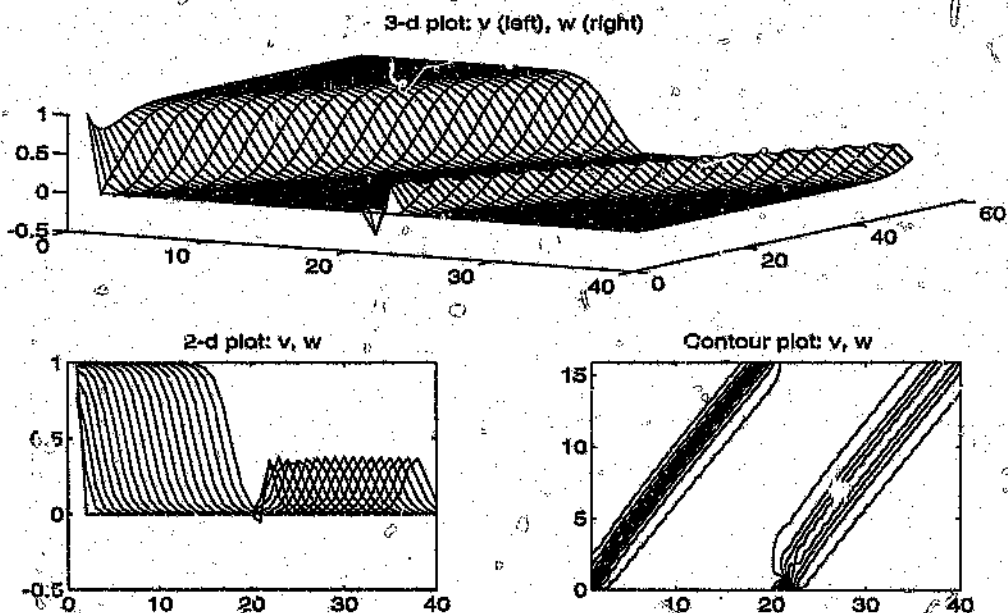


Figure 4.23: DASL solution to MCL Problem 1: 3-d plot ( $v(x,t)$ ,  $w(x,t)$ ), time evolution ( $v(x)$ ,  $w(x)$  for various  $t$ ) and contour plot ( $v(x,t)$ ,  $w(x,t)$ ).

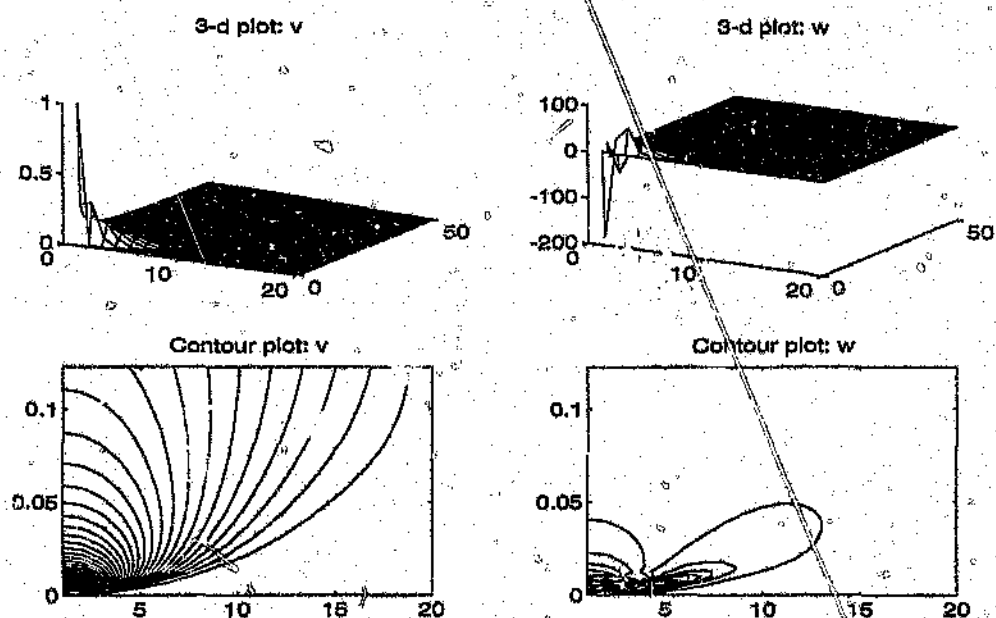


Figure 4.24: DASSL solution to MOL Problem 3: 3-d and contour plots

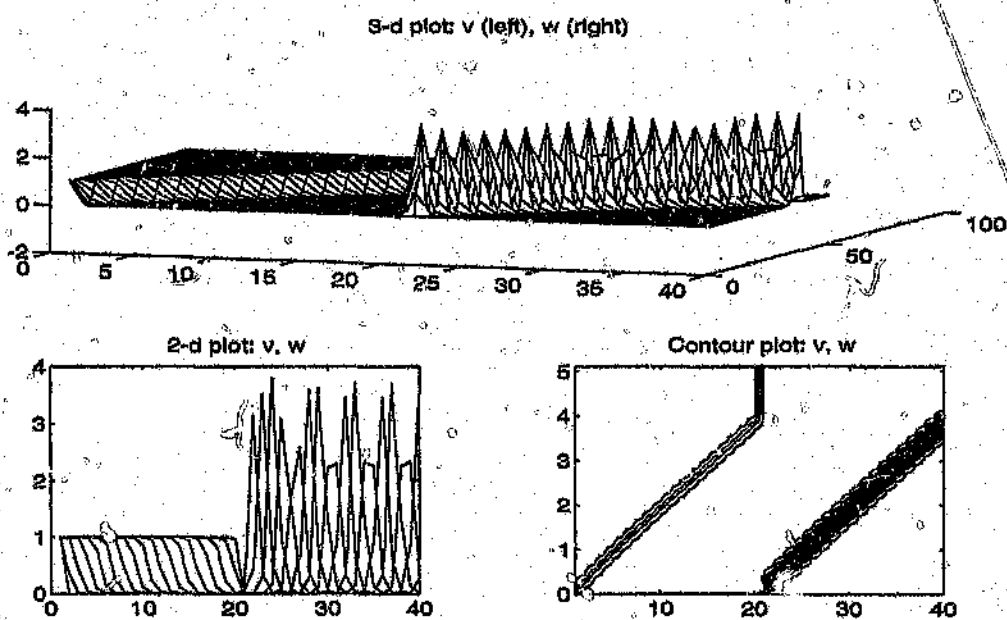


Figure 4.25: DASSL solution to MOL Problem 4: 3-d plot, time evolution and contour plot

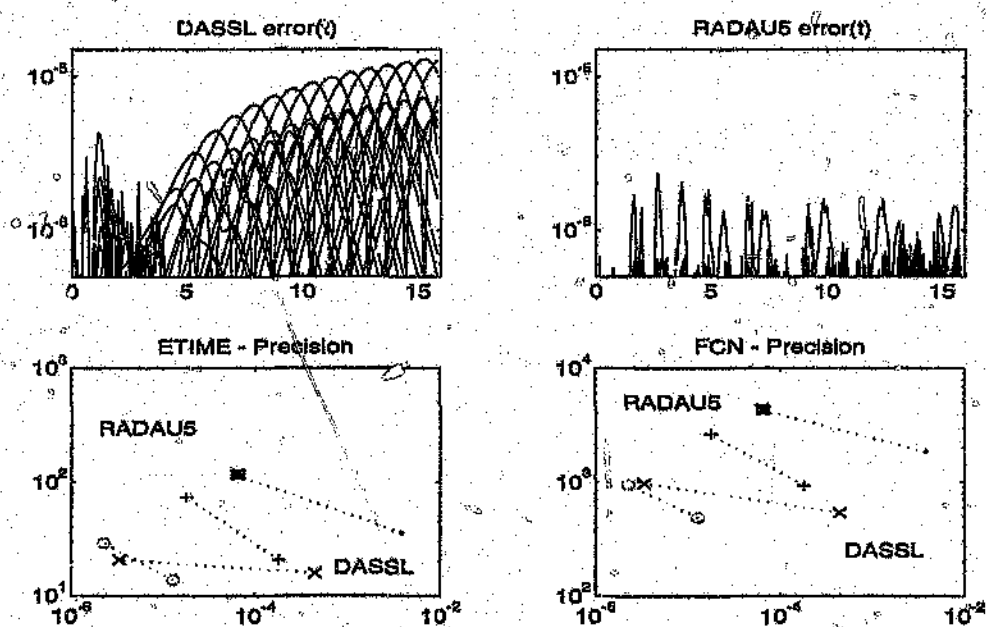


Figure 4.26: Global absolute errors for MOL Problem 1 and work-precision diagrams.

## 4.11 Small Contrived Problems

For this final experiment we solve numerically two linear variable-coefficient DAEs of theoretical interest [41] (cf. §2.1.2).

### 4.11.1 Irregular Solvable DAE

#### Formulation

System I below has a singular pencil for all  $t$ , and is thus irregular:

$$\begin{pmatrix} 1 & t \\ 0 & 0 \end{pmatrix} x' + \begin{pmatrix} 0 & 0 \\ 1 & t \end{pmatrix} x = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}. \quad \} I$$

A linear time-dependent transformation of variables yields the regular index 2 constant-coefficient DAE (Hessenberg of size 2) System II.

$$x = \begin{pmatrix} t & 1 \\ -1 & 0 \end{pmatrix} y \Rightarrow \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} y' + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} y = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}. \quad \} II$$

The unique true solution, easily found by index reduction, is

$$y = \begin{pmatrix} f_1 - f_2' \\ f_2 \end{pmatrix}, \quad \text{ie. } x = \begin{pmatrix} t(f_1 - f_2') + f_2 \\ -(f_1 - f_2') \end{pmatrix}.$$

#### Solution

We attempted to solve both Systems I and II (to within  $\text{TOL} = 10^{-4}$ ) numerically, with

$$\begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} 2t^2 \\ t+1 \end{pmatrix}, \quad t \in [-0.5, T].$$

(The choice of  $t_0 \neq 0$  was in order to retain generality that would have been lost for  $t_0 = 0$ .) Consistent i.c.'s were obtained from the true solution.

As expected for the irregular DAE System I, DASSL and LSODI failed at  $t_0$  due to repeated corrector convergence test failures. (System I is unsuitable for RADAU5 due to the time-varying mass matrix  $A$ .) The successful solution statistics for System II are given in Table 4.22. Note the high order of global accuracy obtained (up to the double precision used) and the remarkable ease of computation. RADAU5 used only one JAC evaluation since it solves the DAE in linear form and therefore its Jacobian in this case is constant. DASSL solves the fully implicit form and has a time-varying Jacobian, which therefore requires repeated evaluation. Of significance is the fact that DASSL solved the higher-index DAE without adjustment or undue effort. This is not unexpected, as Theorem 3.2.1 predicts no order loss in this case.

This problem verifies the claim made in §2.1.2 that an irregular DAE can be transformed to a regular one, and that these numerical methods used require regularity, in addition to solvability, for successful integration.

| T  | Code   | STEPS | FCN | JAC | Max. error in y(1) |               | Max. error in y(2) |               |
|----|--------|-------|-----|-----|--------------------|---------------|--------------------|---------------|
|    |        |       |     |     | abs                | rel           | abs                | rel           |
| 2  | DASSL  | 17    | 17  | 16  | $O(10^{-14})$      | $O(10^{-16})$ | $O(10^{-15})$      | $O(10^{-16})$ |
|    | RADAU5 | 9     | 36  | 1   | $O(10^{-15})$      | $O(10^{-16})$ | $O(10^{-15})$      | $O(10^{-16})$ |
| 20 | DASSL  | 19    | 19  | 18  | $O(10^{-13})$      | $O(10^{-16})$ | $O(10^{-15})$      | $O(10^{-16})$ |
|    | RADAU5 | 10    | 40  | 1   | $O(10^{-14})$      | $O(10^{-16})$ | $O(10^{-13})$      | $O(10^{-16})$ |

Table 4.22: Solution statistics for irregular solvable DAE System II

### 4.11.2 Variable Index DAE

#### Formulation

The equation

$$\begin{pmatrix} 0 & \phi \\ \psi & 0 \end{pmatrix} x' + x = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix},$$

$$\text{with } \phi(t) = \begin{cases} \phi_+(t) > 0 & \text{for } t > 0 \\ 0 & \text{for } t \leq 0 \end{cases}, \quad \psi(t) = \begin{cases} 0 & \text{for } t \geq 0 \\ \psi_-(t) > 0 & \text{for } t < 0 \end{cases},$$

and  $\phi$  and  $\psi$  infinitely differentiable, is solvable on  $(-\infty, \infty)$ . The pencil formed by the coefficients is nilpotent of index 2 if  $t \neq 0$  and of index 1 if  $t = 0$ . The unique solution is

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} f_1 - \phi f_2' \\ f_2 - \psi f_1' \end{pmatrix}.$$

Note that  $A$  changes rank at  $t = 0$  and changes nullspace as  $t$  crosses 0, so that no smooth coordinate changes  $P, Q$  can be found which transform the system to a semi-explicit DAE with explicit algebraic constraints.

#### Solution

We chose

$$f = (t^2, t^3)^T, \quad \phi_+(t) = \sin^3(t), \quad \psi_-(t) = \sin^4(t), \quad t \in [-0.5, 0.5],$$

and requested  $\text{tol} = 10^{-4}$ . Consistent i.c.'s were obtained from the true solution. Note that  $\phi$  is  $C^2$  and  $\psi$  is  $C^1$  (both not  $C^\infty$ ). RADAU5 is unsuitable for time-variant  $A$ , but DASSL solved the problem successfully without higher-index adjustment or computational difficulty, even when the index and nullspace change at  $t = 0$  (see Figures 4.27 and 4.28).

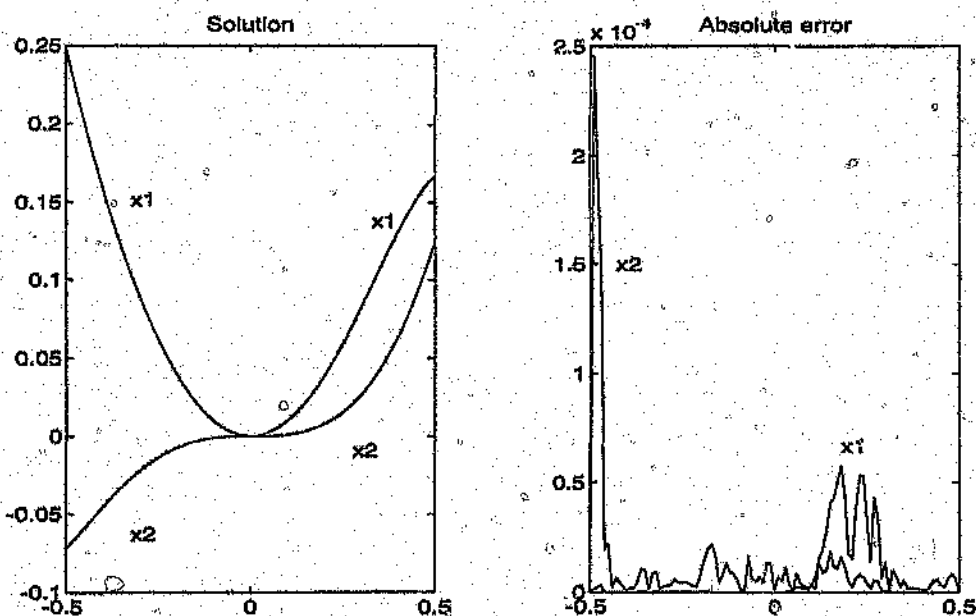


Figure 4.27: DASSL solution to variable index DAE.

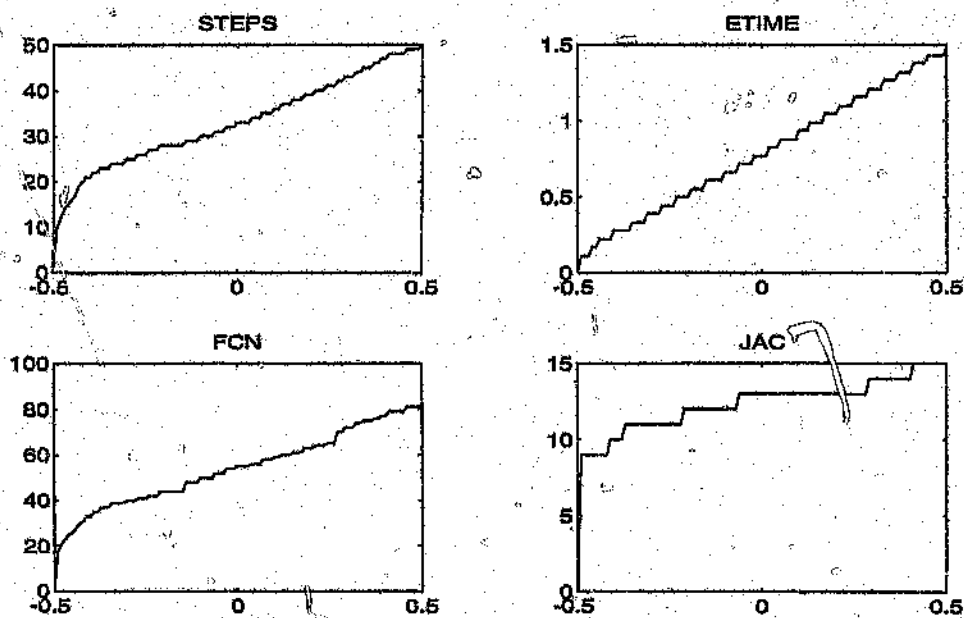


Figure 4.28: DASSL solution statistics for variable index DAE

## 4.12 Conclusion

### 4.12.1 Report on Techniques

#### Scaling

The steplength-based variable scaling and row scaling suggestions of §3.2.3 were largely unsuccessful, most likely since source code was not rewritten to include current steplength in TOL and RES routines at each step (only at each call in a multi-call mode). Although short output intervals were requested, resulting in frequent steplength reevaluation, this proved insufficient especially in the sensitive initial stage. Some variable scaling did result in successful integration but the obtained accuracy was usually intolerable, and the approach of scaling index  $m$  components by  $h^{1-m}$  (based on scaling algebraic variables by  $h^{-1}$  (§3.2.3)) is clearly inadequate. Row scaling was attempted for all higher-index problems, but with no notable improvement.

#### TOL for Precision

From the work-precision diagrams, it appears that LSODI and DASSL have roughly the same interpretation of the TOL request, with the obtained global accuracy falling behind TOL. In this respect, RADAU5 proved more reliable, with global accuracy confirming the TOL request, often to within an order of magnitude.

#### Code Comparison

Before presenting a conclusive report on relative code performance, we warn against exclusive use of results such as effective order, which, although providing an uncomplicated standard for comparison between methods and with theory, oversimplify the issue and presuppose robustness, which is often lacking, eg.

1. The high effective orders obtained by LSODI (eg. Figure 4.12) is overshadowed by the difficulty of obtaining a successful solution over a range of error tolerances (eg. Figure 4.20).
2. The concepts of effective order and work/precision ratios rely on the underlying linearity of work-precision curves, at least for some precision range. This was not always the case (eg. Figure 4.15).

### 4.12.2 Code Comparison

Table 4.23 summarizes the results of the experiments according to two basic criteria:

1. **Success:** Did the code solve the problem at all?
2. **Quality:** If so, how good was the solution?

The answer to (1) is determined by suitability and robustness. For (2), which depends primarily on accuracy, economy and predictability, the codes are listed (by initial) in decreasing order of 'quality', using '>' to denote preference. (An important omission here is that LSODI and, to some extent, DASSL were given information on the initial condition gained from RADAU5.) From Table 4.23 it is apparent that RADAU5 generally

| Problem                                           |       | Success |       |       | Quality |
|---------------------------------------------------|-------|---------|-------|-------|---------|
| Name                                              | Index | RADAU5  | DASSL | LSODI |         |
| 2-Transistor Amplifier                            | 1     | ✓       | ✓     | ×     | R>D     |
| 2-Phase Plug Flow                                 | 1     | ✓       | ✓     | ✓     | D>R>L   |
| Discharge Pressure Control                        | 2     | ✓       | ×     | ×     | R       |
| Plane Pendulum                                    | 1     | ✓       | ✓     | ✓     | D>R,L   |
|                                                   | >1    | ✓       | ✓     | ×     | R>D>L   |
| Van der Pol Oscillator<br>(or very stiff index 0) | 0     | ✓       | ✓     | ✓     | R,D,L   |
|                                                   | 1     | ✓       | ✓     | ✓     | R>D>L   |
| Ring Modulator<br>(or very stiff index 0)         | 0     | ✓       | ✓     | ✓     | R,D>L   |
|                                                   | 2     | ✓       | ✓     | ×     | R,D     |
| Cascade of Diff-Amps                              | 2     | ✓       | ✓     | ×     | R,D     |
|                                                   | 3     | ✓       | ×     | ×     | R,D     |
|                                                   | 4     | ✓       | ×     | ×     | R       |
| MOL Transmission Line                             | 0,1   | ✓       | ✓     | ×     | R>D     |

Table 4.23: Comparison of code performance

outperformed both LSODI and DASSL in these experiments, which, it may be argued, are biased towards RADAU5 as they derive mainly from literature presenting methods for solving linear or linearly implicit DAEs. In fact the constant-coefficient linearly implicit restriction is a severe one, and DASSL would be the obvious choice for any more general problem.

For index 0 problems without extreme stiffness, the three codes performed comparably, and presumably, would any decent stiff ODE solver.

RADAU5's main advantage over DASSL is its higher-index scaling, for which, unfortunately, there is theoretical support only for  $\nu \leq 2, 3$ . DASSL was successful for the index 2 ring modulator problem without any higher-index remedy, but this is an exception. For index 0 or 1 linearly implicit problems, no clear preference is given: sometimes DASSL was better, sometimes RADAU5, and sometimes it was too close to tell.

That LSODI was not designed to be robust on all index 1 linearly implicit DAEs is reflected in the results, and higher-index problems are clearly beyond its scope.

## Chapter 5

# Critique

In this dissertation, an attempt has been made to convey the entire process of solving initial value DAEs numerically, from the underlying theory to the development of numerical techniques and the implementation of available software for solving problems from a variety of application areas. It is hoped that it will prove useful as a guide to the field, requiring no other prerequisite than a graduate course in numerical analysis. We consider the achievements of the dissertation to be:

- Literature survey: incorporating, or at least citing, the main works by the major authors in the field, as well as subtle points and obscure methods.
- The observation that the unifying concept of a network formulation of physical systems is fundamentally suited to the solution of the resulting DAE.
- Experimentation: collection, formulation, solution and analysis of a variety of DAEs, not restricted to index 1, verifying and illustrating theoretical predictions.
- Software comparison: examining the suitability (with respect to applicability, accuracy and robustness) and economy of some of the available DAE codes, based on the above experiments.
- The observation that the general theory and available methods and software do not cater effectively for general higher index DAEs arising in applications, such as the index 3 Euler-Lagrange equations, the index 2 Navier-Stokes and (some) network equations and the arbitrarily high index DAEs possible with singular perturbations. Clearly much work is still to be done.
- The collection of different uses of the index, a concept which is only vaguely defined in much of the literature.

We see the major weakness of the dissertation to be that in seeking to provide an overview rather than focusing on a specific topic, no real groundbreaking work has been done. The analysis of the experiments of Chapter 4 could have been more thorough, and some problems were treated more fully than others. This is justified, as the primary objective

there was the illustration of DAE behaviour and the application of the methods and conceptual tools presented earlier, and not the successful solution of the problems tackled. Some work has been omitted for the sake of brevity, importantly:

- Proofs of some important results and theorems in Chapters 2 and 3. These are either standard results outside the scope of this dissertation, or results requiring a deeper understanding of specific subtopics within numerical DAE or ODE theory than is possible here. Some proofs are included, and these support the discussion without hindering it.
- The solution of the classic space shuttle problem [14, p.157] and other nonlinear DAEs.
- The use of other codes, such as LIMEX (extrapolation) and DAIS (consistent initialization).
- The solution of large problems and the use of parallelization. (A parallel version of DASSL is in preparation.)
- The boundary value DAE problem [6, 28, 60, 61]. We do mention that the code COLDAE, an extension of the code COLSYS/COLNEW, attempts to solve semi-explicit multipoint boundary-value DAEs of index at most 2, using projected collocation at Gaussian points with automatic mesh selection and error control. COLDAE was written by U. Ascher and R. Spiteri and is now available from NETLIB.
- Recent work on Taylor-type methods, which are still being developed and are not yet available for use, shows that these methods are very promising for the accurate solution to higher-index DAEs. Much of this dissertation has focused on the failure of general codes when solving higher-index problems, and this seems sure to be remedied in the near future.

## Appendix A

# Further Reading

Apart from the references cited in the text, the following related material was also read in preparation.

- **Section 1.1: Introduction**

- 1.1.1 [4, 21, 26, 27, 29, 52, 98] — Electrical circuits.
- 1.1.3 [13, 50, 53, 70, 93] — Control theory.

- **Chapter 2: Theory**

- 2.0 [85, 86] — Existence and uniqueness.

- **Chapter 3: Numerical Solution**

- 3.2 [3, 12, 53] — BDFs.
- 3.3 [88, 90, 92] — IRKs.
- 3.4 [62, 64, 74, 77] — Other methods.

## Appendix B

# Sample Fortran Driver

### B.1 Amplifier Driver Code

```
C -----
C --- DASSL DRIVER FOR PROBLEM 1: 2-TRANSISTOR AMPLIFIER ---
C -----
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON /parms/ ue,ub,uf,alpha,beta,pi,r0,r(9),c(5)
      INTEGER*2 ihr,imin,isec,i100th,ihr0,imin0,isec0,i100th0
C --- Declarations
      EXTERNAL res, jac
      PARAMETER (neq=8, maxord=5, ml=2, mu=1)
      PARAMETER (lrw=200)
      PARAMETER (liw=20+neq)
      DIMENSION y(neq), yprime(neq), rwork(lrw), iwork(liw)
      DIMENSION info(15)
C --- Set time0
      CALL GETTIM (ihr0, imin0, isec0, i100th0)
      DO 1 i=1,15
        1 info(i) = 0
C --- Output parameters
      OPEN(6,FILE='sample.out')
      OPEN(10,FILE='data.out')
      OPEN(8,FILE='sample.err')
C --- Solver parameters
C --- Multi- or single-call (multi)
      info(3) = 0
C --- Jacobian parameters
      iwork(1) = ml
      iwork(2) = mu
C --- Required Tolerance (for vectors: info(2)=1)
      tol = 1.d-5
      rtol = tol
      atol = tol
C --- Problem parameters
      pi = 3.14159265358979d0
      ue = 0.1d0
      ub = 6.0d0
      uf = 0.026d0
      alpha = 0.99d0
      beta = 1.0d-6
      r0 = 1.d3
      DO 10 i=1,9
```

```

10  r(1) = 9.d3
    DO 30 i=1,5
20  c(i) = i*1.d-6
C --- Initial conditions
    t = 0.d0
    y(1) = 0.d0
    y(2) = ub/(r(2)/r(1)+1.d0)
    y(3) = ub/(r(2)/r(1)+1.d0)
    y(4) = ub
    y(5) = ub/(r(5)/r(5)+1.d0)
    y(6) = ub/(r(5)/r(5)+1.d0)
    y(7) = ub
    y(8) = 0.d0
C --- The following were obtained analytically from the DAE at t0
    DO 30 i=1,neq
30  yprime(i) = 0.d0
    yprime(3) = -y(3)/(r(3)*c(2))
    yprime(6) = -y(6)/(r(7)*c(4))
    WRITE (10,*) neq+1
    nerr = 0
    ttimes = 1000
    tend = 0.2d0
    dtout = tend/ttimes
    tout = dtout
C --- Main loop
    DO 50 iout=1,ttimes
C --- Double-precision DASSL routine call
        CALL ddassl(res,neq,t,y,yprime,tout,info,rtol,atol,idid,
            rwork,lrw,iwork,liw,rpar,ipar,jac)
C --- Write output data
        WRITE (10,*) tout
        DO 40 i=1,neq
40      WRITE (10,*) y(i)
        WRITE (10,*) ue*DSIN(2.0d2*pi*tout)
        hu = rwork(7)
        nqu = iwork(8)
        WRITE (6,1010) t,y(8),ue*DSIN(2.d2*pi*tout),nqu,hu
        IF (idid.LT.0) GO TO 60
50      tout = tout + dtout
60  CONTINUE
C --- Write integration statistics
    IF (idid.LT.0) nerr = nerr + 1
    nst = iwork(11)
    nfe = iwork(12)
    nje = iwork(13)
    netf = iwork(14)
    nctf = iwork(15)
C --- Calculate elapsed time
    CALL GETTIM (ihr, imin, isec, i100th)
    etime = 60.d0*(60.d0*(ihr-ihr0)+(imin-imin0))+(isec-isec0)
        +(i100th-i100th0)/100.d0
    WRITE (6,1020) neq,nst,nfe,nje,nerr,netf,nctf,idid,tol,etime
190 CONTINUE
    IF (nerr.GT.0) THEN
        WRITE (6,1030)
    ELSE
        WRITE (6,1040)
    ENDIF

```

```

1010 FORMAT(1X,'t=',e10.5,' y(8)=',e12.5,' uet=',e12.5,
      ' order=',i1,' h=',e8.3)
1020 FORMAT(/,1X,' Integration statistics',/
      1X,' System size      = ',i8/
      1X,' Number of steps = ',i8/
      1X,' Number of F-s     = ',i8/
      1X,' Number of J-s     = ',i8/
      1X,' Error overrun    = ',i8/
      1X,' Number of etf-s   = ',i8/
      1X,' Number of ctf-s   = ',i8/
      1X,' Success          = ',i8/
      1X,' Error tolerance  = ',e10.5/
      1X,' Etime           = ',e10.5)
1030 FORMAT (/,' *****DDASSL PASSED ALL TESTS*****')
1040 FORMAT (/,' *****DDASSL FAILED SOME TESTS*****')
STOP
END

SUBROUTINE res(t,y,yprime,delta,ires,rpar,ipar)
IMPLICIT REAL*8(A-H,O-Z)
COMMON /parms/ ue,ub,uf,alpha,beta,pi,r0,r(9),c(6)
DIMENSION y(8),yprime(8),delta(8)
uet = ue*DSIN(2.0d2*pi*t)
f23 = beta*(DEXP((y(2)-y(3))/uf)-1.d0)
f56 = beta*(DEXP((y(5)-y(6))/uf)-1.d0)
C --- delta = f - k*y'
delta(1) = (y(1)/r0-uet/r0)/c(1)
      -(-yprime(1)+yprime(2))
delta(2) = ((1/r(1)+1/r(2))*y(2)-(alpha-1.d0)*f23-ub/r(2))/c(1)
      -(yprime(1)-yprime(2))
delta(3) = (f23-y(3)/r(3))/c(2)
      -(yprime(3))
delta(4) = (ub/r(4)-y(4)/r(4)-alpha*f23)/c(3)
      -(yprime(4)-yprime(5))
delta(5) = (-ub/r(6)+(1/r(5)+1/r(6))*y(5)-(alpha-1.d0)*f56)/c(3)
      -(yprime(4)-yprime(5))
delta(6) = (f56-y(6)/r(7))/c(4)
      -(yprime(6))
delta(7) = (ub/r(8)-y(7)/r(8)-alpha*f56)/c(5)
      -(yprime(7)-yprime(8))
delta(8) = (y(8)/r(9))/c(5)
      -(yprime(7)-yprime(8))
RETURN
END

SUBROUTINE jac(t,y,yprime,pd,cj,rpar,ipar)
IMPLICIT REAL*8(A-H,O-Z)
COMMON /parms/ ue,ub,uf,alpha,beta,pi,r0,r(9),c(6)
DIMENSION y(8),yprime(8),pd(6,8)
C --- Banded storage
C --- pd = dg/dy + cj*dg/dy' = df/dy - cj*A
fdash23 = beta*DEXP((y(2)-y(3))/uf)
fdash56 = beta*DEXP((y(5)-y(6))/uf)
pd(4,1) = 1/(r0*c(1))
      + cj
pd(3,2) =
      - cj
pd(5,1) =
      - cj
pd(4,2) = ((1/r(1)+1/r(2))-(alpha-1.d0)*fdash23)/c(1) + cj
pd(3,3) = (alpha-1.d0)*fdash23/c(1)

```

```

pd(5,2) = fdash23/c(2)
pd(4,3) = (-fdash23-1/r(3))/c(2)      - cj
pd(6,2) = -alpha*fdash23/c(3)
pd(5,3) = alpha*fdash23/c(3)
pd(4,4) = -1/(r(4)*c(3))              - cj
pd(3,5) =                               + cj
pd(5,4) =                               - cj
pd(4,5) = ((1/r(5)+1/r(6))-(alpha-1.d0)*fdash56)/c(3) + cj
pd(3,6) = (alpha-1.d0)*fdash56/c(3)
pd(5,5) = fdash56/c(4)
pd(4,6) = (-fdash56-1/r(7))/c(4)      - cj
pd(6,5) = -alpha*fdash56/c(5)
pd(5,6) = alpha*fdash56/c(5)
pd(4,7) = -1/(r(8)*c(5))              - cj
pd(3,8) =                               + cj
pd(5,7) =                               - cj
pd(4,8) = 1/(r(9)*c(5))                + cj
RETURN
END

```

## B.2 Amplifier Output File

```

t=.20000E-03 y(8)= -.75359E-03 uet= .12533E-01 order=3 h=.256E-04
t=.40000E-03 y(8)= .34486E-02 uet= .24869E-01 order=4 h=.256E-04
t=.60000E-03 y(8)= .44398E-02 uet= .36812E-01 order=5 h=.230E-04
t=.80000E-03 y(8)= .44374E-02 uet= .48175E-01 order=3 h=.154E-04
t=.10000E-02 y(8)= .44248E-02 uet= .58779E-01 order=2 h=.162E-04
t=.12000E-02 y(8)= .44150E-02 uet= .68455E-01 order=3 h=.308E-04
t=.14000E-02 y(8)= .44052E-02 uet= .77051E-01 order=4 h=.555E-04
t=.16000E-02 y(8)= .43954E-02 uet= .84433E-01 order=4 h=.499E-04
t=.18000E-02 y(8)= .43857E-02 uet= .90483E-01 order=5 h=.499E-04
t=.20000E-02 y(8)= .43760E-02 uet= .95106E-01 order=3 h=.499E-04
...
t=.19900E+00 y(8)= .12512E+01 uet= -.58779E-01 order=2 h=.888E-04
t=.19920E+00 y(8)= .12484E+01 uet= -.48175E-01 order=3 h=.888E-04
t=.19940E+00 y(8)= .12457E+01 uet= -.36812E-01 order=3 h=.888E-04
t=.19960E+00 y(8)= .12428E+01 uet= -.24869E-01 order=3 h=.888E-04
t=.19980E+00 y(8)= .12400E+01 uet= -.12533E-01 order=2 h=.799E-04
t=.20000E+00 y(8)= .12370E+01 uet= .16411E-08 order=2 h=.799E-04

```

### Integration Statistics

```

System size      =      8
Number of steps  =    6282
Number of F-s    =   15689
Number of J-s    =   4957
Err. pr overrun  =      0
Number of etf-s  =   2543
Number of ctf-s  =      0
Success          =      3
Error tolerance  = .10000E-04
Etime           = .43180E+02

```

\*\*\*\*\*DDASSL FAILED SOME TESTS\*\*\*\*\*

# Bibliography

- [1] R.C. AIKEN, ed., *Stiff Computation*, Oxford University Press, 1985.
- [2] R.K. ALEXANDER & J.J. COYLE, "Runge-Kutta methods and differential-algebraic systems," *SIAM J. Numer. Anal.*, v. 27, no. 3, 1990, pp. 735-752.
- [3] U. ASCHER, "On symmetric schemes and differential-algebraic equations," *SIAM J. Sci. Stat. Comput.*, v. 10, no. 5, 1989, pp. 937-949.
- [4] U. ASCHER, "On numerical differential algebraic problems with application to semiconductor device simulation," *SIAM J. Numer. Anal.*, v. 26, no. 3, 1989, pp. 517-538.
- [5] U. ASCHER & L.R. PETZOLD, "Projected implicit Runge-Kutta methods for differential-algebraic equations," *SIAM J. Numer. Anal.*, v. 28, no. 4, 1991, pp. 1097-1120.
- [6] U. ASCHER & L.R. PETZOLD, "Numerical methods for boundary value problems in differential-algebraic equations," *Recent Developments in Numerical Methods and Software for ODEs/DAEs/PDEs*, ed. G.D. Byrne & W.E. Schiesser, World Scientific, 1992, pp. 125-135.
- [7] G. BACIU, J.C.K. CHOU, & H.K. KESAVAN, "Constrained multibody systems: graph-theoretic Newton-Euler formulation," *IEEE Trans. Systems, Man and Cybernetics*, v. 20, no. 5, 1990, pp. 1025-1048.
- [8] A. BARRLUND, "Constrained least squares methods for linear timevarying DAE systems," *Numer. Math.*, v. 60, 1991, pp. 145-161.
- [9] A. BARRLUND & B. KÅGSTRÖM, "Analytical and numerical solutions to higher index linear variable coefficient DAE systems," *J. Comput. Appl. Math.*, v. 31, 1990, pp. 305-330.
- [10] A. BELLEN, C.W. GEAR & E. RUSSO, eds., "Numerical methods for ordinary differential equations," *Lecture Notes in Math.*, v. 1386, Springer-Verlag, 1988.
- [11] F.H. BRANIN JR. & K. HUSEYIN, eds., *Problem Analysis in Science and Engineering*, Academic Press, New York, 1977, pp. 1-112.

- [12] R.K. BRAYTON, F.G. GUSTAVSON & G.D. HACHTEL, "A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas," *Proc. IEEE*, v. 60, no. 72, pp. 98-108.
- [13] K.E. BRENNAN, "Numerical simulation of trajectory prescribed path problems by the backward differentiation formulas," *IEEE Trans. Automatic Control*, AC-31, no. 3, 1986, pp. 266-269.
- [14] K.E. BRENNAN, S.L. CAMPBELL & L.R. PETZOLD, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Elsevier Science Publishing Co., Inc., New York, 1989.
- [15] K.E. BRENNAN & B.E. ENGQUIST, "Backward differentiation approximations of nonlinear differential-algebraic systems," *Math. Comput.*, v. 51, no. 184, 1988, pp. 659-676.
- [16] K.E. BRENNAN & L.R. PETZOLD, "The numerical solution of higher index differential/algebraic equations by implicit methods," *SIAM J. Numer. Anal.*, v. 26, no. 4, 1989, pp. 976-996.
- [17] W.L. BROGAN, *Modern Control Theory*, Prentice-Hall, 1985.
- [18] A. BUIUM, "Differential-algebraic groups of finite dimension," *Lecture Notes in Math.*, v. 1506, Springer-Verlag, 1992.
- [19] K. BURRAGE & L.R. PETZOLD, "On order reduction for Runge-Kutta methods applied to differential/algebraic systems and to stiff systems of ODEs," *SIAM J. Numer. Anal.*, v. 27, no. 2, pp. 447-456.
- [20] G.D. BYRNE & A.C. HINDMARSH, "Stiff ODE solvers: a review of current and coming attractions," *J. Comput. Physics*, v. 70, 1987, pp. 1-62.
- [21] D.A. CALAHAN, *Computer-Aided Network Design*, rev. ed., McGraw-Hill, New York, 1972.
- [22] S.L. CAMPBELL, *Singular Systems of Differential Equations*, Pitman, London, 1980.
- [23] S.L. CAMPBELL, "The numerical solution of higher index linear time varying singular systems of differential equations," *SIAM J. Sci. Stat. Comput.*, v. 6, 1985, pp. 334-348.
- [24] S.L. CAMPBELL, "Least squares completions for nonlinear differential algebraic equations," *Num. Math.*, v. 65, no. 1, 1993, pp. 77-94.
- [25] S.L. CAMPBELL & W.J. TERRELL, "Observability of linear time-varying descriptor systems," *SIAM J. Matrix Anal. Appl.*, v. 12, no. 3, 1991, pp. 484-496.
- [26] L.O. CHUA & P-M LIN, *Computer Aided Analysis of Electronic Circuits*, Prentice-Hall, 1985.
- [27] L.O. CHUA, L.A. DESOER & E.S. KUH, *Linear and Nonlinear Circuits*, McGraw-Hill, 1987.

- [28] K.D. CLARK & L.R. PETZOLD, "Numerical solution of boundary value problems in differential-algebraic systems," *SIAM J. Sci. Stat. Comput.*, v. 10, no. 5, 1989, pp. 915-936.
- [29] L.A. DESOER & E.S. KUH, *Basic Circuit Theory*, McGraw-Hill, 1969.
- [30] P. DEUFLHARD, E. HAIRER & J. ZUGCK, "One-step and extrapolation methods for differential-algebraic systems," *Numer. Math.*, v. 51, 1987, pp. 501-516.
- [31] S.R. DUNBAR & H.C. OTHMER, "On a nonlinear hyperbolic equation describing transmission lines, cell movement, and branching random walks," *Nonlinear Oscillations in Biology and Chemistry, Lecture Notes in Biomathematics*, v. 66, Springer-Verlag, pp. 274-289.
- [32] W.H. ENRIGHT, T.E. HULL & B. LINDBERG, "Comparing numerical methods for stiff systems of ODEs," *BIT*, v. 15, 1975, pp. 10-48.
- [33] R. FRANK, J. SCHNEID & C.W. UEBERHUBER, "Order results for implicit Runge-Kutta methods applied to stiff systems," *SIAM J. Numer. Anal.*, v. 22, 1985, pp. 515-534.
- [34] C. FÜHRER & B.J. LEIMKUHLER, "Numerical solution of differential-algebraic equations for constrained mechanical motion," *Numer. Math.*, v. 59, 1989, pp. 55-69.
- [35] F.R. GANTMACHER, *The Theory of Matrices II*, Chelsea, New York, 1960.
- [36] C.W. GEAR, "Simultaneous solution of differential-algebraic equations," *IEEE Trans. Circuit Theory*, CT-18, 1971, pp. 89-95.
- [37] C.W. GEAR, "Maintaining solution invariants in the numerical solution of ODEs," *SIAM J. Sci. Stat. Comput.*, v. 7, 1986, pp. 734-743.
- [38] C.W. GEAR, "Differential-algebraic equation index transformations," *SIAM J. Sci. Stat. Comput.*, v. 9, 1988, pp. 39-47.
- [39] C.W. GEAR, "DAEs: ODEs with constraints and invariants," Collection: Numerical methods for ordinary differential equations (L'Aquila, 1987), *Lecture Notes in Math.*, v. 1386, Springer-Verlag, 1988, pp. 54-68.
- [40] C.W. GEAR, "Differential algebraic equations, indices and integral algebraic equations," *SIAM J. Numer. Anal.*, v. 27, no. 6, 1990, pp. 1527-1534.
- [41] C.W. GEAR & L.R. PETZOLD, "Differential/algebraic systems and matrix pencils," *Lecture Notes in Math.*, v. 973, Springer-Verlag, 1983, pp. 75-89.
- [42] C.W. GEAR & L.R. PETZOLD, "ODE methods for the solution of differential/algebraic systems," *SIAM J. Numer. Anal.*, v. 21, no. 4, 1984, pp. 367-384.
- [43] C.W. GEAR, G.K. GUPTA & B. LEIMKUHLER, "Automatic integration of Euler-Lagrange equations with constraints," *J. Comp. Appl. Math.*, v. 12/13, 1985, pp. 77-90.

- [44] G.G. GUNDERSEN & I. LAINE, "Existence of meromorphic solutions of algebraic differential equations," *Mathematica Scandinavica*, v. 67, no. 1, 1990, pp. 35-55.
- [45] G.D. HACHTEL, R.K. BRAYTON & F.G. GUSTAVSON, "The sparse tableau approach to network analysis and design," *IEEE Trans. Circuit Theory*, CT-18, 1971, pp. 101-113.
- [46] E. HAIRER, CH. LUBICH & M. ROCHE, "Error of Runge-Kutta methods for stiff problems studied via differential algebraic equations," *BIT*, v. 28, 1988, pp. 678-700.
- [47] E. HAIRER, CH. LUBICH & M. ROCHE, "The numerical solution of differential-algebraic systems by Runge-Kutta methods," *Lecture Notes in Math.*, v. 1409, Springer-Verlag, 1989.
- [48] E. HAIRER & G. WANNER, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics, v. 14, Springer-Verlag, 1991.
- [49] A.C. HINDMARSH, "LSODE and LSODI, two new initial value ordinary differential equation solvers," *ACM-SIGNUM Newsletters*, 15, 1980, pp. 10-11.
- [50] D. HINRICHSSEN & J. O'HALLORAN, "A complete characterization of orbit closures of controllable singular systems under restricted system equivalence," *SIAM J. Control and Optimization*, v. 28, no. 3, 1990, pp. 602-623.
- [51] C-W HO, A.E. RUEHLI & P.A. BRENNAN, "The modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems*, CAS-22, no. 6, 1975, p. 504-509.
- [52] J. KATZELNELSON, "AEDNET — Simulator for nonlinear networks," *Proc. IEEE*, v. 54, 1966, p. 1536.
- [53] J.B. KEIPER & C.W. GEAR, "The analysis of generalized backward difference formula methods applied to Hessenberg form differential-algebraic equations," *SIAM J. Numer. Anal.*, v. 28, no. 3, 1991, pp. 833-858.
- [54] P. KUNKEL & V.L. MEHRMANN, "Numerical solution of Riccati differential algebraic equations," *Lin. Alg. Appl.*, v. 137/138, 1990, pp. 39-66.
- [55] P. KUNKEL & V.L. MEHRMANN, "A new class of discretization methods for the solution of linear differential-algebraic equations," *Materialien LXII, FSP Mathematisierung*, Universität Bielefeld, 1992.
- [56] P. KUNKEL & V.L. MEHRMANN, "Canonical forms for linear differential-algebraic equations with variable coefficients," *J. Comput. Appl. Math.*, (to appear).
- [57] P. KUNKEL & V.L. MEHRMANN, "A new look at matrix pencils," *Lin. Alg. Appl.*, (to appear).
- [58] J.D. LAMBERT, *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*, J. Wiley, 1991.

- [59] B. LEIMKUHLER, L.R. PETZOLD & C.W. GEAR, "Approximation methods for the consistent initialization of differential-algebraic equations," *SIAM J. Numer. Anal.*, V. 28, no. 1, 1991, pp. 205-226.
- [60] M. LENTINI & R. MÄRZ, "The conditioning of boundary value problems in transferable differential-algebraic equations," *SIAM J. Numer. Anal.*, v. 27, no. 4, 1990, pp. 1001-1015.
- [61] M. LENTINI & R. MÄRZ, "Conditioning and dichotomy in differential algebraic equations," *SIAM J. Numer. Anal.*, v. 27, no. 6, 1990, pp. 1519-1526.
- [62] M-Y LIN LEE, "Estimation of the error in the reduced basis method solution of differential algebraic equation systems," *SIAM J. Numer. Anal.*, v. 28, no. 2, 1991, pp. 512-528.
- [63] W. LINIGER, "Multistep and one-leg methods for implicit mixed differential algebraic systems," *IEEE Trans. Circuits and Systems*, CAS-26, 1979, p. 755.
- [64] L. LOPEZ, "One-step collocation methods for differential-algebraic systems of index 1," *J. Comp. Appl. Math.*, v. 29, 1990, pp. 145-159.
- [65] P. LÖTSTEDT, "Mechanical systems of rigid bodies subject to unilateral constraints," *SIAM J. Appl. Math.*, v. 42, 1982, pp. 281-296.
- [66] P. LÖTSTEDT & L.R. PETZOLD, "Numerical solution of nonlinear differential equations with algebraic constraints I: convergence results for backward differentiation formulas," *Math. Comput.*, v. 46, 1986, pp. 491-516.
- [67] CH. LUBICH, "Linearly implicit extrapolation methods for differential-algebraic equations," *Numer. Math.*, v. 55, 1989, pp. 197-211.
- [68] CH. LUBICH, "On projected Runge-Kutta methods for differential-algebraic equations," *BIT*, v. 31, no. 3, 1991, pp. 545-550.
- [69] CH. LUBICH & M. ROCHE, "Rosenbrock methods for differential-algebraic systems with solution-dependent singular matrix multiplying the derivative," *Computing*, v. 43, no. 4, 1990, pp. 325-342.
- [70] D. LUENBERGER, "Dynamic equations in descriptor form," *IEEE Trans. Automatic Control*, AC-22, 1977, pp. 312-321.
- [71] M.S. MAHMOUD, M.F. HASSAN & M.G. DARWISH, *Large-Scale Control Systems: Theories and Techniques*, M. Dekker, 1985.
- [72] R.E. O'MALLEY, JR., "On nonlinear singularly perturbed initial value problems," *SIAM Rev.*, v. 30, 1988, pp. 193-212.
- [73] R.E. O'MALLEY, JR., "Singular perturbations and optimal control," *Mathematical Control Theory*, ed. W.A. Coppel, Springer-Verlag, v. 680, 1978, pp. 170-218.

- [74] R. MÄRZ, "On initial value problems in differential-algebraic equations and their numerical treatment," *Computing*, v. 35, 1985, pp. 13-37.
- [75] R. MÄRZ, "On one-leg methods for differential-algebraic equations," *Circuits, Systems and Signal Processing*, v. 5, no. 1, 1986, pp. 87-95.
- [76] R. MÄRZ, "Some new results concerning index-3 differential-algebraic equations," *J. Math. Anal. Appl.*, v. 140, no. 1, 1989, pp. 177-199.
- [77] U. MIEKKALA, "Dynamic iteration methods applied to linear DAE systems," *J. Comp. Appl. Math.*, v. 25, 1989, pp. 133-151.
- [78] A. OSTERMANN, "A half-explicit extrapolation method for differential-algebraic systems of index 3," *IMA J. Numer. Anal.*, v. 10, no. 2, 1990, pp. 171-180.
- [79] C.C. PANTELIDES, "The consistent initialization of differential-algebraic systems," *SIAM Sci. Stat. Comput.*, v. 9, 1988, pp. 213-231.
- [80] L.R. PETZOLD, "Differential/algebraic equations are not ODEs," *SIAM J. Sci. Stat. Comput.*, v. 3, no. 3, 1982, pp. 367-384.
- [81] L.R. PETZOLD, "Order results for implicit Runge-Kutta methods applied to differential/algebraic systems," *SIAM J. Numer. Anal.*, v. 23, 1986, pp. 837-852.
- [82] L.R. PETZOLD & P. LÖTSTEDT, "Numerical solution of nonlinear differential equations with algebraic constraints II: practical implications," *SIAM J. Sci. Stat. Comput.*, v. 7, 1986, pp. 720-733.
- [83] S. REICH, "On a geometrical interpretation of differential-algebraic equations," *Circuits, Systems and Signal Processing*, v. 9, no. 4, 1990, pp. 367-382.
- [84] P. RENTROP, M. ROCHE & G. STEINEBACH, "The application of Rosenbrock-Wanner type methods with stepsize control in differential-algebraic equations," *Numer. Math.*, v. 55, 1989, pp. 545-563.
- [85] W.C. RHEINOLDT, "Differential-algebraic systems as differential equations on manifolds," *Math. Comp.*, v. 43, 1984, pp. 473-482.
- [86] W.C. RHEINOLDT, "On the existence and uniqueness of solutions of nonlinear semi-implicit differential-algebraic equations," *Nonlinear Analysis, Theory, Methods & Applications*, v. 16, no. 7/8, 1991, pp. 647-661.
- [87] M. ROCHE, "Rosenbrock methods for differential algebraic equations," *Numer. Math.*, v. 52, 1988, pp. 45-63.
- [88] M. ROCHE, "Implicit Runge-Kutta methods for differential algebraic equations," *SIAM J. Numer. Anal.*, v. 26, no. 4, pp. 963-975.
- [89] S.S. SASTRY & C.A. DESOER, "Jump behaviour of circuits and systems," *IEEE Trans. Circuits and Systems*, CAS-28, 1981, pp. 1109-1124.

- [90] C. SCHNEIDER, "Rosenbrock-type methods adapted to differential-algebraic systems," *Math. Comp.*, v. 56, no. 193, 1991, pp. 201-213.
- [91] R. SCOTT, "The electrophysics of a nerve fiber," *Rev. Mod. Phys.*, v. 47, 1975, pp. 487-533.
- [92] L.F. SHAMPINE, "Implementation of implicit formulas for the solution of ODE's," *SIAM J. Sci. Stat. Comput.*, v. 1, 1980, pp. 103-118.
- [93] R.F. SINCOVEC, A.M. ERISMAN, E.L. YIP & M.A. EPTON, "Analysis of descriptor systems using numerical algorithms," *IEEE Trans. Automatic Control*, AC-26, 1981, pp. 139-147.
- [94] D.R. SMITH, *Singular-Perturbation Theory: An Introduction with Applications*, Cambridge University Press, 1985.
- [95] K. STREHMEL, R. WEINER & I. DANNEHL, "On error behaviour of partitioned linearly implicit Runge-Kutta methods for stiff and differential algebraic systems," *BIT*, v. 30, 1990, pp. 338-375.
- [96] F. TAKENS, "Constrained differential equations," *Dynamical Systems — Warwick 1974*, Springer-Verlag, v. 468, 1975, pp. 80-82.
- [97] J. WITTENBURG, *Dynamics of Systems of Rigid Bodies*, Teubner, Stuttgart, 1977.
- [98] M. YANILMAZ & V. EVELEIGH, "Numerical device modelling for electronic circuit simulation," *IEEE Trans. Computer-aided Design*, v. 10, no. 3, 1991, p. 366.





**Author: Aspoas Michael A.**

**Name of thesis: On the numerical solution of differential-algebraic equations.**

***PUBLISHER:***

**University of the Witwatersrand, Johannesburg**

**©2015**

***LEGALNOTICES:***

**Copyright Notice:** All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed or otherwise published in any format, without the prior written permission of the copyright owner.

**Disclaimer and Terms of Use:** Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page) for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.