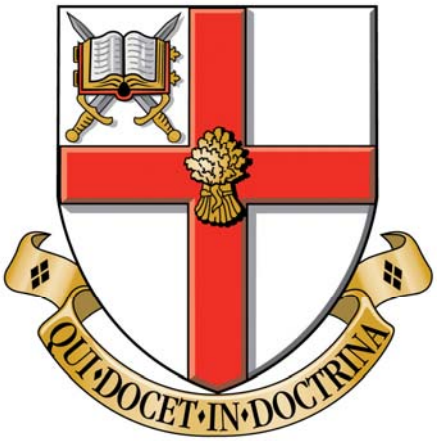


University of Chester

A REVIEW OF THE METHODS FOR THE SOLUTION OF DAES

PATRICIA MARY LUMB

Dissertation submitted to Chester College for the Degree of Master of Science (Mathematics) in part fulfilment of the award.

FOUR MODULE DISSERTATION

JUNE 1999

ABSTRACT

Differential-Algebraic equation systems (DAEs) occur in a variety of applications in science and engineering and interest in them has grown considerably during the latter part of the twentieth century.

The aims of this thesis are:-

- to provide the interested reader with a comprehensible and informative introduction to DAEs, whilst assuming no prior knowledge of the subject area.
- to give an indication to the reader with a DAE to solve whether or not they can hope to be successful.
- to introduce the reader to possible methods of solution (either analytical, numerical or both)

This work is original and has not been submitted previously in support of any qualification or course.

ACKNOWLEDGEMENTS

I would like to thank Dr N. J. Ford for his guidance and encouragement during the preparation and writing of this thesis.

CONTENTS

PREFACE

The scientist, engineer and the industrial or academic mathematician are likely to have acquired some knowledge of both differential and algebraic equations through the mathematical components of their education. However, they are less likely to have encountered DAEs either during their mathematical education or in their scientific or engineering training.

In Section 1 we introduce the reader to the basic types of DAE, to the Hessenberg form of a DAE and to the important concept of the index. We highlight similarities and differences between ODEs and DAEs and give examples of DAE applications. Section 2 focuses on the solution of a DAE. After explaining what we mean by a solvable DAE we move on to consider which types of DAEs can be solved using established techniques. We indicate how particular classes of DAEs can be solved analytically and briefly introduce numerical approaches to solving DAEs. The purpose of Section 3 is to provide the reader with the information about numerical methods applied to ODEs which is relevant to the material which we cover in Section 4. In the latter we give information about which numerical method is appropriate for which DAE type, including comments on some of the problems encountered in their solution, and also we include a brief introduction to the computer codes available and their use. In Section 5 we provide further evidence of the applications of DAEs. This is likely to be of particular interest to the academic mathematician who is seeking a justification for the introduction of, or the interest in, the DAE, as well as to all readers interested in the difficulties encountered in their solution.

We provide references to the source material throughout the thesis and include examples of DAEs as an aid to understanding the theory.

SECTION 1: INTRODUCING DIFFERENTIAL-ALGEBRAIC EQUATIONS

1.1 WHY DO WE NEED TO INTRODUCE DIFFERENTIAL-ALGEBRAIC EQUATIONS(DAEs)?

We begin by considering the first order system

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

where $\mathbf{y}$, $\mathbf{y}'$ and $\mathbf{F}$ are n-dimensional vectors and $\mathbf{F}$ is assumed to be sufficiently smooth.

If $\dfrac{\partial \mathbf{F}}{\partial \mathbf{y}'}$ is non-singular then the system can, in principle, be written in the explicit form

$$\mathbf{y}' = \mathbf{F}(t,\mathbf{y}(t)) \tag{1.2}$$

to produce a system of ordinary differential equations, (ODEs), to which we can apply standard ODE theory and numerical techniques.

Rewriting the system in this form is not always possible and in an increasing number of applications preference is being given to working directly with the system (1.1).

1.2 INTRODUCING DIFFERENTIAL-ALGEBRAIC EQUATIONS

Marz in [12], after describing DAEs as "singular ODEs", introduces them as "special implicit ordinary differential equations (ODE) $\mathbf{f}(\mathbf{x}'(t), \mathbf{x}(t), t) = \mathbf{0}$ where the partial Jacobian $\mathbf{f}_{\mathbf{y}'}(\mathbf{y},\mathbf{x},t)$ is singular for all values of its arguments."

In [4], [18] and [19] DAEs are introduced as systems of the form

$$\mathbf{0} = \mathbf{F}(t,\mathbf{y},\mathbf{y}') \quad \text{where } \frac{\partial \mathbf{F}}{\partial \mathbf{y}'} \text{ may be singular.}$$

We recall that the Jacobian Matrix can be expressed in the form $\mathbf{g}_{\mathbf{x}}$ or $\dfrac{\partial\, \mathbf{g}}{\partial\, \mathbf{x}}$ and that it is defined by

$$\left(\frac{\partial g}{\partial x}\right)_{\mathrm{i,j}} = (\mathrm{g_x})_{\mathrm{i,j}} = \frac{\partial \mathrm{g}_i}{\partial \mathrm{x}_j} \quad \text{for } 1 \le \mathrm{i} \le \mathrm{n},\ 1 \le \mathrm{j} \le \mathrm{k} \tag{1.3}$$

In an expanded form this can be written as

$$\frac{\partial \mathbf{g}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_k} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & & \frac{\partial g_2}{\partial x_k} \\ \vdots & & & \vdots \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} & \cdots & \frac{\partial g_n}{\partial x_k} \end{pmatrix} \qquad (1.4)$$

A DAE combines differential equations with algebraic equations, and we illustrate this in the following example.

Example 1.1

In [4] Newton's equations of motion for a simple, rigid pendulum of length L are given in the form

$$\ddot{x} = \lambda x \qquad (1.5a)$$

$$\ddot{y} = \lambda y - g \qquad (1.5b)$$

$$0 = x^2 + y^2 - L^2 \qquad (1.5c)$$

We observe that the first two equations are differential equations and that the third is an algebraic equation, resulting from a constraint on the variables x and y.

DAEs arise naturally in many applications and have been known by a variety of names, depending on the area of application. They may be known, for example, as descriptor systems (e.g. in circuit analysis), generalised state space (e.g. in system theory), constrained systems, reduced order model and non-standard systems. In the simulation of a physical problem the model often takes the form of a DAE, consisting of a collection of relationships between the variables involved and some of their derivatives (as in equation (1.5)). Previously, the solution of DAEs has involved a reformulation of the system as an explicit ordinary differential equation (ODE) followed by the use of one of the many software packages available. However, if a direct solution of the DAE is possible then the scientist or engineer can more readily explore the effect of modelling changes and parameter variation. (Changing parameter values can alter the relationship between variables and require different explicit models with solution manifolds of different dimensions. The variables

usually have a physical significance, hence changing the model to ODEs may produce less meaningful variables). In addition the need to accommodate boundary conditions and the possibility of simplifying the underlying problem are avoided, and it may be possible to exploit the system structure. In the study of more complex systems a direct solution will avoid the need to repeat the reduction of a DAE to an ODE when the model is altered, thus speeding up the solution. A direct solution of the DAE also makes it easier to interface modelling software with design software.

1.3: BASIC TYPES OF DAEs

1.3.1 LINEAR CONSTANT COEFFICIENT DAEs are regarded as the best understood class of DAEs, (see [4]), and consist of a system of the form

$$A\mathbf{x}'(t) + B\mathbf{x}(t) = \mathbf{f}(t) \qquad (1.6)$$

where A, B are square matrices of real or complex numbers, A singular, and t is a real variable. We note that if A is nonsingular then the equation is an ODE, since we are able to find A^{-1}, multiply throughout by A^{-1} and solve the resulting equation by ODE methods. The authors of [4] assume that the vectors are real for notational convenience but state that the results are the same for the complex case.

Example 1.2

The system
$$x_2' - 2x_1 = 0 \qquad (1.7a)$$
$$3x_2' - 6x_1 + x_2 = t^2 \qquad (1.7b)$$

is a linear, constant coefficient DAE which can be written in the form (1.6) with

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 3 \end{pmatrix}, B = \begin{pmatrix} -2 & 0 \\ -6 & 1 \end{pmatrix} \text{ and } \mathbf{f}(t) = \begin{pmatrix} 0 \\ t^2 \end{pmatrix}$$

The solution to the system (1.7) is $x_1 = t$, $x_2 = t^2$.

1.3.2 LINEAR TIME-VARYING DAEs are of the form

$$A(t)\,\mathbf{x}'(t) + B(t)\,\mathbf{x}(t) = \mathbf{f}(t) \qquad (1.8)$$

with A(t) singular for all t

Most of the established mathematical models seen by the authors of [4] have led to either linear constant coefficient or nonlinear DAEs. However, although mathematical models leading to linear time-varying DAEs have also been observed by the same authors, a more important feature of DAE systems of the form (1.8) is that they "exhibit much of the behaviour which distinguishes general DAEs from linear constant coefficient DAEs." Up to the time when [4] was written some techniques and results which seem appropriate for nonlinear systems, (in that they correctly predict some of their behaviour), had only been completely and rigorously proved for linear time-varying DAEs. The latter type have already proved to be important by improving the understanding of more general systems and are likely to "play an even greater role in future analytical and numerical work with nonlinear DAEs" [4].

Example 1.3 The DAE system

$$\begin{aligned} tx_1' + x_2' &= t^2 \\ tx_1 + x_2 &= t \end{aligned} \qquad (1.9)$$

is of the form (1.8) with $A = \begin{pmatrix} t & 1 \\ 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 & 0 \\ t & 1 \end{pmatrix} and\ \mathbf{f}(t) = \begin{pmatrix} t^2 \\ t \end{pmatrix}$.

The solution of system (1.9) is $\mathbf{x} = \left((1-t^2), t^3\right)^{\mathrm{T}}$.

The semi-explicit linear DAE of the form

$$x_1'(t) + B_{11}(t)x_1(t) + B_{12}(t)x_2(t) = f_1(t) \qquad (1.10a)$$

$$B_{21}(t)x_1(t) + B_{22}(t)x_2(t) = f_2(t) \qquad (1.10b)$$

is a special case of (1.8) with $A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$.

1.3.3 GENERAL (OR FULLY IMPLICIT) NONLINEAR DAEs *are of the form*

$$\mathbf{F}(t, \mathbf{y}(t), \mathbf{y}'(t)) = 0 \qquad (1.11)$$

where **F** is nonlinear. If (1.11) is linear in the derivative

i.e. $A(t, \mathbf{y}(t))\mathbf{y}'(t) + \mathbf{f}(t, \mathbf{y}(t)) = \mathbf{0}$

then the system may be referred to as linearly implicit.

1.3.4 SEMI-EXPLICIT NONLINEAR DAEs are of the form

$$\begin{aligned} \mathbf{x}'(t) &= \mathbf{f}(t,\mathbf{x},\mathbf{y}) \\ \mathbf{0} &= \mathbf{g}(t,\mathbf{x},\mathbf{z}) \end{aligned} \tag{1.12}$$

DAEs in the form (1.12) can also be described as an ODE with constraints or they may be considered as a special case of (1.11). We note that in some applications a system in the form

$$\mathbf{F}(\mathbf{x}'(t),\mathbf{x}(t),\mathbf{y}(t),t) = \mathbf{0} \tag{1.13a}$$

$$\mathbf{G}(\mathbf{x}(t),\mathbf{y}(t),t) = \mathbf{0} \tag{1.13b}$$

where $\mathbf{F}_{\mathbf{x}'}$ is nonsingular, may be referred to as semi-explicit.

Constant coordinate changes (i.e. not time dependent) are permitted [4] and a fully-implicit linear constant coefficient DAE can always be transformed to a semi-explicit linear constant coefficient DAE by a constant coordinate change.

1.4 THE INDEX OF A DAE

If we apply analytical differentiations to a given system, and eliminate where needed, the process will yield an explicit ODE system for all the unknowns unless the problem is singular. In [2] the index of the DAE is stated to be the number of differentiations for this transformation to take place. We note that this process avoids performing numerical differentiations by differentiating analytically, and that the index referred to here is also known as the differential index, to distinguish it from the perturbation index which is defined in [9] and which is motivated by the loss of smoothness in solutions to higher-index DAEs.

Definition 1.1 In [19] we find the following definition for the index of $\mathbf{F}(t,\mathbf{y},\mathbf{y}') = \mathbf{0}$.

The index of a solvable DAE is the smallest nonnegative integer m such that **F** has m continuous derivatives and the nonlinear system

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

$$\frac{d}{dt}\mathbf{F}(t,\mathbf{y},\mathbf{y}',\mathbf{y}'') = \frac{\partial \mathbf{F}}{\partial \mathbf{y}}\mathbf{y}' + \frac{\partial \mathbf{F}}{\partial \mathbf{y}'}\mathbf{y}'' + \frac{\partial \mathbf{F}}{\partial \mathbf{t}} = \mathbf{0}$$

. .

. .

$$\frac{d^m}{dt^m}\mathbf{F}(t,\mathbf{y},\mathbf{y}',\mathbf{y}'',\ldots,\mathbf{y}^{(m+1)}) = \mathbf{0}$$

can be solved pointwise for $\mathbf{y}'$ uniquely in terms of **y** and t: $\mathbf{y}' = \phi(\mathbf{y},t)$.

Example 1.4

Consider $\mathbf{y} = \mathbf{f}(t)$ (1.14)

If we differentiate (1.14) once we obtain $\mathbf{y}' = \mathbf{f}'(t)$ which is an ODE for **y**. One differentiation was needed $\therefore$ index = 1.

Example 1.5

Consider $y_1 = f(t)$ (1.15a)

$y_2 = y_1'$ (1.15b)

$y_3 = y_2'$ (1.15c)

Differentiating (1.15a) gives $y_1' = f'(t)$

.. (1.15b) .. $y_2' = y_1'' \Rightarrow y_2' = f''(t)$

.. (1.15c) .. $y_3' = y_2'' \Rightarrow y_3' = f'''(t)$

Three differentiations are needed to obtain

$$\mathbf{y}' = (y_1', y_2', y_3')^T = \left(f'(t), f''(t), f'''(t)\right)^T$$

$\therefore$ Index of (1.15) is three.

The role of the index has been crucial in understanding the structure of the DAE, although differentiation of the system is rarely carried out in practice. It is a rough measure of the degree of singularity in the system, and in general the higher the index the more difficult the problem is numerically. We note that DAEs with an index greater than 1 are often referred to as higher-index DAEs, and that the index of an ODE is zero.

The index of a DAE is not only dependent upon the form of the DAE but also on the solution and on the initial conditions, which must be consistent i.e. must satisfy the constraints of the system. (More information about consistent initial conditions will be given in section 4.4). Higher index DAEs may include some hidden constraints. The following examples are included to illustrate these points.

Example 1.6 To illustrate the dependence of the index on the solution we will consider the DAE system

$$y_2' = y_1 \qquad (1.16a)$$

$$0 = (y_3 - 1)(2 - y_3) \qquad (1.16b)$$

$$0 = y_2(y_3 - 1) + y_1(2 - y_3) - t^2 \qquad (1.16c)$$

(1.16b) yields two solutions for $y_3(t)$, namely $y_3 = 1$ or $y_3 = 2$.

We will first consider $y_3 = 2$, and substitute this value into (1.16c) to obtain

$$0 = y_2 - t^2$$

$\Rightarrow$ $y_2 = t^2$ (1.17)

Using (1.16a) we find $y_1 = 2t$, giving the solution to the DAE as $\mathbf{y}(t) = \begin{pmatrix} 2t \\ t^2 \\ 2 \end{pmatrix}$.

In this case we find that the index of the DAE is 2 since we need to differentiate (1.17) to obtain $y_2'(t) = 2t$, and also (1.16a) to obtain $y_1'(t) = 2$.

We now consider the case when $y_3 = 1$.

(1.16c) becomes $0 = y_1 - t^2$

$$\therefore \quad y_1 = t^2 \tag{1.18}$$

Using $y_1 = t^2$ in (1.16a) gives

$$y_2' = t^2 \tag{1.19}$$

Integration of (1.19) gives $y_2(t) = \frac{t^3}{3} + y_2(0)$.

In this case the solution to the DAE is $\mathbf{y}(t) = \left(t^2, \frac{t^3}{3} + y_2(0), 1\right)^{\mathrm{T}}$

The index in this case is 1 since the only differentiation required is that of (1.18) to obtain $y_1'(t)$. We note the need for an initial value, (or other value of y at a given t), in the case $y_3 = 1$.

Example 1.7 To illustrate how a higher-index DAE may include some hidden constraints we consider the DAE

$$y_1(t) = t^3 + t + 2 \tag{1.20a}$$

$$y_2(t) = y_1' \tag{1.20b}$$

The index of the system can be seen to be 2. Differentiation of (1.20a) leads to

$$y_1' = 3t^2 + 1 \tag{1.21}$$

Hence, in addition to the requirement to satisfy $y_1(0) = 2$ and $y_2(0) = y_1'(0)$ we need to satisfy $y_1'(0) = 1$.

Example 1.8 This example illustrates the dependence of the index on the initial conditions.

$$y_2' = y_1 \tag{1.22a}$$

$$y_3' = 0 \tag{1.22b}$$

$$0 = y_2(y_3 - 1) + y_1(2 - y_3) - t^2 \tag{1.22c}$$

(1.22b) $\Rightarrow$ $y_3(t)$ is constant, say $y_3(t) = k$.

Substitution of $y_3(t) = k$ in (1.22c) gives $0 = y_2(k-1) + y_1(2-k) - t^2$.

If k = 1, we have the system $y_2' = y_1$ (1.23a)

$$0 = y_1 - t^2 \quad (1.23b)$$

which has index 1, since one differentiation of (1.23b) is needed to obtain

$$\mathbf{y}' = (2t, t^2 0)^T.$$

However, if k = 2, we obtain the system $y_2' = y_1$ (1.24a)

$$0 = y_2 - t^2 \quad (1.24b)$$

This system has index 2 since we need to differentiate (1.24b) to obtain y_2' and (1.24a) to obtain y_1'.

Any DAE in the form (1.11) can be written in the semi-explicit form (1.12) by defining $\mathbf{y}' = \mathbf{z}$. The index is consequently increased by 1 but we note that merely rewriting the DAE in the form

$$\mathbf{y}' = \mathbf{z}$$
$$\mathbf{0} = \mathbf{F}(t, \mathbf{y}, \mathbf{z})$$

does not make it easier to solve. We observe that the class of fully implicit index-1 DAEs in the form (1.11) is equivalent to the class of semi-explicit index-2 DAEs of the form (1.12).

1.5 HESSENBERG FORMS OF DAEs

Many higher-index problems encountered in practical applications can be expressed as ODEs coupled with constraints. In systems called Hessenberg forms of the DAE the algebraic and differential variables are explicitly identified for higher-index DAEs as well, and in principle, the algebraic variables can be eliminated using the same number of differentiations..

[19] refers to the system $\mathbf{F}(t,\mathbf{y},\mathbf{y}') = 0$ being Hessenberg of index m if it can be written in the form

$$\mathbf{x}_1' = \mathbf{F}_1(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m,t) \qquad (1.25a)$$

$$\mathbf{x}_2' = \mathbf{F}_2(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_{m-1},t) \qquad (1.25b)$$

. .

. .

$$\mathbf{x}_i' = \mathbf{F}_i(\mathbf{x}_{i-1},\mathbf{x}_i,\ldots,\mathbf{x}_{m-1},t) \qquad (1.25c)$$

$$0 = \mathbf{F}_m(\mathbf{x}_{m-1},t) \qquad (1.25d)$$

where the matrix $\left(\frac{\partial \mathbf{F}_m}{\partial \mathbf{x}_{m-1}}\right)\left(\frac{\partial \mathbf{F}_{m-1}}{\partial \mathbf{x}_{m-2}}\right).....\left(\frac{\partial \mathbf{F}_2}{\partial \mathbf{x}_1}\right)\left(\frac{\partial \mathbf{F}_1}{\partial \mathbf{x}_m}\right)$ is nonsingular.

(We need to add $3 \le i \le r-1$ to (1.25c)).

Example 1.9 The Hessenberg form of index 2 is

$$\mathbf{x}_1' = \mathbf{F}_1(\mathbf{x}_1,\mathbf{x}_2,t)$$

$$\mathbf{0} = \mathbf{F}_2(\mathbf{x}_1,t) \quad \text{with} \quad \left(\frac{\partial \mathbf{F}_2}{\partial \mathbf{x}_1}\right)\left(\frac{\partial \mathbf{F}_1}{\partial \mathbf{x}_2}\right) \quad \text{nonsingular.}$$

Example 1.10 The Hessenberg form of index 4 is

$$\mathbf{x}_1' = \mathbf{F}_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, t)$$
$$\mathbf{x}_2' = \mathbf{F}_2(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, t)$$
$$\mathbf{x}_3' = \mathbf{F}_3(\mathbf{x}_2, \mathbf{x}_3, t)$$
$$\mathbf{0} = \mathbf{F}_4(\mathbf{x}_3, t)$$

with $\left(\frac{\partial \mathbf{F}_4}{\partial \mathbf{x}_3}\right)\left(\frac{\partial \mathbf{F}_3}{\partial \mathbf{x}_2}\right)\left(\frac{\partial \mathbf{F}_2}{\partial \mathbf{x}_1}\right)\left(\frac{\partial \mathbf{F}_1}{\partial \mathbf{x}_4}\right)$ nonsingular.

If we consider the linear time-varying DAE in the form (1.8) then [4] states that this system is in Hessenberg form of size r if it can be written as

$$\begin{pmatrix} I & 0 & . & . & 0 \\ 0 & I & & & . \\ . & & & & . \\ . & & & I & . \\ 0 & . & . & . & 0 \end{pmatrix}\begin{pmatrix} \mathbf{x}_1' \\ . \\ . \\ . \\ \mathbf{x}_r' \end{pmatrix} + \begin{pmatrix} B_{11} & B_{12} & & B_{1,r-1} & B_{1,r} \\ B_{21} & & & B_{2,r-1} & 0 \\ 0 & & & & . \\ . & & & & . \\ 0 & . & 0 & B_{r,r-1} & 0 \end{pmatrix}\begin{pmatrix} \mathbf{x}_1 \\ . \\ . \\ . \\ \mathbf{x}_r \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ . \\ . \\ . \\ \mathbf{f}_r \end{pmatrix} \qquad (1.26)$$

where the $\mathbf{x}_i$ are vectors, B_{ij} are matrices and the product $B_{r,r-1}B_{r-1,r-2}.....B_{1,r}$ is nonsingular.

Example 1.11 A simple example of a DAE in Hessenberg form of size 2 is

$$x_1' + B_{11}x_1 + B_{12}x_2 = f_1$$
$$B_{21}x_1 = f_2$$

with $B_{21}B_{12}$ nonsingular.

1.6 SIMILARITIES AND DIFFERENCES BETWEEN DAEs AND ODEs

The supposition that DAEs were considered to be essentially similar to regular implicit ODEs was challenged by conflicting computation results, (e.g. Sincovec et al, 1981), and a more thorough investigation of DAEs began. According to [4] C. W. Gear et al initiated a discussion on DAEs with the publication of their famous paper in 1981.

In [2] integration is described as a "smoothing" process and differentiation as an "antismoothing process". An ODE involves integration, hence smoothing, but a DAE involves both integrations and differentiations which may be "intertwined in a complex manner". (see page 232). We observe that all ODEs are DAEs but that the converse is not true.

For initial value ODEs there is a theorem which guarantees solution, existence, uniqueness and continuous dependence on initial data for a large class of problems (see [2], for example) but no corresponding theorem exists in such generality for boundary value ODEs and we cannot expect a simple, corresponding theorem for DAEs. In fact, by 1988, theorems on the uniqueness and existence of a solution and expressions for closed-form solutions had only been been resolved fully for linear constant coefficient DAE systems, the solution of even linear variable coefficient DAE systems proving to be a much more intricate problem (see[3]).

We note that we need, for an ODE of order m, m initial or boundary conditions in order to specify the solution, and illustrate this in example 1.12.

Example 1.12 Consider the ODE

$$y''(x) = f(x) \tag{1.27}$$

We integrate (1.27) to obtain

$$y'(x) = \int_0^x f(s)ds + y'(0) \tag{1.28}$$

and integrate (1.28) to give

$$y(x) = \int_0^x \left\{ \int_0^s f(u)du + y'(0) \right\} ds + y(0) \tag{1.29}$$

We observe that, since the system has 2 degrees of freedom, we need to know 2 values, say y(0) and y'(0), to specify the solution for this ODE of order 2.

However, as we illustrate in example 1.13, for simple DAEs, the solutions is determined by the equation.

Example 1.13

For the DAE $\begin{aligned} x' &= z + t^2 \\ 0 &= x - t^2 \end{aligned}$ the solution is $x = t^2, z = 2t - t^2$, without the need to specify any conditions.

If we consider the simple index two problem

$$y_1 = f(t)$$
$$y_2 = y_1'$$

then the solution is $\begin{aligned} y_1 &= f(t) \\ y_2 &= f'(t) \end{aligned}$.

The solution depends on the forcing function f(t) and it's first derivative, both evaluated at the current time only i.e. it does not depend upon the initial value or on the past history of f(t). In general, for systems of index m the solution involves derivatives of order (m-1) of the forcing function.

A major practical difference when we come to discuss the numerical solutions of ODEs and DAEs is the need to start the solution of a DAE system with a consistent set of initial conditions. E.g. Consider example 1.13. If x(0) = c = a constant is imposed then we find that this is inconsistent with the DAE unless c = 0. However, in the latter case this gives no further information i.e. it is superfluous. This aspect of DAEs is considered further in section 4.4.

A problem formulated as an initial value ODE can be readily solved numerically using an appropriate (computer) code. However, for a DAE we find that the processes of formulating the problem and solving it numerically are combined in that a DAE formulation of the problem is likely to need more user attention and intervention. As well as the usual behaviour associated with ODEs, DAEs can exhibit additional behaviour such as the bifurcation of solutions (see [4] for further information on bifurcation points).

1.7 APPLICATIONS OF DAEs IN THE "REAL WORLD"

We now move on to discuss where DAEs arise. We find that they are a natural way of describing a wide variety of physical systems and are particularly useful in

- constrained variational problems
- network modelling
- classical singular perturbation theory
- solving partial differential equations (PDEs) by the method of lines

1.7.1 CONSTRAINED VARIATIONAL PROBLEMS

These include constrained mechanical systems and optimal control problems with unconstrained controls. The latter involve a process and a cost and the problem to be solved is to choose the control in order to minimise the cost subject to the given process and specified initial or boundary conditions. Constrained mechanical systems arise in real-time vehicle simulation and design, in computer-aided design of mechanical systems and in robotics.

Example 1.14 A robot arm moving with an end point in contact with a surface is governed by the following equations, given in [4].

$$M(x)x'' + G(x,x') = U + B^T(x)\lambda \qquad (1.31a)$$

$$0 = \phi(x) \qquad (1.31b)$$

with $B = \phi(x)$, $x \in \Re^n$, $\lambda \in \Re^m$, $U \in \Re^n$ and M is the mass matrix.

G characterises the Coriolis, centrifugal and gravitational effects of the robot.
U = input (control) torque vector at the joints.
c defines the contact surface and $B^T\lambda$ is the contact force vector.

1.7.2 NETWORK MODELLING

Here we start with a collection of quantities and known, or desired, relationships between them. Examples are found in

- circuit analysis
- trajectory prescribed path control
- chemical process simulation

In circuit analysis the currents in the branches and the voltages at nodes, or voltage drops on branches, are of physical interest. These quantities are related by laws, e.g. Kirchoff's node (current) and loop (voltage) laws. For RLC circuits the equations are often linear, constant coefficient and will be semi-explicit. However, if devices such as diodes or nonlinear resistors are included then the system will be nonlinear. We note that computer-aided design stimulated researchers to study DAEs and their direct numerical solution and that, according to the authors of [4], the first paper on the numerical solution of DAEs appeared in the IEEE Transactions on Circuit Theory in 1971.

Trajectory prescribed path control applications include the design of a safe re-entry profile for the space shuttle and aiding the performance analysis for the design of a space vehicle. Prescribed path control also arises when there are invariants present in the solution to a system of ODEs.

Example 1.15 (taken from [4])

A process is governed by

$$\mathbf{x}' = \mathbf{f}(\mathbf{x},\mathbf{u},t) \tag{1.32a}$$

where **x** are the state variables and **u** are the control variables. **u** is chosen so that the trajectory **x** follows some prescribed path

$$\mathbf{g}(\mathbf{x},\mathbf{u},t) = 0 \tag{1.32b}$$

We note that in practice **u** is frequently absent from (1.32b) and that this can lead to numerical difficulties when solving the semi-explicit DAE.

1.7.3 CLASSICAL SINGULAR PERTURBATION THEORY

Classical singular perturbation theory is a rich source of DAEs. In a given model there may be various small parameters which may be set equal to zero in an attempt to simplify the model or to obtain a first order approximation to its behaviour. The resulting system is often a DAE. [4] gives the classical singular perturbation in the form

$$\mathbf{x}' = \mathbf{f}(\mathbf{x},\mathbf{y},t,\varepsilon) \qquad 0 < \varepsilon << 1 \tag{1.33a}$$

$$\varepsilon\,\mathbf{y}' = \mathbf{g}(\mathbf{x},\mathbf{y},t,\varepsilon) \tag{1.33b}$$

which is an example of a stiff ODE. (see section 3.1.2 for further discussion on stiffness) Even if the solution is needed for all $t \geq 0$ the solution can be added to a boundary layer correction term corresponding to a fast time scale. Applications can be found in fluid dynamics, the study of nonlinear oscillations with large parameters and in chemical kinetics with slow and fast reactions.

1.7.4 METHOD OF LINES

These problems, in which the spatial derivatives are discretized by finite differences or finite elements to obtain a DAE system, lead to an explicit ODE but many well-posed problems which are of practical interest are more readily handled as a DAE. The addition of a physical property to be satisfied, e.g.incompressibility, as a constraint in the spatial discretization, leads to a DAE and may enable the numerical solution to satisfy the constraint as well as the analytical solution. Examples of this method are found in

- combustion and chemical kinetics modelling
- chemical vapour decomposition
- the fabrication of integrated circuits
- incompressible fluid flows
- gas transfer in piping networks

SECTION 2: SOLVING DIFFERENTIAL-ALGEBRAIC EQUATIONS

We will now turn our attention to solving DAEs. We will consider systems of the form

$$\mathbf{0} = \mathbf{F}(t, \mathbf{y}, \mathbf{y}') \tag{2.1}$$

2.1 WHAT IS MEANT BY A SOLVABLE DAE?

Solvability has been defined in various ways by different people. In [18] and [19] Petzold defines solvability for (2.1) as:-

"The DAE is solvable for t in a finite interval I, if for each t, there is a manifold M_t, through every point of which there is a smooth solution which exists over all of I, solutions never bifurcate and two solutions beginning at distinct values never intersect in I."

We find alternative definitions of solvability in [4] and [11]. Naturally they ensure that solvable problems have solutions which do not bifurcate and which are uniquely determined by their value at $t = t_0$. We now give an example of a non-solvable DAE.

Example 2.1 We consider the system, from [13], which describes a nonlinear resistor network.

$$x_1' - x_3^{\,2} = 0 \tag{2.2a}$$

$$x_2' + x_3 = 0 \tag{2.2b}$$

$$x_1 + x_2 x_3 + x_3^{\,3} = 0 \tag{2.2c}$$

One solution of this DAE is given as

$$x_*(t) = \left(2(\frac{t}{6}+1)^3, -3(\frac{t}{6}+1)^2, \frac{t}{6}+1\right)^{\mathrm{T}}$$

We can see that this is a solution by noting that

$$x_1' = \left(\frac{t}{6}+1\right)^2 \quad \text{so that} \quad x_1' = x_3^2,$$

$$x_2' = -3(2)\left(\frac{t}{6}+1\right).\frac{1}{6} = -\left(\frac{t}{6}+1\right) \quad \text{so that} \quad x_2' = -x_3.$$

We can readily show that (2.2c) is satisfied. We note that $x_*(0) = (2,-3,1)^{\mathrm{T}}$.

However, $x_{**}(t) = (2+t,-3-t,1)^{\mathrm{T}}$ is also a solution, since $x_1' = 1, x_2' = -1$, along with $x_{**}(t)$ can be shown to satisfy (2.2). We note that in this case also $x_{**}(0) = (2,-3,1)^{\mathrm{T}}$. Consequently this singular, index-1 DAE displays bifurcation phenomena.

$\therefore$ the DAE is not solvable along its singular curves, which take the form

$$x_2 + 3x_3^2 = 0$$

since differentiation of (2.2c) with respect to t gives

$$x_1' + x_2' x_3 + x_2 x_3' + 3x_3^2 x_3' = 0$$

and substitution from (2.2a) and (2.2b) gives

$$x_3^2 + (-x_3)x_3 + x_2 x_3' + 3x_3^2 x_3' = 0$$

which implies

$$x_3'(x_2 + 3x_3^2) = 0\,.$$

Having given an example of a nonsolvable DAE we now move on to consider, in the next subsection, which types of DAE are known to be solvable or for which a method of solution exists if the DAE is solvable.

2.2 GUIDANCE ON WHICH DAEs CAN BE SOLVED CURRENTLY

The purpose of this section is to answer the following questions:-

1. Is it possible to determine whether a DAE is solvable by inspecting its system of equations?
2. Is it always necessary to determine the index of the DAE before a decision about its solvability is made?
3. Is it possible to be certain whether or not a DAE can be solved? (i.e. can conclusive evidence be found in the research undertaken to date to make a definite statement about its solvability?)
4. Is it easier to make a decision for some types of DAEs than for others?

In section 1.2 we saw that DAEs could be classified into different types. We now give a visual representation of determining whether a DAE is solvable in Figure 2.2, along with references to the original publication in Figure 2.1.

The first step is to decide which type of DAE you are interested in and for ease of reference the following key has been adopted.

Type A:	Linear Constant Coefficient DAE	(see 1.3.1)
Type B:	Linear Time-Varying DAE	(see 1.3.2)
Type C:	Semi-explicit DAE	(see 1.3.4)
Type D:	A DAE in Hessenberg form	(see 1.5)

The emphasis in the research material which we have considered focuses largely on the numerical solution to DAEs. Discussion about analytical solutions to DAEs is mainly concerned with linear constant coefficient and linear time-varying DAEs, (i.e. types A and B). By saying that a DAE is numerically solvable authors are implying that if the DAE has a solution then current software is available to find it , e.g. in [19] it is stated that "The class of DAEs which we consider has the property that solutions exist and are in some sense unique."

FIGURE 2.1A ORIGIN OF RESEARCH FINDINGS FOR FIGURE 2.2A

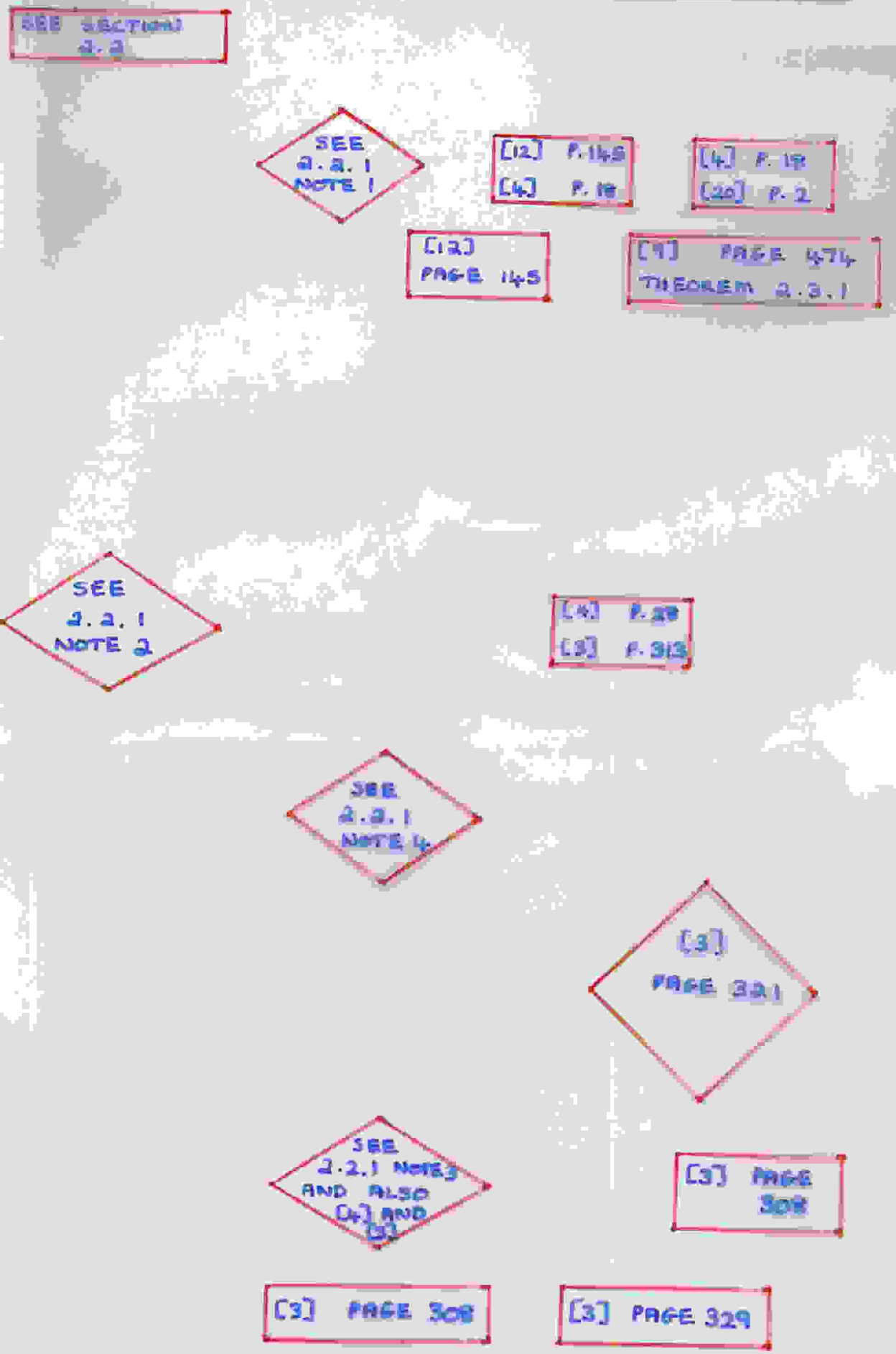


FIGURE 2.2A TO DETERMINE WHETHER OR NOT A DAE IS SOLVABLE

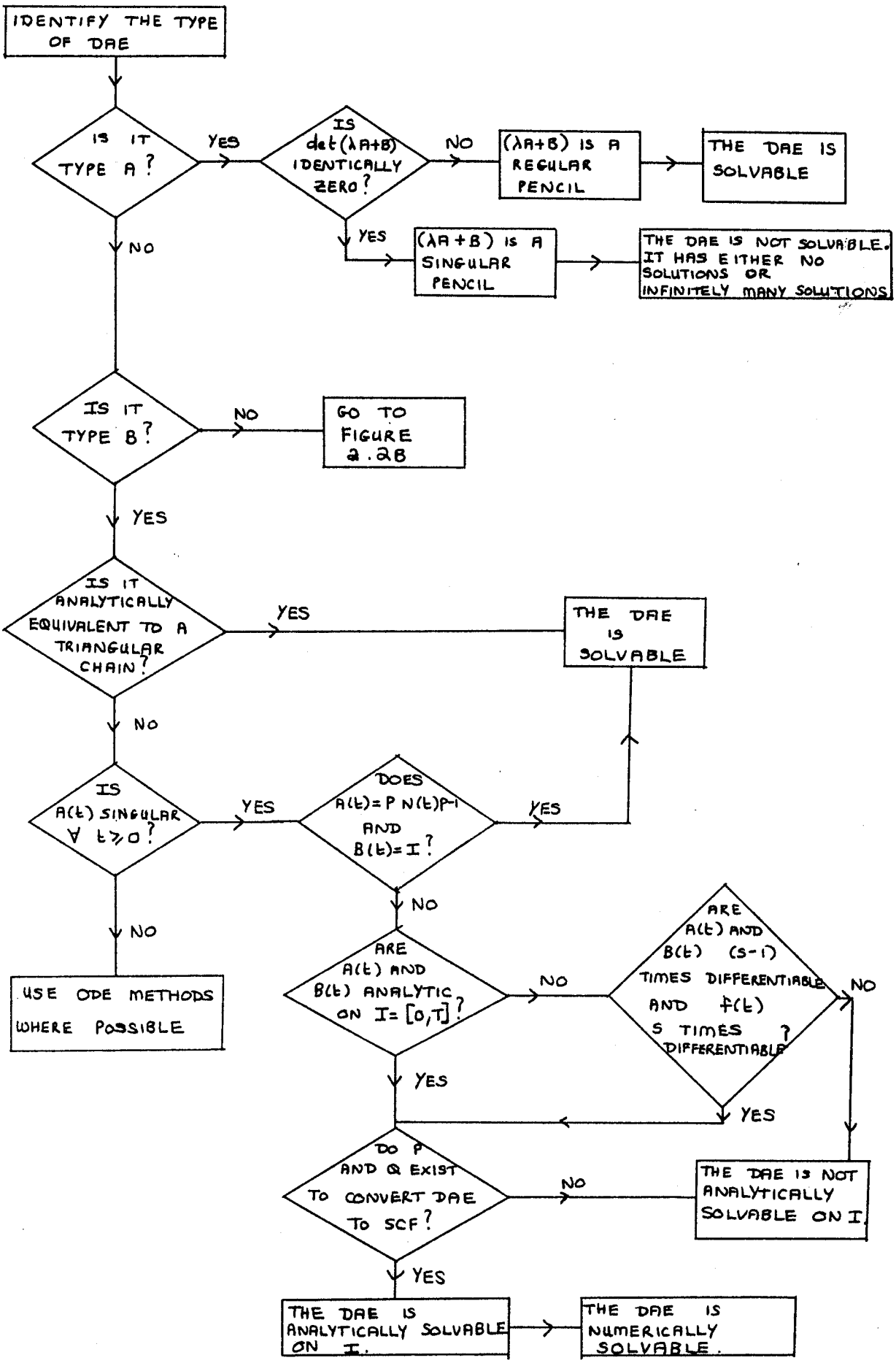


FIGURE 2.1 B

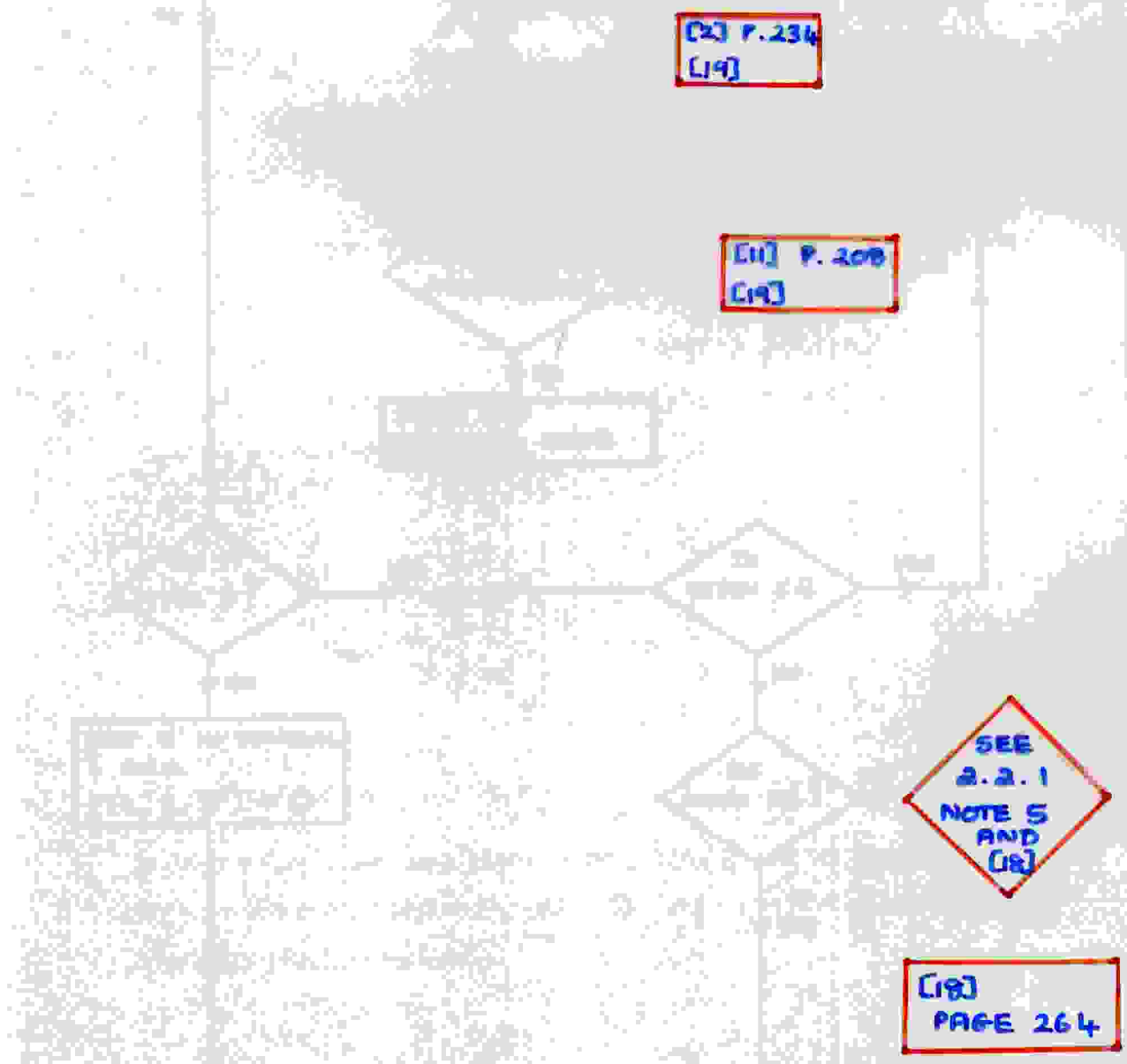
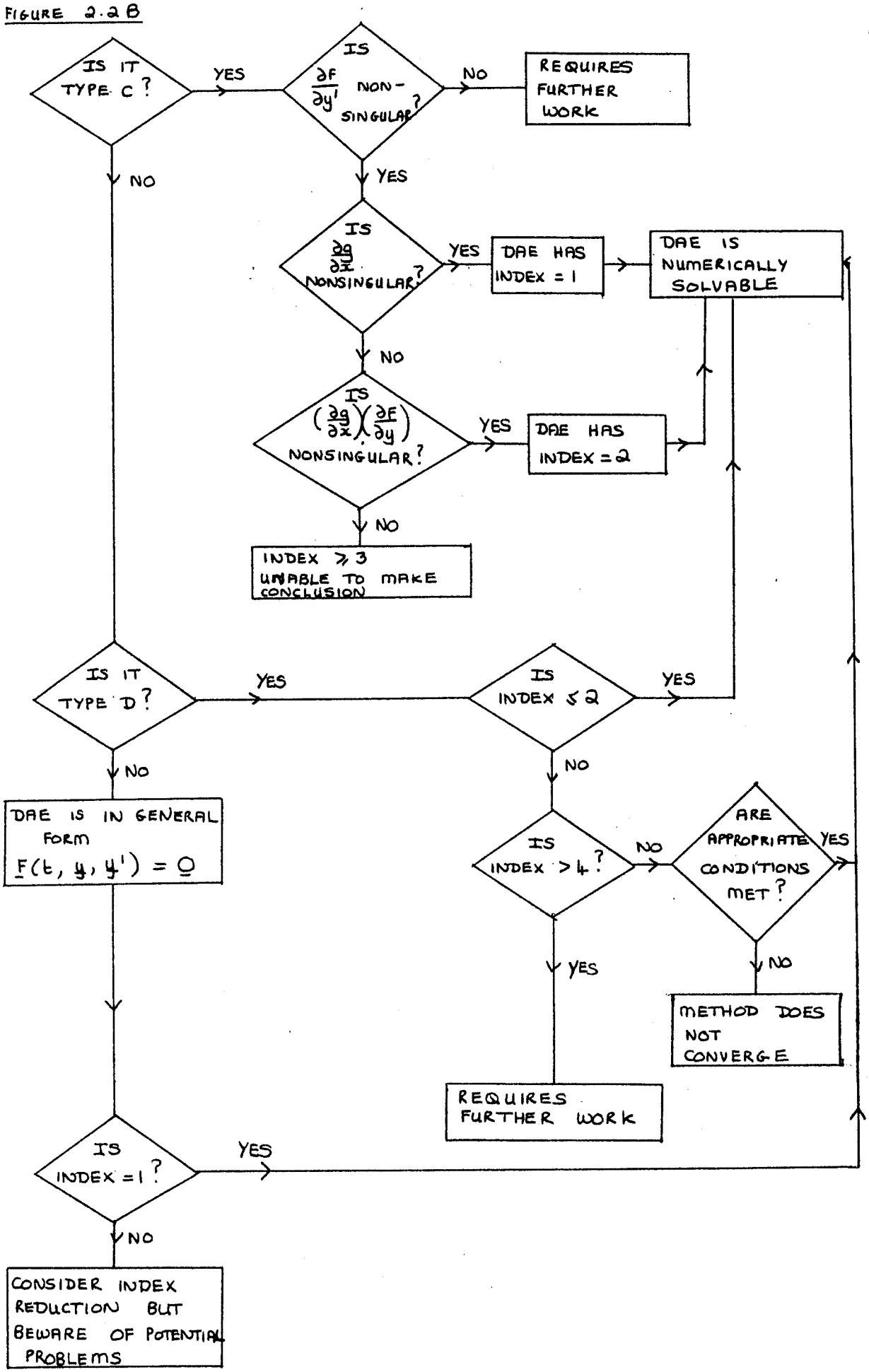


FIGURE 2.2B


IS IT TYPE C?
YES
IS $\frac{\partial F}{\partial y'}$ NON-SINGULAR?
NO
REQUIRES FURTHER WORK
NO
YES
IS $\frac{\partial g}{\partial x}$ NONSINGULAR?
YES
DAE HAS INDEX = 1
DAE IS NUMERICALLY SOLVABLE
NO
IS $\left(\frac{\partial g}{\partial x}\right)\left(\frac{\partial F}{\partial y}\right)$ NONSINGULAR?
YES
DAE HAS INDEX = 2
NO
INDEX ≥ 3 UNABLE TO MAKE CONCLUSION
IS IT TYPE D?
YES
IS INDEX ≤ 2
YES
NO
DAE IS IN GENERAL FORM
$\underline{F}(t, \underline{y}, \underline{y}') = \underline{0}$
IS INDEX > 4?
NO
ARE APPROPRIATE CONDITIONS MET?
YES
NO
METHOD DOES NOT CONVERGE
YES
REQUIRES FURTHER WORK
IS INDEX = 1?
YES
NO
CONSIDER INDEX REDUCTION BUT BEWARE OF POTENTIAL PROBLEMS

We consider the questions raised at the beginning of this section with reference to Figure 2.2. We observe that if, on inspection of the DAE system it can be seen to be in Hessenberg form then a decision can be made if the index is known. However, if the DAE is recognised to be a linear constant coefficient DAE, i.e. type A, then knowledge of the index is not required as solvability depends upon whether the matrix pencil is regular or not. We note that, in relation to question 4, the analytical solvability of a linear time-varying DAE is more complicated to determine than that of a linear constant coefficient DAE.

In general, merely inspecting the DAE will not be sufficient and further mathematical analysis will be needed before a decision can be made regarding its analytical or numerical solvability.

2.2.1 SUMMARY OF DEFINITIONS AND ADDITIONAL EXPLANATORY NOTES FOR FIGURE 2.2

1. If λ is a complex parameter then (λA + B) is called a matrix pencil.
2. [4] states that the DAE $A(t)\mathbf{x}'(t) + B(t)\mathbf{x}(t) = \mathbf{f}(t)$ is an r th order (lower) triangular chain if it is in the form

$$\begin{pmatrix} A_{11} & 0 & . & . & . & 0 \\ A_{21} & A_{22} & . & & & . \\ . & & . & . & & . \\ . & & & . & . & . \\ . & & & & . & 0 \\ A_{r1} & . & . & . & . & A_{rr} \end{pmatrix} \begin{pmatrix} x_1' \\ . \\ . \\ . \\ . \\ x_r' \end{pmatrix} + \begin{pmatrix} B_{11} & 0 & . & . & . & 0 \\ B_{21} & B_{22} & . & & & . \\ . & & . & . & & . \\ . & & & . & . & . \\ . & & & & . & 0 \\ B_{r1} & . & . & . & . & B_{rr} \end{pmatrix} \begin{pmatrix} x_1 \\ . \\ . \\ . \\ . \\ x_r \end{pmatrix} = \begin{pmatrix} f_1 \\ . \\ . \\ . \\ . \\ f_r \end{pmatrix}$$

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 index-1, index zero, in Hessenberg form, or in standard canonical form.

3. In [4] we find that the system $A(t)\mathbf{x}'(t) + B(t)\mathbf{x}(t) = \mathbf{f}(t)$ is in standard canonical form (SCF) if it is in the form

$$\begin{pmatrix} I & 0 \\ 0 & N(t) \end{pmatrix}\mathbf{x}' + \begin{pmatrix} c(t) & 0 \\ 0 & I \end{pmatrix}\mathbf{x} = \mathbf{f}(t)$$

where N is strictly lower (or upper) triangular.

P and Q are analytic and nonsingular and are such that transformations of the form **x** = Q(t)**y**, and left multiplication of the equation by P(t) exist which transform the linear time-varying DAE to SCF everywhere on I. i.e. P and Q are such that

$PAQ\mathbf{y}' + (PAQ' + PBQ)\mathbf{y} = P\mathbf{f}$ is analytically equivalent to

$A(t)\mathbf{x}'(t) + B(t)\mathbf{x}(t) = \mathbf{f}(t)$.

4. The singular system

$$A(t)\mathbf{x}'(t) + \mathbf{x}(t) = \mathbf{f}(t) \quad \text{where} \quad A(t) = PN(t)P^{-1}$$

with P s x s, nonsingular and constant; N(t) analytic, s x s and strictly upper (or lower) triangular,

has a unique solution. (see [3]).

5. The order of the variable step size BDF method needs to be greater than 1 for Hessenberg index-3 DAEs and greater than 3 for Hessenberg index-4 DAEs. (Further details on BDF methods can be found in sections 3 and 4).

2.3 ANALYTICAL SOLUTIONS TO DAEs

First we consider analytical solutions to linear constant coefficient and linear time-varying DAEs. We follow this with examples of solving DAEs analytically.

2.3.1 LINEAR CONSTANT COEFFICIENT SYSTEMS

We consider the system

$$A\mathbf{x}' + B\mathbf{x} = \mathbf{f}(t) \tag{2.3}$$

which is solvable iff ($\lambda A + B$) is not identically zero, i.e. the matrix pencil is regular.

Consider $\mathbf{x} = (x_1, x_2)^T, \mathbf{f}(t) = (f_1, f_2)^T$.

Nonsingular P and Q exist such that $PAQ = \begin{pmatrix} I & 0 \\ 0 & N \end{pmatrix}$ and $PBQ = \begin{pmatrix} C & 0 \\ 0 & I \end{pmatrix}$.

[3] states that the solution to the equivalent system

$$y_1'(t) + Cy_1(t) = f_1(t) \tag{2.4a}$$

$$Ny_2'(t) + y_2(t) = f_2(t) \tag{2.4b}$$

is given by

$$y_1(t) = e^{-tc}y_1(0) + \int_0^t e^{(s-t)}cf_1(s)ds \tag{2.5a}$$

$$y_2(t) = \sum_{i=0}^{v-1}(-N)^i f_2^{(i)}(t) \tag{2.5b}$$

where N is nilpotent of degree v, i.e. $N^{v-1} \neq 0$ but $N^v = 0$.

We will consider example 1.2

where $A = \begin{pmatrix} 0 & 1 \\ 0 & 3 \end{pmatrix}$, $B = \begin{pmatrix} -2 & 0 \\ -6 & 1 \end{pmatrix}$ and $\mathbf{f}(t) = \begin{pmatrix} 0 \\ t^2 \end{pmatrix}$

We can see that $\lambda A + B = \begin{pmatrix} -2 & \lambda \\ -6 & 3\lambda + 1 \end{pmatrix}$

Since det(λA + B) = -2(3λ + 1) + 6λ

= -2

≠ 0

then the matrix pencil is regular, and hence the DAE is solvable.

In this simple example v = 1, N = 0, c = 0 since we can write the system in the form

$$x_1'(t) = t$$
$$x_2(t) = t^2.$$

We note that $f_1(t) = t$ and $f_2(t) = t^2$

Applying (2.5a) gives

$$x_1(t) = e^{-0}x_1(0) + \int_0^t 0ds = x_1(0) + t \qquad (2.6)$$

and using (2.5b) we obtain

$$x_2(t) = \sum_{i=0}^{0} (-N)^i f_2^{(i)}(t) = f_2^{(0)}(t) = t^2 \qquad (2.7)$$

We note that the initial condition $x_1(0)$ in (2.6) can be chosen arbitrarily but that $x_2(0)$ must be zero, (from(2.7)), in order to obtain a consistent initial condition of the DAE system.

2.3.2 LINEAR VARIABLE COEFFICIENT SINGULAR SYSTEMS

Solving linear variable coefficient systems of the form

$$A(t)\mathbf{x}'(t) + B(t)\mathbf{x}(t) = \mathbf{f}(t) \qquad (2.8)$$

has proven to be much more intricate than solving systems of the form (2.3). We are no longer able to equate solvability with the regularity of the matrix pencil λA(t) + B(t). We illustrate this, using examples from [2], by demonstrating that non-regularity of the pencil does not imply nonsolvability in example 2.2, followed by example 2.3 which shows that regularity of the pencil does not imply solvability of the DAE.

Example 2.2 Consider a DAE in the form (2.8) with

$$\mathrm{A}(t)=\begin{pmatrix}1 & t\\ 0 & 0\end{pmatrix} \quad \text{and} \quad \mathrm{B}(t)=\begin{pmatrix}0 & 0\\ 1 & t\end{pmatrix} \quad \text{and} \quad \mathbf{f}_1(t)=\begin{pmatrix}f_1\\ f_2\end{pmatrix}.$$

We find that $\lambda\mathrm{A}(t) + \mathrm{B}(t) = \begin{pmatrix}\lambda & \lambda t\\ 1 & t\end{pmatrix}$

from which we can show that $\det(\lambda\mathrm{A}(t) + \mathrm{B}(t)) = 0$

$\therefore$ $\lambda\mathrm{A}(t) + \mathrm{B}(t)$ is not a regular pencil.

However, by rewriting the system in the form

$$x_1'(t)+tx_2'(t)=f_1(t) \tag{2.9a}$$

$$x_1(t)+tx_2(t)=f_2(t) \tag{2.9b}$$

and differentiating (2.9b) to give

$$x_1'(t)+x_2(t)+tx_2'(t)=f_2'(t)$$

we are able to obtain the solution

$$\begin{aligned}x_2(t)&=f_2'(t)-f_1(t)\\ x_1(t)&=f_2(t)-tf_2'(t)+tf_1(t)\end{aligned}$$

Hence nonregularity of the matrix pencil $\not\Rightarrow$ nonsolvability of the DAE, since we have a DAE with a singular pencil which is solvable.

We note that alternatively we can write $x=\begin{pmatrix}t & 1\\ -1 & 0\end{pmatrix}y$.

In this case (2.8) becomes $\begin{pmatrix}1 & t\\ 0 & o\end{pmatrix}\left\{\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\mathbf{y}+\begin{pmatrix}t & 1\\ -1 & 0\end{pmatrix}\mathbf{y}'\right\}+\begin{pmatrix}0 & 0\\ 1 & t\end{pmatrix}\begin{pmatrix}t & 1\\ -1 & 0\end{pmatrix}\mathbf{y}=\mathbf{f}$

which we can simplify to $\begin{pmatrix}0 & 1\\ 0 & 0\end{pmatrix}\mathbf{y}'+\begin{pmatrix}\mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{1}\end{pmatrix}\mathbf{y}=\mathbf{f}$

which is a solvable, linear constant coefficient index-2 DAE.

Example 2.3 Consider a DAE in the form (2.8) with

$$A(t)=\begin{pmatrix}-t & t^2\\ -1 & t\end{pmatrix},\quad B(t)=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$

We find that the matrix pencil is $\begin{pmatrix}-\lambda t+1 & \lambda t^2\\ -\lambda & \lambda t+1\end{pmatrix}$

and we are able to show that $\det\begin{pmatrix}-\lambda t+1 & \lambda t^2\\ -\lambda & \lambda t+1\end{pmatrix} = (-\lambda t + 1)(\lambda t + 1)-(-\lambda^2 t^2) = 1 \neq 0.$

We can thus conclude that the matrix pencil is regular.

However, if we consider $\mathbf{x}=\phi(t)\begin{pmatrix}t\\ 1\end{pmatrix}$ where $\phi(t)$ is any scalar function, then the LHS of (2.8) with A(t) and B(t) as given becomes $\begin{pmatrix}-t & t^2\\ -1 & t\end{pmatrix}\left\{\phi'(t)\begin{pmatrix}t\\ 1\end{pmatrix}+\phi(t)\begin{pmatrix}1\\ 0\end{pmatrix}\right\}+\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\phi(t)\begin{pmatrix}t\\ 1\end{pmatrix}$

which equals $\begin{pmatrix}0\\ 0\end{pmatrix}\phi'(t)+\phi(t)\begin{pmatrix}-t\\ -1\end{pmatrix}+\begin{pmatrix}t\\ 1\end{pmatrix}\phi(t)=0$

Hence the DAE is not solvable, since an initial value does not uniquely determine a solution, and we have a DAE with no solution but with a regular matrix pencil.

We now return to solving systems in the form (2.8).

If we can rewrite the DAE in SCF (see section 2.2.1 Note 3) with $P(t)A(t)Q(t) = \begin{pmatrix}I & 0\\ 0 & N(t)\end{pmatrix}$

AND $P(t)A(t)Q'(t)+P(t)B(t)Q(t)=\begin{pmatrix}c(t) & 0\\ 0 & I\end{pmatrix}$ AND P, Q analytic and nonsingular,

then [3] gives the analytic solution as

$$\mathbf{x}=Q(t)\mathbf{y}(t)$$

where $\mathbf{y}(t)=\left(y_1(t)^T, y_2(t)^T\right)^T$

and $y_1(t)=\Phi(t)\Phi^{-1}(0)y_1(0)+\Phi(t)\int_0^t\Phi^{-1}(s)f_1(s)ds$

$$y_2(t)=\sum_{i=0}^{s-1}\left(-N(t)D\right)^i f(t).$$

We note that:-

- Φ is a continuously differential fundamental solution matrix of $y_1'(t) + c(t)y_1(t) = 0$
- $y_1(0)$ is a given initial condition which can be chosen arbitrarily
- D is the differentiation operator, and N(t) is an analytic sxs matrix-valued function.
  i.e. (-N(t)D) is defined as (-N(t)D) x(t) = -N(t)x'(t) and

$$(-N(t)D)^{t+1}x(t) = (-N(t)D)^t(-N(t)x'(t))$$

2.3.3 FURTHER EXAMPLES OF SOLVING DAEs ANALYTICALLY

Example 2.4 We will consider a DAE of the form

$$A(t)\mathbf{x}'(t) + \mathbf{x}(t) = \mathbf{f}(t) \qquad (2.10)$$

where A(t) = PN(t)P^{-1} with P sxs, nonsingular and constant, N(t) analytic, sxs and strictly upper triangular. We know from section 2.2 that this system is solvable. We will take the following example from [3] as an illustration of how this approach can be used.

$$N(t) = \begin{pmatrix} 0 & t & 0 & 2t^2 \\ 0 & 0 & t-1 & 2t-1 \\ 0 & 0 & 0 & t \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad P(t) = \begin{pmatrix} -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 \end{pmatrix},$$

and $f(t) = \begin{pmatrix} -e^{-t} \\ -e^{-t} \\ -e^{-t} \\ -e^{-t} \end{pmatrix}$ which is equivalent to $-e^{-t}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

We note that det P = 8 $\therefore$ P is nonsingular

We find that $P^{-1} = \frac{1}{2}\begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & -1 \end{pmatrix}$

We can show that

$$A(t) = PN(t)P^{-1} = \frac{1}{2}\begin{pmatrix} 2t^2+3t-1 & t & -1 & 2t^2+2t \\ -2t^2+3t-1 & -t & 2t-1 & -2t^2+2t \\ -2t^2-t+1 & -t & 1 & -2t^2 \\ -2t^2-3t+1 & -t & 1 & -2t^2-2t \end{pmatrix}$$

and that det A = 0

$\therefore$ A is singular.

Returning to (2.10) and using $A(t) = PN(t)P^{-1}$ we can write

$$PN(t)P^{-1}\mathbf{x}'(t) + \mathbf{x}(t) = \mathbf{f}(t)$$

Premultiplying by P^{-1} will give

$$N(t)P^{-1}\mathbf{x}'(t) + P^{-1}\mathbf{x}(t) = P^{-1}\mathbf{f}(t) \tag{2.11}$$

We can show, by considering each term separately, that

$$N(t)P^{-1}\mathbf{x}'(t) = \frac{1}{2}\begin{pmatrix} -2t^2x_1' - tx_2' + tx_3' - 2t^2x_4' \\ (1-2t)x_1' + (1-t)x_3' - tx_4' \\ -tx_1' - tx_4' \\ 0 \end{pmatrix} \tag{2.12}$$

$$P^{-1}\mathbf{x}(t) = \frac{1}{2}\begin{pmatrix} -x_1 + x_2 \\ -x_2 + x_3 \\ -x_3 + x_4 \\ -x_1 - x_4 \end{pmatrix} \tag{2.13}$$

and $$P^{-1}\mathbf{f}(t) = e^{-t}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.14}$$

Using (2.11) and substituting from (2.12), (2.13) and (2.14) we obtain the following four equations:-

$$-2t^2x_1' - tx_2' + tx_3' - 2t^2x_4' - x_1 + x_2 = 0 \tag{2.15a}$$

$$(1-2t)x_1' + (1-t)x_3' - tx_4' - x_2 + x_3 = 0 \tag{2.15b}$$

$$-tx_1' - tx_4' - x_3 + x_4 = 0 \tag{2.15c}$$

$$-x_1 - x_4 = 2e^{-t} \tag{2.15d}$$

From (2.15d) we obtain

$$x_1 + x_4 = -2e^{-t} \tag{2.16a}$$

which we can differentiate to obtain

$$x_1' + x_4' = 2e^{-t}$$

if we substitute into (2.15c) we obtain

$$-t(2e^{-t}) = x_3 - x_4 \tag{2.16b}$$

Rewriting (2.15b) in the form

$$(x_1' + x_3') - t(x_1' + x_3') - t(x_1' + x_4') - x_2 + x_3 = 0$$

we can show that

$$x_2 - x_3 = -2t^2 e^{-t} \qquad (2.16c)$$

Rewriting (2.15a) in the form

$$-2t^2(x_1' + x_4') - t(x_2' - x_3') - x_1 + x_2 = 0$$

we can show that

$$-2t^3 e^{-t} = x_1 - x_2 \qquad (2.16d)$$

We now have (2.16), which is a system of 4 equations in 4 unknowns, and which can be solved to give

$$x_1 = e^{-t}(-1 - t - t^2 - t^3)$$

$$x_2 = e^{-t}(-1 - t - t^2 + t^3)$$

$$x_3 = e^{-t}(-1 - t + t^2 + t^3)$$

$$x_4 = e^{-t}(-1 - t - t^2 + t^3)$$

The expressions which we have for $x_1(t)$, $x_2(t)$, $x_3(t)$ and $x_4(t)$ are in agreement with the solution given in [3].

Changing A(t) into the form $PN(t)P^{-1}$ has eased the solution of the DAE since if we start with A(t) in its original form we do not obtain equations which are readily solvable.

Example 2.5 The equations (2.17) are taken from [2], (page 267).

$$x_1' = \left(\alpha - \frac{1}{2-t}\right)x_1 + (2-t)\alpha z + \frac{3-t}{2-t} \qquad (2.17a)$$

$$x_2' = \left(\frac{1-\alpha}{t-2}\right)x_1 - x_2 + (\alpha - 1)z + 2e^t \qquad (2.17b)$$

$$0 = (t+2)x_1 + (t^2 - 4)x_2 - (t^2 + t - 2)e^t \qquad (2.17c)$$

where α is a parameter.

The DAE is in Pure Index-2 form (or Hessenberg form).

In seeking a solution we notice that in (2.17c) the coefficients of x_1, x_2 and e^t have a total of zero

i.e. $(t + 2) + (t^2 - 4) - (t^2 + t - 2)e^t = 0$

$\therefore$ A possible solution involves $x_1 = x_2 = e^t$ (2.18)

If we substitute from (2.18) into (2.17b) we obtain

$$e^t = \left(\frac{1-\alpha}{t-2}\right)e^t - e^t + (\alpha - 1)z + 2e^t$$

$$\Rightarrow \qquad \left(\frac{1-\alpha}{t-2}\right)e^t + (\alpha - 1)z = 0$$

$$\Rightarrow \qquad z = \frac{e^t}{(t-2)} \quad \text{, providing } \alpha \neq 1.$$

We rearrange (2.17a) into the form

$$x_1' - \left(\alpha - \frac{1}{2-t}\right)x_1 - (2-t)\alpha z = \frac{3-t}{2-t}$$

Evaluating the LHS using the expressions already obtained for x_1, x_2 and z gives

$$\text{LHS} = e^t - \left(\alpha - \frac{1}{2-t}\right)e^t - (2-t)\alpha\left(\frac{e^t}{t-2}\right)$$

$$= e^t\left(\frac{3-t}{2-t}\right)$$

In conclusion we find that (2.17a) and the solution given in [2], which is the same as the solution deduced above, are inconsistent. A multiplying factor of e^t in the last term on the RHS of (2.17a) would resolve the inconsistency.

Example 2.6 This is an example of an index-3 DAE ($\forall \eta \in \Re$), taken from [12]. Consider

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & \eta t & 1 \\ 0 & 0 & 0 \end{pmatrix}\mathbf{x}'(t) + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \eta + 1 & 0 \\ 0 & \eta t & 1 \end{pmatrix}\mathbf{x}(t) = \mathbf{q}(t) \qquad (2.19)$$

By writing $\mathbf{x}(t) = (x_1, x_2, x_3)^{\mathrm{T}}$ and $q(t) = (q_1, q_2, q_3)^{\mathrm{T}}$

we can rewrite the DAE as

$$x_2'(t) + x_1(t) = q_1(t) \qquad (2.20a)$$

$$\eta t x_2'(t) + x_3'(t) + (\eta + 1)x_2(t) = q_2(t) \qquad (2.20b)$$

$$\eta t x_2(t) + x_3(t) = q_3(t) \qquad (2.20c)$$

The approach we will use in this example involves differentiaion of the constraint, i.e. the algebraic equation, followed by substitution into the differential equation. More detail about this technique is given in section 2.6.

Differentiation of (2.20c) gives

$$\eta x_2(t) + \eta t x_2'(t) + x_3'(t) = q_3'(t) \qquad (2.21)$$

from (2.20b) and (2.21) we deduce that

$$x_2(t) = q_2(t) - q_3'(t)$$

We can now substitute for x_2'(t) in (2.20a) to obtain

$$x_1(t) = q_1(t) - q_2'(t) + q_3''(t)$$

and for x_2(t) in (2.20c) to obtain

$$x_3(t) = q_3(t) - \eta t q_2(t) + \eta t q_3'(t)$$

(We notice that in this example the value of η = 0 would reduce the DAE to a linear constant coefficient DAE.)

The matrix pencil would be $\begin{pmatrix} 1 & \lambda & 0 \\ 0 & \lambda\eta t + \eta + 1 & \lambda \\ 0 & \eta t & 1 \end{pmatrix}$

and we can show that det(λA + B) = η + 1, which is not identically zero.

$\therefore$ the matrix pencil is regular and the DAE is solvable.

We now move on the consider non-analytical methods of solving DAEs and turn our attention to numerical methods of solution.

2.4 AN INTRODUCTION TO NUMERICAL APPROACHES TO SOLVING DAEs

We begin by noting that when numerical methods are applied to a DAE we assume that the DAE "makes sense", i.e. it is solvable, and that the method is implementable, i.e. the nonlinear system of equations solved at each time step has a solution. (see section 3 for further information on numerical methods).

Numerical approaches to solving DAEs can be roughly divided into two groups

1. Direct discretization of the given system,
2. Methods which involve a reformulation (e.g. index reduction) combined with a discretization.

We need to be able to identify which of the currently available numerical techniques will work for a given problem since, in the views of the authors of [4], none of them work for all DAEs. Direct discretization methods are preferred, when possible, since a reformulation of the DAE may need more input and intervention from the user. Also the system size may be increased, resulting in greater cost. However, the use of direct methods is limited to index-1 and semi-explicit index-2 DAE systems, and is not recommended by the authors of [2] for

non-decoupled DAEs of index higher than 1, hence the popularity of reformulation approaches. We note that by decoupling we mean splitting the solution components **y** into differential and algebraic variables. Reformulation will be considered in greater detail in section 2.5 and direct discretization methods in section 4.1.

Structural characteristics, e.g. the index, are important when considering the convergence and stability properties of numerical methods. In general the higher the index the more difficult the problem is to solve numerically. Advantages and disadvantages of index reduction will be discussed in section 2.6. Section 3 and section 4 provide more extensive detail about numerical approaches to solving DAEs.

2.5 THE REFORMULATION OF DAEs

We will now consider advantages and disadvantages of reformulating a DAE since there is often a choice of formulation for a particular problem, and this choice can influence the performance of a numerical method. We may have choices in both the selection of the equations to be satisfied and in the variables. We often have the option to reduce the index but we note that a better implementation is not guaranteed.

We are able to write any DAE of the form $\mathbf{F}(t,\mathbf{y},\mathbf{y}') = \mathbf{0}$ in the semi-explicit form, but with the index increased by 1, by defining $\mathbf{y}' = \mathbf{z}$ to obtain the system

$$\mathbf{y}' = \mathbf{z}$$

$$\mathbf{0} = \mathbf{f}(t,\mathbf{y},\mathbf{z}).$$

However, this reformulation does not ease the solution of the problem, but it does show that the class of fully implicit index-1 DAEs is equivalent to the class of semi-explicit index-2 DAEs.

To illustrate what is meant by a reformulation of a problem we will consider the following DAE system found in [2].

$$y_1' = \lambda y_1 - y_4 \tag{2.22a}$$

$$y_2' + y_3' = (2\lambda - \sin^2 t)(y_2 + y_3) + \frac{1}{2}(y_2 - y_3)^2 \tag{2.22b}$$

$$0 = y_2 - y_3 - 2\sin t(y_1 - 1) \tag{2.22c}$$

$$0 = y_2 + y_3 - 2(y_1 - 1)^2 \tag{2.22d}$$

Here λ is a parameter and initial conditions are given as $y_1(0) = 2, y_2(0) = 1$.

The constant or time invariant, nonsingular transformation

$$x_1 = y_1 \tag{2.23a}$$

$$x_2 = \frac{1}{2}(y_2 + y_3) \tag{2.23b}$$

$$z_1 = \frac{1}{2}(y_2 - y_3) \tag{2.23c}$$

$$z_2 = y_4 \tag{2.23d}$$

is used in [2] to convert (2.22) into the semi-explicit form (2.24)

$$x_1' = \lambda x_1 - z_2 \qquad (2.24a)$$

$$x_2' = (2\lambda - \sin^2 t)x_2 + z_1^2 \qquad (2.24b)$$

$$0 = z_1 - \sin t(x_1 - 1) \qquad (2.24c)$$

$$0 = x_2 - (x_1 - 1)^2 \qquad (2.24d)$$

We note that although (2.24) is in semi-explicit form it is not in Hessenberg form. A further transformation using $z_1 = \sin t(x_1 - 1)$, (from (2.24c), yields (2.25) which is Hessenberg index-2.

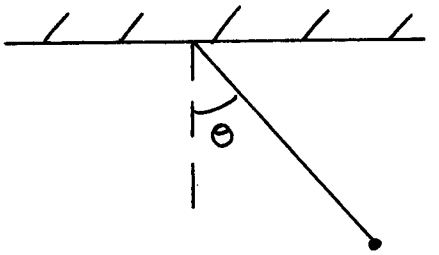
$$x_1' = \lambda x_1 - z_2 \qquad (2.25a)$$

$$x_2' = (2\lambda - \sin^2 t)x_2 + \sin^2 t(x_1 - 1)^2 \qquad (2.25b)$$

$$0 = x_2 - (x_1 - 1)^2 \qquad (2.25c)$$

We note that in (2.24) z_1 and z_2 are algebraic variables, that z_1 is an index-1 variable and that z_2 is an index-2 variable, (One differentiation is needed to find z_1' but two differentiations are needed to find z_2'). This example has illustrated a change to semi-explicit form (2.24) followed by a change to the Hessenberg of (2.25).

Next we will consider the reformulation of a second order ODE subject to constraints into a Hessenberg index-3 DAE, and illustrate this by considering the motion of a simple pendulum.



We will let the mass of the pendulum bob =1 and let the length of the pendulum = 1.

We will assume that the motion is not affected by friction so that the motion is governed by the simple second-order nonlinear ODE

$$\ddot{\Theta} = -g \sin \Theta$$

We will now express the motion of the tiny pendulum bob in terms of the cartesian co-ordinates (x_1, x_2), using $\lambda(t)$ as a Lagrange multiplier.

Newton's equations of motion give

$$\ddot{x}_1 = -\lambda x_1 \tag{2.26a}$$

$$\ddot{x}_2 = -\lambda x_2 - g \tag{2.26b}$$

subject to the constraint $x_1^2 + x_2^2 = 1$.

We can rewrite the two second-order ODEs as four first-order ODEs by defining $v_1 = x_1', v_2 = x_2'$.

We then have

$$v_1' = -\lambda x_1 \tag{2.27a}$$

$$v_2' = -\lambda x_2 - g \tag{2.27b}$$

$$x_1' = v_1 \tag{2.27c}$$

$$x_2' = v_2 \tag{2.27d}$$

which with the constraint in the form

$$0 = x_1^2 + x_2^2 - 1 \tag{2.27e}$$

gives a first-order DAE system which is Hessenberg index-3. Further discussion of this problem can be found in section 5.

2.6 SOLVING HIGHER INDEX DAEs

We have a choice of several techniques when attempting to solve higher index systems. These include:-

- Solve the system in its original form
- Differentiate the constraint one or more times and solve the resulting lower index system
- Use index reduction techniques
- Eliminate Lagrange multipliers analytically
- Use penalty functions or regularisations.

Solving the problem in its original form has the advantage that the system is easy to formulate, and, since we do not rewrite the system in any way or differentiate the constraints, the sparsity of the system is preserved and we satisfy the constraints at each step. On the other hand we can encounter difficulties when using a variable step size BDF code to solve the system. (See sections 3 and 4 for further information on variable step size BDF codes). Although many of these difficulties can be overcome the authors of [4] do not consider it an easy matter to obtain an accurate numerical solution.

If we solve the DAE by differentiating the constraint one or more times, to reduce the index, we introduce the problem that the lower index formulation of the DAE system does not ensure that the constraints are satisfied on each step. We note that the essential concept here is that the DAE is equivalent to an ODE with an invariant. We will illustrate this by further discussion of the equations for the motion of a simple pendulum in the form (2.27).

If we differentiate the constraint, i.e. (2.27e), we obtain

$$0 = x_1 x_1' + x_2 x_2'$$

which, using (2.27c) and (2.27d), can be written as

$$0 = x_1 v_1 + x_2 v_2 \tag{2.28}$$

which is known as the velocity constraint.

If we differentiate (2.28) we obtain

$$0 = x_1 v_1' + x_1' v_1 + x_2' v_2 + x_2 v_2'$$

which, using (2.27c) and (2.27d), can be written as

$$0 = v_1^2 + x_1 v_1' + v_2^2 + x_2 v_2' \tag{2.29}$$

Using (2.27a), (2.27b) and (2.29) we find that

$$0 = v_1^2 + v_2^2 - \lambda - g x_2 \tag{2.30}$$

From (2.30), which is known as the acceleration constraint, we can find an expression for λ,

i.e. $$\lambda = v_1^2 + v_2^2 - g x_2 \tag{2.31}$$

We can substitute from (2.31) into (2.27) and, by rearranging the order, obtain the following ODE, which has resulted from an unstabilized index reduction.

$$x_1' = v_1 \tag{2.32a}$$

$$x_2' = v_2 \tag{2.32b}$$

$$v_1' = -(v_1^2 + v_2^2 - g x_2) x_1 \tag{2.32c}$$

$$v_2' = -(v_1^2 + v_2^2 - g x_2) x_2 - g \tag{2.32d}$$

The constraints on the position and velocity levels give the invariant equations

$$0 = x_1^2 + x_2^2 - 1$$

$$0 = x_1 v_1 + x_2 v_2$$

which are now additional to the ODE (2.32)

We note that [2] states that "index reduction of a DAE leads to an ODE with an invariant," and that "an ODE with an invariant $\begin{matrix}\mathbf{x}' = \hat{\mathbf{f}}(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) = \mathbf{0}\end{matrix}$ is equivalent to the Hessenberg index-2 DAE

$$\begin{aligned}\mathbf{x}' &= \hat{\mathbf{f}}(\mathbf{x}) - \mathrm{D}(\mathbf{x})\mathbf{z} \\ \mathbf{0} &= \mathbf{h}(\mathbf{x})\end{aligned}$$

i.e. the relationship between DAEs and ODEs with invariants goes both ways. Here **D**(**x**) is any bounded matrix function such that HD, where **H** = **hx** is boundedly invertible for all t.

We now return to the problem of the constraints not being satisfied on each step, and note also that the amount by which the constraint is not satisfied may increase from step to step, i.e. a "drift" in the algebraic constraints is present. Small step sizes can be used to help to keep the "drift" small, as can fairly stringent error tolerances in an automatic code. (See section 4 for an introduction to computer software for solving DAEs.)

If the algebraic constraints (in the DAE) reflect important physical properties then the "drift" can have serious implications. (see [4]). Gear proposed an idea which involved introducing constraints which have been "lost" back into the system by augmenting the system as the index reduction proceeds. In this way all original equations and their successive derivatives are retained in the process and a consistent, but overdetermined, index-1 DAE is obtained. However, consistency is generally lost when the system is discretized and special techniques are required for its numerical solution.

In [15] index reduction techniques using a dummy derivative are discussed. We will outline the method using an example from [15].

Consider $\dot{x}=y$ (2.33a)

$\dot{y} = z$ (2.33b)

$x = f(t)$ (2.33c)

We note that (2.33) is an index-3 DAE with solution $x = f(t).y = \dot{f}(t), z = \ddot{f}(t)$ and that it can be thought of, according to the authors of [15], as prototypical for prescribed-trajectory problems in mechanics, in which the calculation of the forces required for the system to accomplish the desired action is required. In such an application x, y and z would represent position, velocity and force per unit mass respectively, and f(t) is the prescribed trajectory for the system.

We now consider the overdetermined but consistent system obtained by augmenting the original system (2.33) by successive derivatives of (2.33a) and (2.33c), i.e.

$$
\begin{aligned}
x &= f(t) \\
\dot{x} &= \dot{f}(t) \\
\ddot{x} &= \ddot{f}(t) \\
y &= \dot{x} \\
\dot{y} &= \ddot{x} \\
z &= \dot{y}
\end{aligned}
$$

We now eliminate $\ddot{x}$ by putting $x'' = \ddot{x}$ i.e. use a dummy derivative x'' for $\ddot{x}$, and note that the dummy derivative is a purely algebraic variable and is not subject to discretization. In a similar way we replace $\dot{x}$ by x' and $\dot{y}$ by y' and thus we obtain the augmented but over-determined system

$$
\begin{aligned}
x &= f(t) \\
x' &= \dot{f}(t) \\
x'' &= \ddot{f}(t) \\
y &= x' \\
y' &= x'' \\
z &= y'
\end{aligned}
$$

The system is purely algebraic and hence index-1, and it is mathematically equivalent to (2.33). No initial conditions can be imposed and no discretization is required for its numerical solution. The authors of [15] note that this system is special, but also that it demonstrates that the system can be solved numerically without discretizing all derivatives. The general case is discussed inn [15] but the merits of this technique are felt, by the authors, to be the fact that the dummy derivatives are identified and excluded from discretization. As a result over-discretization of the DAE is avoided and the differentiations inherent in a high-index DAE are carried out analytically rather than numerically. In addition we note that since the algebraic equations are still present in their original form there will be no "drift" away from the solution manifold of the DAE, and hence no need for constraint stabilization.

Another idea, proposed by Gear, for reducing the index in semi-explicit systems of the form

$$\mathbf{x}' = \mathbf{f}(\mathbf{x},\mathbf{y},t)$$
$$\mathbf{0} = \mathbf{g}(\mathbf{x},\mathbf{y},t)$$

is to replace the undifferentiated variables **y** by $\mathbf{w}'$. The index of the system is reduced by 1 but the solution of the new DAE system yields only **x** and **w** and consequently the computed **w** will need to be differentiated to obtain **y**.

The strategy of eliminating Lagrange multipliers analytically to obtain a standard ODE is illustrated by considering example (1.1) again. We will introduce x = L sinϕ and y = - L cosϕ, thus the algebraic constraint is satisfied. By differentiating with respect to t we can obtain

$$L\ddot{\phi} = -g\sin\phi$$

which is an alternative form of the ODE

$$\ddot{\phi} + \frac{g}{L}\sin\phi = 0$$

which in system theory and mechanics is known as the state space form or the Lagrange equation of the second kind. We note that the authors of [14] consider that this approach may be difficult to implement in general or for very large systems.

The regularization of a DAE is defined in [4] to be the introduction of a small parameter into the DAE in such a way that the solution of the perturbed system approaches the solution of the unperturbed system as the parameter tends to zero. However, the authors of [4] suggest that, although approaches based on some regularization have definitely proven successful in some applications, at the time of writing of [4] results on regularization as a general technique for solving higher index systems are inconclusive.

SECTION 3 NUMERICAL METHODS AND ODEs

3.1 THE ROLE OF NUMERICAL METHODS IN SOLVING ODEs

Many problems in physics , engineering, biology, chemistry etc. can be modelled by systems of ODEs. However, as is stated by the author of [10] the number of instances where an exact solution can be found by analytical methods is very limited, and the only general class of systems for which exact solutions can always be found (subject to being able to find a particular integral) consists of linear constant coefficient systems of the form

$$y' = \mathrm{A}y + \mathrm{F}(x) \tag{3.1}$$

where A is a constant matrix.

Exact analytical solutions can be found for particular linear variable coefficient or nonlinear systems, but in general it is necessary to use either an approximate or numerical method to solve an ODE.

We will focus our attention on numerical methods but note that an approximate method, such as solutions in series or solutions which only hold asymptotically for large x, frequently produces an approximate general solution whereas a numerical method produces a particular solution satisfying given initial or boundary conditions.

In the numerical methods which we consider the idea of a discretization is involved. By this we mean that "the continuous interval [a,b] of x is replaced by the discrete point set $\{x_n\}$ defined by $x_n = a + nh$, $n = 0, 1, 2, \ldots\ldots$, $N = \left(\frac{b-a}{h}\right)$" (see [10]).

The parameter h is called the step length and y_n is used to denote an approximation to the solution $y(x_n)$ at x_n, i.e. $y_n \approx y(x_n)$. Numerical methods may take many forms, e.g.

$$y_{n+2} - y_{n+1} = \frac{h}{3}\left[3f(x_{n+1}, y_{n+1}) - 2f(x_n, y_n)\right] \tag{3.2}$$

and

$$y_{n+1} - y_n = \frac{h}{2}(k_1 + k_2) \tag{3.3a}$$

where

$$k_1 = f(x_n, y_n) \tag{3.3b}$$

$$k_2 = f\left(x_n + h, y_n + \frac{1}{2}hk_1 + \frac{1}{2}hk_2\right) \tag{3.3c}$$

Many numerical methods can be written in the general form (see [10])

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\phi_f(y_{n+k}, y_{n+k-1}, \ldots, y_n, x_n; h) \qquad (3.4)$$

where the subscript f on the RHS indicates that the dependence of ϕ on $y_{n+k}, y_{n+k-1}, \ldots, y_n, x_n$ is through the function f(x, y). Two conditions are imposed on (3.4), (see [10]). The first of these places a condition on the method and the second, a condition on the problem for it to be solvable. These conditions are not found to be restrictive since all methods we consider satisfy the first and the second follows from the assumption that the problem satisfies a Lipchitz condition.

The choice of which numerical method to use depends partly on the accuracy requirements, noting that the error for a given step size is smaller for higher order methods but also that the cost of each step is higher for a higher-order method.

3.1.1 CONVERGENCE, CONSISTENCY AND STABILITY OF NUMERICAL METHODS (applied to ODEs)

We consider the general method in the form (3.4). We require that the numerical solution $\left\{y_n, n = 0,1,\ldots,N = \frac{b-a}{h}\right\}$ where y_n is defined by (3.4) to become the exact solution y(x) of the ODE; this in general terms is the meaning of convergence but a more precise definition is given in [10] . the method is said to convergent if "for all initial value problems satisfying the hypotheses of Theorem 1.1," (the general existence and uniqueness theorem for ODEs), "we have that

$$\max_{0\le n\le N} \|y(x_n) - y_n\| \to 0 \quad \text{as} \quad h \to 0$$

We define the residual R_{n+k}, which is a measure of accuracy, by

$$R_{n+k} = \sum_{j=0}^{k} \alpha_j y(x_{n+j}) - h\phi_f(y(x_{n+k}), y(x_{n+k-1}), \ldots, y(x_n), x_n; h) \qquad (3.5)$$

and note that R_{n+k} is essentially the local truncation error. (see [5] for further details).

Consistency is defined in [10] as;-

"The method is said to be consistent if, for all initial value problems satisfying the hypotheses of Theorem 1.1, the residual R_{n+k}, defined by (3.5) satisfies $\lim_{\substack{h\to 0 \\ x=a+nh}} \frac{1}{h} R_{n+k} = 0$."

We note that convergence implies consistency but that consistency does not imply convergence.

Various types of stability exist. "Totally stable" is said by the authors of [10] to be equivalent to being "properly posed". A method is said to zero-stable, (see [10] for example), if it satisfies the root condition, i.e. "all the roots of the first characteristic polynomial have modulus less than or equal to unity, and those of modulus unity are simple", (i.e. it is a von Neumann polynomial). If the region of absolute stability of the method contains the entire left half-plane then the method is said to be A-stable (see [2] for example). The region of absolute stability of a method is "that region of the complex z-plane such that applying the method for the test equation $y' = \lambda y$, with z = hλ from within this region, yields an approximate solution satisfying the absolute stability requirement

$$|y_n| \leq |y_{n-1}| , \qquad n = 1, 2, \ldots\ldots"$$

(see [2] for further details).

The necessary and sufficient conditions for the method (3.4) to be convergent are that it is consistent and zero-stable (see [10] for example).

3.1.2 STIFFNESS: WHAT IS MEANT BY STIFFNESS?

- "Stiff equations are problems for which explicit methods don't work." [9]
- "Stiffness occurs when stability requirements, rather than those of accuracy, constrain the step length." (We note that this statement is not entirely accurate since stiffness is concerned with the accumulation of error). [10]
- "Stiffness occurs when some components of the solution decay much more rapidly than others." [10]

Stiffness cannot be defined satisfactorily in precise mathematical terms, (see [10]), even for the class of linear constant coefficient systems. A variety of qualitative statements have been made in an attempt to define the notion of stiffness, e.g.

"An IVP is stiff in some interval [0, b] if the step size needed to maintain stability of the forward Euler method is much smaller than the step size required to represent the solution accurately." (see [2])

and

"the numerical method may be restricted to using very small steps, in the case that its absolute stability region is limited, because of the presence of a fast time scale in the differential equation." (see [1]

When applying a numerical method the choice of step size should ideally be dictated, according to [2], by approximation accuracy requirements. However, for many methods h must also be chosen sufficiently small to satisfy an absolute stability restriction. If the latter restriction dictates a much smaller step size than the first then the problem is referred to as stiff.

Stiffness is often described by scientists in terms of multiple time scales. A problem is considered to be "stiff" if the solution modes or phenomena which change on fast scales are stable. Examples of sources of stiffness are

1. a controller which is designed to bring a system rapidly back to a steady state
2. chemical reactions which occur much more rapidly than others in chemically reacting systems.

In mathematical terms, in addition to the differential equation stiffness depends upon the accuracy criterion, the length of the interval of integration and the region of absolute stability of the method.

[17] notes that differential-algebraic equations are very similar to stiff systems.

e.g. If $\varepsilon >> 0$ then

$$\left(\mathrm{E} - \varepsilon \mathrm{A}\right)\mathbf{z}'(t) = \mathrm{A}\mathbf{z}(t) + \mathbf{g}(t)$$

is a stiff system near to

$$\mathrm{E}\mathbf{z}'(t) = \mathrm{A}\mathbf{z}(t) + \mathbf{g}(t).$$

3.2 COMPARISONS BETWEEN DIFFERENT TYPES OF NUMERICAL METHODS

A numerical method leads to a difference equation involving a number of consecutive approximations $y_{n+j}, j = 0,1,....,k$. Using this the sequence $\{y_n | n = 0,1,2,...,N$ can be computed sequentially. K is the step number of the method e.g. if k = 1 then the method is a one-step method and if k > 1 then the method is a multistep or k-step method.

e.g. $y_{n+2} - y_{n+1} = \frac{h}{3}\left[3f(x_{n+1}, y_{n+1}) - 2f(x_n, y_n)\right]$ is an example of a 2-step method since y_{n+1} and y_n are needed to compute y_{n+2}.

For a one-step method we require one initial given value, say y_0, but for a k-step method we require k initial values, say $y_0, y_1, y_2, ..., y_{k-1}$ to start the method off.

Euler's rule is the simplest of all numerical methods, and it uses the difference equation $y_{n+1} = y_n + hf_n$. It is linear in y_n and f_n and because it is a one-step method changing the step length h presents no difficulty. However, it has very low accuracy.

A linear multistep method involves only linear combinations of $y_{n+j}, f(x_{n+j}, y_{n+j}), j = 0,1,\ldots,k$. These methods achieve higher accuracy by retaining linearity with respect to y_{n+j} and f_{n+j}, j = 0, 1, ...,k but lose the one-step format. Considerable difficulties are encountered when we want to change the step length.

Runge-Kutta methods also developed from Euler's rule, but, in contrast to the linear multistep methods the one-step form is retained but the linearity is sacrificed. Consequently, changing the step length is not problematic but the structure of the local error is more complex making it hard to decide when to change the step length. We note that although these methods are said in [19] to be advantageous over multistep methods for some systems, e.g. systems with frequent discontinuities, care must be taken when choosing a Runge-Kutta method which is appropriate for DAEs since in general they do not achieve the same order of accuracy for DAEs as they do for ODEs.

An implicit Runge-Kutta method attains a higher order for a given number of stages than an explicit Runge-Kutta method. Many of the most commonly used implicit Runge-Kutta methods are based on quadrature methods, and according to [2] these can be divided into several classes. Gauss methods are the maximum order methods; Radau methods correspond to quadrature rules where one end of the interval is included; Lobatto methods involve sampling the function at both ends of the interval. Implicit Runge-Kutta methods can also be categorised according to whether they are collocation methods. The basic principle behind collocation methods is to choose a function (usually a polynomial) from a simple space and a set of collocation points. We then require the function to satisfy the given problem equations at the collocation points.

Multistep methods attributed to Adams were derived by integrating the polynomial which interpolates past values of f but BDF methods are derived by differentiating this polynomial and setting the derivative at t_n equal to $f(t_n, y_n)$.

3.3 NUMERICAL METHODS: FURTHER DETAIL

We will now restrict ourselves to the numerical methods relevant to the material in section 4, and give a brief outline of each method, noting relevant features concerning order, stability and convergence. In Table 3.1 we provide information about different types of methods and in Table 3.2 we include information about specific methods, each time indicating a reference for it's source. Since the focus of this thesis is on DAEs the reader should consult material specifically written about numerical methods and ODEs e.g. [1] or [10] for further information.

3.3.1 LINEAR MULTISTEP METHODS

The general form of a linear multistep method, or linear k-step method, is given in [2] by

$$\sum_{j=0}^{k}\alpha_j \mathbf{y}_{n-j} = h\sum_{j=0}^{k}\beta_j \mathbf{f}_{n-j} \qquad (3.6)$$

where α_j, β_j are the method's coefficients. We note that the method is implicit if $\beta_0 \neq 0$ and explicit if $\beta_0 = 0$. It is assumed that $\alpha_0 \neq 0$ and $|\alpha_k| + |\beta|_k \neq 0$ and to eliminate arbitrary scaling we set $\alpha_0 = 1$. The past k integraions steps are assumed to be equally spaced.

The most popular linear multistep methods are based on polynomial interpolation, and the methods typically come in families. The most popular for non-stiff problems is the Adams family and for stiff problems it is the backward differentiation formula (BDF).

For all Adams methods we set $\alpha_0 = 1, \alpha_1 = -1, \alpha_j = 0$, j > 1. The explicit Adams methods, also called Adams-Bashforth methods, use the difference equation

$$y_n = y_{n-1} + h\sum_{j=1}^{k}\beta_j f_{n-j}$$

where $$\beta_j = (-1)^{j-1}\sum_{i=j-1}^{k-1}\binom{i}{j-1}\gamma_i$$

$$\gamma_i = (-1)^i \int_0^1 \binom{-s}{i} ds \qquad \text{(see [2])}$$

The implicit Adams methods, also called Adams-Moulton methods, use the difference equation

$$y_n = y_{n-1} + h\sum_{j=0}^{k}\beta_j f_{n-j}$$

A table of coefficients, β_j, can be found in [2] for methods up to order 6.

We note that k = 1, $\beta_1 = 0$ gives the Backward Euler method, and k = 1 with $\beta_1 \neq 0$ gives the implicit trapezoidal method. Adams-Moulton methods have smaller error constants and use one fewer step than the Adams-Bashforth of the same order. Their stability regions are much larger but they have the disadvantage of being implicit. Adams-Moulton methods are often used together with Adams-Bashforth methods for the solution of non-stiff ODEs in a type of implementation called predictor-corrector.

In [2] we find the characteristic polynomials defined as

$$\rho(\xi) = \sum_{j=0}^{k} \alpha_j \xi^{k-j}$$

$$\sigma(\xi) = \sum_{j=0}^{k} \beta_j \xi^{k-j}$$

for the k-step linear multistep method given by (3.6).

The linear multistep method is consistent iff $\rho(1) = 0, \rho'(1) = \sigma(1)$, stable if all k roots of $\phi(\xi)$ satisfy $|\xi_i| \leq 1$ with $|\xi_i| = 1$ requiring that ξ_i is a simple root and asymptotically stable if all roots satisfy $|\xi_i| < 1$, where $\phi(\xi) = \sum_{j=0}^{k} \alpha_j \xi^{k-j} = 0$.

The method is convergent to order p if the root condition is satisfied and the method and the initial values are accurate to order p. It is strongly stable if all roots of $\rho(\xi) = 0$ are inside the unit circle except for the root ξ =1 and weakly stable if it is 0-stable but not strongly stable.

(see [2]).

3.3.2 BDF METHODS

The difference equation, in scaled form (i.e. $\alpha_0 = 1$), for the BDF methods is given in [2] as

$$\sum_{i=0}^{k} \alpha_i y_{n-i} = h\beta_0 f(t_n, y_n) \qquad \text{where} \quad \alpha_0 = 1. \tag{3.7}$$

Backward Euler is the first BDF method and its associated difference equation is $y_{n+1} - y_n = hf_{n+1}$. This can be seen to follow from (3.7) if k = 1, since (3.7) would become

$$\sum_{i=0}^{1} \alpha_i y_{n-i} = h\beta_0 f(t_n, y_n) \tag{3.8}$$

which can be rewritten as

$$\alpha_0 y_n + \alpha_1 y_{n-1} = h\beta_0 f(t_n, y_n) \tag{3.9}$$

and since $\alpha_1 = -1.\beta_0 = 1$ (see[2]) this becomes

$$y_n - y_{n-1} = hf(t_n, y_n)$$

According to the authors of [2] the distinguishing feature of BDF is that **f**(t, **y**) is only evaluated at the right end of the current step (t_n, y_n). They are usually implemented together with a modified Newton method to solve the nonlinear system at each time step.

TABLE 3.1 ORDER, CONVERGENCE AND STABILITY FOR GENERAL NUMERICAL METHODS APPLIED TO ODEs

METHOD	TYPE	ORDER	CONVERGENCE	STABILITY	NOTES	REF	PAGE
GAUSS METHODS (IMPLICIT)	S-STAGE	2S		A-STABLE	• MAXIMUM ORDER METHODS • COLLOCATION METHODS • SYMMETRIC METHODS	[2] [10] [1]	 190 215
LOBATTO METHODS	S-STAGE	2S - 2		A-STABLE	• BOTH ENDS OF INTERVAL INCLUDED • SYMMETRIC METHODS • COLLOCATION METHODS	[1] [2] [10]	215 191
RADAU METHODS	S-STAGE	2S - 1			• ONE END OF INTERVAL IS INCLUDED • COLLOCATION METHODS • PARTICULARLY SUITABLE FOR SOLVING STIFF INITIAL VALUE ODEs	[2] [1] [10]	103 215 191
ADAMS-BASHFORTH (EXPLICIT ADAMS METHODS)	K-STEP		see [2] chapter 5		• MOST POPULAR AMONG EXPLICIT MULTISTEP METHODS • VERY SMALL REGIONS OF ABSOLUTE STABILITY	[2]	
ADAMS-MOULTON (IMPLICIT ADAMS METHODS)	K-STEP	$p = k + 1$ for $k \neq 1$	see [2] chapter 5		• LARGER STABILITY REGION THAN ADAMS-BASHFORTH • USED WITH ADAMS-BASHFORTH FOR NON-STIFF ODEs (predictor-corrector)	[2] [10]	127 86
BDF (IMPLICIT)	K-STEP		see [2] section 5.2.3	O-stable for $k<7$, unstable for $k\geq 7$	• MOST POPULAR FOR STIFF PROBLEMS • ADDITIONAL STARTING VALUES MUST BE $O(h^p)$ ACCURATE FOR A METHOD OF ORDER p IF FULLL CONVERGENCE ORDER IS TO BE REALISED	[2] [4] [8] [16]	140 42 380 263

TABLE 3.1 (continued)

METHOD	TYPE	ORDER	CONVERGENCE	STABILITY	NOTES	REF	PAGE
LINEAR MULTISTEP METHOD	K-STEP	≤ (k + 1) if strongly stable	see [8], chapter III.4 and [2], section 5.2.3	A-stable for given $\hat{h}$ if for that $\hat{h}$ all the roots of the stability polynomial satisfy $\lvert r_s \rvert < 1$, s=1,2,...k and to be absolutely unstable for that $\hat{h}$ otherwise	• ORDER OF AN A-STABLE METHOD IS ≤2 • CONSISTENT IF IT HAS ORDER p≥1 • AN EXPLICIT M.S.M CANNOT BE A-STABLE • A-STABILITY IS VERY DIFFICULT TO ATTAIN • STABILITY POLYNOMIAL IS $\pi(r,\hat{h}) = \rho(r) - \hat{h}\sigma(r)$ where $\hat{h} = h\lambda$	[2] [10]	70
EXPLICIT RUNGE-KUTTA METHOD	S-STAGE	variable, ≤s see [1], page 87 see [5]	see [8] chapter II.3	• $\lvert R(z) \rvert \leq 1$ is the stability condition where $y_n = R(z)y_{n-1}$	• INAPPROPRIATE FOR STIFF PROBLEMS • ALL S-STAGE WITH ORDER s HAVE THE SAME REGION OF ABSOLUTE STABILITY • FOR ORDER <s THE ABSOLUTE STABILITY REGION DEPENDS TO SOME EXTENT ON THE METHOD'S COEFFICIENTS	[2] [10]	90 177
IMPLICIT RUNGE-KUTTA METHOD			see [8] chapter II.3		• COLLOCATION METHODS • APPROPRIATE FOR STIFF PROBLEMS		

TABLE 3.2 ORDER AND STABILITY FOR SPECIFIC METHODS APPLIED TO ODEs

METHOD	TYPE	ORDER	STABILITY	NOTES	REF	PAGE
FORWARD EULER	• A-B • R-K	1	O-STABLE	• VERY SMALL REGION OF ABSOLUTE STABILITY • CONVERGENT OF ORDER 1	[2]	41
BACKWARD EULER	• R • A-M • BDF	1	• A-STABLE • 0-STABLE		[2] [2] [2]	56 60 52
TRAPEZOIDAL RULE	• L • A-M • R-K	2	• A-STABLE • 0-STABLE	A SYMMETRIC METHOD	[2]	60
IMPLICIT MIDPOINT METHOD	• 1-STAGE GAUSS • R-K	2	• A-STABLE	A SYMMETRIC METHOD	[2]	68
EXPLICIT MIDPOINT METHOD	• R-K • 2-STAGE	2		CONSISTENT	[2]	
EXPLICIT TRAPEZOIDAL METHOD	• R-K • 2-STAGE	2				
CLASSICAL FOURTH-ORDER RUNGE-KUTTA METHOS	• R-K	4		CLOSELY RELATED TO SIMPSON'S QUADRATURE RULE		

KEY

A-B ADAMS-BASHFORTH
R-K RUNGE-KUTTA
L LOBATTO
R RADAU
A-M ADAMS-MOULTON

3.4 NEWTON'S METHOD

We include this section since, according to the authors of [2], variants of Newton's method are used in nearly all modern stiff ODE (computer) codes. We note that for a nonlinear equation $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ a sequence of iterates is defined as follows:-

$\mathbf{x}^0$ is an initial guess

$\mathbf{x}^{\nu}$ is a current iterate

Neglecting higher-order terms in the Taylor expansion we can define the next iterate $\mathbf{x}^{\nu+1}$ by the linear equation

$$\mathbf{0} = \mathbf{g}(\mathbf{x}^{\nu}) + \mathbf{g}'(\mathbf{x}^{\nu})(\mathbf{x}^{\nu+1} - \mathbf{x}^{\nu})$$

Replacing the first derivative of **g** by the Jacobian matrix $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ we obtain the iteration

$$\mathbf{x}^{\nu+1} = \mathbf{x}^{\nu} - \left(\frac{\partial \mathbf{g}(\mathbf{x}^{\nu})}{\partial \mathbf{x}}\right)^{-1} \mathbf{g}(\mathbf{x}^{\nu}) \qquad \nu = 0, 1, \ldots\ldots$$

However, computing a matrix inverse is not considered to be good practice in [2] and it is generally better to solve the linear system for the difference δ between $\mathbf{x}^{\nu+1}$ and $\mathbf{x}^{\nu}$, and then update. Hence δ is computed by solving the linear system $\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)\delta = -\mathbf{g}(\mathbf{x}^{\nu})$ with the Jacobian matrix evaluated at $\mathbf{x}^{\nu}$.

The next Newton iterate is obtained by $\mathbf{x}^{\nu+1} = \mathbf{x}^{\nu} + \delta$.

For the nonlinear system $\mathbf{g}(\mathbf{y}_n) = \mathbf{y}_n - \mathbf{y}_{n-1} - hf(t_n, \mathbf{y}_n) = 0$ Newton's method gives

$$\mathbf{y}_n^{\nu+1} = \mathbf{y}_n^{\nu} - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)^{-1} \mathbf{g}(\mathbf{y}_n^{\nu})$$

$$\mathbf{y}_n^{\nu} - \left(\mathrm{I} - \mathrm{h}\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^{-1} \left(\mathbf{y}_n^{\nu} - \mathbf{y}_{n-1} - h\mathbf{f}(t_n, \mathbf{y}_n^{\nu})\right), \quad \nu = 0, 1, \ldots.$$

$\left(\mathrm{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)$ is known as the iteration matrix and is evaluated at the current iterate $\mathbf{y}_n^{\nu}$.

We note that the costs of forming it and solving the linear system, (for $\delta = \mathbf{y}^{v+1} - \mathbf{y}_n^v$), are often the major factor in the costs of solving the problem.

The initial guess can be taken as $y_n^0 = y_{n-1}$ although better ones are often available.

Newton's method is iterated until the error estimate due to terminating the iteration is less than a user-specified tolerance, e.g. $\left|\mathbf{y}_n^{v+1} - \mathbf{y}_n^v\right| \leq$ NTOL.

SECTION 4: NUMERICAL APPROACHES TO SOLVING DAEs

We now discuss numerical methods applied to DAEs. We consider the regularization of the DAE

$$\mathbf{x}' = \mathbf{f}(t,\mathbf{x},\mathbf{z}) \tag{4.1a}$$

$$\mathbf{0} = \mathbf{g}(t,\mathbf{x},\mathbf{z}) \tag{4.1b}$$

in which (4.1b) is replaced by

$$\varepsilon\,\mathbf{z}' = \mathbf{g}(t,\mathbf{x},\mathbf{z}) \tag{4.1c}$$

where $0 \le \varepsilon << 1$.

The regularised ODE (4.1a) and (4.1c) is very stiff, hence we are led to consider methods for stiff ODEs for the direct discretization of the limit DAE.

4.1 DIRECT DISCRETIZATION METHODS

In a direct discretization method we approximate $\mathbf{y}$ and $\mathbf{y}'$ by a discretization formula e.g. multistep or Runge-Kutta. We begin with the simplest method, namely the Backward Euler Method.

4.1.1 THE BACKWARD EULER METHOD

If we consider only the simplest class of nonlinear DAEs i.e. semi-explicit index-1, we find that the backward Euler retains the order, stability and convergence properties from the ODE case. Consider

$$\mathbf{x}' = \mathbf{f}(t,\mathbf{x},\mathbf{z}) \tag{4.2a}$$

$$\mathbf{0} = \mathbf{g}(t,\mathbf{x},\mathbf{z}) \tag{4.2b}$$

where $\mathbf{g}_z$ is nonsingular.

the DAE (4.2) is equivalent to the ODE

By the implicit function theorem $\tilde{\mathbf{g}}$ exists such that $\mathbf{z} = \tilde{\mathbf{g}}(t,\mathbf{x})$

Hence

$$\mathbf{x}' = \mathbf{f}\big(t,\mathbf{x},\tilde{\mathbf{g}}(t,\mathbf{x})\big) \tag{4.3}$$

We will now apply Backward Euler to (4.2)

$$\frac{\mathbf{x}_n - \mathbf{x}_{n-1}}{h_n} = \mathbf{f}(t_n, \mathbf{x}_n, \mathbf{z}_n) \tag{4.4a}$$

$$\mathbf{0} = \mathbf{g}(t_n, \mathbf{x}_n, \mathbf{z}_n) \tag{4.4b}$$

Hence $\exists\ \tilde{\mathbf{g}}$ such that $\mathbf{z}_n = \tilde{\mathbf{g}}(t_n, \mathbf{x}_n)$

$$\therefore \quad \frac{\mathbf{x}_n - \mathbf{x}_{n-1}}{h_n} = \mathbf{f}(t_n, \mathbf{x}_n, \tilde{\mathbf{g}}(t_n, \mathbf{x}_n)) \tag{4.5}$$

We note that (4.5) is just the Backward Euler discretization of (4.3).

We now consider the general DAE

$$\mathbf{0} = \mathbf{F}(t, \mathbf{y}, \mathbf{y}') \tag{4.6}$$

and apply the Backward Euler method to (4.6) to give

$$\mathbf{0} = \mathbf{F}\left(t_n, \mathbf{y}_n, \frac{\mathbf{y}_n - \mathbf{y}_{n-1}}{h_n}\right)$$

which generally gives a system of m nonlinear equations for y_n at each time step n.

The following example illustrates that even this simple method does not always work.

Example 4.1 (taken from [2])

$$\begin{pmatrix} 0 & 0 \\ 1 & \eta t \end{pmatrix} \mathbf{y}' + \begin{pmatrix} 1 & \eta t \\ 0 & 1+\eta \end{pmatrix} \mathbf{y} = \begin{pmatrix} q(t) \\ 0 \end{pmatrix} \tag{4.7}$$

This is a linear index-2, variable coefficient DAE which depends upon a parameter η. We can rewrite (4.7) as

$$y_1 + \eta t y_2 = q(t) \tag{4.8a}$$

$$y_1' + \eta t y_2' + (1+\eta) y_2 = 0 \tag{4.8b}$$

If we differentiate (4.8a) we obtain

$$y_1' + \eta t y_2' + \eta y_2 = q'(t) \tag{4.9}$$

We now rearrange (4.9) and substitute in (4.8b) to give the solution

$$y_2 = -q'(t). \tag{4.10}$$

Upon substitution from (4.10) into (4.8a) we obtain

$$y_1 = q(t) + +\eta t q'(t),$$

and we note that this exact analytical solution is well defined for all values of η.

We will now consider a direct discretization of (4.7) using Backward Euler.

$$\begin{pmatrix} 0 & 0 \\ 1 & \eta t_n \end{pmatrix}\left(\frac{\mathbf{y}_n - \mathbf{y}_{n-1}}{h_n}\right) + \begin{pmatrix} 1 & \eta t_n \\ 0 & 1+\eta \end{pmatrix}\mathbf{y}_n = \begin{pmatrix} q(t_n) \\ 0 \end{pmatrix} \tag{4.11}$$

We rewrite (4.11) to give

$$y_{1,n} + \eta t_n y_{2,n} = q(t_n) \tag{4.12a}$$

$$\frac{y_{1,n} - y_{1,n-1}}{h_n} + \eta t_n\left(\frac{y_{2,n} - y_{2,n-1}}{h_n}\right) + (1+\eta)y_{2,n} = 0 \tag{4.12b}$$

Using (4.12a) we find that (4.12b) is equivalent to

$$\frac{q(t_n)}{h_n} - \frac{q(t_{n-1})}{h_n} + (1+\eta)y_{2,n} = 0 \tag{4.13}$$

We can see from (4.13) that $y_{2,n}$ is undefined for η = -1, in contrast with the analytical solution where no restriction is placed on η.

4.1.2 BDF AND GENERAL MULTISTEP METHODS

The constant step-size BDF method applied to a general nonlinear DAE in the form (4.6) is given in [2] by

$$\mathbf{F}\left(t_n, \mathbf{y}_n, \frac{1}{\beta_0 h}\sum_{j=0}^{k}\alpha_j \mathbf{y}_{n-j}\right) = \mathbf{0}$$

where β_0 and α_j, j = 0, 1, ..., k are the coefficients of the BDF method.

4.1.3 RUNGE-KUTTA METHODS

The s-stage implicit Runge-Kutta method is defined in [2] by

$$\mathbf{0} = \mathbf{F}(t_i, \mathbf{Y}_i, \mathbf{K}_i)$$

$$t_i = t_{n-1} + C_i h \qquad i = 1,2, \ldots, s$$

$$\mathbf{Y}_i = \mathbf{y}_{n-1} + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j$$

and $$\mathbf{y}_n = \mathbf{y}_{n-1} + h\sum_{i=1}^{s} b_i \mathbf{k}_i \,.$$

We note that the coefficient matrix A = (a_{ij}) is assumed to be nonsingular.

4.2 NUMERICAL METHODS FOR DAEs

Historically the first numerical methods to be implemented for DAEs were the BDF. These methods are suitable for solving a wide variety of DAE problems (see [19]) and form the basis for the most widely used computer codes for DAEs. Higher index DAEs pose problems. For example, convergence even with Backward Euler may not happen (see [19]). In addition this method can be unstable for small step sizes (see [19]). Algorithms requiring the extensive use of symbolic or automatic differentiation are, according to [19], "the only known means for dealing with the most general high-index DAE numerically." Results on the instability of numerical ODE methods when applied directly to high-index linear systems have been discouraging but some important classes of high-index nonlinear problems have been identified for which the numerical methods are stable and accurate.

Table 4.1 gives an indication of which numerical methods can be used to solve different types of DAEs. Reasons are given where the methods are known not to be suitable for a particular type of DAE and the origin of the information is shown. Further information regarding the convergence, order etc. of a particular method applied to different DAE types is given in tables A to E, together with a reference for the source of the information.

TABLE 4.1 APPROPRIATE NUMERICAL METHODS FOR SOLVABLE DAES

	NUMERICAL METHOD				
TYPE OF DAE	K-STEP CONSTANT STEP-SIZE BDF	VARIABLE STEP-SIZE BDF	GENERAL MULTISTEP METHODS	RUNGE-KUTTA METHODS	BACKWARD EULER
LINEAR CONSTANT COEFFICIENT	TABLE A ROW 1	TABLE B ROW 1	TABLE C ROW 1	TABLE D ROW 1	
LINEAR TIME-VARYING	TABLE A ROW 2	TABLE B ROW 2			TABLE E ROW 1
SEMI-EXPLICIT INDEX-1	TABLE A ROW 3	TABLE B ROW 6	TABLE C ROW 2	TABLE D ROW 2	TABLE E ROW 2
SEMI-EXPLICIT INDEX-2	TABLE A ROW 4	TABLE B ROW 3			TABLE E ROW 3
HESSENBERG INDEX-1	TABLE A ROW 6	TABLE B ROW 10	TABLE C ROW 4	TABLE D ROW 4	
HESSENBERG INDEX-2		TABLE B ROW 10	TABLE C ROW 3	TABLE D ROW 3	
HESSENBERG INDEX-3	TABLE A ROW 5	TABLE B ROW 4			
HESSENBERG INDEX-4		TABLE B ROW 5			
HESSENBERG INDEX-V		TABLE B ROW 10			
GENERAL INDEX-1	TABLE A ROW 6	TABLE B ROW 6	TABLE C ROW 3	TABLE D ROW 4	TABLE E ROW 4
TRANSFERABLE INDEX-1	TABLE A ROW 7		TABLE C ROW 4		
GENERAL INDEX ≥ 2			**XXX SEE NOTE 1**		**XXX SEE NOTE 2**
LINEAR CONSTANT COEFFICIENT INDEX V		TABLE B ROW 7			
UNIFORM INDEX-1	TABLE A ROW 8	TABLE B ROW 8			
NONLINEAR SEMI-EXPLICIT INDEX-2	TABLE A ROW 9	TABLE B ROW 9			
FIXED INDEX-2 0R 3 SEMI-EXPLICIT	TABLE A ROW 10				

NOTE 1 Not stable and convergent - see [4], page 46

NOTE 2 Not convergent - see [19], page 127

TABLE A K-step constant step size BDF

ROW	DAE TYPE	NOTES	REF	PAGE
1	LINEAR CONSTANT COEFFICIENT	For k < 7 (index-v), it is convergent and stable; accurate to order $O(h^k)$ after a maximum of (v - 1)k + 1 steps	[3] [4] [7] [14]	314 44 720 244
2	LINEAR TIME-VARYING	the system needs to be expressible in SCF; for k < 7 it is transient stable	[3]	314, 318 and 320
3	SEMI-EXPLICIT INDEX-1	An order k method converges with order k	[14]	246
4	SEMI-EXPLICIT INDEX-2	With sufficiently accurate initial values it converges to order $O(h^k)$, under appropriate assumptions, after (k + 1) steps	[18] [19]	263 127
5	HESSENBERG INDEX-3	for k < 7 it converges with kth order accuracy after k + 1 steps under conditions on starting values and algebraic equations	[4]	57
6	GENERAL INDEX-1	for k < 7 it converges to $O(h^k)$ if all initial values are correct to $O(h^k)$ and if the Newton iteration on each step is solved to accuracy $O(h^{k+1})$.	[2] [18]	266 263
7	TRANSFERABLE INDEX-1	for k < 7 it is stable for small enough h	[4]	70
8	UNIFORM INDEX-1	for k < 7 it converges to $O(h^k)$ if all initial values are correct to $O(h^k)$ and if the Newton iteration on each step is solved to accuracy $O(h^{k+1})$	[4] [18] [19]	51 263 127
9	NONLINEAR SEMI-EXPLICIT INDEX-2	convergent and globally accurate to $O(h^k)$ after (k + 1) steps - subject to conditions on the errors.	[4]	55
10	FIXED INDEX-2 AND 3	for $g_y \cdot f_u$ nonsingular it converges to $O(h^k)$ accuracy after (k + 1) steps.	[3]	315

TABLE B Variable Step Size BDF

ROW	DAE TYPE	NOTES	REF	PAGE
1	LINEAR CONSTANT COEFFICIENT	unsuitable for DAE with arbitrary index	[7]	720
2	LINEAR TIME-VARYING	possibility of stability problems even with index-2	[7]	722
3	SEMI-EXPLICIT INDEX-2	With sufficiently accurate initial values it is suitable provided that the method is stable for standard ODEs;	[2] [4] [18]	267 56 264
4	HESSENBERG INDEX-3	converges if order of the method is > 1	[18] [19]	264 128
5	HESSENBERG INDEX-4	convergent if the order of the method >3 BUT there are practical problems to deal with.	[18] [19]	264 128
6	GENERAL INDEX-1	for k < 7 it is convergent provided the method is stable for standard ODEs (see table A, row 6)	[2] [4]	266 54
7	LINEAR CONSTANT COEFFICIENT INDEX-V	If the ratio of adjacent steps is kept bounded then the global error in the solution is $O(h_{\max}^q)$ where q = min(k, k - v +2)	[4]	45
8	UNIFORM INDEX-1	convergent if method is stable for standard ODEs.	[4]	54
9	NONLINEAR SEMI-EXPLICIT INDEX-2	convergent if the method is stable for standard ODEs	[4]	56
10	HESSENBERG INDEX-V	has order (k - v + 2) for index v provided that the step size sequence is stable for ODEs	[4]	57

TABLE C General Multistep Methods

ROW	DAE TYPE	NOTES	REF	PAGE
1	LINEAR CONSTANT COEFFICIENT	Asymptotically stable under certain conditions	[20]	5
2	SEMI-EXPLICIT INDEX-1	stable and convergent to the same order of accuracy as for standard non-stiff ODEs	[4] [18]	47 264
3	HESSENBERG INDEX-2 AND GENERAL INDEX-1	coefficients must satisfy order conditions, in addition to those for ODEs, to achieve order >2	[2]	267
4	TRANSFERABLE INDEX-1	stable if the method is stable for explicit initial value problems	[4]	70

TABLE D Runge-Kutta Methods

ROW	DAE TYPE	NOTES	REF	PAGE
1	LINEAR CONSTANT COEFFICIENT	Asymptotically stable under strict stability conditions	[20]	7
2	SEMI-EXPLICIT INDEX-1	additional set of order conditions needed; the constraint equations are satisfied exactly at each step	[2] [18]	269 266
3	HESSENBERG INDEX-2	additional order conditions needed to achieve order > 2 - and other problems	[2]	269
4	GENERAL INDEX-1	additional order conditions needed to achieve order > 2; stable if method's coefficients satisfy $\|r\| < 1$	[2] [3] [18]	269 315 254, 265

TABLE E Backward Euler

ROW	DAE TYPE	NOTES	REF	PAGE
1	LINEAR TIME-VARYING	stability problems; Taylor type methods called (l, j) methods	[3] [3]	314 315
2	SEMI-EXPLICIT INDEX-1	first-order accurate, stable and convergent if $\mathbf{g}_z$ is non-singular	[2]	265
3	SEMI-EXPLICIT INDEX-2	convergent; solution accurate to O(h) after 2 steps have been taken	[2]	266
4	GENERAL INDEX-1	Decouple the system into differential and algebraic parts - BUT problems!	[2]	265, 275

4.3 AN INTRODUCTION TO THE COMPUTER CODES USED TO SOLVE DAEs

Solving problems from applications is the most compelling reason for developing methods and analysis. However, the codes used to implement the methods need to be efficient, robust, easy to use and well documented.

Several codes are available for solving index-one DAEs, but in the views of the authors of [4] the most widely used production code at this time is DASSL (Differential-Algebraic Systems SoLver), which is designed for solving IVPs in the implicit form $\mathbf{F}(t, \mathbf{y}, \mathbf{y}') = \mathbf{0}$ which are index zero or one.

DASSL approximates the derivative using the kth order BDF, where k ranges from 1 to 5. It selects the order k and stepsize h_{n+1} at every step and its choice is based on the behaviour of the solution. A variable stepsize, variable order fixed leading coefficient implementation of BDF formulae is used to advance the solution from one time step to the next. The standard version of DASSL solves the DAE from time `T' to `TOUT' where TOUT is specified by the user. In some problems it would be more natural to stop the code at the root of some function and an extension of DASSL, DASSLRT, is able to do this.

Three approaches can be used to extend fixed stepsize multistep methods to variable stepsize, the three formulations being referred to as fixed coefficient, variable coefficient and fixed leading coefficient. Fixed coefficient methods can be implemented very efficiently for smooth problems but for problems requiring frequent stepsize adjustments they are inefficient and possibly unstable. The most stable implementation, in the views of the authors of [4], is that produced by variable coefficient methods but, due to the fact that they have a tendency to require more evaluations of the Jacobian matrix in intervals when the stepsize is changing, they are usually considered, in the views of the authors of [4], to be less efficient than a fixed coefficient implementation for most problems. A compromise between these two methods is given by the fixed leading coefficient formulation, offering less stability and in general fewer Jacobian evaluations.

The code LSODI is, in contrast to DASSL, a fixed coefficient implementation of the BDF formulae. The question of which formulation is best for a general purpose code remains "an open question for stiff ODEs and DAEs". [see (4)]

How does a DAE code decide whether to accept or reject a step? This, in the view of the author's of [4] is one of the most important questions concerning the reliability of a DAE code. Two sources of error concern us. Firstly, the local truncation error of the method and secondly the error in interpolating to find the solution between mesh points for output purposes. The local truncation error is the amount by which the solution to the DAE fails to satisfy the BDF formula. (DASSL estimates the principal term of this error)

The code must decide which order method is to be used on the next step, irrespective of whether or not the present step is rejected. We note that DASSL and LSODI differ in their strategies used to select the order of the method.

After a step has been accepted or rejected, and the order of the method to be used for the next step has been decided, DASSL decides upon the stepsize to use on the next step. In all modern ODE solvers, the stepsize selection strategy reduces the stepsize as a response to instability. Consequently, codes do not produce a faulty solution due to the instability for large stepsizes. Instead the stepsize is reduced and they become very inefficient. (see [4])

DASSL keeps count of the number of failures since the last step, and after three consecutive error test failures the order is reduced to one and the stepsize is reduced by a factor of one quarter on every subsequent failure.

Selecting the initial stepsize is not straightforward, even for ODE codes. DASSL's strategy yields a successful step for a zero th order method and is, in the views of the authors of [4], quite conservative. DASSL also contains an option for the user to select the initial stepsize. Further detailed information concerning DASSL can be found in [4].

DASSL, according to [4], solves most DAE systems from applications without difficulty, but some problems have been encountered with some systems. These include the problem of the finite difference Jacobian calculation, inconsistent initial conditions which may cause DASSL to fail on the first step, a higher index formulation of a problem, a singular iteration matrix and inappropriate error tolerances.

Table 4.1 introduces the codes currently available for solving DAEs. We include information on the basis of the codes, on applications and note problems encountered and/or other relevant information.

DASSL is a powerful code which can handle a wide variety of problems. However, for some problems it is unsuitable, and along with the need for a more efficient code, research into developing extensions of DASSL continues. Extensions to DASSL include:-

1. DASSLRT - a root finding version, (see [18], Page 268 and [4], Page 136)
2. DASSAC - for sensitivity analysis (see [18], Page 268 and [4], Page 137)
3. DASPK - for large scale problems e.g. PDE systems in 2 or 3 dimensions. We note that the user must supply the preconditioner, and that we need to combine iterative methods with discretizations developed for DAEs. (see [18], Page 269 and [2], Page 290)
4. DASSLSO and DASPKSO - for sensitivity analysis. (see[4], Page 229)

TABLE 4.2

CODE	METHOD BASED ON	APPLICATION	PROBLEMS/NOTES	REF	PAGE
DASSL	BDF methods - uses a fixed leading coefficient implementation	Index-0 or Index-1 Initial value problems in fully implicit form $F(t, y, y') = 0$.	1. It can be difficult to distinguish between a failure due to inconsistent initial conditions and one due to a higher index formulation. 2. Requires consistent initial conditions. 3. Improvements are currently being researched	[4] [4] [18] [2]	 270 290
RADAU5	Three-stage Radau II A method of order 5	• Index-1, 2, or 3 of the form $By' = f(t, y)$, $y(t_0) = y_0$ where B is a constant, square matrix which may be singular. • Non stiff linearly implicit DAEs.	For higher index systems the user must be able to identify the higher index variables.	[4] [2] [19]	215 290 131
LIMEX	Extrapolation of semi-explicit Euler method. (Non BDF code)	• Linearly implicit DAEs of the form $A(t, y)y' = f(t, y)$. • Semi-explicit index-1 • Problems arising in combustion modelling where there are frequent discontinuities in time.	It attempts to diagnose failures in the first step but the techniques used cannot distinguish between systems of higher index and inconsistent initial conditions for an index-1 problem	[4] [4]	108 117
LSODI	BDF methods - uses a fixed coefficient implementation of the BDF formulae.	• Linearly implicit DAEs of the form $A(t, y)y' = f(t, y)$.	1. The user must supply a subroutine for evaluating the matrix A times a vector. 2. It differs from DASSL in several - most notably in the stepsize and order selection and error control strategies.	[4] [4]	116 117
FACSIMILE	BDF methods	Semi-explicit index-1		[4]	117
SPRINT	BDF methods	• Linearly implicit DAEs of the form $A(t, y)y' = f(t, y)$. • A wide variety of applications using the method of lines	It uses the filtered error estimate described in [4]	[4]	117
MEXX	Extrapolation methods based on half-explicit Euler or half-explicit midpoint formulae.	• Equations of motion of constrained multibody systems. • Non-stiff mechanical systems:- index-1 or 2	It exploits the structure of mechanical systems for greater efficiency.	[4] [2] [19]	216 290 131
COLDAE	Projected Gauss Collocation	• BVPs • Semi-explicit index-2 in the form $x' = f(t, x, z)$, $0 = g(t, x, z)$	An additional singular value decomposition decouples algebraic variable of different indexes if needed.	[2] [2]	290 215

4.4 FINDING CONSISTENT INITIAL CONDITIONS

The authors of [11] state that, when an ODE method such as BDF is used to solve a DAE, "small inconsistencies in the initial values may cause the method to fail or become extremely inefficient." Numerical methods require initial values for all components and although consistent initial values can sometimes be determined from physical considerations their calculation often requires greater understanding of the underlying relationships between variables than is needed to actually state the problem. In consequence such methods for DAEs can create extra challenging work for the user. The authors of [4] regard the determination of consistent initial conditions as "often the most difficult "part of solving a DAE system.

When we use the term initial conditions we are referring to the vector $(\mathbf{x}_0, \dot{\mathbf{x}}_0, \mathbf{y}_0)$. We now consider two examples, both taken from [16].

Example 4.2 Consider the DAE (4.14)

$$\dot{x} = x + y \tag{4.14a}$$

$$0 = x + 2y + a(t) \tag{4.14b}$$

where a(t) is a given continuous and differentiable function of time.

If we differentiate (4.14b) with respect to t we obtain

$$0 = \dot{x} + 2\dot{y} + a'(t)$$

If we set $\dot{y}_0 = -\frac{1}{2}(\dot{x}_0 + a'(t_0))$ then we can satisfy the equation for all initial values of the variable $\dot{x}_0$. i.e. in this case no further constraints are imposed on the initial conditions.

Example 4.3 Consider the DAE (4.15)

$$\dot{x}_1 + \dot{x}_2 = a(t) \tag{4.15a}$$

$$x_1 + x_2^2 = b(t) \tag{4.15b}$$

where a(t) and b(t) are continuous and differentiable functions of time. If we differentiate (4.15b) we obtain

$$\dot{x}_1 + 2x_2\dot{x}_2 = b'(t) \tag{4.16}$$

In this case the initial conditions are $(x_{1_0}, x_{2_0}, \dot{x}_{1_0}, \dot{x}_{2_0})$ and they must also satisfy (4.16) in addition to the original system. i.e. extra constraints are imposed on the initial conditions. From these two examples we can begin to appreciate that it would be useful to identify which constraints, when differentiated, impose further constraints on the vector of the initial conditions. Pantelides in [16] uses a graph-theoretical algorithm to locate those subsets of

the system equations which need to be differentiated. The algorithm does not use any arithmetic operations and consequently does not produce problems associated with numerical algorithms,

e.g. rounding errors. However, a price is paid for this advantage since it is possible that an equation subset which should be differentiated may not be detected as a consequence of the sufficiency but not necessity of one of the criterion involved. (see [16]). In addition the algorithm requires knowledge of analytical expressions for various derivatives of the problem and the authors of [4] regard this as a drawback for large scientific problems.

In [5] the consistent initialisation of a class of DAEs in the form $\mathbf{G}(t.\mathbf{y},\mathbf{y}') = \mathbf{0}$, which includes semi-explicit index-one systems, is considered. Two initialisation problems are considered. For the first problem, referred to as "**Initialisation Problem 1**", such systems are characterised in [5] by splitting the dependent variable vector **y** into a vector **u** of size N_d, called the differential variables, and a vector **v** of size N_a, called the algebraic variables such that the equations have the form

$$\mathbf{u}' = \mathbf{f}(t,\mathbf{u},\mathbf{v})$$
$$\mathbf{0} = \mathbf{g}(t,\mathbf{u},\mathbf{v})$$

in which $\mathbf{g}_\mathbf{v} = \dfrac{\partial \mathbf{g}}{\partial \mathbf{v}}$ is a nonsingular square matrix.

We are concerned with finding the initial value $\mathbf{v}_0$ of **v** when the initial value u_0 for **u** is specified. The class of problems can be generalised to include those in the form

$$\mathbf{f}(t,\mathbf{u},\mathbf{v},\mathbf{u}') = 0$$
$$\mathbf{g}(t,\mathbf{u},\mathbf{v}) = 0$$

(i.e. the ODE subsystem for **u** may be implicit)

where $\mathbf{u},\mathbf{f} \in \Re^{N_d}$ and $\mathbf{v},\mathbf{g} \in \Re^{N_a}$ with $\dfrac{\partial \mathbf{f}}{\partial \mathbf{u}'}$ being square and nonsingular.

"Initialisation Problem 2" is concerned with finding the initial value $\mathbf{y}_0$ for **y** when the initial value $\mathbf{y}'_0$ for $\mathbf{y}'$ is given. We note that in this problem putting $\mathbf{y}'_0 = 0$ corresponds to beginning the DAE solution at a steady state, and that no splitting of **y** into differential and algebraic parts is needed.

[5] generalises the application of the method further.

The requirement, in Initialisation Problem 1, that the differential and algebraic components are separated into blocks in **y** is first dropped and the idea of a permutation matrix P of size N is introduced such that

$$\mathbf{Py} = \mathbf{z} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}, \mathbf{u} \in \Re^{N_d}, \mathbf{v} \in \Re^{N_a}$$

Subsequently problems in which the algebraic constraints are not necessarily identified explicitly are considered. However, even with these further generalisations, the method described in [5] does not include all fully implicit index-one DAEs.

We consider the index-1 DAE system (4.17) which is well posed for any given value of $y_1(t_0)$. (see [5])

$$y_1' + y_2' = g_1(t, y_1) \tag{4.17a}$$

$$y_2 = g_2(t) \tag{4.17b}$$

We note that using $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_1 + y_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix}$ we can rewrite the DAE system as

$$\mathbf{u}' = \mathbf{g}_1(t, \mathbf{u}, \mathbf{v}) \tag{4.18a}$$

$$\mathbf{v} = \mathbf{g}_2(t) \tag{4.18b}$$

in which **u** is a differential variable and **v** is an algebraic variable.

Hence P = $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ is a permutation matrix.

The procedure outlined in [5] would find $v_0 = y_{2,0}$ correctly but the input initial value of y_2 may differ from $\mathbf{g}_2(t_0)$ and both components of $\mathbf{y}_0$ would be changed whilst preserving the value of $\mathbf{u}_0 = \mathbf{y}_1 + \mathbf{y}_2$. i.e. the procedure determines a consistent set of initial values for the transformed variables but not necessarily for the derivatives of the original variables.

An algorithm to solve initialisation problems 1 and 2, in the context of DAE solvers, such as DASSL, DASPK, is given in [5]. Users of DASPK are required to set INFO(11) equal to 0, 1 or 2 depending upon whether the initial values are already consistent, or initialisation problem 1 is to be solved, or initialisation problem 2 is to be solved respectively. The authors of [5] regard the algorithm as "convenient for users" because the extra information required, beyond that needed for solving the DAE system is minimal. Symbolic or automatic differentiation software are not needed and it is immediately applicable, in the views of the authors of [5], to a wide range of problems including very large scale systems.

SECTION 5 APPLICATIONS OF DAEs

We conclude this thesis with a section in which we explore further some of the applications referred to in section 1.7, and highlight some of the problems encountered when attempting to find numerical solutions of the relevant DAEs.

We begin with a discussion about reformulating and solving the general index-3 system for constrained mechanical motion with reference being made to methods discussed in earlier sections of the thesis. We revisit the pendulum problem and consider the differences in numerical results when different formulations of the problem are used and some of the reasons for these differences.

We move on to briefly consider electrical networks in relation to DAEs and include an example giving rise to a DAE.

5.1 CONSTRAINED MECHANICAL MOTION

Equations of constrained mechanical motion are typically developed from variational principles, resulting in a system which can be considered as a differential equation (DE) on a manifold, and which is mapped to an ODE.

In classical mechanics the resulting system is known as a Lagrange equation of the first kind i.e. a constrained DE - or a DAE. [6] gives the equation as the index-three system

$$\dot{\mathbf{p}} = \mathbf{v} \tag{5.1a}$$

$$\mathbf{M}(\mathbf{p})\dot{\mathbf{v}} = \mathbf{f}(\mathbf{p},\mathbf{v},t) - \mathrm{G}^{\mathrm{T}}(\mathbf{p})\lambda \tag{5.1b}$$

$$0 = g(\mathbf{p}) \tag{5.1c}$$

where $\mathbf{p},\mathbf{v} \in \Re^{n_p}$, $\mathbf{M}(\mathbf{p})$ is a n_p x n_p regular (symmetric, positive definite) mass matrix, $\mathbf{f}$ is a vector of applied forces and λ represents the n_λ Lagrange multipliers or constraint forces coupled to the system by the n_λ x n_p constraint matrix, $\mathbf{G} = \dfrac{\partial \mathbf{g}}{\partial \mathbf{p}}$.

[4] gives the equations in the form

$$\mathbf{M}(\mathbf{x})\mathbf{x}'' = \mathbf{g}(\mathbf{x},\mathbf{x}',t) + \mathbf{G}^{\mathrm{T}}(\mathbf{x})\lambda \tag{5.2a}$$

$$\mathbf{0} = \phi(\mathbf{x}) \tag{5.2b}$$

where $G = \phi_x$, $x \in \Re^n, \lambda \in \Re^m$ and $m \leq n$.

[19] gives the equations in an equivalent form to (5.2). We note that (5.2) is equivalent to a reformulation of (5.1) obtained by writing $\mathbf{x}' = \mathbf{v}$.

Returning to (5.1) we can rewrite this equation as an ODE, (i.e. obtain the underlying ODE to the DAE), by differentiating the constraint (5.1c) twice with respect to time. This produces n_λ constraint equations at the velocity level

$$\mathbf{0} = \frac{\partial \mathbf{g}}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial t} \qquad \text{or } \mathbf{0} = \mathrm{G}(\mathbf{p})\mathbf{v} \tag{5.3}$$

and n_λ constraint equations on the acceleration level

$$\mathbf{0} = \mathbf{G}(\mathbf{p})\dot{\mathbf{v}} + \mathbf{g}''(\mathbf{p})(\mathbf{v}.\mathbf{v}) \tag{5.4}.$$

We note that if the position-level constraint (5.1c) is replaced by the velocity level constraint (5.3) the resulting system has index-2, and if it is replaced by the acceleration-level constraint the result is an index-1 system i.e. the index has been lowered. However, in the first case the position-level constraint is no longer satisfied and in the second case the solution may drift away from satisfying both the position-level and the velocity-level constraints. The solutions to the underlying ODE are solutions of the original DAE but, due to the difficulty of obtaining accurate initial values and the presence of discretization errors in the numerical solution of the problem, they do not in general satisfy the position constraints. In the view of the authors of [6] the theoretical and practical behaviour of the drift from the position constraints is unpredictable. Stabilisation against the undifferentiated constraints is usually recommended to avoid the creation of a system with different stability properties, possibly an unstable one, which can lead to a very inefficient or even inaccurate numerical method. We remind ourselves that this can be achieved in several ways, e.g. by introducing new variables and adding new algebraic equations to the system or by a method resulting in an overdetermined system.

An alternative option is to reformulate the problem in the state-space form, i.e. as an explicit system of ODEs in a minimal set of variables which completely describe the system. We note that this transformation was necessary until recently but that such a transformation may be an exceptionally difficult problem, since according to [6] the reduction depends on the solution of nonlinear functional equations and their differentiation.

Direct discretization of the index-3 system is a third option. However, this class of DAEs was not solvable by "state-of-the-art numerical software" at the time of writing of [6] and, in the views of the authors of [19], the numerical methods yielded suffer from difficulties with error estimation, step size control and other problems, and consequently they are often not very robust. In addition to other problems the constraints can be highly nonlinear and since they have a strong physical relevance it is usually felt important that the constraints, and sometimes the time derivatives of the constraints, are satisfied very accurately. Other potential difficulties exist in that

1. the constraints can become rank-deficient or nearly rank-deficient,
2. components of the solution may be oscillating at a high frequency , and
3. frequent discontinuities are possible.

In the first case the problem becomes poorly conditioned and numerical methods can experience serious difficulties. Possible solutions include regularising the system or eliminating the redundant constraints but we note that [19] states that there are difficulties with either alternative. The second case can arise from components which are rotating or from natural frequencies of the system. We note that often the long term solution behaviour is more important than the details of the oscillating solution, i.e. the oscillations can be neglected. If the amplitude is small enough then the oscillations may be damped by the numerical method but according to [19] this can result in problems with Newton convergence for the Lagrangian formulation. We note that methods for oscillations which cannot be damped or neglected need to be developed. (see [19]). In addition real time simulation imposes severe requirements on the solution method.

We now include an example from [6] which demonstrates that the growth of the residual of the original constraint is qualitatively different from the growth of the error in the solution components.

Example 5.1

$$x' = y - ax^2 + u(t) \tag{5.5a}$$

$$y = ax^2 \tag{5.5b}$$

(5.5) has the state space form $x' = u(t)$, (which is a Lagrange equation of the second kind), with solution

$$\mathbf{x}(t) = \mathbf{x}_0 + \mathbf{U}(t) \qquad \text{where } \mathbf{U}(t) = \int_0^t u(\tau)d\tau \text{ and } \mathbf{x}_0 = \mathbf{x}(0) \tag{5.6}$$

It follows that

$$y = (\mathbf{x}_0 + \mathbf{U}(t))^2 \tag{5.7}$$

However, if we differentiate the constraint, thus reducing the index, we obtain

$$y' = 2axx' \tag{5.8}$$

or $$y' = 2ax(y - ax^2 + u(t)) \tag{5.9}$$

to give the system

$$x' = y - ax^2 + u(t) \tag{5.10a}$$

$$y' = 2ax(y - ax^2 + u(t)) \tag{5.10b}$$

The system (5.10) has the solution

$$x(t) = U(t) + x_0 + \int_0^t (y - ax^2)dt \tag{5.11}$$

or

$$x(t) = U(t) + x_0 + t(y_0 - ax_0^2) \tag{5.12}$$

since (5.8) can be integrated to give $y = ax^2 + c$ with solution

$$y = ax^2 + y_0 - ax_0^2 \tag{5.13}$$

We can write (5.12) as

$$x(t) = \varepsilon_0 t + \mathrm{U}(t) + x_0 \tag{5.14}$$

where $\varepsilon_0 = y_0 - ax_0^2$ = residual of the position constraint.

Using (5.14) gives

$$y(t) = a(\varepsilon_0 t + \mathrm{U}(t) + x_0)^2 + \varepsilon_0 \tag{5.15}$$

If the initial conditions are consistent then $\varepsilon_0 = 0$. In this case (5.7) and (5.15) give the same solution. However the two problems have quite different numerical properties. (see [6])

We will now return to the pendulum problem, discussed in section 2.5

$$x'' = -\lambda x$$

i.e. $$y'' = -\lambda y - g$$

$$0 = x_1^2 + x_2^2 - L^2$$

We have already noted how to reformulate the system as an index-3 semi-explicit nonlinear DAE in section 2.5

i.e . in the form

$$x_1' = v_1 \tag{5.16a}$$

$$x_2' = v_2 \tag{5.16b}$$

$$v_1' = -\lambda x_1 \tag{5.16c}$$

$$v_2' = -\lambda x_2 - g \tag{5.16d}$$

$$0 = x_1^2 + x_2^2 - L^2 \tag{5.16e}$$

The velocity constraint, obtained through differentiation of (5.16e) is

$$0 = x_1 v_1 + x_2 v_2 \tag{5.17}$$

Equations (5.16a) -(5.16d) and (5.17) form an index-two DAE in semi-explicit form.

Another index-two formulation is obtained by introducing a new variable μ and replacing (5.16a) and (5.16b) by

$$x_1' = v_1 + x_1 \mu \tag{5.18a}$$

$$x_2' = v_2 + x_2 \mu \tag{5.18b}$$

The system (5.18) and (5.16c) - (5.16e) is known as the stabilised index-2 formulation of the problem.

We now consider the numerical solutions to three different formulations of the problem, namely:-

1. the index-3 formulation
2. the index-1 formulation
3. the stabilised index-2 formulation

We will consider the data given for these problems in [4]. We are using L = 1, g = 1 and consistent initial values $x_1(0) = 1, x_2(0) = 0, v_1(0) = 0, v_2(0) = 1$ and λ (0) = 1 at t = 0.

FORMULATION	TYPE OF CODE USED	NOTES	PAGE In [4]	TABLE N0. IN [4]
Index-3	A simple code implementing the two-step BDF method, with constant stepsize, for a sequence of fixed stepsizes.	The rate of convergence remained second order in all the variables	155	6.2.1
Index-1	DASSL for a sequence of local error tolerance inputs. (The drift in the constraints is measured at t =1).	• the position constraint is satisfied to a lower accuracy than RTOL (the relative error tolerance -specified by the user) • For RTOL ≤ 1.E-9 there is some drift in satisfying the velocity constraint • No drift in satisfying the acceleration constraint - as expected	155-156	6.2.2
Stabilised index-2	DASSL	• Algebraic variables have been excluded from the integration error control estimates • All variables were included in the convergence test for the corrector iteration • The problem of drift is eliminated	155-156	6.2.3 6.2.4

5.2 ELECTRICAL NETWORKS

DAEs arising in this field tend to be large and sparse, and are frequently linear, although some circuit devices introduce nonlinearities. By convention, current is the net flow of positive charge (from the positive node to the negative node). The voltage drop across each branch is defined to be difference between the voltage at the positive node and that at the negative node.

We note that energy is stored in a capacitor, in the form of a charge, and in the form of a magnetic field in an inductor. Resistors are used to produce a loss or gain of power in a branch.

The author of [14] gives Kirchhoff's laws in the form:-

(i) The algebraic sum of all currents meeting at any node in a circuit is zero.

(ii) Around any closed loop in a circuit the sum of the voltage drops is equal to the sum of the voltage sources in that loop.

We recall that the voltage and currents are related by

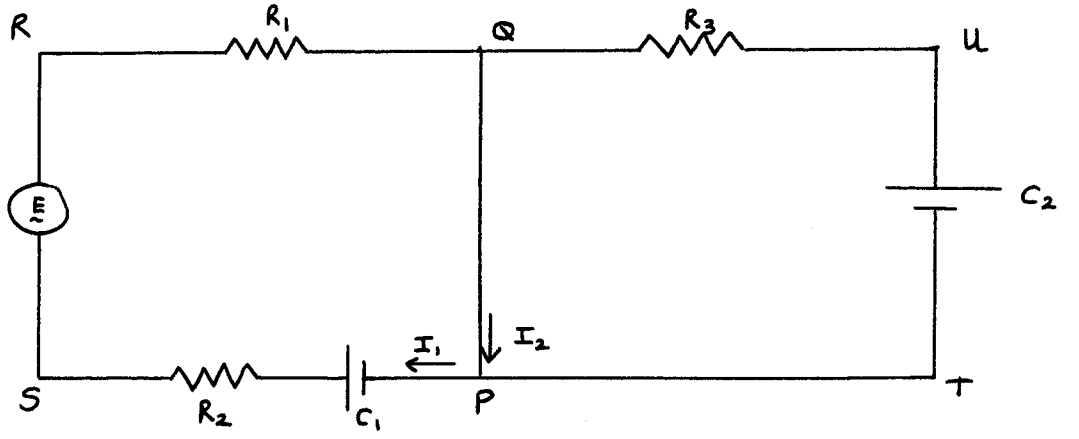
$$V = I\,R \quad \text{for a resistor (Ohm's law)} \tag{5.22}$$

and by

$$I = C\frac{dV}{dt} \quad \text{for a capacitor.} \tag{5.23}$$

Example 5.2

We consider an electrical network, taken from [14], with resistors R_1, R_2, R_3, capacitors C_1, C_2, currents I_1, I_2, I_3, voltage drops $V_1, V_2, V_3. V_4, V_5$ respectively, and voltage source E, as shown in the following diagram.



P is an example of a node and SPQR and PTUQ are examples of loops in a circuit.

We can use the first of Kirchhoff's laws at the node P to write

$$I_1 - I_2 - I_3 = 0,$$

and the second law can be used to give

$$E = V_1 + V_2 + V_4$$

where the voltage drops at R_1, R_2 and C_1 are equal to V_1, V_2, V_4 respectively. We have, using (5.22) and (5.23)

$$V_1 = I_1 R_1$$
$$V_2 = I_2 R_2$$
$$V_3 = I_3 R_3$$
$$I_3 = C_2 \frac{dV_5}{dt}$$
$$I_1 = C_1 \frac{dV_4}{dt}$$

where V_3 and V_5 are the voltage drops at R_3 and C_2. Using the second of Kirchhoff's laws we obtain $0 = V_3 + V_5$ for the circuit PTUQ, and no further useful information is found from applying it to circuit STUR.

We now have 8 equations, 2 of which are differential equations and the remaining 6 are algebraic equations (derived from Kirchhoff's laws and Ohm's law).

$$V_4' = C_1^{-1} I_1$$
$$V_5' = C_2^{-1} I_3$$
$$0 = V_1 + V_2 + V_4 - E$$
$$0 = V_3 + V_5$$
$$0 = I_1 - I_2 - I_3$$
$$0 = V_1 - R_1 I_1$$
$$0 = V_2 - R_2 I_2$$
$$0 = V_3 - R_3 I_3$$

This is a DAE which can be expressed in the form

$$\frac{d}{dt}\begin{bmatrix} V_4 \\ V_5 \end{bmatrix} = \begin{bmatrix} C_1^{-1} I_1 \\ C_2^{-1} I_3 \end{bmatrix}$$

$$0 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 1 & 0 & 0 & 0 & 0 & -R_1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -R_2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -R_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ I_1 \\ I_2 \\ I_3 \end{pmatrix} - \begin{pmatrix} E \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

This is a linear constant coefficient DAE. It is typically large and sparse. (see [2])

5.3 THE NEXT GENERATION OF DAE RESEARCH?

DAE research continues to be stimulated by the desire to solve more complex problems and the increasing use of computer models. Interest in, and literature on, DAEs is to be found in many disciplines and is concerned with both advancing the knowledge about DAEs and improving upon the numerical methods by which to solve them.

Research into numerical methods continues.

The characteristics of numerical methods which preserve symplecticity and the construction of a new class of symplectic methods have attracted an avalanche of research since 1988 (see [8]). Such methods are particularly desirable for applications involving long time integration, examples of which are found in molecular dynamics and celestial mechanics simulations. Methods which can preserve certain features of the flow of constrained conservative mechanical systems have recently been developed. These can retain the symplectic and reversible structures of the flow and can achieve a better qualitative behaviour than non-preserving methods.

"There is an increasing interest in large complex models involving many different types of equations" [4].

Advantages of a direct solution of a model using a DAE are encouraging research into "DAEs " which include other types of equation than just ODEs and algebraic equations e.g. Integral equations, delay systems and PDEs and composites of these systems. Also, interest is growing in differential equations with algebraic inequality constraints which arise, for example, in the field of optimal control or prescribed path control.

BIBLIOGRAPHY

[1] URI M ASCHER, ROBERT M. M. MATTHEIJ, ROBERT D. RUSSELL
Numerical Solution of Boundary Value Problems for Ordinary Differential Equations
SIAM, 1995.

[2] U. M. ASCHER, L. R. PETZOLD
Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations.
SIAM, 1998.

[3] A. BARRLUND AND B. KÅGSTRÖM
Analytical and numerical solutions to higher index linear variable coefficient DAE systems.
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS 31, (1990), 305 - 330.

[4] K. E. BRENAN, S. L. CAMPBELL, L. R. PETZOLD
Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.
SIAM, 1996.

[5] PETER N. BROQN, ALAN C. HINDMARSH, AND LINDA R. PETZOLD
Consistent Initial Condition Calculation for Differential-Algebraic Systems.
SIAM J. SCI. COMPUT. VOL 19, NO 5, (1998), 1495 - 1512.

[6] C. FÜHRER and B. J. LEIMKUHLER
Numerical Solution of Differential-Algebraic Equations for Constrained Mechanical Motion.
NUMER. MATH. 59, (1991), 55 - 69.

[7] C. W. GEAR AND L. R. PETZOLD
ODE Methods for the Solution fo Differential/Algebraic Systems
SIAM. J. NUMER. ANAL. VOL21, NO4, (1984), p 716 - 728.

[8] E. HAIRER, S. P. NØRSETT, G. WANNER
Solving Ordinary Defferential Equations I: Nonstiff Problems
SPRINGER-VERLAG, 1993.

[9] E. HAIRER, G. WANNER
Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.
SPRINGER-VERLAG, 1991.

[10] J. D. LAMBERT
Numerical Methods for Ordinary Defferential Systems: The Initial Value Problem.
WILEY, 1991.

[11] B. LEIMKUHLER, L. R. PETZOLD, and C. W. GEAR
Approximation Methods for the Consistent Initialization of Differential-Algebraic Equations.
SIAM. J. NUMER. ANAL. VOL 28, NO 1, (1991), p 205 - 226.

[12] ROSWITHA MARZ
Numerical Methods for Differential Algebraic Equations
ACTA NUMERICA (1991), p 141 - 198.

[13] R. MARZ AND C. TISCHENDORF
Solving More General Index-2 Differential Algebraic Equations
COMPUTER MATH.APPLIC. VOL 28, NO 10 - 12, (1994), p 77 - 105.

[14] R. M. M. MATTHEIJ nad J. MOLENAAR
Ordinary Differential Equations in Theory and Practice.
JOHN WILEY, & SONS,1996.

[15] SVEN ERIK MATTSSON AND GUSTAF SODERLIND
Index Reduction in Differential-Algebraic Eqations using Dummy Derivatives.
SIAM. J. SCI. COMPUT. VOL 14, NO 3, (1993), p 677 - 692.

[16] CONSTANTINOS C. PANTELIDES
The Consistent Initialization of Differential-Algebraic Systems
SIAM. J. SCI. STAT. COMPUT. VOL 9, NO 2, (1988), p 213 - 231.

[17] LINDA PETZOLD
Differential/Algebraic Equations are not ODE's
SIAM. J. SCI. STAT. COMPUT, VOL 3, NO 3, (1982), p 367 - 384.

[18] LINDA PETZOLD
Numerical Methods for DAEs - Current Status and Future Directions. In J. R. CASH AND I. CLADWELL (Eds) Computational Ordinary Differential Equations
OXFORD UNIVERSITY PRESS, 1992.

[19] LINDA PETZOLD
Numerical Solution of Differential-Algebraic Equations
ADVANCES IN NUMERICAL ANALYSIS VOL. IV, (1995), p 123 - 138.

[20] WENJIE ZHU and LINDA R. PETZOLD
Asymptotic Stability of Linear Delay Differential Algebraic Equations and Numerical Methods
Department of Computer Science, University of Minnesota, TR 96 - 055, (1996).