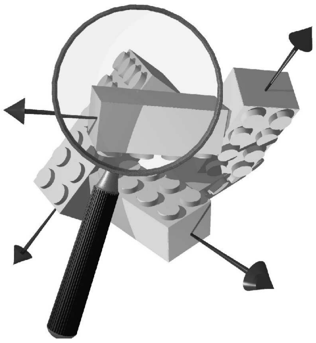
IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

May 6, 1999
CBE

# Numerical Solution of Differential Algebraic Equations



**Editors: Claus Bendtsen**
**Per Grove Thomsen**

**TECHNICAL REPORT**

**IMM-REP-1999-8**

**IMM**

# Contents

# Preface

These lecture notes have been written as part of a Ph.D. course on the numerical solution of Differential Algebraic Equations. The course was held at IMM in the fall of 1998. The authors of the different chapters have all taken part in the course and the chapters are written as part of their contribution to the course.

We hope that coming courses in the Numerical Solution of DAE's will benefit from the publication of these lecture notes.

The students participating is the course along with the chapters they have written are as follows:

- Astrid Jørdis Kuijers, Chapter 4 and 5 and Section 1.2.1.
- Anton Antonov AntonovChapter 2 and 6.
- Brian ElmegaardChapter 8 and 10.
- Mikael Zebbelin PoulsenChapter 2 and 3.
- Falko Jens WagnerChapter 3 and 9.
- Erik Østergaard, Chapter 5 and 7.

CHAPTER 1

# Introduction

The (modern) theory of numerical solution of ordinary differential equations (ODEs) has been developed since the early part of this century – beginning with Adams, Runge and Kutta. At the present time the theory is well understood and the development of software has reached a state where robust methods are available for a large variety of problems. The theory for Differential Algebraic Equations (DAEs) has not been studied to the same extent – it appeared from early attempts by Gear and Petzold in the early 1970'es that not only are the problems harder to solve but the theory is also harder to understand.

The problems that lead to DAEs are found in many applications of which some are mentioned in the following chapters of these lecture notes. The choice of sources for problems have been influenced by the students following this first time appearance of the course.

## 1.1. Definitions

The problems considered are in the most general form a fully implicit equation of the form

$$\mathbf{F}(\mathbf{y}', \mathbf{y}) = \mathbf{0}, \tag{1.1}$$

where $\mathbf{F}$ and $\mathbf{y}$ are of dimension $n$ and $\mathbf{F}$ is assumed to be sufficiently smooth. This is the autonomous form, a non-autonomous case is defined by

$$\mathbf{F}(x, \mathbf{y}', \mathbf{y}) = 0.$$

Notice that the non-autonomous equation may be made autonomous by adding the equation $x' = 1$ and we do therefore not need to consider the non-autonomous form seperately[1].

A special case arises when we can solve for the $\mathbf{y}'$-variable since we (at least formally) can make the equation explicit in this case and obtain a system of ODEs. The condition to be fullfilled is that $\frac{\partial \mathbf{F}}{\partial \mathbf{y}'}$ is nonsingular. When this is not the case the system is commonly known as being differential algebraic and this

[1] this may be subject to debate since the non-autonomous case can have special features

will be the topic of these notes. In order to emphasize that the DAE has the general form Eq. 1.1 it is normally called a *fully implicit DAE*. If $\mathbf{F}$ in addition is linear in $\mathbf{y}$ (and $\mathbf{y}'$) (i.e. has the form $\mathbf{A}(x)\mathbf{y}+\mathbf{B}(x)\mathbf{y}'=0$) the DAE is called *linear* and if the matrices $\mathbf{A}$ and $\mathbf{B}$ further more are constant we have a *constant coefficient linear DAE*.

**1.1.1. Semi explicit DAEs.** The simplest form of problem is the one where we can write the system in the form

$$\begin{aligned}\mathbf{y}' &= \mathbf{f}(\mathbf{y},\mathbf{z})\\ \mathbf{0} &= \mathbf{g}(\mathbf{y},\mathbf{z})\end{aligned} \tag{1.2}$$

and $\mathbf{g}_z$ (the partial derivative $\partial\mathbf{g}/\partial\mathbf{z}$) has a bounded inverse in a neighbourhood of the solution.

Assuming we have a set of consistent initial values $(\mathbf{y}_0,\mathbf{z}_0)$ it follows from the inverse function theorem that $\mathbf{z}$ can be found as a function of $\mathbf{y}$. Thus local existence, uniqueness and regularity of the solution follows from the conventional theory of ODEs.

**1.1.2. Index.** Numerous examples exist where the conditions above do not hold. These cases have general interest and below we give a couple of examples from applications.

1.1.2.1. *Singular algebraic constraint.* We consider the problem defined by the system of three equations

$$\begin{aligned}y_1' &= y_3\\ 0 &= y_2(1-y_2)\\ 0 &= y_1y_2+y_3(1-y_2)-x,\end{aligned}$$

where $x$ is a parameter of our choice. The second equation has two solutions $y_2=0$ and $y_2=1$ and we may get different situations depending on the choice of initial conditions:

1. if $y_2=0$ we get $y_3=x$ from the last equation and we can solve the first equation for $y_1$.
2. Setting $y_2=1$ we get $y_1=x$ from the last equation and $y_3=1$ follows from the first equation.

1.1.2.2. *Implicit algebraic variable.*

$$\begin{aligned}y' &= f(y,z)\\ 0 &= g(y)\end{aligned} \tag{1.3}$$

In this case we have that $g_z = 0$ and the condition of boundedness of the inverse does not hold. However if $g_y f_z$ has a bounded inverse we can do the trick of differentiating the second equation leading to system

$$\begin{aligned} y' &= f(y,z) \\ 0 &= g_y(y) f(y,z) \end{aligned}$$

and this can then be treated like the semi explicit case, Eq. 1.2. This motivates the introduction of *index*.

DEFINITION 1.1 (Differential index). For general DAE-systems we define the index along the solution path as the minimum number of differentiations of the systems, Eq. 1.1 that is required to reduce the system to a set of ODEs for the variable $\mathbf{y}$.

The concept of index has been introduced in order to quantify the level of difficulty that is involved in solving a given DAE. It must be stressed, as we indeed will see later, that numerical methods applicable (i.e. convergent) for DAEs of a given index, might not be useful for DAEs of higher index.

Complementary to the differential index one can define the *pertubation index* [HLR89].

DEFINITION 1.2 (Perturbation index). Eq. 1.1 is said to have perturbation index $m$ along a solution $\mathbf{y}$ if $m$ is the smallest integer such that, for all functions $\hat{\mathbf{y}}$ having a defect

$$\mathbf{F}(\hat{\mathbf{y}}', \hat{\mathbf{y}}) = \delta(x),$$

there exists an estimate

$$\begin{aligned} \|\hat{\mathbf{y}}(x) - \mathbf{y}(x)\| \le C(\|\hat{\mathbf{y}}(0) - \mathbf{y}(0)\| + \max \|\delta(\xi)\| + \\ \ldots + \max \|\delta^{(m-1)}(\xi)\|) \end{aligned}$$

whenever the expression on the right hand side is sufficiently small. $C$ is a constant that depends only on the function $\mathbf{F}$ and the length of the interval.

If we consider the ODE-case Eq. 1.1, the lemma by Gronwall (see [HNW93, page 62]) gives the bound

$$\| \hat{y}(x) - y(x) \| \le C(\| \hat{y}(0) - y(0) \| + \max_{0 \le \xi \le x} \| \int_0^{\xi} \delta(t) dt \|). \tag{1.4}$$

If we interpret this in order to find the perturbation index it is obviously zero.

**1.1.3. Index reduction.** A process called index reduction may be applied to a system for lowering the index from an initially high value down to e.g. index one. This reduction is performed by successive differentiation and is often used in theoretical contexts. Let us illustrate it by an example:

We look at example 1.1.2.2 and differentiate Eq. 1.3 once, thereby obtaining

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

Differentiating Eq. 1.3 once more leads to

$$\begin{aligned} y' &= f(y,z) \\ 0 &= g_{yy}(y)(f(y,z))^2 + g_y(y)(f_y(y,z)f(y,z) + f_z(y,z)z'), \end{aligned}$$

which by rearranging terms can be expressed as

$$\begin{aligned} y' &= f(y,z) \\ z' &= -\frac{g_{yy}(y)(f(y,z))^2 + g_y(y)f_y(y,z)f(y,z)}{g_y(y)f_z(y,z)}. \end{aligned}$$

The two differentiations have reduced the system to one of index zero and we can solve the resulting ODE-system using well known methods.

If we want to find the perturbation index we may look at the equations for the perturbed system

$$\begin{aligned} \hat{y}' &= f(\hat{y},\hat{z}) + \delta(x) \\ 0 &= g(\hat{y}) + \theta(x) \end{aligned}$$

Using differentiation on the second equation leads to

$$0 = g_y(\hat{y})f(\hat{y},\hat{z}) + g_y(\hat{y})\delta(x) + \theta'(x)$$

From the estimates above we can now (by using Gronwalls lemma Eq. 1.4) obtain

$$\begin{aligned} \|\hat{y}(x) - y(x)\| &\le C(\|\hat{y}(0) - y(0)\| + \\ &\qquad \int_0^x (\|\delta(\xi)\| + \|\theta'(\xi)\|)d\xi \\ \|\hat{z}(x) - z(x)\| &\le C(\|\hat{y}(0) - y(0)\| + \\ &\qquad \max\|\delta(\xi)\| + \max\|\theta'(\xi)\|) \end{aligned}$$

All the max- values are to be taken over $0 \le \xi \le x$.

## 1.2. Singular Perturbations

One important source for DAE-problems comes from using singular perturbations [RO88] Here we look at systems of the form

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(\mathbf{y},\mathbf{z}) \\ \varepsilon\mathbf{z}' &= \mathbf{g}(\mathbf{y},\mathbf{z}) \text{ where } 0 < \varepsilon \ll 1. \end{aligned} \tag{1.5}$$

Letting $\varepsilon$ go to zero ($\varepsilon \to 0$) we obtain an index one problem in semi-explicit form. This system may be proven to have an $\varepsilon$-expansion where the expansion coefficients are solution to the system of DAEs that we get in the limit of Eq. 1.5.

**1.2.1. An example.** In order to show the complex relationship between singular pertubation problems and higher index DAEs, we study the following problem:

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}_1(x,\mathbf{y},\mathbf{z},\varepsilon) & \mathbf{y}_0 &= \zeta_0 \\ \varepsilon\mathbf{z}' &= \mathbf{f}_2(x,\mathbf{y},\mathbf{z},\varepsilon) & \mathbf{z}_0 &= \frac{\mathbf{g}(0,\zeta_0,\varepsilon)}{\varepsilon} \end{aligned}$$
$$\text{where } \varepsilon \ll 1,$$

which can be related to the reduced problem (by letting $\varepsilon \to 0$)

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}_1(x,\mathbf{y},\mathbf{z},0) \\ 0 &= \mathbf{f}_2(x,\mathbf{y},\mathbf{z},0). \end{aligned}$$

This is a DAE-system in semi-explicit form and we can differentiate the second equation with respect to $x$ and obtain

$$\begin{aligned} \bar{\mathbf{y}}' &= \mathbf{f}_1(x,\bar{\mathbf{y}},\bar{\mathbf{z}},0) & \bar{\mathbf{y}}_0 &= \zeta_0 \\ 0 &= \mathbf{g}_y(x,\bar{\mathbf{y}},0)\mathbf{f}_2(x,\bar{\mathbf{y}},\bar{\mathbf{z}},0) + \mathbf{g}_x(x,\bar{\mathbf{y}},0). \end{aligned}$$

This shows, that for indexes greater than one the DAE and the stiff problem can not always be related easily.

Singular Pertubation Problems (SPP) are treated in more detail in Subsection 2.2.2.

## 1.3. The Van der Pol equation

A very famous test problem for systems of ODEs is the Van der Pol equation defined by the second order differential equation

$$y'' = \mu(1-y^2)y' - y$$

This equation may be treated in different ways, the most straigtforward is to split the equation into two by introducing a new variable for the derivative ($y_1 = y$, $y_2 = y'$) which leads to

$$\begin{aligned} {y_1}' &= y_2 \\ {y_2}' &= \mu(1 - {y_1}^2)y_2 - y_1 \end{aligned}$$

The system of two ODEs may be solved by any standard library solver but the outcome will depend on the solver and on the parameter $\mu$. If we divide the second of the equations by $\mu$ we get an equation that has the character of a singular perturbation problem. Letting $\mu \to \infty$ we see that this corresponds to $\varepsilon \to 0$ in Eq. 1.5.

Several other approaches may show other aspects of the nature of this problem for example [HW91] introduces the transformation $t = x/\mu$ after a scaling of $y_2$ by $\mu$ we get

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= \mu^2((1 - {y_1}^2)y_2 - y_1) \end{aligned}$$

The introduction of $\varepsilon = 1/\mu^2$ finally results in a problem in singular perturbation form

$$\begin{aligned} y_1' &= y_2 \\ \varepsilon y_2' &= (1 - {y_1}^2)y_2 - y_1 \end{aligned}$$

As explained previously this problem will approach a DAE if $\varepsilon \to 0$ Using terminology from ODEs the stiffness of the problem increases as $\varepsilon$ gets smaller giving rise to stability problems for numerical ODE-solvers that are explicit while we expect methods that are A- or L-stable to perform better.

# Part 1

# Methods

CHAPTER 2

# Runge-Kutta Methods for DAE problems

***Written by Anton Antonov Antonov & Mikael Zebbelin Poulsen***

**Keywords:** *Runge-Kutta methods, order reduction phenomenon, stage order, stiff accuracy, singular perturbation problem, index-1 and index-2 systems,* $\varepsilon$*-embedding, state space method,* $\varepsilon$*-expansion, FIRK and DIRK methods.*

## 2.1. Introduction

Runge-Kutta methods have the advantage of being one-step methods, possibly having high order and good stability properties, and therefore they (and related methods) are quite popular.

When solving DAE systems using Runge-Kutta methods, the problem of incorporation of the algebraic equations into the method exists. This chapter focuses on understanding algorithms and convergence results for Runge-Kutta methods. The specific Runge-Kutta tables (i.e. coefficients) has to be looked up elsewhere, eg. [HW96], [AP98] or [ESF98].

**2.1.1. Basic Runge-Kutta methods.** To settle notation we recapitulate the form of the $s$-stage Runge-Kutta method.

For the autonomous explicit ODE system

$$\mathbf{y}' = \mathbf{f}(\mathbf{y})$$

the method, for advancing the solution from a point $\mathbf{y}_n$ to the next, would look like this:

$$\mathbf{Y}_{n,i} = \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{f}(\mathbf{Y}_{n,j}) \quad ,i = 1,2,\ldots,s \tag{2.1a}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i\mathbf{f}(\mathbf{Y}_{n,i}) \tag{2.1b}$$

The method is a one-step method (i.e. it does not utilize information from previous steps) and it is specified by the matrix $\mathbf{A}$ with the elements $a_{ij}$, and the vector **b** having the elements $b_i$. We call the $\mathbf{Y}_{n,i}$'s the internal stages of the step. In general these equations represents a non-linear system of equations.

The classical theory for determining the order of the local and global error is found in a number of books on ODEs. Both the J. C. Butcher and P. Albrecht approach are shortly described in [Lam91].

**2.1.2. Simplifying conditions.** The construction of implicit Runge-Kutta methods is often done with respect to some *simplifying (order) conditions* , see [HW96, p.71]:

$$B(p): \quad \sum_{i=1}^{s} b_i c_i^{q-1} = \frac{1}{q} \qquad q = 1, \dots, p \tag{2.2a}$$

$$C(\eta): \quad \sum_{j=1}^{s} a_{ij} c_j^{q-1} = \frac{c_i^q}{q} \qquad i = 1, \dots, s \quad q = 1, \dots, \eta \tag{2.2b}$$

$$D(\zeta): \quad \sum_{i=1}^{s} b_i c_i^{q-1} a_{ij} = \frac{b_j}{q}(1 - c_j^q) \quad j = 1, \dots, s \quad q = 1, \dots, \zeta \tag{2.2c}$$

Condition $B(p)$ is the quadrature condition, and $C(\eta)$ could be interpreted as the quadrature condition for the internal stages. A theorem of Butcher now says

THEOREM 2.1. *If the coefficients $b_i$, $c_i$ and $a_{ij}$ of a Runge-Kutta method satisfies $B(p)$, $C(\eta)$ and $D(\zeta)$ with $p \leq \eta + \zeta$ and $p \leq 2\eta + 2$, then the method is of order $p$.*

*Comment:* You could say "at least order $p$", or - that to discover the properties of the method, - try to find a $p$ as high as possible.

**2.1.3. Order reduction, stage order, stiff accuracy.** When using an implicit Runge-Kutta method for solving a stiff system, the order of the error can be reduced compared to what the classical theory predicts. This phenomenon is called *order reduction*.

The cause of the classical order "failing", is that the step size is actually "chosen" much to large compared to the time scale of the system. We know though, and our error estimates often tells the same, that we get accurate solutions even though choosing such big step-sizes. This has to do with the fact that in a stiff system we are not following the solution curve, but we are actually following the "stable manifold"

The order we observe is (at least) described by the concept of *stage order*. The stage order is the minimum of $p$ and $q$ when the Runge-Kutta method satisfies $B(p)$ and $C(q)$ from Eq. 2.2.

A class of methods not suffering from order reduction are the methods which are *stiffly accurate*. Stiff accuracy means that the final point is actually one of the internal stages (typically the last stage), and therefore the method satisfies

$$b_i = a_{si}, \qquad i = 1, \ldots, s$$

**2.1.4. Collocation methods.** *Collocation* is a fundamental numerical method. It is about fitting a function to have some properties in some chosen *collocation points*. Normally the collocation function would be a polynomial $\mathbf{P}(x)$ of some degree $r$.

The collocation method for integrating a step (like Eq. 2.1a) would be to find coefficients so that

$$\begin{aligned} \mathbf{P}(x_n) &= \mathbf{y}_n \\ \mathbf{P}'(x_n + c_i h) &= \mathbf{f}(x_n + c_i h, \mathbf{P}(x_n + c_i h)) \qquad , i = 1, 2, \ldots, s \\ \mathbf{y}_{n+1} &= \mathbf{P}(x_n + h) \end{aligned}$$

Collocation methods are a relevant topic when discussing Runge-Kutta methods. It turns out that some special Runge-Kutta methods like Gauss, Radau IIA and Lobatto IIIA, are in fact collocation methods.

In [Lam91, p. 194] it is shown why such a method is a Runge-Kutta method. The principle is that the $c_i$ in the collocation formula matches the ones from the Runge-Kutta method, and the internal stages of the Runge-Kutta method match the collocation method as $\mathbf{Y}_i = \mathbf{P}(x_n + c_i h)$. The "parameters" of the collocation method are the $c_i$'s and therefore define the $\mathbf{A}$-matrix of the identical (implicit) Runge-Kutta method.

## 2.2. Runge-Kutta Methods for Problems of Index 1

The traditional form of the index-one problem looks like this:

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}, \mathbf{z}) \tag{2.3a}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{y}, \mathbf{z}) \tag{2.3b}$$

with the assumption that

$$\mathbf{g}_z(\mathbf{y}, \mathbf{z}) \text{ is invertible} \tag{2.3c}$$

for every point at the solution curve. If $\mathbf{g}_z(\mathbf{y}, \mathbf{z})$ were not invertible the system would have index at least 2.

This implies that Eq. 2.3b has a unique solution $\mathbf{z} = \mathbf{G}(\mathbf{y})$ by the Implicit Function Theorem. Inserted into Eq. 2.3a gives

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}, \mathbf{G}(\mathbf{y})) \tag{2.4}$$

- the so called *state space form*. We see that, if algebraic manipulations could lead to Eq. 2.4, then we could solve the problem as a normal explicit ODE problem. This is in general not possible.

We should at this time mention another form of the implicit ODE system: The form

$$\mathbf{M}\mathbf{y}' = \mathbf{f}(\mathbf{y})$$

typically occurs when you can't separate differential and algebraic equations like in Eq. 2.3, and some of the equations contain more than one "entry" of the primes $y_i'$ from $\mathbf{y}'$, see eg. [HW96, p. 376 ff.].

If $\mathbf{M}$ is invertible/regular, this system is actually just a normal ODE system (index-0), and traditional methods can be used. If $\mathbf{M}$ is singular the system has index at least one.

If the system is of index-1 then a linear transformation of the variables can give the system a form like Eq. 2.3.

**2.2.1. State Space Form Method.** Because of the possibility of (at least "implicitly") rewriting the problem to the state space form Eq. 2.4, we could imagine using the Runge-Kutta method, by solving Eq. 2.3b (for $z$) in each of the internal stages and at the final solution point:

$$\left.\begin{aligned} \mathbf{Y}_{n,i} &= \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{f}(\mathbf{Y}_{n,j}, \mathbf{Z}_{n,j}) \\ \mathbf{0} &= \mathbf{g}(\mathbf{Y}_{n,i}, \mathbf{Z}_{n,i}) \end{aligned}\right\} \quad , i = 1, 2, \ldots, s$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i\mathbf{f}(\mathbf{Y}_{n,i}, \mathbf{Z}_{n,i})$$

$$\mathbf{0} = \mathbf{g}(\mathbf{y}_{n+1}, \mathbf{z}_{n+1})$$

These methods have the same order as the classical theory would predict for Runge-Kutta methods with some $\mathbf{A}$ and $\mathbf{b}$, and this holds for both $\mathbf{y}$ and $\mathbf{z}$.

Furthermore these methods do not have to be implicit. The explicit state space form methods are treated in [ESF98, p. 182], here called *half-explicit Runge-Kutta methods*.

**2.2.2. The $\varepsilon$-embedding method for problems of index 1.** Discussions of index-1 problems are often introduced by presenting the *singular perturbation problem* (SPP) already encountered in Section 1.2. A key idea (see eg. [HW96, p. 374]) is the following: Transform the DAE to a SPP, by introducing $\varepsilon$. Deriving the method for a SPP and then putting $\varepsilon = 0$. This will give a method for DAE of index 1:

$$\mathbf{Y}_{n,i} = \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{f}(\mathbf{Y}_{n,j},\mathbf{Z}_{n,j}) \qquad i = 1,2,\ldots,s \tag{2.6a}$$

$$\varepsilon\mathbf{Z}_{n,i} = \varepsilon\mathbf{z}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{g}(\mathbf{Y}_{n,j},\mathbf{Z}_{n,j}) \qquad i = 1,2,\ldots,s \tag{2.6b}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i\mathbf{f}(\mathbf{Y}_{n,i},\mathbf{Z}_{n,i}) \tag{2.6c}$$

$$\varepsilon\mathbf{z}_{n+1} = \varepsilon\mathbf{z}_n + h\sum_{i=1}^{s} b_i\mathbf{g}(\mathbf{Y}_{n,i},\mathbf{Z}_{n,i}) \tag{2.6d}$$

In order to eliminate $\varepsilon$ from (2.6) we have

$$h\mathbf{g}(\mathbf{Y}_{n,i},\mathbf{Z}_{n,i}) = \varepsilon\sum_{j=1}^{s} w_{ij}(\mathbf{Z}_{n,j} - \mathbf{z}_n)$$

from Eq. 2.6b, where $\{\omega_{i,j}\} = \{a_{i,j}\}^{-1}$. This means that $\mathbf{A}$ is invertible and therefore the RK methods for $\varepsilon$-embedding must at least be implicit. So lets at last put $\varepsilon = 0$

$$\mathbf{Y}_{n,i} = \mathbf{y}_n + \sum_{j=1}^{s} a_{ij}\mathbf{f}(\mathbf{Y}_{n,j},\mathbf{Z}_{n,j}) \qquad i = 1,2,\ldots,s \tag{2.7a}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{Y}_{n,i},\mathbf{Z}_{n,i}) \qquad i = 1,2,\ldots,s \tag{2.7b}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i\mathbf{f}(\mathbf{Y}_{n,i},\mathbf{Z}_{n,i}) \tag{2.7c}$$

$$\mathbf{z}_{n+1} = (1 - \sum_{j=1}^{s} b_i\omega_{i,j})\mathbf{z}_n + \sum_{i,j=1}^{s} b_i\omega_{i,j}\mathbf{Z}_{n,j}) \tag{2.7d}$$

If we replace Eq. 2.7d with

$$\mathbf{0} = \mathbf{g}(y_{n+1}, z_{n+1}) \tag{2.7e}$$

$\Rightarrow \mathbf{Z}_{n,j} = \mathbf{G}(\mathbf{Y}_{n,j})$ and $\mathbf{z}_{n+1} = \mathbf{G}(\mathbf{y}_{n+1})$ In this case Eq. 2.7 is identical to the solution of the state space form with the same RK method. The same would hold if the method were stiffly accurate. In these cases the error is predicted by the classical order theory referred to previously.

Generally the convergence results are listed in [HW96, p.380]. We should though mention the convergence result if the method is not stiffly accurate, but linearly stable at infinity: Then the method has the minimum of the classical order and the stage order + 1.

We have the following diagram:

$$\begin{tikzcd}
\text{SPP} \arrow[d, "\text{RK}"'] & \text{DAE} \arrow[l, "\varepsilon \leftarrow 0"'] \arrow[r, "\text{z=G(y)}"] \arrow[d] & \text{ODE} \arrow[d, "\text{RK}"'] \\
\text{Solution} \arrow[r, "\varepsilon \rightarrow 0"] & \begin{array}{c}\text{direct approach}\\ \text{indirect approch}\end{array} & \text{Solution} \arrow[l]
\end{tikzcd}$$

FIGURE 2.1. The transformation of the index-1 problem

### 2.3. Runge-Kutta Methods for high-index problems

Not many convergence results exist for high-index DAEs.

**2.3.1. The semi-explicit index-2 system.** For problems of index 2, results mainly exist for the semi-explicit index-2 problem. It looks as follows:

$$\mathbf{y}' = \mathbf{f}(\mathbf{y},\mathbf{z}) \tag{2.8a}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{y}) \tag{2.8b}$$

with

$$\mathbf{g}_y(\mathbf{y})\mathbf{f}(\mathbf{y},\mathbf{z}) \text{ being invertible.} \tag{2.8c}$$

**2.3.2. The ε-embedding method.** This method also applies to the semi-explicit index-2 case, although in a slightly different form: (note that the new variable $\mathbf{l}_{n,j}$ is introduced)

$$\mathbf{Y}_{n,i} = \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{f}(\mathbf{Y}_{n,j},\mathbf{Z}_{n,j}) \quad ,i = 1,2,\dots,s \tag{2.9a}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{Y}_{n,i}) \quad ,i = 1,2,\dots,s \tag{2.9b}$$

$$\mathbf{Z}_{n,i} = \mathbf{z}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{l}_{n,j} \quad ,i = 1,2,\dots,s \tag{2.9c}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{f}(\mathbf{Y}_{n,i}, \mathbf{Z}_{n,i}) \tag{2.9d}$$

$$\mathbf{z}_{n+1} = \mathbf{z}_n + h\sum_{i=1}^{s} b_i \mathbf{l}_{n,j} \tag{2.9e}$$

Convergence results are described in [HW96, p. 495 ff.]. The main convergence results are based on the simplifying conditions

$$B(p) \text{ and } C(q),$$

see Eq. 2.2. An abstract of convergence results are that the local error then behaves like

$$\begin{aligned} LE(y) &= O(h^{q+1}) \\ LE(z) &= O(h^q) \end{aligned}$$

with $p \geq q$. The interesting result for the global error is then that

$$\begin{aligned} GE(y) &= \begin{cases} O(h^{q+1}) & ,\text{if } p > q \\ O(h^q) & ,\text{if } p = q \end{cases} \\ GE(z) &= O(h^q) \end{aligned}$$

Remember that these results deal with the method used for the semi-explicit index-2 problem. The convergence results are, as presented, concentrated useful implications of more general results found in [HW96].

**2.3.3. Collocation methods.** The collocation method, matching an $s$-stage Runge-Kutta method could be applied to Eq. 2.8, as follows: Let $\mathbf{u}(x)$ and $\mathbf{v}(x)$ be polynomials of degree $s$. At some point $x_n$ the polynomial satisfies consistent initial conditions

$$\mathbf{u}(x_n) = \mathbf{y}_n, \qquad \mathbf{v}(x_n) = \mathbf{z}_n \tag{2.10a}$$

Then the collocation polynomials are found to satisfy

$$\left.\begin{aligned} \mathbf{u}'(x_n + c_i h) &= \mathbf{f}(\mathbf{u}(x_n + c_i h), \mathbf{v}(x_n + c_i h)) \\ \mathbf{0} &= \mathbf{g}(\mathbf{u}(x_n + c_i h)) \end{aligned}\right\} \quad ,i = 1, 2, \ldots, s \tag{2.10b}$$

and the step is finally determined from

$$\begin{aligned} \mathbf{y}_{n+1} &= \mathbf{u}(x_n + h) \\ \mathbf{z}_{n+1} &= \mathbf{v}(x_n + h) \end{aligned} \tag{2.10c}$$

It can be shown that this method coincides with the ε-method just mentioned, if the coefficients are chosen correctly. The Gauss-, Radau IIA-, and Lobatto IIIA coefficients could therefore be used in both methods. These coefficients are quite popular, when solving DAEs.

**2.3.4. Order-table for some methods in the index-2 case.** Here we present a table with some of the order results from using the above methods.

| Method | stages | local error y | local error z | global error y | global error z |
|---|---|---|---|---|---|
| Gauss | $s$ odd | $h^{s+1}$ | $h^s$ | $h^{s+1}$ | $h^{s-1}$ |
| Gauss | $s$ even | $h^{s+1}$ | $h^s$ | $h^s$ | $h^{s-2}$ |
| projected Gauss | $s$ | $h^{2s+1}$ | | $h^{2s}$ | |
| Radau IA | $s$ | $h^s$ | $h^{s-1}$ | $h^s$ | $h^{s-1}$ |
| projected Radau IA | $s$ | $h^{2s-1}$ | | $h^{2s-2}$ | |
| Radau IIA | $s$ | $h^{2s}$ | $h^{2s-1}$ | $h^{2s-2}$ | $h^{s-1}$ |
| Lobatto IIIA | $s$ odd | $h^{2s-1}$ | $h^s$ | $h^{2s-2}$ | $h^{s-1}$ |
| Lobatto IIIA | $s$ odd | $h^{2s-1}$ | $h^s$ | $h^{2s-2}$ | $h^s$ |
| Lobatto IIIC | $s$ odd | $h^{2s-1}$ | $h^{s-1}$ | $h^{2s-2}$ | $h^{s-1}$ |

For results on projection methods we refer to the chapter about Projection methods, Chapter 3 or we refer to the text in [HW96, p.502].

## 2.4. Special Runge-Kutta methods

This section will give an overview of some special classes of methods. These classes are defined by the structure of their coefficient matrix **A**. Figure 2.2 gives an overview of these structures. The content is presented with inspiration from [Kvæ92] and [KNO96].

**2.4.1. Explicit Runge-Kutta methods (ERK).** These methods have the nice property that the internal stages one after one are expressed explicitly from former stages. The typical structure of the **A**-matrix would the be

$$a_{ij} = 0 \quad j \geq i$$

These methods are very interesting in the explicit ODE case, but looses importance when we introduce implicit ODEs. In this case the computation of the internal stages would in general imply solving a system of nonlinear equations for each stage, and introduce iterations. Additionally the stability properties of these methods are quite poor.

As a measure of the work load we could use the factorization of the Jacobian. We have an implicit $n$-dimensional ODE system. Since the factorization should be done for each stage the work load for the factorization is of the order $sn^3$.

**2.4.2. Fully implicit Runge-Kutta methods (FIRK).** These methods have no restriction on the elements of **A**. An s-stage FIRK method together with an implicit ODE system, forms a fully implicit nonlinear equation system with a number of equations in the order of $ns$. The work done in factorization of the Jacobian would be of the order $(sn)^3 = s^3n^3$.

**2.4.3. Diagonally Implicit Runge-Kutta methods (DIRK).** In order to avoid the full Jacobian from the FIRK method, one could construct a semi explicit (close to explicit) method, by making the structure of **A** triangular and thereby the Jacobian of the complete nonlinear equation system block-triangular. The condition of the **A** being triangular is traditionally expressed as

$$a_{ij} = 0 \quad , j > i \tag{2.11}$$

The work load could again be related to the factorization of the Jacobian. In each stage we have $n$ equations and the factorization would cost in the order of $n^3$ per stage. This means that the total workload would be of the order $sn^3$ just as for the ERK methods, when used on the implicit ODE system.

But as a major advantage these methods have better stability properties than ERK methods. On the other hand they can not match the order of the FIRK methods with the same number of stages.

As a subclass of the DIRK methods one should notice the singly DIRK method (SDIRK methods) which on top of Eq. 2.11 specifies that

$$a_{ii} = \gamma \quad , i = 1, \ldots, s$$

Yet another class of method is the ESDIRK methods which introduces the start point as the first stage, and hereafter looks like the SDIRK method: the method is defined by Eq. 2.11 and

$$\begin{aligned} a_{11} &= 0 \\ a_{ii} &= \gamma \quad , i = 2, \ldots, s \end{aligned}$$

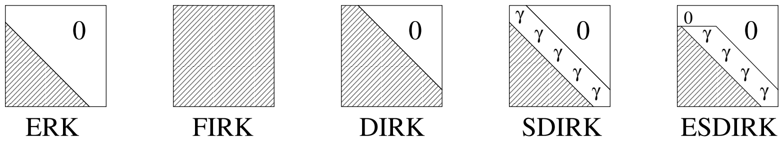



FIGURE 2.2. Overview of the structure of **A** for the different classes of methods

**2.4.4. Design Criteria for Runge-Kutta methods.** The DIRK methods are investigated for solving stiff systems, index-1 and index-2 systems in [Kvæ92]. She uses the ε-expansion shown in the next section to describe the error generated when solving a stiff system.

She summarizes a number of design criteria based on the properties found of importance, before presenting the DIRK methods. These properties are listed below:

- The advancing method should be stiffly accurate.
- The error estimating method should be stiffly accurate.
- The advancing method should be at least $L(0)$-stable.
- The error estimating method should at least be $A(0)$-stable, if possible $A(0)$-stable.
- The stageorder of the method should be as high as possible.

The conditions are presented in preferential order. We will only mention that the fist two conditions are ment to ensure that the order of the method and the error estimate is known also in the these stiff(-like) problems, where methods otherwise can suffer from order reduction.

## 2.5. ε-Expansion of the Smooth Solution

Following [HW96, Chapter VI.2] let us consider the singular perturbation problem Eq. 1.5 with the functions **f** and **g** sufficiently smooth. The functions **f**, **g** and the initial values $\mathbf{y}(0)$, $\mathbf{z}(0)$ may depend smoothly on $\varepsilon$. The corresponding equation for $\varepsilon = 0$ is the reduced problem 2.3a. Again in order to guarantee solvability of Eq. 2.3a, we assume that $\mathbf{g}_z(\mathbf{y},\mathbf{z})$ is invertible.

Now it is natural to be interested in smooth solutions of Eq. 1.5 which are of the form:

$$\mathbf{y}(x) = \mathbf{y}_0(x) + \varepsilon \mathbf{y}_1(x) + \varepsilon^2 \mathbf{y}_2(x) + \dots \tag{2.12}$$

$$\mathbf{z}(x) = \mathbf{z}_0(x) + \varepsilon \mathbf{z}_1(x) + \varepsilon^2 \mathbf{z}_2(x) + \dots \tag{2.13}$$

If we now insert Eq. 2.12 and Eq. 2.13 into Eq. 1.5 and collect the equal powers of $\varepsilon$ we will end up with couples of systems of equations first of which is the system for $\mathbf{y}_0(x)$, $\mathbf{z}_0(x)$. Since $\mathbf{g}_z$ is invertible we can solve the other systems for $\mathbf{z}_1$ , $\mathbf{z}_2$ and so forth. This leads to the following:
for $\varepsilon^0$:

$$\mathbf{y}_0' = \mathbf{f}(\mathbf{y}_0, \mathbf{z}_0) \tag{2.14}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{y}_0, \mathbf{z}_0) \tag{2.15}$$

for $\varepsilon^1$:

$$\mathbf{y}_1' = \mathbf{f_y}(\mathbf{y_0},\mathbf{z_0})\mathbf{y_1} + \mathbf{f_z}(\mathbf{y_0},\mathbf{z_0})\mathbf{z_1} \tag{2.16}$$

$$\mathbf{z}_0' = \mathbf{g_y}(\mathbf{y_0},\mathbf{z_0})\mathbf{y_1} + \mathbf{g_z}(\mathbf{y_0},\mathbf{z_0})\mathbf{z_1} \tag{2.17}$$

...

for $\varepsilon^\nu$:

$$\mathbf{y}_\nu' = \mathbf{f_y}(\mathbf{y_0},\mathbf{z_0})\mathbf{y}_\nu + \mathbf{f_z}(\mathbf{y_0},\mathbf{z_0})\mathbf{z}_\nu + \phi_\nu(\mathbf{y_0},\mathbf{z_0},\ldots,\mathbf{y}_{\nu-1},\mathbf{z}_{\nu-1}) \tag{2.18}$$

$$\mathbf{z}_{\nu-1}' = \mathbf{g_y}(\mathbf{y_0},\mathbf{z_0})\mathbf{y}_\nu + \mathbf{g_z}(\mathbf{y_0},\mathbf{z_0})\mathbf{z}_\nu + \tag{2.19}$$

$$\psi_\nu(\mathbf{y_0},\mathbf{z_0},\ldots,\mathbf{y}_{\nu-1},\mathbf{z}_{\nu-1}) \tag{2.20}$$

As expected, we see from Eq. 2.14 and Eq. 2.15 that $\mathbf{y_0}(x)$, $\mathbf{z_0}(x)$ is a solution of the reduced system. Since $\mathbf{g_z}$ is invertible, the equation Eq. 2.17 can be solved for $\mathbf{z_1}$. By inserting $\mathbf{z_1}$ into the Eq. 2.15 we obtain a linear differential equation for $\mathbf{y_1}(x)$. Hence, $\mathbf{y_1}(x)$ and $\mathbf{z_1}(x)$ are determined. similarly, we get $\mathbf{y_2}(x)$ and $\mathbf{z_2}(x)$ from Eq. 2.18 and Eq. 2.19, etc.

This construction of the coefficients of Eq. 2.12 and Eq. 2.13 shows that we can choose the initial values $\mathbf{y}_j(0)$ arbitrarily, but that there is no freedom in the choice of $\mathbf{z}_j(0)$. Consequently not every solution of Eq. 1.5 could be written in the form Eq. 2.12 and Eq. 2.13.

The same kind of analysis could be made for a Runge-Kutta method. We consider the Runge-Kutta method

$$\begin{bmatrix}\mathbf{y}_{n+1}\\ \mathbf{z}_{n+1}\end{bmatrix} = \begin{bmatrix}\mathbf{y}_n\\ \mathbf{z}_n\end{bmatrix} + h\sum_{i=1}^{s} b_i \begin{bmatrix}\mathbf{k}_{ni}\\ \mathbf{l}_{ni}\end{bmatrix} \tag{2.21}$$

where

$$\begin{bmatrix}\mathbf{k}_{ni}\\ \varepsilon\mathbf{l}_{ni}\end{bmatrix} = \begin{bmatrix}f(\mathbf{Y}_{ni},\mathbf{Z}_{ni})\\ g(\mathbf{Y}_{ni},\mathbf{Z}_{ni})\end{bmatrix} \tag{2.22}$$

and the internal stages are given by

$$\begin{bmatrix}\mathbf{Y}_{ni}\\ \mathbf{Z}_{ni}\end{bmatrix} = \begin{bmatrix}\mathbf{y}_n\\ \mathbf{z}_n\end{bmatrix} + h\sum_{j=1}^{s} a_{ij} \begin{bmatrix}\mathbf{k}_{nj}\\ \mathbf{l}_{nj}\end{bmatrix} \quad ,i = 1,\ldots,s \tag{2.23}$$

We formally expand all occurring quantities into powers of ε with ε-independent coefficients

$$\mathbf{y}_n = \mathbf{y}_n^0 + \varepsilon\mathbf{y}_n^1 + \varepsilon^2\mathbf{y}_n^2 \ldots \tag{2.24}$$

$$\mathbf{Y}_{ni} = \mathbf{Y}_{ni}^0 + \varepsilon\mathbf{Y}_{ni}^1 + \varepsilon^2\mathbf{Y}_{ni}^2 \ldots \tag{2.25}$$

$$k_{ni} = \mathbf{k}_{ni}^0 + \varepsilon\mathbf{k}_{ni}^1 + \varepsilon^2\mathbf{k}_{ni}^2 \ldots \tag{2.26}$$

and similarly for $\mathbf{z}_n$, $\mathbf{Z}_{ni}$ and $\mathbf{l}_{ni}$. Since Eq. 2.22 has the same form as Eq. 1.5 we will end up with exactly the same systems of equations as those we had after inserting the $\varepsilon$-expansions of $\mathbf{y}(x)$ and $\mathbf{z}(x)$ in Eq. 1.5. If we subtract Eq. 2.24 from Eq. 2.12 we will get formally

$$\mathbf{y}_n - \mathbf{y}(x_n) = \sum_{\nu \geq 0} \varepsilon^\nu (\mathbf{y}_n^\nu - \mathbf{y}_\nu(x_n)) \tag{2.27}$$

$$\mathbf{z}_n - \mathbf{z}(x_n) = \sum_{\nu \geq 0} \varepsilon^\nu (\mathbf{z}_n^\nu - \mathbf{z}_\nu(x_n)) \tag{2.28}$$

The next theorem gives rigorous estimate of the remainder in (2.27)-(2.28).

THEOREM 2.2. [HW96, page 428] *Consider the stiff problem Eq. 1.5 for which* $\mu(g_z(x)) \leq -1$ *with the initial values* $\mathbf{y}(0)$, $\mathbf{z}(0)$ *admitting a smooth solution. Apply the Runge-Kutta method Eq. 2.21-Eq. 2.23 of classical order p and stage order q* ($1 \leq q \leq p$)*. Assume that the method is A-stable. that the stability function satisfies* $\mid R(\inf) \mid < 1$*, and that the eigenvalues of the coefficient matrix A have positive real part. Then for any fixed constant* $c > 0$ *the global error satisfies for* $\varepsilon \leq ch$ *and* $\mu \leq q+1$

$$\mathbf{y}_n - \mathbf{y}(x_n) = \Delta\mathbf{y}_n^0 + \varepsilon\Delta\mathbf{y}_n^1 + \cdots + \varepsilon^\nu \Delta\mathbf{y}_n^\nu + O(\varepsilon^{\nu+1}) \tag{2.29}$$

$$\mathbf{z}_n - \mathbf{z}(x_n) = \Delta\mathbf{z}_n^0 + \varepsilon\Delta\mathbf{z}_n^1 + \cdots + \varepsilon^\nu \Delta\mathbf{z}_n^\nu + O(\varepsilon^{\nu+1}/h) \tag{2.30}$$

*Here* $\Delta\mathbf{y}_n^0 = \mathbf{y}_n^0 - \mathbf{y}_0(x_n)$, $\Delta\mathbf{z}_n^0 = \mathbf{z}_n^0 - \mathbf{z}_0(x_n), \ldots$ *are the global errors of the method applied to the systems derived from Eq. 1.5 and Eq. 2.12,Eq. 2.13. The estimates Eq. 2.29, Eq. 2.30 hold uniformly for* $h \leq h_0$. *and* $nh \leq Const$.

COROLLARY 2.1. *Under the assumptions of Theorem 2.2 the global error of Runge-Kutta method satisfies*

$$\mathbf{y}_n - \mathbf{y}(x_n) = O(h^p) + O(\varepsilon h^{q+1}), \qquad \mathbf{z}_n - \mathbf{z}(x_n) = O(h^{q+1}). \tag{2.31}$$

*If in addition* $a_{si} = b_i$ *for all i we have*

$$\mathbf{z}_n - \mathbf{z}(x_n) = O(h^p) + O(\varepsilon h^q) \tag{2.32}$$

The conclusion is that the explained error when solving a stiff system using a Runge-Kutta method, can be explained as a combination of errors from using the same Runge-Kutta method on the derived DAEs.

The table below shows the theoretical error bounds of some methods.

| Method | $a_{si} = b_i$ | $y$-comp. | $z$-comp. |
|---|---|---|---|
| Radau IA | no | $h^{2s-1} + \varepsilon h^s$ | $h^s$ |
| Radau IIA | yes | $h^{2s-1} + \varepsilon^2 h^s$ | $h^{2s-1} + \varepsilon h^s$ |
| Lobatto IIIC | yes | $h^{2s-2} + \varepsilon h^{s-1}$ | $h^{2s-2} + \varepsilon^2 h^{s-1}$ |
| SDIRK IV(16) | yes | $h^4 + \varepsilon h^2$ | $h^4 + \varepsilon h$ |
| SDIRK IV(18) | yes | $h^4 + \varepsilon h^2$ | $h^2$ |

CHAPTER 3

# Projection Methods

***Written by Mikael Zebbelin Poulsen & Falko Jens Wagner***

**Keywords:** *High index DAEs, index reduction, drift-off phenomenon, differential equations on manifolds, coordinate projection, derivative projection, order preserving projection, invariant projection, over determined systems.*

## 3.1. Introduction

When trying to solve a DAE problem of high index with more traditional methods, it often causes instability in some of the variables, and finally leads to breakdown of convergence and integration of the solution. This is nicely shown in [ESF98, p. 152 ff.].

This chapter will introduce projection methods as a way of handling these special problems. It is assumed that we have methods for solving normal ODE systems and index-1 systems.

**3.1.1. Problem.** The general form of the DAE problem treated in this chapter is

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y},\mathbf{z}) \tag{3.1a}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{y}) \tag{3.1b}$$

where $\mathbf{f}:\mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}^n$ and $\mathbf{g}:\mathbb{R}^n\to\mathbb{R}^m$, which is seen to be a DAE system of differentiation index at least 2 (see [HW96, p. 456]).

**3.1.2. Example case.** As an example case of this type of DAE system, a model of the pendulum system will be used (see [ESF98, p. 150]):

$$
\begin{aligned}
\dot{p_x} &= v_x \\
\dot{p_y} &= v_y \\
\dot{v_x} &= -2p_x\lambda/m \\
\dot{v_y} &= -g/m - 2p_y\lambda/m
\end{aligned}
\tag{3.2}
$$

with the restriction

$$0 = p_x^2 + p_y^2 - l^2 \tag{3.3}$$

It is seen that this equation system, Eq. 3.2 and Eq. 3.3 is a specialized form of Eq. 3.1. It is further in the form described in [HW96, p. 456 (1.15)], and therefore has differentiation index 3.

## 3.2. Index reduction

A way of handling the problem of instability is to construct a new equation system by performing *index reduction* on the original DAE system.

To understand index reduction, it can be useful to notice the definition of the differentiation index (see e.g. [HW96, p. 455]). This index gives the number of times $m$, that one will have to differentiate the equation system

$$
\left.
\begin{aligned}
\frac{d^i}{dx^i}\dot{\mathbf{y}} &= \frac{d^i}{dx^i}\mathbf{f}(\mathbf{y},\mathbf{z}) \\
\mathbf{0} &= \frac{d^i}{dx^i}\mathbf{g}(\mathbf{y})
\end{aligned}
\right. \qquad , i = 0, 1, \ldots, m
\tag{3.4}
$$

with $x$ being the independent variable, to be able to rearrange equations to what is called the "underlying ODE" [HW96].

The principle of index reduction is then, that *one* differentiation of the equation system will give a new set of equations, so that a new equation system, with index one lower, can be set up using algebraic manipulations. This reduction can the be continued in successive steps, lowering the index of the system, and enable the use of methods for lower index problems.

**3.2.1. Example on index reduction.** In this example we will see that, in the pendulum case, we can reduce the index by successive differentiation of the algebraic restriction Eq. 3.3 alone.

We differentiate on both sides with respect to time, and substitute using Eq. 3.2 and get

$$
\begin{aligned}
0 &= 2p_x\dot{p}_x + 2p_y\dot{p}_y \\
&= 2p_x v_x + 2p_y v_y \qquad \Leftrightarrow \\
0 &= p_x v_x + p_y v_y
\end{aligned} \tag{3.5}
$$

Using Eq. 3.5 instead of Eq. 3.3 in the equation system changes the problem into an index-2 problem. Further differentiation of Eq. 3.5 gives

$$
\begin{aligned}
0 &= p_x\dot{v}_x + \dot{p}_x v_x + p_y\dot{v}_y + \dot{p}_y v_y \\
&= p_x(-2p_x\lambda/m) + v_x^2 + p_y(-g - 2p_y\lambda/m) + v_y^2 \\
&= v_x^2 + v_y^2 - gp_y - 2l^2\lambda/m
\end{aligned} \tag{3.6}
$$

Using this equation as the algebraic restriction, makes the problem index-1. Note that, isolating $\lambda$ in this equation and reducing $\lambda$ from the equations Eq. 3.2, actually gives an explicit ODE system.

We want though to show how yet another differentiation can give a normal ODE problem, reducing the index to 0.

$$
\begin{aligned}
0 &= 2v_x\dot{v}_x + 2v_y\dot{v}_y - g\dot{p}_y - 2l^2\dot{\lambda}/m \\
&= 2v_x(-2p_x\lambda/m) + 2v_y(-g - 2p_y\lambda/m) - gv_y - 2l^2\dot{\lambda}/m \\
&= -4\lambda/m(v_x p_x + v_y p_y) - 3gv_y - 2l^2\dot{\lambda}/m \\
&= -3gv_y - 2l^2\dot{\lambda}/m
\end{aligned}
$$

which is reached by insertion of (Eq. 3.5). Isolating $\dot{\lambda}$ we get

$$
\dot{\lambda} = -3mgv_y/(2l^2) \tag{3.7}
$$

We see that by using Eq. 3.7 instead of the algebraic equation Eq. 3.3 we get an explicit ODE system. The high index problem is reduced to an index-0 problem.

*Comment: As is, this is not a proof of the original system having index exactly 3, but it is though a proof that the system has index 3 or less. In other words, just because we differentiated 3 times, and we end up with an index-0 problem, it doesn't mean, that the original systems was index-3. We could just as well have been too bad at rearranging the variables.*

**3.2.2. Restriction to manifold.** The index reduction process just exemplified, can be understood in another way.

Observe that the general problem can be seen as an ODE system Eq. 3.1a restricted to a manifold Eq. 3.1b by varying $\mathbf{z}$. This right value $\mathbf{z}^*$ would be the value of $\mathbf{z}$ for which the solution curve would proceed on the manifold. Locally this means that $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y},\mathbf{z})$ should be in the tangent plane of the manifold (see Figure 3.1). We can express this as

$$\mathbf{g}_y(\mathbf{y})\mathbf{f}(\mathbf{y},\mathbf{z}^*) = \mathbf{0}$$

or just

$$\mathbf{g}_y\mathbf{f} = \mathbf{0}. \tag{3.8}$$

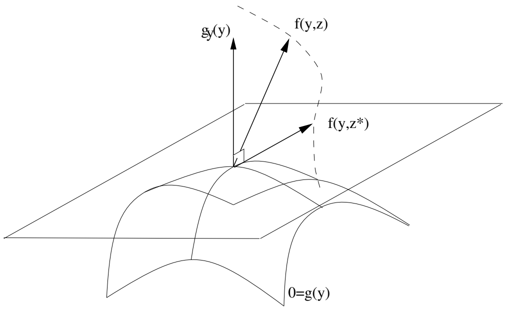



FIGURE 3.1. The manifold and tangent plane, showing how to pick $z = z^*$ so that the solution will proceed on the manifold.

Note that Eq. 3.8 is in fact the differentiation Eq. 3.1b with respect to the independent variable, and therefore corresponds to Eq. 3.5 in the previous example.

If $\mathbf{z}^*$ is uniquely determined from Eq. 3.8 i.e. $\mathbf{g}_y\mathbf{f}_z$ is invertible, Eq. 3.8 together with Eq. 3.1a forms an index-1 system.

If $\mathbf{g}_y\mathbf{f}$ is only a function of $\mathbf{y}$, Eq. 3.8 specifies a manifold like Eq. 3.1b, and the same scheme can be used once more:

$$(\mathbf{g}_y\mathbf{f})_y\mathbf{f} = (\mathbf{g}_{yy}\mathbf{f} + \mathbf{g}_y\mathbf{f}_y)\mathbf{f} = \mathbf{0}$$

or in a more correct notation

$$\mathbf{g}_{yy}(\mathbf{f},\mathbf{f}) + \mathbf{g}_y\mathbf{f}_y\mathbf{f} = \mathbf{0} \tag{3.9}$$

while $\mathbf{g}_{yy}(\cdot,\cdot)$ is a Frechet derivative ([Lam91, p. 158]). Again this is just similar to differentiation of the equation Eq. 3.8.

In general $\mathbf{g}_y\mathbf{f}$ could be a function of $\mathbf{y}$, without the unique determination of $\mathbf{z}^*$ being possible from equation Eq. 3.8 i.e. $(\mathbf{g}_y\mathbf{f})_z$ is not invertible (not of full rank). A way to handle this is described in [HW96, p. 456].

**3.2.3. Implications of reduction.** An exact solution (curve) for the index reduced system is not necessarily a solution for the original system, though the opposite is true. *Comment: In this way, differentiation is only a single implication.*

The index reduced system has more solutions than the original system. To get the solutions to the original system from the reduced system, one has to choose the right initial values/conditions.

Such initial conditions is a point on a solution curve for the original system. It is not a trivial question to find such a point - eg. it is not necessarily sufficient to satisfy the algebraic equations Eq. 3.1b form the original system alone. The problem is called the problem of finding *consistent initial conditions*, and is of relevance even when solving the original system, because most algorithms need such consistent initial conditions.

**3.2.4. Drift-off phenomenon.** Numerically there is another consequence of index reduction, related to the problem just mentioned, but not solved by finding consistent initial conditions. Lets say that the initial conditions are picked according to the above:

When the computation of the numerical solution advances, a new point will be found satisfying a local error condition. This error will be measured and depend on the reduced system and its differentiated algebraic restrictions, eg. Eq. 3.5.

The consequence is that when looking at the defect of the original algebraic restriction, eg. Eq. 3.3, it will develop as the integral of this local error. In this way the points on the computed solution will randomly move away from the manifold specified by the original algebraic restriction.

**3.2.5. Example of drift-off.** Figure 3.2 illustrates how the solutions of the pendulum model changes due to index reduction. In the pendulum case Eq. 3.2, the original index-3 formulation had the algebraic equation Eq. 3.3

$$0 = p_x^2 + p_y^2 - l^2$$

specifying the manifold drawn as the solid curve. The index-2 formulation had the algebraic equation Eq. 3.5

$$0 = p_x v_x + p_y v_y.$$

This equation only specifies that the velocity $(v_x, v_y)$ should be orthogonal to the position vector $(p_x, p_y)$. This is illustrated by the dashed curves in Figure 3.2.

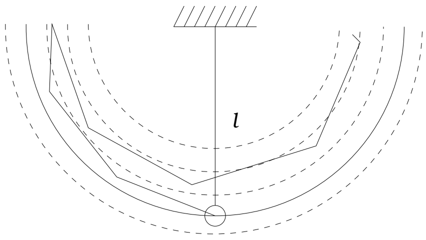



FIGURE 3.2. Expanding set of solutions due to index reduction and illustration of the drift-off phenomenon

We could say that by doing index reduction, we expanded the solution set of the original system to include not only the solutions moving on the solid line, but also all the solutions moving in "parallel" to the solid line.

Illustrated on Figure 3.2, as the piecewise linear curve, is also the drift-off phenomenon. When numerically advancing the solution, the next point is found with respect to the index-2 restriction. The index-2 restriction says, that we shall move in "parallel" with the solid line (along the dashed lines), and not as the index-3 restriction says, to move *on* the solid line. Therefore the solution slowly moves away from (drifts off) the solid line - although "trying" to move in parallel with it.

*Comment: When solving a system the consequence of drift of is not necessarily "worse" than the normal global error. It is though obvious that at least from a "cosmetic" point of view a solution to the pendulum system with shortening length of the pendulum looks bad. But using the model for other purposes might be indifferent to drift-off compared to the global error.*

## 3.3. Projection

The idea is to project the solution points back to the manifold defined by the original system as the steps are computed. This will prevent drift-off, but can also in special cases increase the order of the method.

Let's say that $(\mathbf{y}_{n-1}, \mathbf{z}_{n-1})$ is a point consistent (or "almost consistent") with the original system. Then we take a step using some method on the reduced equation system, and get the point $(\tilde{\mathbf{y}}_n, \tilde{\mathbf{z}}_n)$. This point is then projected on to the manifolds inherent in the original system, giving $(\mathbf{y}_n, \mathbf{z}_n)$ as the next point on the solution curve. This is the general idea, and is illustrated in Figure 3.3.

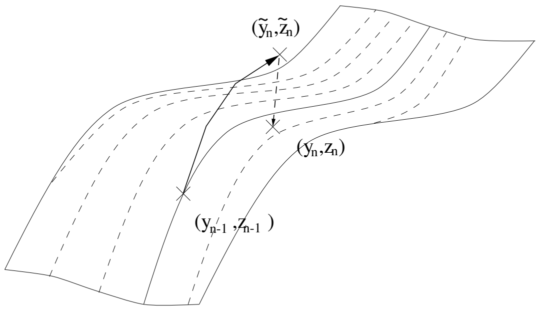



FIGURE 3.3. The general projection method. The broken solid arrow shows the step taken using the reduced system. The dashed arrow shows the projection back to the manifold

Different information from the original and reduced system can be used for various projection methods. As an example it is mentioned in [HW96, p.471] that there is an advantage in determining the velocity constraints and project to these instead of projecting to the position constraints. In general the various methods need not give consistent values, and they don't necessarily preserve the order of the method.

**3.3.1. Coordinate projection.** Coordinate projection uses correction after each step, indifferent of the ODE-part of the differential equation Eq. 3.1a, and only concerned about the manifold Eq. 3.1b, and/or the manifolds specified by the derivatives of this.

Generalizing the idea of [ESF98, p. 165] the method uses the index reduced system to advance the solution from $(\mathbf{y}_{n-1}, \mathbf{z}_{n-1})$ to $(\tilde{\mathbf{y}}_n, \tilde{\mathbf{z}}_n)$. Then the projection is defined by

$$\begin{aligned} \|\tilde{\mathbf{y}}_n - \mathbf{y}_n\|_2 &= \min_{\mathbf{y}_n} \\ \mathbf{g}(\mathbf{y}_n) &= 0 \end{aligned} \tag{3.10}$$

which is a *nonlinear constrained least squares problem* because of the norm chosen. The projection gives the orthogonal projection to the manifold.

In [ESF98, p. 165] the idea is used on different sets of variables and equation in successive steps. Especially equations from the index reduction are included, and it is mentioned in [HW96, p. 471] that this can give better results than using the original algebraic restrictions.

In [ESF98] there are some references to proofs of convergence for various of these projection methods.

**3.3.2. Another projection method and the projected RK method.** This method is explained in [HW96, p. 470]. The method has some nice convergence properties, when applied to index-2 problems.

The ideas is again to use a method on the index-reduced system to advance the solution one step, finding the value $(\tilde{\mathbf{y}}_n, \tilde{\mathbf{z}}_n)$. The method uses projection along the image of $\mathbf{f}_z(\tilde{\mathbf{y}}_n, \tilde{\mathbf{z}}_n)$, and is defined as

$$\begin{aligned} \mathbf{y}_n - \tilde{\mathbf{y}}_n &= \mathbf{f}_z(\tilde{\mathbf{y}}_n, \tilde{\mathbf{z}}_n)\mu \\ \mathbf{g}(\mathbf{y}_n) &= \mathbf{0} \end{aligned} \tag{3.11a}$$

and after this solving for $\mathbf{z}_n$ using

$$\mathbf{g}_y(\mathbf{y}_n)\mathbf{f}(\mathbf{y}_n, \mathbf{z}_n) = \mathbf{0} \tag{3.11b}$$

or some differentiated form of this.

The convergence of the method is shown in [HW96, p. 471] for the index-2 case, where $g_y$ will have full rank. The proof uses the mean value theorem to conclude that, if the error of $\tilde{\mathbf{y}}_n = O(h^p + 1)$, then the error of the projected point will also be $\mathbf{y}_n = O(h^p + 1)$, and finally but important that $\mathbf{z}_n = O(h^p + 1)$.

This method is used in what is called *projected Runge-Kutta methods*. Such a method is simply using a Runge-Kutta method for computation of $(\tilde{\mathbf{y}}_n, \tilde{\mathbf{z}}_n)$, and hence uses this projection method to correct the solution point. The method is described in [HW96, p. 503].

These are not the only projection methods. There are i.e. *simplectic Euler* and *Derivative Projection* [Eic92, p.57] along with other methods for stabilizing the solution of a reduced system.

## 3.4. Special topics

**3.4.1. Systems with invariants.** Some systems can, along with the description of their dynamics, being some ODE system, have additional properties that should be modeled.

The invariant property applies to a system for which some quantity is preserved or constant at every point on the solution curve. The well known example of this, is the conservation of energy in different technical systems. Some methods using projection can use this extra information.

One of these methods is described in [ESF98, p. 173]. Let's assume that $\mathbf{y}_0$ is a (consistent) initial value for the system. For this value we can compute the invariant

$$\phi_0 = \phi(\mathbf{y}_0). \tag{3.12}$$

Computing the next step, at some point $\mathbf{y}_{n-1}$, $\phi(\mathbf{y}_{n-1}) = \phi_0$, one uses some ODE method. The computed point $\tilde{\mathbf{y}}_n$ is then projected as

$$\begin{gathered} \|\tilde{\mathbf{y}}_n - \mathbf{y}_n\|_2 = \min_{\mathbf{y}_n} \\ \phi(\mathbf{y}_n) = \phi_0 \end{gathered} \tag{3.13}$$

to get the next point $\mathbf{y}_n$ on the solution curve. This is seen to be a nonlinear least squares problem.

In [ESF98] there are references to proofs that this method doesn't disturb convergence, and it is described that such a method can minimize the global error.

**3.4.2. Over determined systems.** In the case of systems with invariants, it is obvious that the mentioned systems are actually overdetermined. This can also be the case for the DAE system Eq. 3.2 if all the equations Eq. 3.3 - Eq. 3.6 are included in the model.

In [HW96, p. 477] the case of overdetermined DAE systems is treated. There are different ways of handling these. A simple suggestion is to solve for the least square solution to the system, but one could also think of using some kind of pseudo-inverse, in a Newton-like process.

CHAPTER 4

# BDF-methods

***Written by Astrid Jørdis Kuijers***

**Keywords:** *BDF-methods, multi step-methods, stability properties, A-stability, DASSL, variable step-size.*

In this chapter an introduction to BDF-methods along with a little in general about multi-step-methods will be given. The second part will be about using the BDF-methods on DAEs in different cases. It will go into order, stability and convergence of the BDF-methods used on DAEs with different indexes.

## 4.1. Multi Step-Methods in general and BDF-methods

A general multi step-method may be written as,

$$\frac{1}{h}\sum_{j=0}^{k}\alpha_j\mathbf{y}_{n-j}=\sum_{j=0}^{k}\beta_j\mathbf{f}(x_{n-j},\mathbf{y}_{n-j}),$$

where $h$ refers to the step-size used, $\alpha_j$ and $\beta_j$ are coefficients for respectively the number of backward steps and the function value at the backward steps. $k$ is called the order of the method and denotes how many backward steps are used. The method is an implicit method if $\beta_0 \neq 0$ and if so it will be necessary to calculate the function value in the point that has to be estimated(iteratively). An example of a multi step-method could be,

$$\mathbf{y}_n=\mathbf{y}_{n-2}+\frac{h}{3}(\mathbf{f}_{n-2}+4\mathbf{f}_{n-1}+\mathbf{f}_n),$$

which is called the Milnes-method [NGT93].

**4.1.1. BDF-methods.** The BDF-methods are a group of multi step-methods in which the function value only will be calculated in the point that is going to be found. A variable number of points calculated in the steps before can be used. This means that a BDF-method is an implicit multi-step method where $\beta_0 \neq 0$

but $\beta_{1,\ldots,k} = 0$. BDF stands for backward differentiating formulas. It is called this because of the general formula, that can be written as:

$$hy'_n = \sum_{j=1}^{k} \frac{1}{j} \nabla^j y_n,$$

where $\nabla$ is the backward differens operator.

Another way to write it, is to write it the way it is written in the multi step-part

$$\frac{1}{h} \sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = \beta_0 \mathbf{f}(x_n, \mathbf{y}_n).$$

The simplest BDF-method is the *implicit Euler* method [NGT93],

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\mathbf{f}(x_n, \mathbf{y}_n)$$

which is of order 1.

For BDF-methods the stability for solving ODEs can be investigated by investigating the linear test equation:

$$y' = \lambda y$$

For the method to be stable it requires that the roots of the polynomial

$$\rho(y) - h\lambda\sigma(y) = 0$$

are lying inside the unit circle see [ESF98] page 106. $\sigma$ and $\rho$ are defined in 5.2.

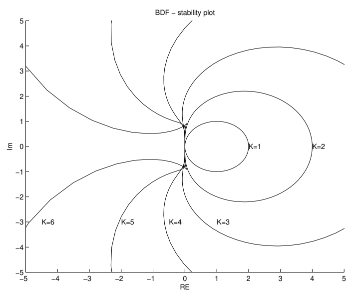



FIGURE 4.1. Stability regions for BDF-methods

In Figure 4.1 the regions of the complex plane, where h$\lambda$ leads to a stable discretization, are pictured. BDF-methods are stable for all $\lambda$ with Re$\lambda < 0$ and $|\arg(-\lambda)| < \alpha$ where $\alpha$ is given as the angle in Figure 4.1 for which Re$\lambda < 0$. This is called A-stability or A($\alpha$)-stability.[ESF98] It is seen, that the region, where the method is stable becomes smaller and smaller for higher orders of the method. A method can only be stable up till the order of 6 because for the order of 7 there are no longer a region of A-stability.

Because of their stability-properties they are good candidates for solving DAEs. In the early 70'th C.W. Gear started to write about using BDF-methods in connection with such problems. Now BDF-methods are widely used in computer codes that solves ODEs and DAEs, such as the DASSL and the LSODI codes for solving DAEs. [AP98] Writes: *'There has been much work on developing convergence results for general multi step methods. For solving DAEs the coefficient of the multi step methods must satisfy a set of order conditions which is in addition to the order conditions for ODEs. It turns out that these additional order conditions are satisfied by BDF methods.'*

## 4.2. BDF-methods applied to DAEs

Applying the BDF formula to the fully implicit DAE Eq. 1.1 leads to

$$\mathbf{F}(x_n, \mathbf{y}_n, \frac{\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j}}{\beta h}) = 0.$$

This could be illustrated by the van der Pol equation given in the introduction where $\varepsilon \to 0$. The implicit Euler is used where $\alpha = \beta = 1$:

$$\begin{aligned} \frac{\mathbf{y}_{2,n} - \mathbf{y}_{2,n-1}}{h} - \mathbf{y}_{2,n} &= 0 \\ \mu^2((1-y_1{}^2) * \mathbf{y}_2 - \mathbf{y}_1) &= 0 \end{aligned}$$

In this case the following theorem holds true [BCP89]:

THEOREM 4.1. *In general a BDF-method of order $k \leq 7$ with constant stepsize applied to a constant coefficient linear DAE of index $\nu$ is convergent of order $O(h^k)$ after $(\nu-1)k+1$ steps.*

This will not be proved but an example will be shown. The index-3 system

$$\begin{aligned} \mathbf{y}_1' &= \mathbf{y}_2 \\ \mathbf{y}_2' &= \mathbf{y}_3 \\ 0 &= \mathbf{y}_1 - \mathbf{g}(x) \end{aligned} \tag{4.1}$$

has the exact solution

$$\begin{aligned}\mathbf{y}_1(x) &= \mathbf{g}(x)\\ \mathbf{y}_2(x) &= \mathbf{g}(x)'\\ \mathbf{y}_3(x) &= \mathbf{g}(x)''\end{aligned}$$

If the index-3 system is solved with the implicit Euler method, then the solution will look like

$$\begin{aligned}\mathbf{y}_{1,n} &= \mathbf{g}(x_n)\\ \mathbf{y}_{2,n} &= \frac{\mathbf{y}_{1,n}-\mathbf{y}_{1,n-1}}{h}\\ \mathbf{y}_{3,n} &= \frac{\mathbf{y}_{2,n}-\mathbf{y}_{2,n-1}}{h}.\end{aligned}$$

It is seen that $y_1$ will be determined exactly at every step, but in the first step $y_2$ and $y_3$ will be determined incorrectly. After 2 steps $y_2$ will however be determined with an accuracy or order $O(h)$, as will $y_3$ be after 3 steps[BCP89, page 43].

If a variable step size BDF-method is used, it is necessary to be much more careful with the method used. The accuracy of a BDF-solution would then be of the order [BCP89]

$$O(h_{max}^q) \qquad \text{where q} = min(k, k-\nu+2).$$

Here $\nu$ is the index of the system to be solved and $k$ is the order of the method used. This means for example that a numerical solution of an index-3 system with a BDF-method of order $k = 1$ would have an accuracy of order $O(1)$, and thus be useless. Many computer codes for solving DAEs such as LSODI or DASSL starts with a BDF-method of order 1 (as no previous steps are available initially), and therefore will fail for systems of index-3 or higher.

**4.2.1. Semi-Explicit Index-1 Systems.** When applying linear multi step-methods (and therefore also BDF-methods) to semi explicit index-1 DAEs (see page 2) they are stable and convergent to the same order of accuracy for the DAE as for the underlying ODE. These problems can therefore be solved with any linear multi-step method, which is appropriate for the underlying ODE, assuming that the constrains are satisfied in each step. [BCP89]

**4.2.2. Fully-Implicit Index-1 Systems.** For the fully implicit index-1 system it can be shown that a constant-step size BDF-method with order $k < 7$ with the initial values given correct to the order of $O(h^k)$ converges with an accuracy of order $O(h^k)$ if each Newton iteration is solved to an accuracy of order $O(h^{k+1})$.[BCP89]

If a variable step-size BDF-method is used (with step-size restrictions as for the standard ODE case) then it will also be convergent for the fully implicit index-1 system.[BCP89]

**4.2.3. Semi-Explicit Index-2 Systems.** A general Semi-Explicit Index-2 Systems can be written as,

$$
\begin{aligned}
\mathbf{y}' &= \mathbf{f}(x, \mathbf{y}, \mathbf{z}) \\
0 &= \mathbf{g}(x, \mathbf{y}, \mathbf{z})
\end{aligned}
$$

where $(\frac{\partial \mathbf{g}}{\partial \mathbf{y}'})^{-1}$ exist and is bounded in a neighborhood of the solution. If a Semi-Explicit Index-2 system is solved with a constant step-size BDF-method of order $k \leq 7$ with the initial values given correct to the order of $O(h^k)$ it converges with an accuracy of order $O(h^k)$ after $k+1$ steps if each Newton iteration is solved to an accuracy of order $O(h^{k+1})$.[BCP89]

If a variable step size BDF-method is used the same as in the index-1 case will happen. It will be convergent if the method is implemented in a way so that it is stable for standard ODEs.[BCP89]

**4.2.4. Index-3 Systems of Hessenberg form.** Such a system can be written as:

$$
\begin{aligned}
\mathbf{y}' &= \mathbf{f}_1(x, \mathbf{y}, \mathbf{z}, \mathbf{u}) \\
\mathbf{z}' &= \mathbf{f}_2(x, \mathbf{y}, \mathbf{z}) \\
0 &= \mathbf{g}(x, \mathbf{y})
\end{aligned}
$$

Where the system will be of index-3 if the matrix $\frac{\partial \mathbf{g}}{\partial \mathbf{z}} \frac{\partial \mathbf{f}_2}{\partial \mathbf{y}} \frac{\partial \mathbf{f}_1}{\partial \mathbf{u}}$ is non singular. If a constant step size BDF-method is used with the starting values at an accuracy of order $O(h^{k+1})$, and the algebraic equations are solved to an accuracy of order $O(h^{k+2})$ if $k \geq 2$ or to an accuracy of order $O(h^{k+3})$ if $k = 1$, the solution would after $k+1$ steps converge to an order of $O(h^k)$.

If a variable step-size BDF-method is used the system might fail to converge. This is shown by trying to solve the equations 4.1 with the implicit Euler method. After three steps the solution would become,

$$
\begin{aligned}
\mathbf{y}_{1,n+1} &= \mathbf{g}(x_{n+1}) \\
\mathbf{y}_{2,n+1} &= \frac{1}{h_{n+1}}(\mathbf{g}(x_{n+1}) - \mathbf{g}(x_n)) \\
\mathbf{y}_{3,n+1} &= \frac{1}{h_{n+1}}((\frac{\mathbf{g}(x_{n+1}) - \mathbf{g}(x_n)}{h_{n+1}}) - (\frac{\mathbf{g}(x_n) - \mathbf{g}(x_{n-1})}{h_n})).
\end{aligned} \tag{4.2}
$$

Expressing Eq. 4.2 as

$$\mathbf{y}_{3,n+1} = (\frac{\mathbf{g}(x_{n+1}) - \mathbf{g}(x_n)}{{h_{n+1}}^2}) - (\frac{\mathbf{g}(x_n) - \mathbf{g}(x_{n-1})}{h_n h_{n+1}})$$

it is seen that if $h_{n+1} \to 0$ for $h_n$ fixed then only the denominator of the second term is guaranteed to tend to zero.

**4.2.5. Summary.** From the above Table 4.1 is presented in order to summarize the results for applying BDF-methods to various types of DAEs.

| **DAE Type** | **Accuracy** |
|---|---|
| Semi-Explicit Index-1 Systems | The same as for the underlying ODE |
| Fully-Implicit Index-1 Systems | $O(h^k)$ see 4.2.2 |
| Semi-Explicit Index-2 Systems | $O(h^k)$ after k+1 steps |
| Index-3 Systems of Hessenberg form | dif. eq. $O(h^k)$, Alg. eq. depend on k |

TABLE 4.1. Accuracy obtained when applying BDF-methods to DAEs.

Notice that when using variable step-size there are no guarantee of convergence for DAEs of index higher than 2.

**4.2.6. DAE-codes.** As mentioned earlier there exist codes using BDF-methods for solving DAEs. Most of these codes are designed for solving index-0 or index-1 DAEs. In [BCP89] some of these codes are described and tested. They have focused on the DASSL-code. DASSL uses BDF-methods with variable step size with orders up till 5. It is reported that the DASSL-code successfully has solved a wide variety of scientific problems, and that there are probabilities of testing out where it went wrong. They report that the finite difference calculation of the Jacobian matrix is the weakest part of the code. It also have problems with handling inconsistent initial conditions and discontinuities. A detailed description of DASSL and a little bit of the code LSODI can be found in [BCP89].

CHAPTER 5

# Stabilized Multi-step Methods Using β-Blocking

***Written by Astrid Jørdis Kuijers & Erik Østergaard***

In this chapter β-blocking of BDF-, Adams and general multi-step methods in different ways will be described, and the consequences of β-blocking will be discussed. β-blocking is used to make an otherwise non-convergent method convergent for a given index-2 problem There will be given an example of β-blocking a method for a Hamiltonian system.
For a description of BDF-methods for DAE see chapter 4.

## 5.1. Adams methods

Adams methods are a group of methods. The predictor is always of order one higher than the corrector, to get the same order of convergence for the explicit and the implicit method. For ODEs Adams methods are used up to order 12, which gives a small order of the error. They are not convergent for index-2 DAEs, but with β-blocking it is possible to make them convergent. For ODEs there are a limit of order approximately 12, but it is only due to the precision of computers. In Theory there could be used orders, that are much higher.

## 5.2. β-blocking

β-blocking implies that a normal multi-step method as in Chapter 4 is corrected on its β-side with an extra term. If an index-2 problem is described by

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}_0(\mathbf{y}) + \mathbf{f}_\lambda(\mathbf{y})\lambda \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}), \end{aligned} \tag{5.1}$$

where $\mathbf{f}(\mathbf{y},\lambda) = \mathbf{f}_0(\mathbf{y}) + \mathbf{f}_\lambda(\mathbf{y})\lambda$ depends linearly on $\lambda$, then a β-blocked multi step-method could look like

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n+j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}(\mathbf{y}_{n+j}, \lambda_{n+j}) - h\mathbf{f}_\lambda(\mathbf{y}_{n+k}) \sum_{j=0}^{k} \gamma_j \lambda_{n+j} \tag{5.2}$$

It could be seen, that the $\beta$ term is modified by a term concerning the algebraic variables. In general in connection with multi-step methods the following generating polynomials are defined:

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j \qquad\qquad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j \tag{5.3}$$

With $\beta$-blocking there would be an extra generating polynomial.

$$\tau(\zeta) = \sum_{j=0}^{k} \gamma_j \zeta^j$$

**5.2.1. Why β-blocking.** There could be problems using otherwise convergent and good methods for index-2 DAEs. Instead of using another method, the method could be changed a little by $\beta$-blocking. The root condition states that the roots in the generating polynomial $\sigma(\zeta)$ should be numerically less than 1. By $\beta$-blocking it is the roots of $\sigma(\zeta)$-$\tau(\zeta)$ that have to be numerically less than 1. Therefore it is possible to change a method a little for a given problem to make it convergent. It could also be used to optimizes a method by optimizing $\sigma(\zeta)$-$\tau(\zeta)$ such that the roots become small. This is written in a theorem as in [HW91]:

THEOREM 5.1. *Consider the index-2 problem Eq. 5.1 discretized by Eq. 5.2 with starting values $\mathbf{x}_j - \mathbf{x}(t_j) = O(h^p)$ and $\lambda_j - \lambda(t_j) = O(h^k)$ for $j = 0 \ldots k-1$, where $p = k$ or $p = k+1$ is the order of the pair $(\rho, \sigma)$. Furthermore, let $\tau$ be chosen such that $\sigma - \tau$ has roots only inside the unit circle and $\tau(\mathbf{x}(t_n)) = O(h^k)$ for any sufficient smooth function. Then the discretization Eq. 5.2 is convergent* [CA94], *i.e. for $nh = t$ constant,*

$$\mathbf{x}_n - \mathbf{x}(t_n) = O(h^p) \qquad\qquad \lambda_n - \lambda(t_n) = O(h^k).$$

By $\beta$-blocking it is seen, that it is only the algebraic variables, that are used in the correction by $\tau$, this implies a reduction in the order of the error for the algebraic equations, but for the differential equations the error would be of the same order as for the method without $\beta$-blocking.

**5.2.2. Difference correcting.** Another way of using $\beta$-blocking is to minimize the order of the error of the method. This is simply done be using the error of the method to $\beta$-block it. A normal ODE could be discretized by a BDF-method by

$$h^{-1} \sum_{j=1}^{k} \frac{\nabla^j}{j} \mathbf{y}_{n+1} = \mathbf{f}(\mathbf{y}_{n+1}) \qquad \text{which have an error of order } O(h^k)$$

The error would in principal terms look like [CA94]

$$\frac{\nabla^k \mathbf{f}(\mathbf{y}(t_{n+1}))}{k+1}$$

Therefore the Difference Corrected BDF-method (DCBDF) would become

$$h^{-1}\sum_{j=1}^{k}\frac{\nabla^j}{j}\mathbf{y}_{n+1} = (1-\frac{\nabla^k}{k+1})\mathbf{f}(\mathbf{y}_{n+1})$$

The difference corrected BDF-method would then have an error of order $O(h^{k+1})$, which is one higher than the first BDF-method.

### 5.3. Consequences of β-blocking

Some of the advantages and disadvantages of β-blocking are listed below.

- It is possible to use otherwise non-convergent methods for Index-2 DAE e.g. Adams-methods
- There is only a convergence order reduction on the algebraic variables
- It can improve an existing method
- It is only possible for k≤3 to obtain a convergent β-blocked Adams method for index-2 DAEs

### 5.4. Discretization of the Euler-Lagrange DAE Formulation

Let us again consider the Lagrange system Eq. 5.1. We define a multi-step method using the ρ and σ polynomials as defined in Eq. 5.3. We introduce the Forward Shift Operator $E$, which shifts the solution one step forward,

$$\mathbf{y}_{n+1} = E(\mathbf{y}_n, h). \tag{5.4}$$

$E$ is invertible, i.e., $\mathbf{y}_n = E^{-1}(\mathbf{y}_{n+1}, h)$. With this operator we introduce the difference operators $\rho = E^{-k}\rho(E)$ and $\sigma = E^{-k}\sigma(E)$ for notational convenience, i.e.

$$\rho\mathbf{y}_n = \sum_{i=0}^{k}\alpha_{k-i}\mathbf{y}_{n-i}, \quad \sigma\mathbf{y}_n = \sum_{i=0}^{k}\beta_{k-i}\mathbf{y}_{n-i}. \tag{5.5}$$

With this notation, the usual multi-step discretization of $\mathbf{y}' = \mathbf{f}(\mathbf{y})$ is found as

$$\rho\mathbf{y}'_n = h\sigma\mathbf{f}(\mathbf{y}_n),$$

and applied to Eq. 5.1 we get

$$
\begin{aligned}
\frac{1}{h}\rho\mathbf{y}_n' &= \sigma(\mathbf{f}_0(\mathbf{y}_n) - \mathbf{f}_\lambda(\mathbf{y}_n)\lambda) \\
\mathbf{0} &= \mathbf{g}(\mathbf{y}_n).
\end{aligned} \tag{5.6}
$$

Now, since using the difference operators introduces instability, we add a stabilizing term, namely the β-blocker term consisting of a τ-difference operator according to Section 5.2:

$$
\begin{aligned}
\frac{1}{h}\rho\mathbf{y}_n' &= \sigma(\mathbf{f}_0(\mathbf{y}_n) - \mathbf{f}_\lambda(\mathbf{y}_n)\lambda) + \mathbf{f}_\lambda(\mathbf{y}_n)\tau\lambda_n \\
\mathbf{0} &= \mathbf{g}(\mathbf{y}_n).
\end{aligned} \tag{5.7}
$$

Using the BDF method with Eq. 5.7 we apply the correction

$$
\tau = -\frac{\nabla^k}{k+1} \tag{5.8}
$$

in order to obtain the Difference Corrected BDF method (DCBDF):

$$
\begin{aligned}
\frac{1}{h}\rho\mathbf{y}_n' &= \sigma(\mathbf{f}_0(\mathbf{y}_n) - \mathbf{f}_\lambda(\mathbf{y}_n)\lambda) - \mathbf{f}_\lambda(\mathbf{y}_n)\nabla^k\lambda_n/(k+1) \\
\mathbf{0} &= \mathbf{g}(\mathbf{y}_n).
\end{aligned} \tag{5.9}
$$

**5.4.1. Commutative application of the DCBDF method.** When applying the DCBDF method to a Lagrange system can be done in two ways, each having their start point in the dynamical variables,

$$
\mathbf{y}' = \mathbf{f}:
$$

1. Apply the algebraic constraints by Lagrange's principle to obtain

$$
\begin{aligned}
\mathbf{y}' &= \mathbf{f}_0 - \mathbf{f}_\lambda\lambda \\
\mathbf{0} &= \mathbf{g},
\end{aligned}
$$

   and discretize by use of the DCBDF to obtain

$$
\begin{aligned}
\frac{1}{h}\rho\mathbf{y}' &= \sigma(\mathbf{f}_0 - \mathbf{f}_\lambda\lambda) \\
\mathbf{0} &= \mathbf{g}.
\end{aligned} \tag{5.10}
$$

2. Discretize by use of the DCBDF to obtain

$$
\frac{1}{h}\rho\mathbf{y}' = \sigma\mathbf{f}_0
$$

and apply the algebraic constraints by Lagrange's principle to obtain

$$\begin{aligned} \frac{1}{h}\rho\mathbf{y}' &= \sigma\mathbf{f}_0 - \mathbf{f}_\lambda\lambda \\ \mathbf{0} &= \mathbf{g}. \end{aligned} \tag{5.11}$$

This seems to be commutative, but method 1 is unstable whereas method 2 is stable . Also, note the difference between the output Eq. 5.10 and Eq. 5.11. In the unstable case the $\sigma$ operator takes $\mathbf{f}$ and $\lambda$ as input, whereas $\sigma$ only takes $\mathbf{f}$ as input in the stable case.

**5.4.2. Example: The Trapezoidal Rule.** Let us look at the Trapezoidal integration

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \frac{h}{2}(\mathbf{f}_n + \mathbf{f}_{n-1}) \tag{5.12}$$

which has the polynomials

$$\rho(\zeta) = 1 - \zeta, \quad \sigma(\zeta) = \frac{1}{2}\zeta + \frac{1}{2}. \tag{5.13}$$

Here the strict root condition is not satisfied, since $\sigma$ has a root on the unit circle; $\sigma(-1) = 0$, hence the method is not stable. Using the notation of the difference operators we get

$$\rho_{TR} = \nabla, \quad \sigma_{TR} = (1 + E^{-1})/2. \tag{5.14}$$

Application of the Trapezoidal Rule to the Lagrange system Eq. 5.1 gives

$$\begin{aligned} \frac{1}{h}\nabla\mathbf{y}_n &= \sigma_{TR}(\mathbf{f}_0(\mathbf{y}_n) - \mathbf{f}_\lambda(\mathbf{y}_n)\lambda_n) \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}_n), \end{aligned} \tag{5.15}$$

and we introduce a correction term in agreement with Eq. 5.8 $-\mathbf{f}_\lambda(\mathbf{y}_n)\nabla\lambda_n/2$ (since $k = 1$), i.e., $\tau = \nabla/2$, which gives (using $\mathbf{f}_0 \equiv \mathbf{f}_0(\mathbf{y}_n)$ and $\mathbf{f}_\lambda \equiv \mathbf{f}_\lambda(\mathbf{y}_n)$ to simplify notation)

$$\begin{aligned}
\frac{1}{h}\nabla \mathbf{y}_n &= \sigma_{TR}(\mathbf{f}_0 - \mathbf{f}_\lambda \lambda_n) - \mathbf{f}_\lambda \nabla \lambda_n/2 \\
&= \sigma_{TR}\mathbf{f}_0 - \sigma_{TR}\mathbf{f}_\lambda \lambda_n - \mathbf{f}_\lambda \nabla \lambda_n/2 \\
&= \sigma_{TR}\mathbf{f}_0 - \frac{1}{2}(1+E^{-1})\mathbf{f}_\lambda \lambda_n - \mathbf{f}_\lambda \nabla \lambda_n/2 \\
&= \sigma_{TR}\mathbf{f}_0 - \frac{1}{2}\mathbf{f}_\lambda \lambda_n - \frac{1}{2}E^{-1}\mathbf{f}_\lambda \lambda_n - \mathbf{f}_\lambda \nabla \lambda_n/2 \\
&= \sigma_{TR}\mathbf{f}_0 - \frac{1}{2}\mathbf{f}_\lambda \lambda_n - \frac{1}{2}\mathbf{f}_\lambda(\mathbf{y}_{n-1})\lambda_{n-1} - \mathbf{f}_\lambda(\lambda_n - \lambda_{n-1})/2 \\
&= \sigma_{TR}\mathbf{f}_0 - \mathbf{f}_\lambda \lambda_n + \frac{1}{2}\left[\nabla \mathbf{f}_\lambda\right]\lambda_{n-1}.
\end{aligned}$$

This formulation is implicit in all terms. The correction term is a normal difference approximation, which deviates the order by $O(h)$ in agreement with Theorem 5.1 hence the consistency of the discretization drops one order. This affects primarily $\lambda$ (the constraint variables), since the other variables already are lowered due to the discretization.

## 5.5. Observed relative error

Simulations show , that the properties expressed in the Theorem 5.1 do hold empirically as well. Depicting the step-size vs. the relative error for the position variable and for the algebraic variable shows a loglog connection in agreement with the expectations. It is furthermore seen, that by application of the DCBDF, one obtains an error reduction of the order $O(h)$ on the position variable, which was to be expected according to Theorem 5.1.

# Part 2

# Special Issues

CHAPTER 6

# Algebraic Stability

***Written by Anton Antonov Antonov***

**Keywords:** *Algebraic stability, A-stability, B-stability, AN-stability, BN-stability,* **B***-consistency*

Algebraic stability

This chapter presents the notion of algebraic stability. We will use the method as demonstrated in [But87, Chap. 35].

Linear stability theory explores the equation

$$y'(x) = \lambda y(x)$$

where this equation of course could be considered as system of equations. From it we can make generalizations in two ways

$$y'(x) = q(x)y(x) \tag{6.1}$$

$$y'(x) = f(y(x)) \tag{6.2}$$

which both are described by

$$y'(x) = f(x, y(x)) \tag{6.3}$$

Good stability properties of a method applied to the equation (6.1) are designated by the name "*AN*-stability". Apparently the suffix "N" stands for "non-autonomous". If these properties are good according to (6.2) then they are said to reveal "B-stability". Last, general obedient behavior i.e. nice according (6.3) is called "*BN*-stability" which affirmates the impression that the suffix "N" stands for "non-autonomous".

## 6.1. General linear stability - *AN*-stability

For a single step of a Runge-Kutta method from $x_{n-1}$ to $x_{n-1}+h$, let $z_i = hq(x_{n-1}+hc_i), (i = 1, 2, ..., s)$ so that the stages $Y_1, Y_2, ..., Y_s$ and the final result $y_n$ are computed from the given value $y_{n-1}$ by the equations

$$Y_i = y_{n-1} + \sum_{j=1}^{s} a_{ij} z_i Y_j, i = 1, 2, ..., s \tag{6.4}$$

$$y_n = y_{n-1} + \sum_{i=1}^{s} b_i z_i Y_i \tag{6.5}$$

Let $Z = \text{diag}(z_1, z_2, ..., z_s)$ so Eq. 6.4 and Eq. 6.5 can be rewritten as

$$(\mathbf{I} - \mathbf{AZ})\mathbf{Y} = y_{n-1}\mathbf{e} \tag{6.6}$$

$$y_n = y_{n-1} + \mathbf{b}^T \mathbf{ZY} \tag{6.7}$$

where $\mathbf{Y} = Y_1, Y_2, ..., Y_s{}^T$. Assuming that the linear equations occurring in Eq. 6.6 have solution, which will be the case if $\det((\mathbf{I} - \mathbf{AZ})\mathbf{Y}) \neq 0$, the value of $y_n / y_{n-1}$ is found to be

$$y_n / y_{n-1} = R(\mathbf{Z})$$

where $R(\mathbf{Z})$is given by Eq. 6.10 below and generates the function $r$ given by

$$r(z) = 1 + z\mathbf{b}^T (\mathbf{I} - z\mathbf{A})^{-1}\mathbf{e}.$$

Note that $r(z) = R(z\mathbf{I})$.

DEFINITION 6.1. A Runge-Kutta method is said to be *AN-stable* if for all $z_1, z_2, ..., z_s \in \mathbb{C}^-$ such that $z_i = z_j$ if $c_i = c_j$,

$$\det(\mathbf{I} - \mathbf{AZ}) \neq 0, \tag{6.8}$$

$$\|R(\mathbf{Z})\| \leq 1 \tag{6.9}$$

$$R(\mathbf{Z}) = 1 + \mathbf{b}^T \mathbf{Z}(\mathbf{I} - \mathbf{AZ})^{-1}\mathbf{e} \tag{6.10}$$

It can be seen from Definition 6.1 that *AN*-stability is not necessary a consequence from A-stability[1]. *AN*-stability could be characterized in a fairly satisfactory way. We will need to make use of the following definition which generalizes positive definiteness for two non-symmetric matrices.

DEFINITION 6.2. An $s \times s$ matrix $\mathbf{A}$ is *positive definite* if there exists a diagonal matrix $\mathbf{D} = \text{diag}(d_1, d_2, ..., d_s)$ with $d_1, d_2, ..., d_s > 0$ such that $\mathbf{DA} + \mathbf{A}^T\mathbf{D}$ is positive definite.

[1]A method is called *A*-stable if $r(z) < 1, \forall z \in \mathbb{C}^-$ where $r(z)$ is defined for the standard test problem $y' = qy$ as $y_n = r(z)^n, z = hq$.

It is easy to see that a symmetric matrix is positive definite in the sense of this definition iff it is positive definite in the usual sense. It is also easy to see that if $\mathbf{A}$ is positive definite then it cannot be singular because if $\mathbf{Av} = 0$ then $\mathbf{v}^T(\mathbf{DA} + \mathbf{A}^T\mathbf{D})\mathbf{v} = 0$ and, furthermore, because of the congruence $(\mathbf{A}^{-1})^T(\mathbf{DA} + \mathbf{A}^T\mathbf{D})\mathbf{A}^{-1} = \mathbf{DA}^{-1} + \mathbf{A}^{-1})^T\mathbf{D}$ the matrix $\mathbf{A}^{-1}$ is also positive definite.

For the characterization of *AN*-stability define $\mathbf{B} = \mathrm{diag}(b_1, b_2, ..., b_s)$ so that $b^T = \mathbf{e}^T\mathbf{B}$ and define $\mathbf{M}$ by Eq. 6.11 below so that $\mathbf{M}$ is a symmetric matrix with the $(i, j)$ element equal to $b_i a_{ij} + b_j a_{ji} - b_i b_j$. Because of its importance in a wider context, $\mathbf{M}$ is also known as the *algebraic stability matrix* for the method. In the theorem which follows, from Burrage and Butcher (1979) we restrict ourselves to non-confluent methods in the sense that $c_i \neq c_j$ for $i \neq j$.

THEOREM 6.1. *If non-confluent Runge-Kutta method is AN-stable then*

$$\mathbf{M} = \mathbf{MA} + \mathbf{A}^T\mathbf{B} - \mathbf{bb}^T \tag{6.11}$$

*and* $\mathbf{B} = diag(b_1, b_2, ..., b_s)$ *are each positive semidefinite matrices.*

The reading of the proof could be omitted. It is presented here in order to explain the strange appearance of the matrix $M$. We will just refer to the last formula in the proof.

**Proof:** If $b_i < 0$ for some $i \in \{1, 2, ..., s\}$ then choose $\mathbf{Z}$ so that $z_j = 0$ if $j \neq i$ and $z_i = -\varepsilon$ for $\varepsilon$ a positive real number. We see that

$$\|\mathbf{R}(\mathbf{Z})\| = 1 - \varepsilon b_i + O(\varepsilon^2)$$

which exceeds 1 for $\varepsilon$ sufficiently small.

Having established that $b_1, b_2, ..., b_s \geq 0$, choose $\mathbf{Z} = \mathrm{diag}(i\varepsilon y_1, \ldots, i\varepsilon y_s)$ where $i = \sqrt{-1}, \mathbf{y} = [y_1, y_2, ..., y_s]^T$ is an arbitrary real vector and $\varepsilon$ a non-zero real number. Expand $(\mathbf{I} - \mathbf{AZ})^{-1}$ by the time series in Eq. 6.10 and we find

$$\mathbf{R}(\mathbf{Z}) = 1 + \varepsilon i\mathbf{b}^T - \varepsilon^2 y^T\mathbf{BAy} + O(\varepsilon^3)$$

so that

$$\begin{aligned} \mathbf{R}(\mathbf{Z}) &= \mathbf{R}(\overline{\mathbf{Z}})\mathbf{R}(\mathbf{Z}) \\ = 1 + \varepsilon i(\mathbf{b}^T\mathbf{y} - \mathbf{y}^T\mathbf{b}) &- \varepsilon^2\mathbf{y}^T(\mathbf{BA} + \mathbf{A}^T\mathbf{B} - \mathbf{bb}^T)\mathbf{y} + O(\varepsilon^3) \\ &= 1 - \varepsilon^2\mathbf{y}^T\mathbf{My} + O(\varepsilon^3) \end{aligned}$$

which is greater than 1 for $\varepsilon$ sufficiently small unless $\mathbf{y}^T\mathbf{My} \geq 0$. ■

Since the requirement of the matrix $\mathbf{B} = \mathrm{diag}(b_1, b_2, ..., b_s)$ to be positive definite is quite natural, the condition for the matrix $\mathbf{M}$ to be positive semidefinite is the important one. When the matrix $\mathbf{B}$ is positive definite the method is consistent in the following sense. $b_1, b_2, ..., b_s$ are something like weights for taking an average of the representatives of the derivative in order for the point of the next step to be calculated with this averaged slope. There is no sense for one or more of these weights to be negative. The first of the next two rows of graphics presents a consistent averaging. The second presents a non-consistent one[2].

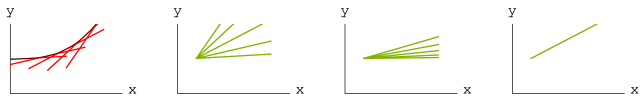



FIGURE 6.1. **B**-Consistent averaging of a Runge-Kutta method for the equation $y' = f(x)$

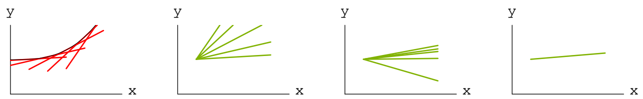



FIGURE 6.2. Non **B**-Consistent averaging

## 6.2. Non-linear stability

We consider the stability of methods with a non-linear test problem $y'(x) = f(y(x))$ in which $f$ fulfills the one-sided Lipschitz condition with one-sided Lipschitz constant 0.

DEFINITION 6.3. The function $f : X \to X$ is said to satisfy *one-sided Lipschitz condition* if there exists a number $\lambda$, known as *one-sided Lipschitz constant* for $f$, such that for all $u, v \in X$,

$$[f(u) - f(v)]^T (u - v) \leq \lambda \|u - v\|^2$$

From the definition above it can be seen that $f(x)$ is decreasing when $\lambda = 0$. If $y_1$ and $y_2$ are solutions of the problem $y = f'(y)$ with initial values at $x_0$ then for $\lambda = 0$ we have that $\|y(x) - z(x)\| \leq \|y(x_0) - z(x_0)\|$, for any $x \geq x_0$ i.e. the primitive of $f$ is a contractive mapping. We consider stability condition that guarantees that computed solution-matrix have similar behavior.

[2]The notion consistency for a Runge-Kutta method is used when it satisfies the condition $\sum_{i=1}^{s} b_i = 1$

DEFINITION 6.4. A Runge-Kutta method is said to be *B-stable* if for any problem $y'(x) = f(y(x))$ where $f$ is such that $[f(u) - f(v)]^T(u - v) \leq 0$ for all $u$ and $v$, then $\|y_n - z_n\| \leq \|y_{n-1} - z_{n-1}\|$ for two solution sequences $\dots, y_n, y_{n-1}, \dots$ and $\dots, z_n, z_{n-1}, \dots$

If the coefficients of the method are

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} \tag{6.12}$$

then the matrix  defined in Eq. 6.11 has the individual element

$$m_{ij} = b_i a_{ij} + b_j a_{ji} - b_i b_j, \qquad i, j = 1, 2, \dots, s. \tag{6.13}$$

DEFINITION 6.5. If for a method defined by Eq. 6.12 and $M$ given by Eq. 6.13 is positive semidefinite and none of $b_1, b_2, \dots, b_s$ is negative, then the method is said to be *algebraically stable*.

The difference between algebraic stability and $AN$-stability is that the later one is for non-confluent methods.

Since $M$ is symmetric, it can be written in the form $\mathbf{M} = \mathbf{N}^T\mathbf{G}\mathbf{N}$ where $\mathbf{G} = \operatorname{diag}(g_1, g_2, \dots, g_s)$ and $\mathbf{N}$ is non-singular. If the method is algebraically stable then

$$g_1, g_2, \dots, g_s \geq 0 \tag{6.14}$$

THEOREM 6.2. *An algebraically stable Runge-Kutta method is B-stable.*

We will skip the rigorous proof but we will explain why we could expect that. In the proof of Theorem 6.1 we saw that the conditions for algebraic stability imply contractive behavior of $R(Z) = y_n/y_{n-1}$ and contractivity of course should imply $\|y_n - z_n\| \leq \|y_{n-1} - z_{n-1}\|$ - the condition in $B$-stability definition. If the definition of $B$-stability were modified to use the test equation Eq. 6.3:

$$y'(x) = f(x, y(x))$$

where for all $x, u, v, [fx, u - f(x, v)]^T(u - v) \leq 0$, then, as it stated in the theorem Theorem 6.3 below, an algebraically stable method still enjoys the modified $B$-stability property. Since the modified property has similar relationship to $B$-stability as $AN$-stability has to $A$-stability it is known as '$BN$-stability'.

DEFINITION 6.6. A Runge-Kutta method is said to be *B-stable* if for any problem $y'(x) = f(x, y(x))$ where $f$ is such that $[f(x,u) - f(x,v)]^T (u - v) \leq 0$ for all $u$ and $v$, then $\|y_n - z_n\| \leq \|y_{n-1} - z_{n-1}\|$ for two solution sequences $\dots, y_n, y_{n-1}, \dots$ and $\dots, z_n, z_{n-1}, \dots$

THEOREM 6.3. *An algebraically stable Runge-Kutta method is BN-stable.*

CHAPTER 7

# Singularities in DEs

***Written by Erik Østergaard***

**Keywords:** *Singularity, impasse-point, projection, augmention*

This section concerns singularities in DAEs and ODEs and how to handle these. Definitions are given and a formalism for projecting the ODE part of the DAE onto the algebraic manifold which handle local singularities correctly is presented.

The method is presented in two ways: Firstly (as done in section 7.2) almost without algebraic symbols in order to establish an overlook of the approach. Secondly (as done in section 7.3) a detailed description of the method including all the algebra is presented.

Finally a pseudo code (algorithm) for implementing the method is given.

## 7.1. Motivation

Consider the DAE initial value problem

$$\begin{aligned} \dot{x}_2 &= 1 && \text{differential (DE) part} \\ x_1^2 + x_2 &= 0 && \text{algebraic (AE) part} \\ \mathbf{x}(0) &= (1,-1) && \text{initial conditions (I.C.),} \end{aligned}$$

which has the unique solution

$$\mathbf{x}(t) = (\sqrt{1-t}, t-1).$$

We see, that $\mathbf{x}(1) = (0,0)$ and that $\lim_{t \to 1_{\div}} \mathbf{x}(t) = \mathbf{x}(1)$. Despite the existence of the solution in $t = 1$, a DAE solver cannot continue the integration beyond $t = 1$ since the derivative

$$\dot{\mathbf{x}}(t) = \left(\frac{-1}{2\sqrt{1-t}}, 1\right)$$

has as singularity at $t = 1$, which makes the step size decrease until the process is stopped. This type of singularities arises often in applications, especially in electrical engineering, where such points are called *impasse points* . The nomenclature is fairly descriptive since these points *are* impassable for numerical integrators. It is now the goal to derive a method for obtaining the solution both at and beyond the impasse point.

The approach is to project the DE part of the DAE into it's own algebraic equation, also named the manifold, in a neighborhood of the impasse point. This results in an ODE with a singularity, and the second step is then to augment the ODE system to an ODE without singularities. It is possible to integrate this system e.g. using the RK method. The computed solution is then transformed back through the two steps to the original DAE. See figure 7.1.

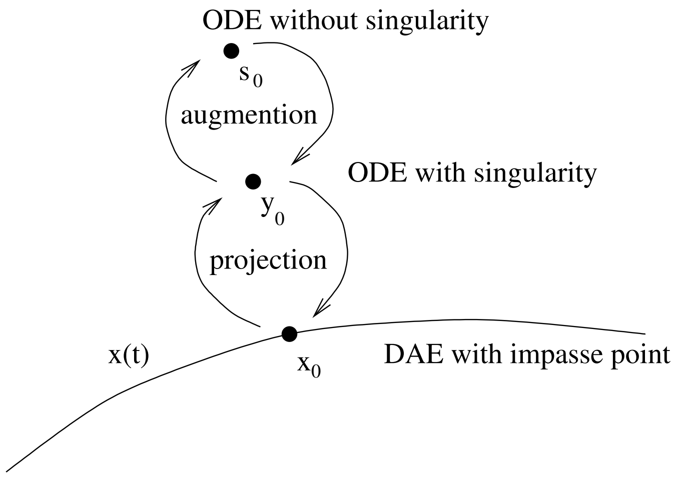



FIGURE 7.1. Visualization of the projection/augmention principle

### 7.2. Description of the method

Consider the autonomous DAE

$$\mathbf{A}(\mathbf{x})\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}),$$

where $\mathbf{x} \in I \subset \mathbb{R}^n, \quad \mathbf{g}: I \to \mathbb{R}^n, \quad \mathbf{A}: I \to \mathbb{R}^{n\times n}$ and $\forall \mathbf{x} \in I: \text{rank}\mathbf{A}(\mathbf{x}) = r < n.$

Let us now consider the DAE only on the manifold defined by the algebraic constraint, i.e. we project the DE part of the DAE onto the manifold using a so called *chart* $\phi$ (see also paragraph 7.3.2). Since we now consider the DAE on the manifold alone, we can handle the resulting expressions as being a pure ODE (see paragraph 7.3.3). Thus we now consider the autonomous ODE

$$\mathbf{B}(\mathbf{y})\dot{\mathbf{y}} = \mathbf{h}(\mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y_0},$$

where $\mathbf{y} \in I \subset \mathbb{R}^n, \quad \mathbf{h}: I \to \mathbb{R}^n, \quad \mathbf{B}: I \to \mathbb{R}^{n\times n}$.

Let us assume, that the mass matrix $\mathbf{B}$ for the ODE has non-full rank in the point $\mathbf{y_0}$, i.e. $\operatorname{rank}\mathbf{B}(\mathbf{y_0}) < n$. The point $\mathbf{y_0}$ is then called a singular point of the ODE. The corresponding point belonging to the DAE-space is $\mathbf{x_0} = \phi(\mathbf{y_0})$, and it is called an impasse point for the DAE.

In analogy with naming $\mathbf{y_0}$ a singularity for the ODE, $\mathbf{x_0}$ is called a singularity for the DAE.

Now, by choosing suiting normalizations, transformations and orthonormal basises (see paragraph 7.3.4) we are able to form an ODE

$$\frac{d\eta}{ds}(s) = \mathbf{v}(\eta(s)),$$

which does not have a singularity in the point $s_0$ corresponding to the singular point $\mathbf{y_0}$, and which matches the solution of the ODE in an area around the singular point. This process is called augmention .

It is now possible to integrate the nice non-singular ODE using standard methods (a RK is satisfactory) in order to pass the singular point.

Finally, we augment the solution back into the ODE containing the singular point and project the solution here into the manifold of the DAE using the chart.

## 7.3. Detailed description of the method

**7.3.1. Singularities in ODEs.** Consider the autonomous ODE

$$\mathbf{B}(\mathbf{y})\dot{\mathbf{y}} = \mathbf{h}(\mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y_0}, \tag{7.1}$$

where $\mathbf{y} \in I \subset \mathbb{R}^n, \quad \mathbf{h}: I \to \mathbb{R}^n, \quad \mathbf{B}: I \to \mathbb{R}^{n\times n}$.

DEFINITION 7.1. The point $\mathbf{y_0}$ is called a regular point if $\operatorname{rank}\mathbf{B}(\mathbf{y_0}) = n$. The point $\mathbf{y_0}$ is called a singular point if $\operatorname{rank}\mathbf{B}(\mathbf{y_0}) < n$ and if $\mathbf{y_0}$ is a limit point of regular points.

We introduce the operator $\alpha(\mathbf{y}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ as

$$\alpha(\mathbf{y})(\mathbf{u},\mathbf{v}) = \langle \mathbf{h}(\mathbf{y}),\mathbf{v}\rangle\langle (D\mathbf{B}(\mathbf{y})\mathbf{u})\mathbf{u},\mathbf{v}\rangle, \tag{7.2}$$

where $\langle\cdot,\cdot\rangle$ is the well known natural inner product and $D$ is the Jacobian operator. $\mathbf{u}$ and $\mathbf{v}$ are arbitrarily chosen such that $u \in \ker\mathbf{B}(\mathbf{y}) \setminus \{\mathbf{0}\}$ and $v \in \ker\mathbf{B}(\mathbf{y})^T \setminus \{\mathbf{0}\}$. This function can be shown ([PR94a], [PR94b], and [Rhe84]) to be nonzero. We will use $\alpha(\mathbf{y})$ as shorthand for $\alpha(\mathbf{y})(\mathbf{u},\mathbf{v})$.

DEFINITION 7.2. The singular point $\mathbf{y_0}$ is accessible if $\alpha(\mathbf{y_0}) < 0$. The singular point $\mathbf{y_0}$ is inaccessible if $\alpha(\mathbf{y_0}) > 0$.

Accessibility is defined as the existence of a final $t = t^*$ such that there exists an interval $J_1 = ]-t^*, 0]$ where the system has a solution $\mathbf{y_1}(t)$, $\mathbf{y_1}(t) \to \mathbf{y_0}$ for $t \to 0^-$.

Likewise, inaccessibility is defined as the existence of a final $t = t^*$ such that there exists an interval $J_2 = [0, t^*[$ where the system has a solution $\mathbf{y_2}(t)$, $\mathbf{y_2}(t) \to \mathbf{y_0}$ for $t \to 0^+$.

At the singular point, $\|\dot{\mathbf{y}}(t)\| \to \infty$ for $t \to 0$, thus the integrator fails since the step length tends to zero.

**7.3.2. Projection: A little more on charts.** Consider the manifold $M = \{\mathbf{y} \in \mathbb{R}^n \mid g(\mathbf{y}) = 0\}$ and the vector field $\mathbf{v} : M \to \mathbb{R}^n$. We define the ODE on $M$ as

$$\mathbf{y}' = \mathbf{v}(\mathbf{y}), \quad \mathbf{y} \in M \subset \mathbb{R}^n.$$

Consider also a lower dimensional manifold $E \subset \mathbb{R}^{n-r}$. We define the local chart as

$$\phi_{\mathbf{i}} : M_i \subset M \to E_i \subset E. \tag{7.3}$$

A solution of (7.3.2) can then be found by the following method:

1. Project $\mathbf{v}$ onto $E_i$ via the chart $\phi_{\mathbf{i}}$ by multiplying $\mathbf{v}(\mathbf{y})$ with $D\phi_{\mathbf{i}}(\mathbf{y})$ (see 7.11).
2. Apply standard methods to the projected field (e.g. a RK solver).
3. Pull the solution back on the manifold $M_i$ using ${\phi_{\mathbf{i}}}^{-1}$.

**7.3.3. Singularities in DAEs.** Again, consider the autonomous DAE

$$\mathbf{A}(\mathbf{x})\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}), \tag{7.4}$$

where $\mathbf{x} \in I \subset \mathbb{R}^n, \quad \mathbf{g} : I \to \mathbb{R}^n, \quad \mathbf{A} : I \to \mathbb{R}^{n \times n}$ and $\forall \mathbf{x} \in I : \mathrm{rank}\mathbf{A}(\mathbf{x}) = r < n$.

We introduce the so called *local chart* $\phi^{-\mathbf{1}}$ that defines a bijective projection of the algebraic part $W$ of the DAE onto a manifold with lower dimension. Set $W = \{\mathbf{x} \in I : \mathbf{g}(\mathbf{x}) \in \mathrm{rge}\mathbf{A}(\mathbf{x})\}$. The chart $\phi(\mathbf{y}) = \mathbf{x}$ maps $\mathbf{y} \in U \subset \mathbb{R}^r$ into $\mathbf{x} \in V \subset W$.

DEFINITION 7.3. The point $\mathbf{x_0}$ is an accessible or inaccessible impasse point (i.e. singular point) of the DAE if and only if the point $\mathbf{y_0} = \phi^{-\mathbf{1}}(\mathbf{x_0})$ is an accessible or inaccessible singular point of the reducing projection of the DAE to an ODE locally to $\mathbf{x_0}$.

We introduce the projection operator $\mathbf{P_i}$. The operator is connected to the local chart $\phi_\mathbf{i}$, and $\mathbf{P_i}$ projects the vector field in $\mathbb{R}^n$ onto the set rge$\mathbf{A}(\mathbf{x_0})$. This defines the functions

$$\begin{aligned} \mathbf{y} &= {\phi_\mathbf{i}}^{-1}(\mathbf{x}) \\ \mathbf{B}(\mathbf{y}) &= \mathbf{P_i}\left(\mathbf{A}(\phi_\mathbf{i}(\mathbf{y}))\cdot D\phi_\mathbf{i}(\mathbf{y})\right) \\ \mathbf{h}(\mathbf{y}) &= \mathbf{P_i}\left(\mathbf{g}(\phi_\mathbf{i}(\mathbf{y}))\right), \end{aligned} \tag{7.5}$$

which establishes the projected ODE

$$\mathbf{B}(\mathbf{y})\dot{\mathbf{y}} = \mathbf{H}(\mathbf{y}). \tag{7.6}$$

It can be shown ([PR94a], [PR94b] and [Rhe84]), that the reduction (7.5) of the DAE (7.2) to the ODE (7.6) on the manifold holds the same properties (existence, uniqueness, smoothness etc.) as the original DAE.

LEMMA 7.1. *An impasse point is a limit point of non-impasse points.*

Since the projection is bijective (and even diffeomorfic), and since singular points are limit points of regular points according to the definition 7.1, the lemma is easily proved.

**7.3.4. Augmention: Making the singular ODE a nonsingular ODE.** Given the ODE

$$\mathbf{B}(\mathbf{y})\dot{\mathbf{y}} = \mathbf{h}(\mathbf{y}) \tag{7.7}$$

with a singular point $\mathbf{y_0}$. Let $\mathbf{y} = \mathbf{y}(t)$ be a $C^1$ solution to (7.7). Let $\tau = \tau(s)$ be a $C^1$ function with $\tau'(s) > 0$ define a transformation of the independent variable $t$ as $t = \tau(s)$. Finally, let $\eta(s) = \mathbf{y}(t) = \mathbf{y}(\tau(s))$. By application of the chain rule and from (7.7) we see, that $\eta$ satisfies

$$\mathbf{B}(\eta)\frac{d\eta}{ds} = \frac{d\tau}{ds}\mathbf{h}(\eta). \tag{7.8}$$

Since $\frac{d\mathbf{y}}{dt}\mid_{\mathbf{y}=\mathbf{y_0}} = \infty$ we choose $\tau$ such that $\frac{d\tau}{ds}\mid_{\mathbf{y}=\mathbf{y_0}} = 0$ and $\frac{d\eta}{ds}\mid_{\mathbf{y}=\mathbf{y_0}} < \infty$. We specify $\tau$ implicitly by the normalization $\mathbf{c}^T(d\eta/ds) = 1$, where $\mathbf{c} \in \mathbb{R}^n$ is chosen to suit the problem. We define the functions $\mathbf{v}(s)$ and $\gamma(\mathbf{y})$ by

$$\begin{bmatrix} \mathbf{B}(\mathbf{y}) & -\mathbf{h}(\mathbf{y}) \\ \mathbf{c}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}(\mathbf{y}) \\ \gamma(\mathbf{y}) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}. \tag{7.9}$$

$c \in \mathbb{R}^n$ is chosen such that the matrix in (7.9) evaluated at the singular point $y_0$ is regular. We will demonstrate, that this augmention of the original ODE fulfills our purpose.

We define an orthonormal basis of the algebraic part named $\mathbf{g_2}(\mathbf{x})$ as $\ker D\mathbf{g_2}(\mathbf{x})$,

$$\mathbf{U}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^{r\times n}, \tag{7.10}$$

and set $\mathbf{U_c} = \mathbf{U}(\mathbf{x_c}) \in \mathbb{R}^{r\times n}$. This ensures existence of the chart $\phi^{-1}$. We note that the Jacobian of the chart then can be found as

$$D\phi(\mathbf{y}) = \mathbf{U}(\phi(\mathbf{y}))[\mathbf{U_c}^T\mathbf{U}(\phi(\mathbf{y}))]^{-1}. \tag{7.11}$$

We form the matrix system

$$\begin{bmatrix} \mathbf{B_c} & -\mathbf{h_c} & \mathbf{e}^* \\ \mathbf{c_1}^T & 0 & 0 \\ \mathbf{c_2}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} \\ w_1 & w_2 \\ z_1 & z_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.12}$$

This system is build using the singular value decomposition (SVD) method. The system is used in order to ensure that the DAE singularity is transformed to the equilibrium point $\mathbf{x_c}$ of the augmented ODE system. Output is the vectors $\mathbf{e}^*, \mathbf{c_1}, \mathbf{c_2}$ which form an orthonormal basis by

(7.13)

$$\mathbf{V}_L^T(\mathbf{B_c}, -\mathbf{h_c})\mathbf{V}_R = (\Sigma, 0) \quad \text{(from the SVD decomp.)}$$
$$\mathbf{V_L} \in \mathbb{R}^{r\times r}, \mathbf{V_R} \in \mathbb{R}^{r+1\times r+1}, \Sigma = \operatorname{diag}(\sigma_1, ..., \sigma_r), \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq 0.$$

Recall that $\mathbf{e}_k^m, k = 1, ..., m$ denotes the unit basis vectors of $\mathbb{R}^m$. Let

$$\begin{pmatrix} \mathbf{u_1} \\ \omega_1 \end{pmatrix} = \mathbf{V_R}\mathbf{e}_{r+1}^{r+1}, \quad \begin{pmatrix} \mathbf{u_2} \\ \omega_2 \end{pmatrix} = \mathbf{V_R}\mathbf{e}_{r+1}^{r} \tag{7.14}$$

which gives us the augmention elements $\mathbf{e}, \mathbf{c_1}, \mathbf{c_2}$:

$$\mathbf{e}^* = \mathbf{V_L}\mathbf{e}_r^r, \quad \mathbf{c_1} = \frac{\mathbf{u_1}}{||\mathbf{u_1}||}, \quad \mathbf{c_2} = \frac{\mathbf{u_2}}{||\mathbf{u_2}||}. \tag{7.15}$$

Finally $\mathbf{v}$ and $\gamma$ is evaluated by solving (7.12) and using

$$\mathbf{v}(\mathbf{y}) \;=\; z_2\mathbf{v_1} - z_1\mathbf{v_2} \tag{7.16}$$
$$\gamma(\mathbf{y}) \;=\; z_2 w_1 - z_1 w_2. \tag{7.17}$$

From (7.9) it is seen that $\mathbf{v} \neq \mathbf{0}\ \forall \mathbf{y}$ and that $\gamma \neq 0\ \forall \mathbf{y} \neq \mathbf{y_0}$, that $\gamma$ is monotone and that $\gamma(\mathbf{y_0}) = 0$. Hence the problem

$$\frac{dt}{ds} = \gamma(\mathbf{y}(t)), \quad t(0) = 0 \tag{7.18}$$

has a unique, monotonically increasing solution $\tau$ in a neighborhood of $t = t_0$. Also, since $\gamma$ changes sign when passing the singular point it can be used as an indicator for whether we have passed the singular point.

From (7.9) we have

$$\mathbf{B}(\mathbf{y})\mathbf{v}(\mathbf{y}) - \mathbf{h}(\mathbf{y})\gamma(\mathbf{y}) = \mathbf{0}. \tag{7.19}$$

The ODE (7.7) then gives us

$$\mathbf{B}(\mathbf{y})\mathbf{v}(\mathbf{y}) - \mathbf{B}(\mathbf{y})\dot{\mathbf{y}}\gamma(\mathbf{y}) = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{v}(\mathbf{y}) - \dot{\mathbf{y}}\gamma(\mathbf{y}) = \mathbf{0}. \tag{7.20}$$

Applying the chain rule we find

$$\frac{d\eta}{ds}(s) = \gamma(\eta(s))\frac{d\mathbf{y}}{dt}(\tau(s)), \tag{7.21}$$

which in (7.20) gives the nice nonsingular ODE

$$\mathbf{B_c}\dot{\mathbf{y}} = \mathbf{h_c} \quad \underset{\text{equiv.}}{\longleftrightarrow} \quad \frac{d\eta}{ds}(s) = \mathbf{v}(\eta(s)). \tag{7.22}$$

Now we can use a one step method e.g. a RK solver to integrate (7.22) in order to find $\eta$ and thereby finding $\mathbf{y}$. The legacy of using a RK solver is shown in [KN92].

## 7.4. Implementing the algorithm

We here present an algorithm (see figure 7.2) for integration beyond impasse points. This method needs a few comments.

We assume, that we have an algorithm that performs normal DAE integration until an impasse point is found. We then switch to using the functions in figure 7.2 in order to pass the impasse point, where after we can use the standard DAE method again. Remember, that impasse points in this context are supposed to be limit points of non-impasse points.

In the previous sections we established existence of the functions and definitions that forms the projection and augmention . These formalisms *only* exist as symbols and not as accessible objects. This means, that every time we need an evaluation of $\eta$, we need all the previous evaluations as well. Since this among other things consists of QR and SVD factorizations, the price for doing this is rather high – and the factorizations cannot be ensured to be reused in the next steps. Thus we need to perform the two steps illustrated in figure 7.1 for every RK step we apply. In comparison to the projection/augmention calculations, the RK step is a low price to pay.

Also, since the projection/augmention only exist locally, multi-step methods are non applicable. Only one-step methods can be applied.

Now to the implementation, figure 7.2. We rewrite the DAE as

$$\begin{bmatrix} \mathbf{A}(\mathbf{x}) \\ \mathbf{0} \end{bmatrix} \dot{\mathbf{x}} = \begin{bmatrix} \mathbf{g_1}(\mathbf{x}) \\ \mathbf{g_2}(\mathbf{x}) \end{bmatrix} \tag{7.23}$$

and denote $\mathbf{x_c}$ as the center of the local coordinate system belonging to the local chart. $\mathbf{x_c}$ is assumed close to the impasse point $\mathbf{x_0}$.

Finally, for ease of notation we QR-factorize the Jacobian of the algebraic part of the DAE as

$$D\mathbf{g_2}(\mathbf{x_c})^T = [\mathbf{Q_1}\ \mathbf{Q_2}] \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}. \tag{7.24}$$

```
Implementation of the method
proc SingDAE(x_k, tol)
   x_c := x_k
   Compute QR factorization of (7.24). Output: R, Q_1, Q_2
   U_c := Q_2
   B_c := A(x_c)U_c
   h_c := g_1(x_c)
   Compute augmentation vectors in (7.15)
   (v, γ) := Eval_v
   y_1 := RK(y_0)
   if (signγ_0 ≠ signγ_1)
     then
           We have passed the singular point
      else
           Do it again
   fi
   x_{k+1} := Eval_φ(y_1)
   return(x_{k+1})
end
```

---

```
Evaluation of the chart φ (the tangential coordinate system)
proc Eval_φ(y, R, Q_1)
   x := x_c + U_c y
   while ok = FALSE do
          Solve R^T z = g_2(x) for z
          x := x − Q_1 z
   od
   return(x)
end
```

---

```
Evaluation of the augmented system
proc Eval_v(Q_1, R, e*, c_1, c_2)
   x := Eval_φ(y)
   Compute Dφ(y) by (7.11)
   B_c := A(x)Dφ(y)
   Solve the augmented system (7.12)
   Compute v(y) and γ(y) from(7.17)
   return(v, γ)
end
```

FIGURE 7.2. Implementation of the projection/augmention method

CHAPTER 8

# ODEs with invariants

***Written by Brian Elmegaard***

**Keywords:** *Invariants, Hamiltonian systems, symplectic systems, symplectic methods, reformulation of ODEs with invariants, stabilization of ODEs with invariants, restricted three-body system.*

Given a fully determined system of ODEs

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}, t) \tag{8.1}$$

an *invariant* expression, i.e. a function, $c = c(\mathbf{y}, \mathbf{y}', t)$, which has a constant value for given initial values of the variables, may be recognized. An algebraic solution to the ODE system will be exact with the invariant having a constant value.
This may not be the case for a numerical solution to the discretized system. For some systems it may be more important to fulfill the invariant requirement than the ODE.

## 8.1. Examples of systems with invariants

In many systems invariants arise from a requirement of having conservation of energy. Some cases are the Hamiltonian systems occurring in molecular dynamics. The Hamiltonian system is given as:

$$\begin{array}{l} \mathbf{q}_i' = \frac{\partial H}{\partial \mathbf{p}_i} \\ \mathbf{p}_i' = \frac{\partial H}{\partial \mathbf{q}_i} \end{array} \text{, for } i = 1, l \tag{8.2}$$

where $q$ and $p$ are position and momentum of the $l$ particles, respectively. $H$ denotes the Hamiltonian which is the total energy of the system and must be conserved.
In spacecraft dynamics it is also important to preserve the energy of the planet-spaceship system. An example of such a system is given in section 8.3.
A very basic example of a system with an invariant is the simple pendulum rotating about a fixed point at one end. The length of the rod of the pendulum has

to be constant, but different examples [AP98], [ESF98] show that this is not the case with any applied method.

## 8.2. Solving ODEs with invariants

**8.2.1. Methods preserving invariants.** Some methods may inherently preserve special invariants. For example Runge-Kutta methods will preserve linear constraints. Methods preserving constraints of up to quadratic order exist.
If the divergence of $\mathbf{f}$ ($\mathbf{divf} = \sum_{i=1}^{n} \partial \mathbf{f}/\partial \mathbf{x_i}$) is zero for a system of ODEs the system is termed as being *symplectic*. One example of a symplectic system is a Hamiltonian system. *Symplectic methods* preserve the divergence of the system and may be applied in this case.

**8.2.2. Reformulation as a DAE.** The ODE system with an invariant

$$\mathbf{y}' = \mathbf{f}(\mathbf{y},t) \tag{8.3a}$$

$$\mathbf{c}(\mathbf{y},\mathbf{y}',t) = \mathbf{0} \tag{8.3b}$$

may be reformulated as a DAE

$$\mathbf{y}' = \mathbf{f}(\mathbf{y},t) - \mathbf{D}(\mathbf{y})\mathbf{z} \tag{8.4a}$$

$$\mathbf{0} = \mathbf{c}(\mathbf{y},\mathbf{y}',t) \tag{8.4b}$$

The function $\mathbf{D}(\mathbf{y})$ is to be chosen so $\mathbf{CD}$ with $\mathbf{C} = \frac{\partial \mathbf{c}}{\partial \mathbf{y}}$ is boundedly invertible. The choice is often $\left(\frac{\partial \mathbf{C}}{\partial \mathbf{y}}\right)^T$, which will define an orthogonal projection of the solution onto the constraint manifold [AP98]. There may for a given system be reasons for choosing other projections.
The DAE reformulation approach may, however, not be advantageous because the DAE will be of index-2 or higher. To solve an index-2 DAE one should be very careful when selecting the solution method.

**8.2.3. Stabilization of the ODE system.** To avoid the DAE one may apply stabilization of the solution to the ODE system. Poststabilization of the solution is what is done in projection methods (see chapter 3). One may also embed the stabilization in the ODE system by formulating it as:

$$\mathbf{y}' = \mathbf{f}(\mathbf{y},t) - \gamma\mathbf{F}(\mathbf{y})\mathbf{c}(\mathbf{y}) \tag{8.5}$$

The solution to the system is stable, i.e. it is tending towards vanishing invariant changes from any initial point, provided $\gamma > \frac{\gamma_0}{\lambda_0}$ and $\mathbf{CF}$ is positive definite. $\lambda_0$ is the smallest eigenvalue of $\mathbf{CF}$. $\gamma_0$ is a constant such $\|\mathbf{Cf}(\mathbf{y},t)\| \le \gamma_0\|\mathbf{c}(\mathbf{y})\|$. The matrix function $\mathbf{F}$ may be chosen as $\mathbf{F} = \mathbf{D}(\mathbf{CD})^{-1} \Rightarrow \lambda_0 = 1$. In many cases $\gamma_0$ is zero. In such cases the invariant is termed *integral*.

## 8.3. Flying to the moon

In [DM70] a *restricted three-body problem* involving motion and interaction of the earth, the moon and a space ship is presented. The trajectory of the space-

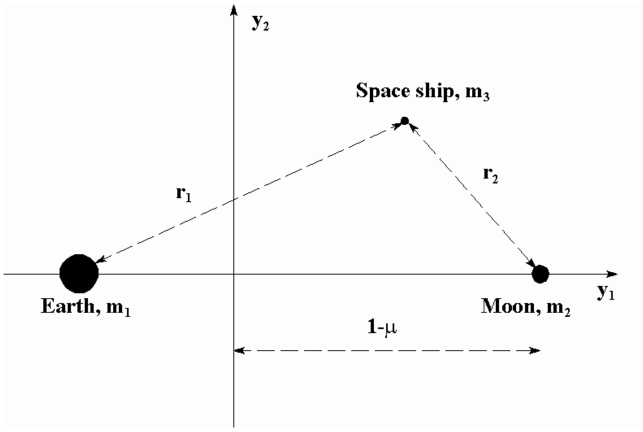



FIGURE 8.1. The restricted three-body problem for earth, moon and spaceship

ship is formulated from Newton's second law in a rotating coordinate system $(y_1, y_2)$ with origin in the center of mass of earth, $m_1$ and moon, $m_2$. The mass ratio is $\frac{m_2}{m_1+m_2} = 0.01215$. The trajectory is defined by two ordinary differential equations:

$$y_1'' = y_1 + 2y_2' - \frac{(1-\mu)(y_1+\mu)}{r_1^3} - \frac{\mu(y_1 - 1 + \mu)}{r_2^3} \tag{8.6}$$

$$y_2'' = y_2 - 2y_1' - \frac{(1-\mu)y_2}{r_1^3} - \frac{\mu y_2}{r_2^3} \tag{8.7}$$

where

$$r_1 = \sqrt{(y_1+\mu)^2 + y_2^2} \tag{8.8}$$

$$r_2 = \sqrt{(y_1 - 1 + \mu)^2 + y_2^2} \tag{8.9}$$

There is an important invariant known as the Jacobi integral in this system:

$$J = \frac{1}{2}({y_1'}^2 + {y_2'}^2 - y_1^2 - y_2^2) - \frac{1-\mu}{r_1} - \frac{\mu}{r_2} \tag{8.10}$$

Below some simulation results obtained with the ode45 method of matlab are presented. The simulations are based on a proposed problem in [DM70]: *Try to make a spaceship positioned near the earth fly to the moon.*
The graphs presented are initialized at the same position $(-\mu/2, 0.1)$ and with the same velocity in the $y_2$-direction $y_2' = 4$. The $y_1$-velocity is increased to produce results where the spacecraft end at the Earth, goes around the Earth, reaches a stable point between the to bodies, reaches the moon and goes to outer space. The most interesting result from the modeling point-of-view is that the Jacobi integral is not at all constant. This means that the applied method is not applicable for this problem.

**8.3.1. Going nowhere.** Figure 8.2 show the trajectory of the spaceship at low initial speed. The ship returns to the Earth, and the trajectory would in reality end there. The value of the Jacobi Integral changes a lot over the path as shown figure 8.3.

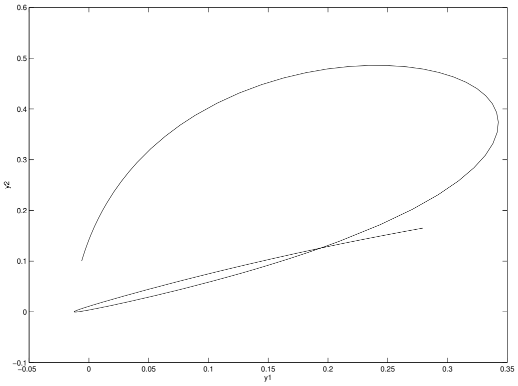



FIGURE 8.2. Trajectory of spaceship for $y_1' = 0.5$

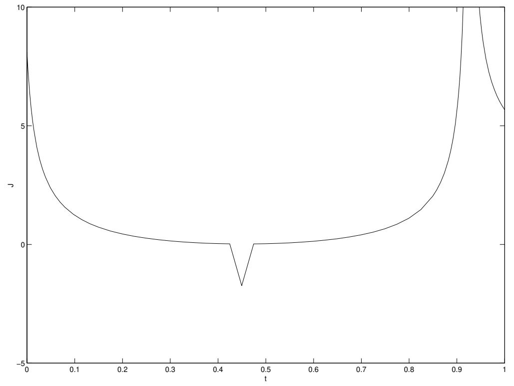



FIGURE 8.3. Jacobi Integral for $y_1' = 0.5$

**8.3.2. Round about the Earth.** In figure 8.4 the spacecraft goes around the earth and is eventually getting back. Figure 8.5 showing the Jacobi Integral values shows that changes in the integral value are distinct whenever the ship is near the planet.

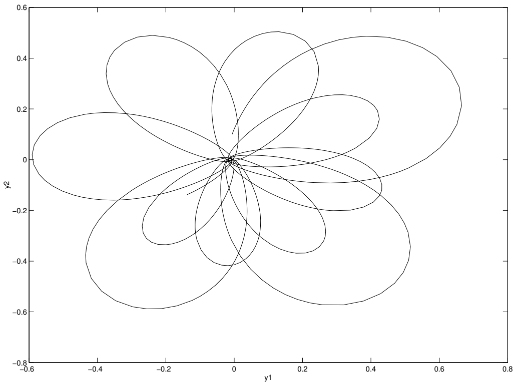



FIGURE 8.4. Trajectory of spaceship for $y_1' = 1.1$

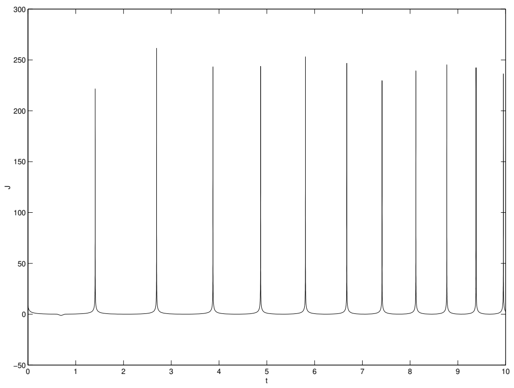



FIGURE 8.5. Jacobi Integral for $y_1' = 1.1$

**8.3.3. Lunar landing.** In figure 8.6 the spaceship reaches the moon. It is however questionable if this also would be the case in a real three-body system, due to the change of the integral over the path (figure 8.7).

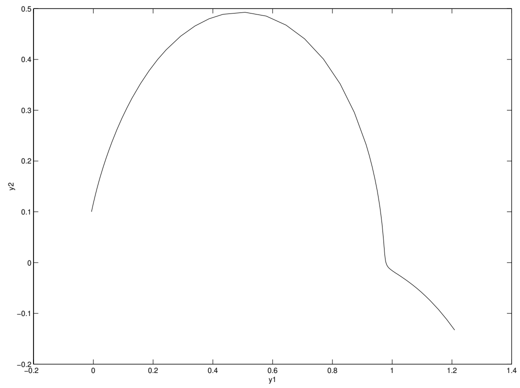



FIGURE 8.6. Trajectory of spaceship for $y_1' = 1.4$

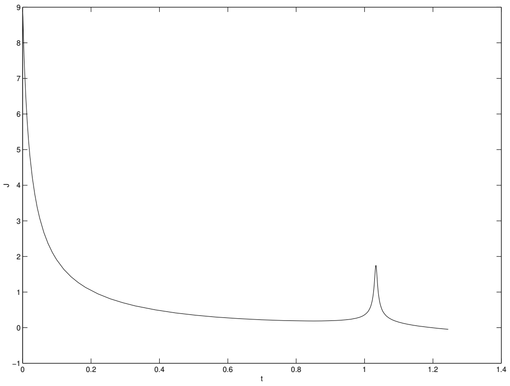



FIGURE 8.7. Jacobi Integral for $y_1' = 1.4$

**8.3.4. What's out there?** In figure 8.8 the spaceship leaves the earth-moon system and goes into outer space. The values of the integral are shown to be non-constant also for this example in figure 8.9.

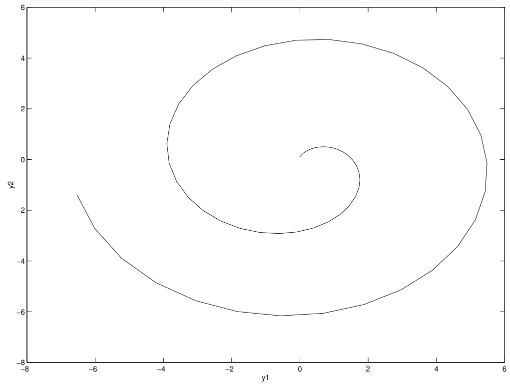



FIGURE 8.8. Trajectory of spaceship for $y_1' = 2.0$

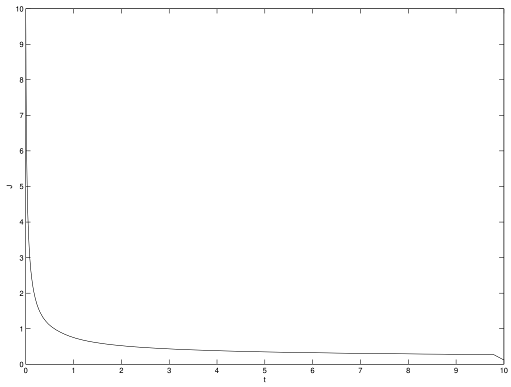



FIGURE 8.9. Jacobi Integral for $y_1' = 2.0$

### 8.4. Comments on Improvement of the Solution

In section 8.2 ways to avoid the badness of the solutions presented above have been described. These approaches may be introduced to overcome the problems encountered in the three-body system. A few comments on how to apply the improved methods may be in order here:

**Methods preserving invariants:** are not readily available for this system.

**Reformulation as a DAE:** will require a higher-index method for an implicit DAE system. Such a method has not been available.

**Poststabilization of the ODE system:** is the recommended approach according to [AP98]. One problem which has not been discussed in this reference is how to deal with the system if it is implicit in the derivatives and not only a function of the variables in $\mathbf{y}$. This is the case for the system examined and a way to overcome this complication has not been found in the literature available.

# Part 3

# Applications

CHAPTER 9

# Multibody Systems

***Written by Falko Jens Wagner***

a

**Keywords:** *Multibody Systems, mechanical systems, Pendulum example, truck model, rigid bodies, elastic bodies, joints, degrees of freedom, multibody system analysis, suspension model*

Multibody systems is one area, in which methods for solving DAEs are of special interest. This chapter is about multibody systems, why they result in DAE systems and what kind of problems, that can arise when dealing with multibody systems and formulating their corresponding DAE system.

## 9.1. What is a Multibody System?

A multibody system is a mechanical system, that consists of one or more bodies. (Strictly, a multibody system should consist of more than one body, but essentially a system consisting of only one body is treated in the same way. So this is a matter of generalization.)

The bodies can either be rigid or elastic. They can have a mass and/or a torque. Somehow they are connected with each other (or they would not be considered being in the same system). These mass-less connections can either be force-elements, like springs, dampers or friction, or non-elastic joints, i.e. translational or rotational joints.

Joints give a restriction in the movement and can transfer forces from one body to another. Joints are reducing the degrees of freedom for the system [Eic92, page 12].

While the force elements result in explicitly given forces, joints result in implicit given forces (like before: they are only transferring forces from one body to another).

The equations necessary for describing a multibody system are the equation of motion (Newton's 2. law)

$$\mathbf{M}(\mathbf{p})\mathbf{p}'' = \sum \mathbf{f}(\mathbf{p}, \mathbf{p}') \tag{9.1}$$

where **M** is the mass matrix, **p** denotes the position and **f** are the forces (both internal and external) acting on the system.

These equations (Eq. 9.1) are the result of external forces acting on the system or due to force-elements such as springs, dampers or friction. As it can be seen in (Eq. 9.1), the forces can be a function of both position and velocity. Friction forces are typically a function of velocity, whereas spring forces are a function of the position (i.e. elongation) of the spring.

In addition to the equation of motion, there can be a restriction

$$\mathbf{g}(\mathbf{p}) = 0 \tag{9.2}$$

on the position, caused by the different joints.

### 9.2. Why this interest in Multibody Systems?

In general, computer simulations are used to investigate systems of the real world, that would otherwise not have been possible to analyze, either because it is too expensive, takes too long time or is too dangerous, etc..

When a physical systems has to be simulated, the way of seeing the system as consisting of several bodies, that are interacting which each other in certain ways, is one formalism, that is very easy to understand and to simulate. For example, the bodies are typically represented only in one point, namely their center of mass.

When modeling a physical system using the concept of multiple bodies connected by force elements or joints, the resulting equation system (using Eq. 9.1 and Eq. 9.2) will end up being a differential algebraic equation system.

In addition to that, multibody systems are easy to understand and therefore serve as good examples for the concept of differential algebraic equation systems.

### 9.3. A little bit of history repeating

Solving DAE systems is not a trivial task, and good and stable numerical methods have only been implemented for a short period of time.

Up until the sixties the equations of motion for multibody systems where solved by hand on the basis of the Euler-Lagrange formulation [Eic92, page 14]:

$$
\begin{aligned}
\frac{d}{dt}\left(\frac{\partial \mathbf{T}(\mathbf{p},\mathbf{p}')}{\partial \mathbf{p}'}\right) - \frac{\partial \mathbf{T}(\mathbf{p},\mathbf{p}')}{\partial \mathbf{p}} &= \mathbf{Q}(\mathbf{p},\mathbf{p}') + \mathbf{G}^{\mathbf{T}}\mathbf{p})\lambda \\
\mathbf{g}(\mathbf{p}) &= 0
\end{aligned} \tag{9.3}
$$

where $\mathbf{T}$ is a function for the kinematic energy, $\mathbf{p}$ is the position, $\lambda$ is the Lagrange multiplier, $\mathbf{Q}$ represents the external forces acting on the system, $\mathbf{G}$ are (here) gravitational forces and $\mathbf{g}$ are the algebraic constraints on the position. An analytic solution was then looked for. But as the systems got bigger and more complex, it was necessary to automate the process of stating the equations of motion. Programs for this purpose where developed, that could analyze the structure of the system, and automatically derive the equations of motion on the basis of the description of the system-elements.

Differentiating (Eq. 9.3) and substituting with $\mathbf{f} = \mathbf{Q} - \mathbf{T}_{\mathbf{p}'\mathbf{p}}\mathbf{p}' + \mathbf{T}_{\mathbf{p}}$ yields the implicit-linear differential equation

$$
\mathbf{M}(\mathbf{p})\mathbf{p}'' = \mathbf{f}\mathbf{p},\mathbf{p}') + \mathbf{G}^{\mathbf{T}}(\mathbf{p})\lambda \tag{9.4}
$$

which together with

$$
\mathbf{g}(\mathbf{p}) = 0
$$

forms a DAE system, that is in the form of (Eq. 9.1 and Eq. 9.2).

Then there are different formalisms to solve these systems, which basically are the theme of these notes.

## 9.4. The tasks in multibody system analysis

The process of transforming a real world system into a multibody dynamics model is not much unlike the process of general modeling and simulation, see Figure 9.1 for process diagram.

First of all the mechanical system in question has to be analyzed with respect to simulation purpose, i.e. *"What function/behavior of the system do we want to investigate?"*.

According to [ESF98, page 13] the following 5 aspects of investigation can be considered:

- Kinematic analysis
- Static equilibrium
- Dynamic simulation
- Linear system analysis

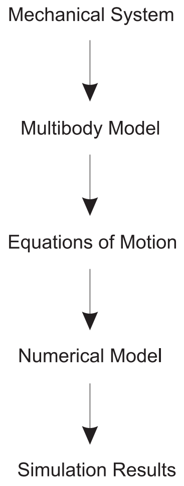



FIGURE 9.1. Process of Multibody System Simulation.

- Design and optimal control

Depending of the kind of the analysis, the abstraction takes place accordingly, and the system is split into individual *bodies*, that are connected by different *joints*. This job has to be done by the engineer, since it requires a lot of experience and *"feeling"* for the system and the concept of multibody system modeling.

After the multibody model has been established, the different formalisms can be used in order to derive the equations of motion, using information about the structure of the system and the individual system elements.

Once the equations of motion have been stated, the mathematical model has to be transformed into something, that is solvable by an appropriate numerical method, see section 9.6 about which problems that can arise, when formulating a multibody system model.

When the mathematical model has been boiled down to something reasonably solvable, the numerics takes over and calculates the solution to the numerical model.

The results have then to be analyzed and the results have to be compared to the simulation objectives. The question is whether the simulation results reflect the problems that had to be investigated. If no, the model has to be adapted in a way so that the simulation objectives can be fulfilled. This is often a highly iterative process.

## 9.5. Example of a multibody system

The following examples of a multibody system are used to illustrate the structure of multibody systems.

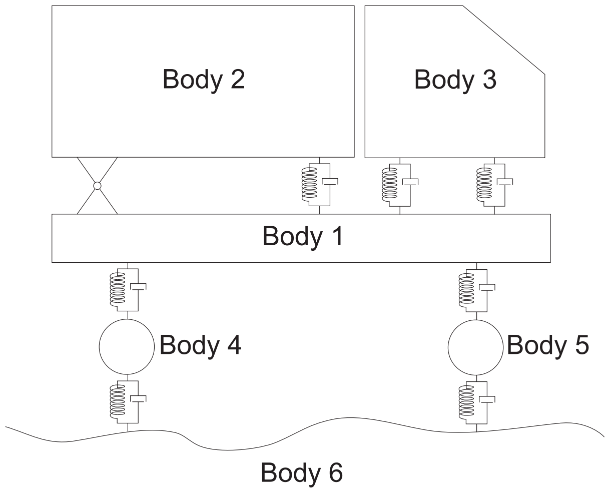



FIGURE 9.2. Example of a multibody system

**9.5.1. The multibody truck.** This figure (Figure 9.2) shows, how a truck is modeled using the multibody concept. A model is always a simplification of the real world, and so is this truck.

The truck model consists of 5 bodies. Body 1 is the chassis of the truck, body 2 is the load area, while body 3 is the cockpit. Bodies 4 and 5 represent the wheels.

For analyzing the movement of the truck the ground has been included in the system as body 6. The wheel suspension is represented by springs and dampers. The same model is used for the contact between the wheels and the road.

The load area has a rotational joint at the back, which allows the loading area to be tilted. The other end of the load area, as the cockpit, has a spring-damper suspension.

This is the multibody model of the truck.

After defining the bodies, the equations of motion are applied. These equations won't be stated here, but can be found in [ESF98, page 20], from which this example is taken.

Instead a part of the truck model is used as an example for analysis. Figure 9.3 shows the wheel suspension of the truck.

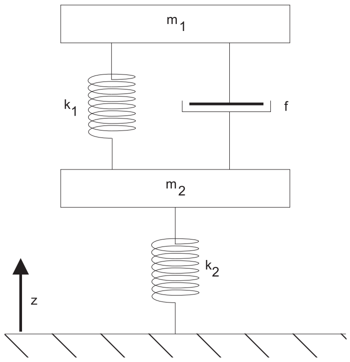



FIGURE 9.3. Wheel suspension

Here $m_1$ is the mass of the truck (or the part of the truck, which load rests on this wheel). $m_2$ is the mass of the wheel, $f$ is the damping coefficient taken by the damper, $k_1$ is the spring constant of the suspension and $k_2$ is the spring constant of the rubber wheel itself (since the tire is very elastic, the oscillation of it should be taken into account).

The mathematical model for this system is given by Eq. 9.1 and becomes (see also [MKZ92, p.45]):

$$
\begin{aligned}
m_1 y_1'' + f(y_1' - y_2') + k_1(y_1 - y_2) &= 0 \\
m_2 y_2'' + f(y_2' - y_1') + k_1(y_2 - y_1) + k_2(y_2 - z) &= 0
\end{aligned}
\tag{9.5}
$$

Solving these equations (Eq. 9.5) for $y_1''$ and $y_2''$ and substituting with $v_1 = y_1'$ and $v_2 = y_2'$ yields the equation system :

$$
\begin{aligned}
y_1' &= v_1 \\
y_2' &= v_2 \\
v_1' &= -\frac{f}{m_1}(v_1 - v_2) - \frac{k_1}{m_1}(y_1 - y_2) \\
v_2' &= -\frac{f}{m_2}(v_2 - v_1) - \frac{k_1}{m_2}(y_2 - y_1) - \frac{k_2}{m_2}(y_2 - z) = 0
\end{aligned} \tag{9.6}
$$

A simulation with the following parameters and initial values

| | |
|---|---|
| $m_1$ | 500 |
| $m_2$ | 50 |
| $k_1$ | 7500 |
| $k_2$ | 150000 |
| $f$ | 2250 |
| $z$ | 0 |

FIGURE 9.4. Parameters for wheel suspension model

| | |
|---|---|
| $y_1$ | -0.05 |
| $y_2$ | 0 |
| $v_1$ | 0 |
| $v_2$ | 0 |

FIGURE 9.5. Initial values for wheel suspension model

gives the results printed in Figure 9.6.

The graph shows the elongation (deformation) of the two springs in the system (Figure 9.3). Since the system is damped, the oscillations will vanish after a few seconds.

**9.5.2. The pendulum.** Another multibody example, which is used throughout these lecture notes, is the pendulum (Figure 9.7).

The equations of the pendulum are as follows (taken from [ESF98, page 150]):

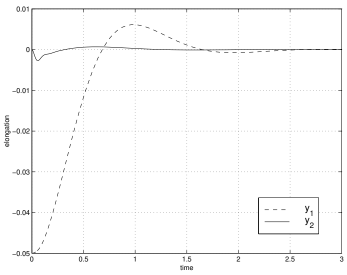



FIGURE 9.6. Simulation results for wheel suspension model

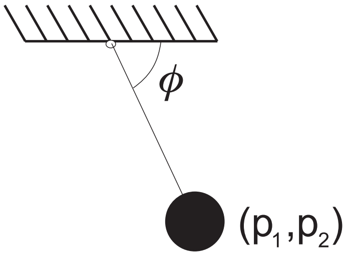



FIGURE 9.7. Pendulum

$$p_1' = v_1 \tag{9.7a}$$

$$p_2' = v_2 \tag{9.7b}$$

$$v_1' = -\lambda p_1 \tag{9.7c}$$

$$v_2' = -\lambda p_2 \tag{9.7d}$$

$$0 = {p_1}^2 + {p_2}^2 - 1 \tag{9.7e}$$

$$0 = p_1 v_1 + p_2 v_2 \tag{9.7f}$$

$$0 = {v_1}^2 + {v_2}^2 - \lambda({p_1}^2 + {p_2}^2) - p_2 g \tag{9.7g}$$

The first 4 equations (Eq. 9.7a-Eq. 9.7d) represent the equation of motion for the pendulum, written in $(x,y)$-coordinates (or $(p_1, p_2)$) and transformed to 1. order differential equations.

Together with a restriction to the length of the pendulum (Eq. 9.7e) the corresponding DAE system becomes an index-3 formulation. Differentiating the restriction (Eq. 9.7e) yields the restriction (Eq. 9.7f), which is equal to Eq. 3.5). This is an index-2 formulation, while a further differentiation to the restriction (Eq. 9.7g) will give the resulting DAE system index 1 (se also Eq. 3.6). Further differentiation of the index-1 problem yields an ordinary differential equation system, or simply index-0. This is also called the State-Space Form (Eq. 9.8). In some literature ([Eic92, ]) this form is also called the minimal-coordinates form.

$$\begin{aligned} \phi_1' &= \phi_2 \\ \phi_2' &= -g\phi_1 \end{aligned} \tag{9.8}$$

As one can see from Eq. 9.8, this form is an ordinary differential equation system.
See section 3.2.1 for the analytical results of index reduction.

**9.5.3. Numerical solution.** A simulation of the index-0 or State-Space Form is shown in Figure 9.8.

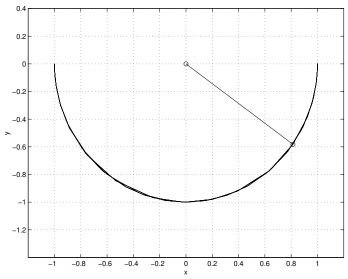



FIGURE 9.8. Solution for index-0 formulation.

A simulation of the index-1 formulation is shown in Figure 9.9.

For both simulations the Matlab function "ode23tb" (stiff systems) is used for integration. In the latter example we see the effect of drift-off because of the preceding index reduction (see section 3.2.1).

Different numerical methods are more or less efficient when it comes to solving the DAE form (Eq. 9.7a-Eq. 9.7g). In general it is harder to solve the index-3 formulation than the index-1 formulation.

## 9.6. Problems

One factor, that causes problems for many numerical methods, is algebraic loops. When deriving the mathematical model from a model, the structure of the

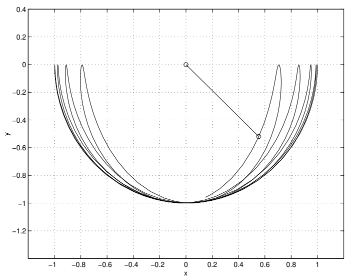



FIGURE 9.9. Solution for index-1 formulation.

model together with the formalism can result in algebraic loops in the mathematical model. This means, that certain variables are over-specified. This can cause the numerical method to break down.

There are different algorithms, that can eliminate algebraic loops and make the problem solvable.

## 9.7. Multibody Software

Dymola is a program package, that can be used to simulate multibody systems. It is developed by Dynasim in Lund, Sweden, and runs on both UNIX and Windows machines. Its graphical user interface gives good support to the multibody concept.

CHAPTER 10

# DAEs in Energy System Models

***Written by Brian Elmegaard***

**Keywords:** *Modeling energy systems, control volume, lumped models, network analysis, Nordsieck form of BDF, step-size control, discontinuities in DAEs.*

Energy systems are in this context technical installations in which fluids are employed to transport energy between the mechanical devices and thereby determine the work of the installation. Common examples of energy systems are power plants, combustion engines, refrigerators, air conditioning, district heating and many industrial production processes. The devices used in energy systems are for instance heat exchangers, pumps, turbines, valves, compressors and fans. Many models of these systems are used for optimization and steady state operation, but more and more often dynamic simulation becomes of interest.

## 10.1. Models of Energy System Dynamics

One assumption that distinguishes energy system models from models of fluid mechanics is the lumpedness of devices in energy systems. In a lumped model a whole device is enclosed in a *control volume* (see figure 10.1) which is designated as a *component*. To form a whole system the components are connected in a *network* by equalizing inlet to one and outlet from the preceding one.

The models are used to determine pressures, mass flows and temperatures (enthalpies/energies) in the system. The system of equations will be stiff, because the time constants for pressure, mass and temperature changes are different in orders of magnitude. Temperature changes are slow compared to mass changes which in turn are slower than pressure changes. Pressure changes propagate at the speed of sound, and may be assumed to be instantaneous.
The resulting system of equations will consist of

- mass, energy and momentum balances for all control volumes
- averages for inlet/outlet values as representative for internal component values

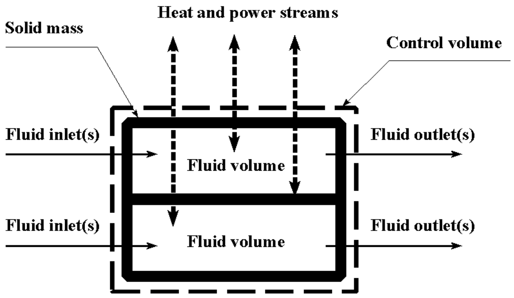



FIGURE 10.1. The control volume

- constitutive relations which correlates the working principle of each device

The constitutive relations are most often empirical relations for transfer of energy, pressure losses, chemical reactions ...

Another characteristic property of energy systems is the thermodynamic state relations which anywhere in a plant connect the state variables pressure, temperature, enthalpy, entropy, density and so on. These relations are the reason why only pressure and enthalpy are needed as global variables in a system. Enthalpy is chosen because it is generally applicable whereas temperature does not supply sufficient information during phase transition which is of great importance in many systems.
The mathematical formulation of the conservation laws are as follows:
Conservation of mass is applied for any separate fluid flow through a component.

$$\sum_{\text{flows}} \dot{m}_i = \frac{dM}{dt} \tag{10.1}$$

$\dot{m}$ is a mass flow. In formulation in- and outlet flows will have opposite signs. $M = V\bar{\rho}$ is the mass of a volume, $V$, of a fluid in a component. $\bar{\rho}$ is an average of the density of the fluid in the component and $t$ is time.

Conservation of energy only accounts for thermal energy of the fluid. Changes in potential or kinetic energy may often be neglected.

$$\sum_{\text{flows}} \dot{m}_i h_i + \dot{Q} + \dot{P} + \dot{W} = \sum_{\text{masses}} \frac{dU_i}{dt} + \frac{dU_S}{dt} \tag{10.2}$$

$h$ is the enthalpy of a fluid, $\dot{Q}$, $\dot{P}$ and $\dot{W}$ are heat flow, electrical power and mechanical power respectively. $U_i = M_i u_i$ indicates internal energy of a fluid mass with specific internal energy $u_i$. $U_s$ is likewise the internal energy of the material of the device.

Conservation of momentum is due to the lumping assumption subject to a constitutive description by frictional, gravitational and accelerational pressure changes and power input/output.

$$\Delta p = f(\dot{m}, \Delta H, \Delta\rho, \dot{P}, \dot{W}) \tag{10.3}$$

$p$ is pressure, $\Delta H$ is the height difference between inlet and outlet of the component.

The resulting system of equations is a semi-explicit index-1 DAE. The equation system could naturally be written in the more familiar, general DAE form

$$\mathbf{M}\mathbf{y}' = \mathbf{f}(t, \mathbf{y}, \mathbf{z}) \tag{10.4}$$

where $\mathbf{M}$ is a constant matrix.

However, this formulation does not indicate the exact nature of the system. Each equation may depend on more than one of the derivatives, but the system may be formulated so each equation is explicit in one of the derivates. Such a formulation is provided below:

$$\begin{array}{rcl} \mathbf{y}' & = & \mathbf{F}(t, \mathbf{y}, \mathbf{z}, \mathbf{y}'^{\star}) \\ \mathbf{0} & = & \mathbf{G}(t, \mathbf{y}, \mathbf{z}) \end{array} \tag{10.5}$$

$\mathbf{y}$ contains differential variables, $\mathbf{y}'$ their time derivatives, and $\mathbf{z}$ the algebraic variables. $\mathbf{y}'^{\star}$ on the right hand side of the differential equations indicates that time derivatives of other variables may be present in the equations.

It is worth to mention that discontinuities are often present in energy systems. One common example is the opening and closing of a valve.

## 10.2. Example Simulations with DNA

The following examples have been carried out with the program DNA which is a general energy system simulator [Lor95]. DNA is an abbreviation for Dynamic Network Analysis, implying that the energy system is regarded as a network of components, similar to the way electrical circuits are often modeled.

**10.2.1. Numerical methods of DNA.** This section describes the numerical methods implemented in DNA. The methods have been carefully selected as the most appropriate for application to the energy system models described above.

A BDF-method (Eq. 10.6) has been chosen as integration method because it is most efficient for the semi-explicit index-1 case. In the scalar case the BDF of order $k$ for step $n$ is given as:

$$\sum_{i=0}^{k} \alpha_i y_{n-i} = \hat{h}\beta_k f(t_n, y_n) \tag{10.6}$$

with $\hat{h}$ being the time step. For further details of BDF-methods consult chapter 4

The order of the method has been limited to four to have a large enough stability region.

The method is implemented as a predictor-corrector method with truncation error estimation and time step control. The error estimate may for multi-step methods be determined by Milne's Estimate, provided the methods are of the same order:

$$\varepsilon = \frac{C_{k+1}^{C}}{C_{k+1}^{C} - C_{k+1}^{P}} \left( y_n^C - y_n^P \right) \tag{10.7}$$

where $C^C$ and $C^P$ are error constants of the corrector and the predictor respectively. $y^C$ and $y^P$ are the values of corrector and predictor.

As BDF-methods are based on backward differences, time step changes require cumbersome interpolation. Due to this the Nordsieck formulation of the BDF-method has been chosen. The Nordsieck formulation is based on a Taylor expansion back in time of the solution to the equation system. For each differential variable in the system a Nordsieck vector is defined by:

$$\mathbf{Y}(j) = \frac{\hat{h}^j}{j!} \cdot y_i{}^{(j)} \quad \text{for} \quad j = 0, k-1 \tag{10.8}$$

This simplifies step size control because each entry of the Nordsieck vector only have to be multiplied by the ratio, $\alpha$ between a new time step and a rejected time step.

$$\mathbf{Y}(j) = \frac{(\alpha \cdot \hat{h})^j}{j!} \cdot y_i{}^{(j)} \quad \text{for} \quad j = 0, k-1 \tag{10.9}$$

Also Milne's estimate may be calculated in an easy way due to the Nordsieck formulation. Firstly, the predicted value should be observed to be given by the corrected value of the previous time step.

$$\mathbf{Y}_{i,n+1}^{P} = \mathbf{P} \cdot \mathbf{Y}_{i,n}^{C} \,\text{for variable}\, i \tag{10.10}$$

where the $\mathbf{P}$ is the Pascal Matrix with elements defined recursively by

$$\mathbf{P}(r,s) = \begin{cases} \mathbf{P}(r-1,s) + \mathbf{P}(r-1,s-1), & \text{for } r < s \text{ and } r > 1 \\ 1, & \text{for } r = s \text{ and } r = 1 \\ 0, & \text{otherwise} \end{cases}$$

A new variable $w$ is introduced to the system as $w_i = Y(2)_i^C - Y(2)_i^P$. $Y(2)$ equals $hy_i'$. The difference in the error estimate may be expressed as $w_i\mathbf{l}(1)$, with $\mathbf{l}$ being a constant vector dependent on the order of the method.

The size of a new time step is calculated as

$$\hat{h}_{n+1} = c\hat{h}_n \sqrt[k+1]{\frac{\varepsilon_{BDF}}{d}} \tag{10.11}$$

$\varepsilon_{BDF}$ is the maximum allowed truncation error, $d$ is a norm of the actual truncation error estimate, $c$ is a constant that limits the step size change to avoid too frequent changes of the sign of the step size ratio.

The equation system may be formulated on residual form including equations for the corrector values and the differences in the error estimate. This forms a non-linear equation system, which may be solved iteratively. In DNA a modified Newton method has been implemented.

Discontinuities should be handled in a robust way. The way to do so, is to determine the point in time where the discontinuity is situated, integrate up to this point and restart the method at the discontinuity [ESF98]. Discontinuities may be divided into time-dependent and variable-dependent ones. The first kind may be explicitly specified in the system. The other kind is more difficult to handle, because it may be defined in a complicated way by differential and/or algebraic variables. To handle such a discontinuity switch variables have to be included in the system. These variables are defined to change sign at the discontinuity. When the integration method encounters a change of sign in a switch it should by iteration determine the exact location of the discontinuity and hereby efficiently find the correct point to restart at. For efficiency reasons, it is important that the method is restarted at a time instance where the switch has changed its sign. The procedure of obtaining the exact location of the switch is based on knowledge of the previously calculated values of the switch variable. As basis two points, one before and one after the switch are calculated. From these two values of the switch variable the value of a better approximation to the switch time may be determined. This new value may then be applied in an iteration procedure until a precise value of the sign change of the switch variable has been obtained. The approximation may be determined by iteration in the switch variable until

$z_s(t) = 0$. However, a more efficient approach is to interpolate the inverse function $D(z_s) = z_s^{-1}(t)$. The value of this function at $z_s = 0$ will be an approximation to the switch time.

**10.2.2. Example 1: Air flowing through a pipe.** This first example is rather simple, but it gives an impression of the equations applied for heat transfer calculations. The model describes the behavior of hot air flowing through a pipe initially at a low temperature. Due to the temperature difference heat is transferred from the fluid to the wall of the pipe.
Mass balance

$$\dot{m}_i + \dot{m}_o = \frac{dM}{dt} \tag{10.12a}$$

Energy balance

$$\dot{m}_i \cdot h_i + \dot{m}_o \cdot h_o - \frac{dU}{dt} = \frac{dU_s}{dt} \tag{10.12b}$$

Internal energy of the air

$$U = M \cdot \frac{u_i + u_o}{2} \tag{10.12c}$$

Total mass of fluid in the pipe

$$M = V \cdot \rho \tag{10.12d}$$

Heat transfer is governed by a heat transfer coefficient, $k$, and the transferring area, $A$. The main problem of the model is this equation because of the log mean temperature difference, which may easily end up trying to take logarithms to negative numbers during Newton iterations.

$$\dot{m}_i \cdot h_i + \dot{m}_o \cdot h_o - \frac{dU}{dt} = kA \frac{(T_i - T_s) - (T_o - T_s)}{\ln \frac{T_i - T_s}{T_o - T_s}} \tag{10.12e}$$

For air flows one could apply ideal gas laws for the fluid property calculations

$$T = f_1(p, h)$$

$$\rho = f_2(p, h)$$

Figure 10.2 shows the progress of mass of air in the pipe and temperatures of fluid and pipe material. Considering DAEs and application of numerical methods to them, figure 10.3 may be be of more interest. It shows the time step determined as optimal for the fourth order BDF of DNA. The conclusion is that the time step is generally increasing because the solution approaches a steady state. About time step 80 the step is almost kept constant. The reason for this is that the Newton method about this point only converges to a value near, but lower than, the allowed maximum.

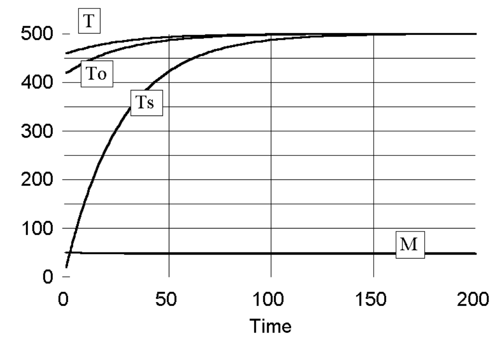



FIGURE 10.2. Temperature of air in pipe, $T$, air at outlet, $T_o$, pipe wall, $T_s$ and mass of air in pipe $M$.

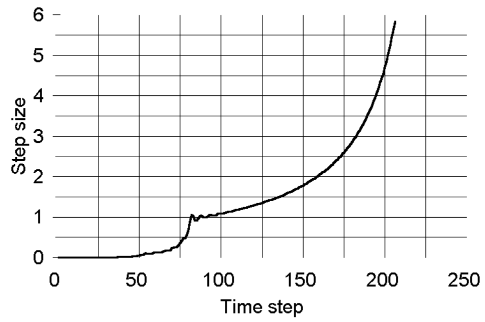



FIGURE 10.3. Time step size for the air flow model

**10.2.3. Example 2: Steam temperature control.** In a power plant boiler the temperature of steam to the turbine has to be controlled carefully, due to material limitations. This is done by injecting water to the steam leaving the boiler and hereby lower the temperature. This process is named attemperation. The following example shows such a temperature control. The purpose of the example is to show the discontinuity handling of the integration routine. Therefore, the temperature of the inlet steam is set to a value outside the proportional gain of the PI-controller. This causes the controller to increase the control signal until it is one and stay at that value. The model is shown in figure 10.4. In the model

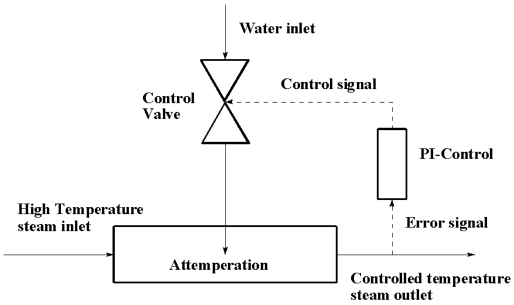



FIGURE 10.4. Steam attemperation

mass and energy balances are of course applied. Furthermore the model involves the following equations.
Attemperator error signal is the difference between the actual temperature and the set point of the temperature.

$$e = T_o - T_{SP} \tag{10.13a}$$

PI-control:
The control signal is the sum of an integral and a proportional part. However it is limited to be between zero and one.

$$z_c = m_p + m_I, \qquad \text{for} \qquad 0 \le z_c \le 1 \tag{10.13b}$$

Proportional control signal.

$$m_p = K_p \cdot e \tag{10.13c}$$

Integral control signal.

$$\frac{m_I}{dt} = K_I \cdot e \tag{10.13d}$$

Switch variable.

$$z_s = \begin{cases} 1 - (m_p + m_I) & , \quad m_p + m_I \le 1 \\ (m_p + m_I) - 1 & , \quad m_p + m_I > 1 \end{cases} \tag{10.13e}$$

Control valve:
The opening of a valve determines the mass flow of water to have the correct steam temperature.

$$\dot{m} = C \cdot z_c \cdot \sqrt{\frac{p_i - p_o}{v_1}} \tag{10.13f}$$

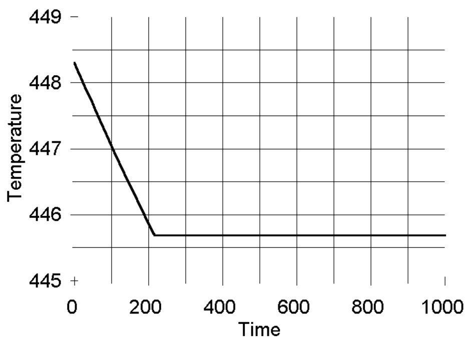



FIGURE 10.5. Temperature at outlet of attemperator

Figure 10.5 shows the temperature of the steam leaving the attemperator. The discontinuity is easily seen from this graph. In figure 10.6 the control signals are shown. It is seen that the integrating signal is increasing after the discontinuity, and so is the sum of proportional and integrating control. However, the controller's output signal cannot exceed one and remains so. The graph also shows the switch variable value. It is seen that this variable reaches zero at the discontinuity. After the discontinuity the sign of the switch variable is changed so it is positive due to the implementation. It then increases as the sum of the control signals.

The iterative procedure to determine the location of the discontinuity is displayed in table 10.1. The first entry is the step before the switch changes sign. The second entry is the step the determines that a discontinuity is going to be passed. The three following steps show how the approximation to the switch point becomes more and more precise and eventually is of the order of the wanted precision.

Figure 10.7 shows the step size chosen in DNA. It is seen that at the discontinuity the integration is restarted at order one with the initial step size. This

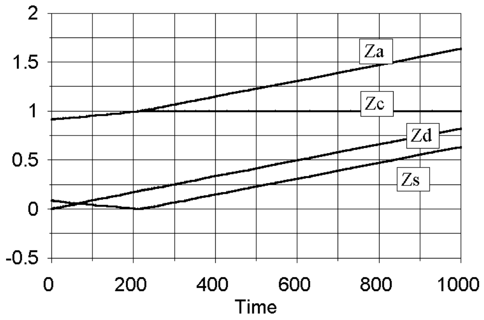



FIGURE 10.6. Control signal, $Z_c$, control signal value demanded by the system, $Z_a$, integrating control signal, $Z_d$, and switch variable value, $Z_s$.

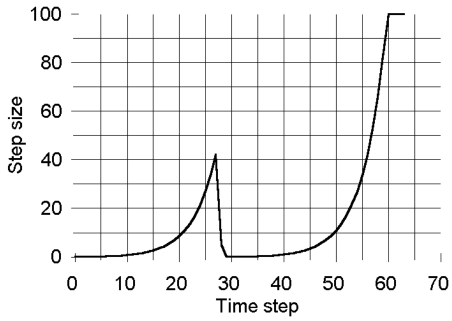



FIGURE 10.7. Step size for attemperator model

choice of step after the restart may be optimized. For instance, could the optimal value found at the first time step, which is first order, have been selected to improve the speed of the method.

| Time | Switch variable value |
|---|---|
| 211.55 | $0.19{\cdot}10^{-2}$ |
| 264.50 | -0.0176 |
| 216.9019 | -2.0e-5 |
| 216.8363 | 5e-10 |
| 216.8363 | -1e-15 |

TABLE 10.1. Iterating to locate the exact switching point.

# Bibliography


[AP98] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1998.

[BCP89] K.E. Brenan, S.L. Campbell, and L.R. Petsold. *Numerical Solutions of Initial-Value Problems in Differential-Algebraic Equations*. North-Holland, 1989.

[But87] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations*. John Wiley & Sons, 1987.

[CA94] G. Söderlind C. Arevalo, C. Führer. Stabilized Multistep Methods for Index 2 Euler-Lagrange DAEs. *BIT-Numerical Mathematics*, December 1994.

[DM70] James W. Daniel and Ramon E. Moore. *Computation and Theory in Ordinary Differential Equations*. W.H. Freeman and Company, 1970.

[Eic92] Edda Eich. *Projizierende Mehrschrittverfahren zur numerischen Lösung von Bewegungsgleichungen rechnishcer Mehrkörpersysteme mit Zwangsbedingungen und Unstetigkeiten*. VDI Verlag, 1992.

[ESF98] E. Eich-Soellner and C. Führer. *Numerical Methods in Multibody Dynamics*. B. G. Teubner, 1998.

[HLR89] E. Hairer, C. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*. Lecture Notes in Mathematics. Springer-Verlag, 1989.

[HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer-Verlag, second edition, 1993.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer-Verlag, 1991.

[HW96] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer-Verlag, second revised edition edition, 1996.

[KN92] D. Stoffer K. Nipp. Research Report No. 92-14: Invariant manifolds of numerical integration schemes applied to stiff systems of singular perturbation type – Part I: *RK*-methods. Seminar für Angewandte Mathematik, Eidgenössische Technische Hochschule, Switzerland., December 1992.

[KNO96] Anne Kværnø, Syvert Paul Nørsett, and Brynjulf Owren. Runge-kutta research in trondheim. Technical report, Division of Mathematical Science, The Norwegian University of Science and Technology, 1996.

[Kvæ92] Anne Kværnø. More and, to be hoped, better DIRK methods for the solution of stiff ODEs. Technical report, Division of Mathematical Science, The Norwegian University of Science and Technology, 1992.

[Lam91] John Denholm Lambert. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 1991.

[Lor95] B. Lorentzen. *Power Plant Simulation*. PhD thesis, Laboratory for Energetics, Technical University of Denmark, 1995.

[MKZ92] D. Matko, R. Karba, and B. Zupancic. *Simulation and Modelling of Continuous Systems*. Prentice Hall, 1992.

[NGT93] H.B. Nielsen and P. Grove Thomsen. *Numeriske metoder for sædvanlige differentialligninger, hæfte 66* . Numerisk Institut, 1993.

[PR94a] W.C. Rheinboldt P.J. Rabier. On the Computation of Impasse Points of Quasi-Linear Differential-Algebraic Equations. *Math. Comp.*, 62(205):133–154, January 1994.

[PR94b] W.C. Rheinboldt P.J. Rabier. On the Computation of Impasse Points of Quasi-Linear Differential-Algebraic Equations. *Jour. Math. Anal. App.*, (181):429–454, 1994.

[Rhe84] W.C. Rheinboldt. Differential-Algebraic Systems as Differential Equations on Manifolds. *Math. Comp.*, 43(168):473–482, October 1984.

[RO88] jr. R.E. O'Malley. On Nonlinear singularly perturbed initial value problems. *SIAM Review*, 30:193–212, 1988.

# Index